ISSN: 0976-9102 (ONLINE)
DOI: 10.21917/ijivp.2019.0276

COMPUTATIONAL MODEL FOR CREATING NEURAL NETWORK DATASET OF EXTRACTED FEATURES FROM IMAGES CAPTURED BY MULTIMEDIA SECURITY DEVICES

Etemi Joshua Garba and Darious Tienhua Chinyio
Department of Computer Science, Nigerian Defence Academy, Nigeria

Abstract

Whenever multimedia security devices, such as the Closed-Circuit Television (CCTV), capture images, the decisive analysis is usually left for the human expert to determine the content therein and suggest the necessary action to be taken. However, the application of Artificial Intelligence (AI) helps in complementing human efforts in carrying out such analysis. This is achievable if the required input dataset is created by extracting features from the identifiable objects in the images. The extraction of such features is based on regional properties of objects within an image – using technique such as the Gray-Level Co-occurrence Matrix (GLCM). This dataset is consequently used in AI platforms that are based on Artificial Neural Network (ANN) and Neuro-Fuzzy systems (specifically in Adaptive Neuro-Fuzzy Inference System (ANFIS)). This paper is presenting a computational model for creating training and testing datasets. For the simulation of the model, Matlab was used, however, the computational model is realizable via other programming and numerical computing environments.

Keywords:

Computational Model, Features Extraction, Training and Testing Datasets

1. INTRODUCTION

The computational model in this paper describes how a dataset is created as output from a given input of image file captured by a multimedia security device. The model proposes some steps of computations that form an algorithm. Each step in the algorithm represents a unit of computation that could consist of a single instruction or a set of instructions in form functions. Though the final output of the model is a dataset; however, the result at the end of every step of the algorithm is outputted for debugging, validation and verification. Over the years, the need to automate the process of recognizing patterns and characters has increasingly become imperative especially within security organizations.

To achieve pattern and character recognition, the algorithms must be intelligent enough to distinguish between the foreground (that contains most relevant patterns or characters) and the background (that could be made of unwanted patterns that constitute noise). Therefore, whenever an image is captured, it has to undergo some sort of preprocessing. Noisy, inconsistent or incomplete images need to be reprocessed in order to improve the effectiveness of features extraction techniques. Inconsistencies, especially due to variations in contrast and light intensities need to be fixed be relevant features are extracted from the image. the median filtering technique is prevalently used in noise elimination and reduction.

Once the noise has been taken care of, the image needs to be enhanced using the histogram equalization method. This results in a balanced dynamic range of gray levels - such that images do not seem over bright (whitewashed) or over darkened. Features from images are extracted and then stored as vectors or scalar values that form a dataset. It is the dataset that is afterwards used as input for the neural network or neuro-fuzzy system. However, it is imperative to note that only relevant features should be considered for extraction. One of the commonly used methods for extracting texture features is the Gray-level co-occurrence matrix (GLCM) which describes patterns of gray-level repetition. This method was introduced by Haralick et al. [17]. The co-occurrence matrix contains information about the orientation and distance between the pixels. If given a two dimensional image $f(x,y)$ with the size of $M \times N$, then the co-occurrence matrix is constructed using Eq.(1).

$$P(i,j|d,\theta) = \#\{[(x_1,y_1),(x_2,y_2)] \in M \times N | d,\theta, f(x_1,y_1) = i, f(x_2,y_2) = j\} \quad (1)$$

where, # denotes the number of the elements of the set. While d and θ denote the distance and angle between x_1, y_1 and x_2, y_2 respectively. Once the matrix is constructed, several features could be extracted. Some of these features are contrast, correlation, energy, and homogeneity [18]. See the following Eq.(2)-Eq.(5).

$$Contrast = \sum_{i=1}^{L-1}\sum_{j=1}^{L-1}(i-j)^2 P(i,j|d,\theta) \quad (2)$$

$$Correlation = \frac{\sum_{i=1}^{L-1}\sum_{j=1}^{L-1} i \cdot j \cdot P(i,j,d,\theta) - \mu_x \mu_y}{\sigma_x \sigma_y} \quad (3)$$

$$Energy = \sum_{i=1}^{L-1}\sum_{j=1}^{L-1}[P(i,j,d,\theta)]^2 \quad (4)$$

$$Homogeneity = \sum_{i=1}^{L-1}\sum_{j=1}^{L-1}\frac{P(i,j,d,\theta)}{1+|i-j|} \quad (5)$$

where, μ_x, σ_x and μ_y, σ_y are the mean and standard deviation of pixel value in the row and column directions respectively.

Extraction of patterns or character, especially from images with low resolutions, complex backgrounds and varying textual styles and font sizes is quite a daunting task. Besides using the GLCM method to extract features, it is also possible to extract features based on shape properties like the Euler Number - where the number of pixels for each region that make up the pattern or character is calculated. The unwanted regions especially the non-text regions are discarded. Some of the shape properties and geometric features extracted using the regionprops function include Convex Hull, Eccentricity, Extent, Orientation, Solidity and Eulerumber [19].

2. RELATED WORKS

Feature extraction involves the measurement of the most relevant details of an object that contains either some patterns or characters. To extract the features, various methods/techniques are used [1]. Many authors have acknowledge that the process of feature extraction and character recognition is not an easy task owing to the fact that patterns, especially characters come in various shapes, texture, styles, font types and sizes. It is because of the aforementioned that multiple feature extraction techniques (such as GLCM, Regional Properties, Histogram etc.) are used [2]. The analysis, detection and extraction of the features in images (especially those images with complex background) pose quite a great challenge in the field of digital image processing [3]. In order to successfully extract features, the boundaries of such patterns must be determined. These boundaries define edges that constitute sharp discontinuity in an image. Note that the process of identifying the edge of a pattern/character could be impeded by the presence of noise. Some of the prevalent techniques for edge detection are spatial derivation and image thresholding [4]. The extracted features are useful during the classification of patterns/character at the recognition stage. In their work, Rajashekararadhya and Vanajaranjan [5], [6] proposed diagonal feature extraction scheme for recognizing off-line handwritten characters. By extracting features from a given image, the data to be later used for classification is thus reduced. The data represents properties of the pattern under investigation [7]. The complexity (in terms of amount of memory and processing time required) increases as the number of features extracted grows bigger. Therefore, only features of utmost relevance should be selected and then extracted [8]. Usually, extracted features are stored in form of vectors to be used during classification and recognition stages [9, 10]. According to Kumar and Bhatia [11], the process of feature extraction should be insensitive to irrelevant features so as limit the number of training data which results in more efficient computations. In this paper, GLCM is used as a method for extracting features based the spatial relationship of pixels in the gray-level co-occurrence matrix. The GLCM functions calculate how often pairs of pixel with specific values and in a specified spatial relationship occur in an image. This way, the characteristics of the texture of the pattern in an image is determined [12]. The extracted features are stored in tabular form such that the first columns are the inputs while the last column is the output. This table of the extracted feature is considered as training or testing dataset to be used in neural network or machine learning platform [13], [14]. In this paper, the dataset is expected to be used in ANFIS of Matlab. Sandhya and Pavithira [15] have opined that the dataset to be used for training, evaluation or validation of model should be noise-free. The evaluation of the performance of the model strongly depends on the dataset used in the ANFIS system. Proper selection of the training dataset ensures that the testing dataset would eventually validate the model. The differentiation between the training and testing datasets should be minimal such that the testing error is low at the first epoch; otherwise the model would not be validated. A good dataset allows decrease of testing error at every epoch until it reaches the jump point (beyond which overfitting occurs) [16].

3. METHODS AND MATERIALS

The techniques used in extracting features are based on the Gray-Level Co-occurrence Matrix and the Regional properties of objects within an image. Subsequently, the training and testing datasets are created from the extracted features. The tools and materials used in this paper include:

- Multimedia security device (as hardware tool for capturing images).
- Image files (for creating training and testing dataset).
- MathWorks Matlab (as a multi-paradigm numerical computing environment).
- Microsoft Excel (as software tool for storing and organizing the dataset of the extracted features).

Fig.1. Image of characters used for creating training datasets

The methodology employed in this paper is based on the design of a computational model for extracting features from image files and then storing them in a dataset. This model forms the basis of a software framework that is realizable through the development of fully functional program. In this paper, however, the model is simulated using Matlab - to demonstrate the workability of the proposed model. The model is made of thirteen steps.

Step 1: Capture image using multimedia security device that saves the image to a file on the storage device. The image file formats supported herein include jpg, png, tiff etc.

Fig.2. Original captured image (with noise)

Step 2: Convert image in RGB format into gray scale format.

Fig.3. Comparison between the original colour image (left) and gray scale image (right)

Step 3: Use histogram equalization to enhance the gray scale image. Histogram equalization is a technique that is used for adjusting image intensities in order to enhance its contrast.

Fig.4. Comparison between the gray scale image (left) and the enhanced gray scale image (right)

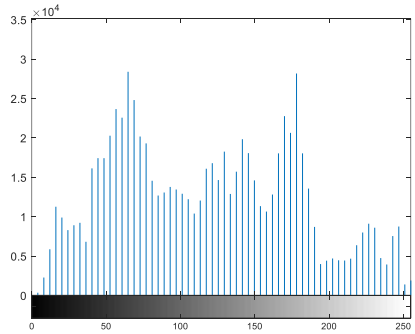

Fig.5. Histogram of the original gray scale image

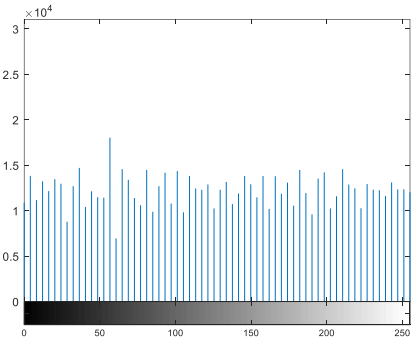

Fig.6. Histogram of the enhanced gray scale image (having balanced distribution of intensities of the shades. The enhanced image helps in better conversion to binary image)

Step 4: Use median filtering to eliminate noises.

Fig.7. Comparison between the enhanced gray scale image (left) and noise filtered gray scale image (right).

Step 5: Convert the noise filtered image to binary image. The binary image is represented using either black (0 values) or white (1 values) shades.

Fig.8. Binary image (enhanced gray image is converted to black [zeros] and white [ones])

Step 6: Determine the number of objects (white regions) and number of holes (black regions). Naturally, the number of objects (foreground) is expected to be greater than the number of holes (background).

Step 7: Removal of boundary connected objects that constitute spurious/false pixels. This helps to reduce the number of unwanted segmentations of the image.

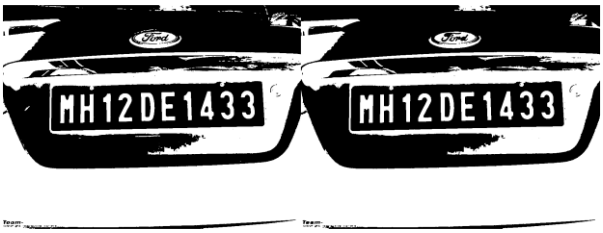

Fig.9. Comparison between the raw binary image (left) and the clear binary image (right)

Step 8: Labeling of all the connected pixels and objects segmentation. This is where patterns/characters are identified.

Step 9: Edge detection and measurement of the properties of the regions of the image and plotting of the bounding box.

Fig.10. Labeling and segmentation of connected pixels with detection of edges of the objects

Step 10: Character and Objects extraction.

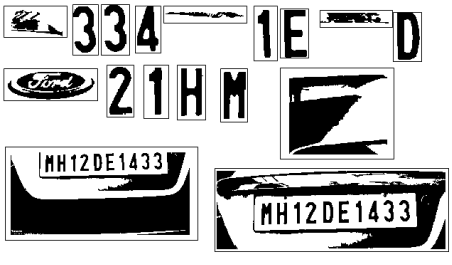

Fig.11. Extracted objects based on the segmentation

Step 11: Morphological manipulation of the binary image. The morphological operations to be used here is dilation. Dilation adds pixels to the boundaries of objects in an image.

Step 12: Skeletonization and thinning of objects/characters. This process constructs the most basic structure of the extracted object/character (regardless of the font type, style and size).

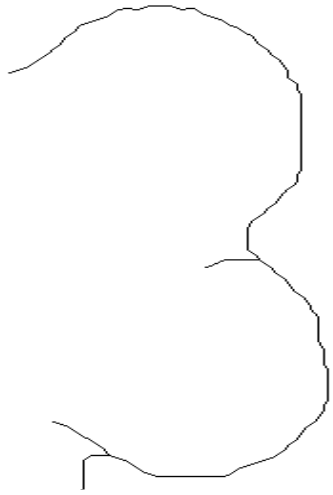

Fig.12. Sample of skeletonized and thinned number "3"

Step 13: Features extraction using various techniques. The features are stored in tabular form to create training and testing datasets.

- Feature 1: Centroid X - Center of mass of the region for the horizontal coordinate (or x-coordinate).
- Feature 2: Centroid Y - Center of mass of the region for the vertical coordinate (or y-coordinate).
- Feature 3: Filled Area - Number of on pixels in filled image that is the same size as the bounding box of the region.
- Feature 4: Equivalent Diameter - Diameter of a circle with the same area as the region that is computed as sqrt(4*Area/pi).
- Feature 5: Solidity - Proportion of the pixels in the convex hull that are also in the region.
- Feature 6: Extent - Ratio of pixels in the region to pixels in the total bounding box.
- Feature 7: Perimeter - Distance around the boundary of the region.
- Feature 8: Correlation - a measure of how correlated a pixel is to its neighbor over the whole image. The range is between -1 and 1.

Note: Features 1 to 7 are based on Regional Properties of objects within an image, while Feature 8 is based on Gray-Level Co-occurrence Matrix (GLCM).

4. RESULTS

The training and testing datasets in Table.1 and Table.2 have been generated using the computational model. The first eight columns represent the input while the last column is for the output. Note that these datasets are considered as small owing to the fact that the number of inputs is quite few. Even though more inputs implies that more features would be extracted that invariably would improve positive recognition; however, computational complexity (as per memory and processing time requirements) normally increase with increase in the number of inputs. The increase in computational complexity is owed to the fact that for the size of the weights between the hidden layer(s) and the output layer may grow so large during training and testing of the model in Fig.16.

Croda et al. [20] discovered that a small dataset does not significantly increase the number of iterations (computational complexity) - regardless of the maximum possible learning rate. As such, using a small dataset does not lead to time convergence problem since the number of iterations is expected to relatively small. Note that there is a proportionate relationship between the size of the dataset [vis-à-vis the number of inputs and outputs] and the number hidden layers (neurons).

Karsoliya [21] stated that the process of deciding the number of hidden layers and number of neurons in each hidden layer is confusing. Karsoliya discovered that if the data is linearly separable, then hidden layer is not required. However, problems which deal with arbitrary decision and accuracy require two or three hidden layers. Therefore, if the number of neurons is few vis-à- vis the complexity of the problem domain, then underfitting may likely occur (because the few neurons are unable to adequately detect the signals in a complicated dataset). On the other hand, overfitting would occur if there are unnecessary neurons present in the network. Over the years several authors have suggested methods for determining the number of neurons in the hidden layer [21][22]. This strongly depends on the size of the dataset. Some of the methods suggest that:

- The number of neurons in a hidden layer should be 2/3 (about 70% to 90%) of the size of the input layer.
- The number of neurons in a hidden layer should be less than twice of the number of neurons in input layer.
- The size of neurons in the hidden layer should be between the input layer size and the output layer size.
- The number of neurons in first and second hidden layers should be kept nearly equal so that the network could be trained easily.

It should be noted that in neuro-fuzzy systems, the number of neurons and number of hidden layers is not only dependent on the size of the inputs of the dataset, but also on the number of membership functions per input. For instance, Fig.14 shows three membership functions for each input.

5. DISCUSSIONS

The generation of the training and testing datasets is an important stage in using ANFIS for pattern/character recognition. The data from the datasets are expected to be fed into the neuro-fuzzy system to model training and validation in Fig.13-Fig.16. The same dataset could be used in a classic neural network in Fig.17 and Fig.18.

ISSN: 0976-9102 (ONLINE)
DOI: 10.21917/ijivp.2019.0276

Table.1. Sample of the training dataset

Input								Output
Centroid X	Centroid Y	Filled Area	Equivalent Diameter	Solidity	Extent	Perimeter	Correlation	
55.5648	85.4922	193	15.6759	0.0424	0.0220	396.8660	0.1555	6
71.1468	101.9602	327	20.4046	0.0235	0.0212	740.9000	0.3833	9
73.7815	97.0728	357	21.3201	0.0246	0.0202	823.4060	0.2697	13
90.2247	94.3499	583	27.2452	0.0280	0.0270	1214.7740	0.0194	14
73.2147	102.3079	6109	21.2303	0.0381	0.0219	446.7040	0.2810	16
77.3795	96.5094	17372	24.6442	0.0260	0.0242	536.9580	0.2866	17
65.8874	95.6623	453	24.0162	0.0244	0.0232	932.8360	0.5941	21
81.7506	94.0668	389	22.2551	0.0222	0.0210	787.6760	0.2238	32

Table.2. Testing dataset of the extracted features

Input								Output
Centroid X	Centroid Y	Filled Area	Equivalent Diameter	Solidity	Extent	Perimeter	Correlation	
34.7840	97.5000	162	14.3619	0.1601	0.0833	325.5020	-0.0173	6
48.8588	95.8980	255	18.0188	0.0272	0.0230	565.2200	0.2598	9
56.8872	98.1907	257	18.0893	0.0266	0.0225	571.4480	0.1823	13
45.8160	96.2747	375	21.8510	0.0375	0.0334	788.1860	-0.0061	14
44.7508	112.2694	2971	19.4461	0.0538	0.0282	434.2340	0.1053	16
44.7639	96.2493	9092	21.9092	0.0409	0.0361	390.4900	0.1162	17
32.5104	97.0903	288	19.1492	0.0303	0.0276	594.2240	0.4102	21
45.5792	92.6545	385	22.1404	0.0352	0.0321	789.6300	0.0770	32

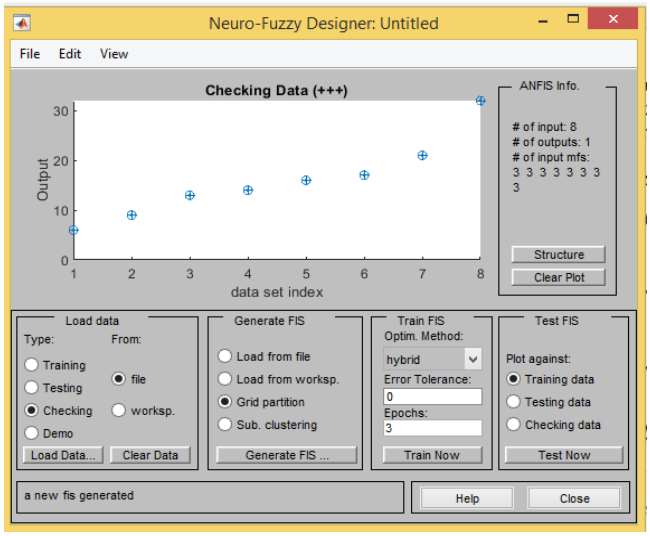

Fig.13. Loading of training and testing data unto the Neuro-Fuzzy Designer.

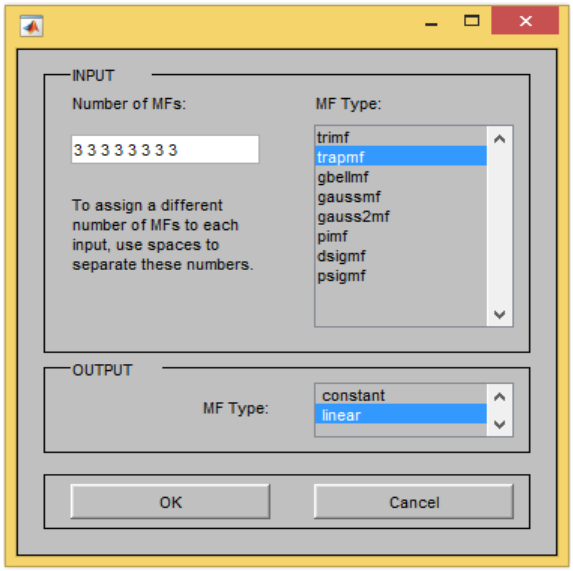

Fig.14. Membership Functions settings

Grid partition partitioning method was selected for automatic generation of the fuzzy inference system (FIS) with the ANFIS designer. The two partition methods, grid partitioning (Fuzzy C-Means Clustering) and subtractive clustering are the available options. The procedure for training the model is summarized as follows:

- Generate FIS.
- Edit Membership Functions settings in Fig.14.
- View the FIS Structure in Fig.15.

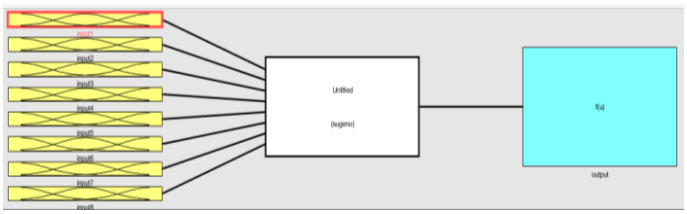

Fig.15. FIS of the training model

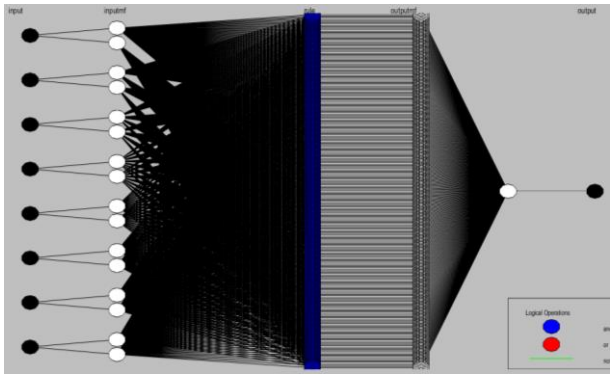

Fig.16. ANFIS structure of the model

For ANFIS training, the two optimization methods available for FIS training are hybrid (mixed least squares and back propagation) and back propagation. The Error Tolerance is used to create a training stopping criterion that is related to the error size. The training will stop after the training data error remains within acceptable tolerance. If the number of epochs is so large, it will result in too much training of the FIS model that would result in over fiting, which makes the model performs badly against new data that it has not seen before. After training, the following details were generated:

ANFIS info:
Number of nodes: 555
Number of linear parameters: 2304
Number of nonlinear parameters: 48
Total number of parameters: 2352
Number of training data pairs: 8
Number of checking data pairs: 8
Number of fuzzy rules: 256
Designated epoch number reached → ANFIS training completed at epoch 2.
Trapezoidal Membership Function:
Minimal training RMSE = 0.014580
Minimal checking RMSE = 14.7505

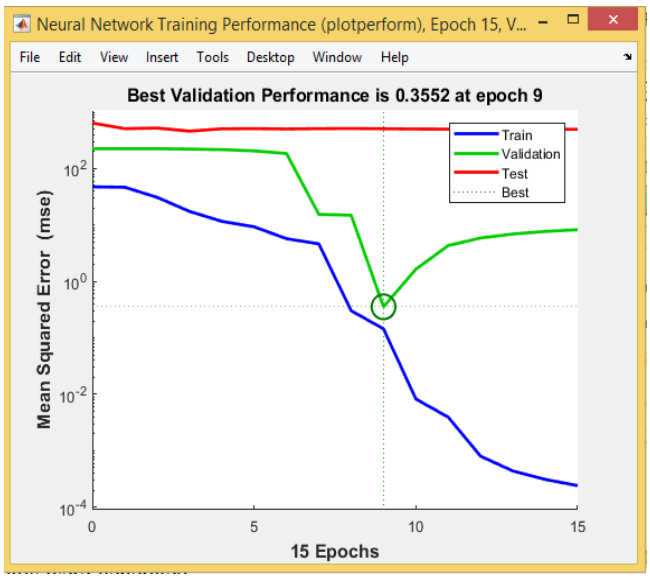

Fig.18. Neural Network Training Performance Graph

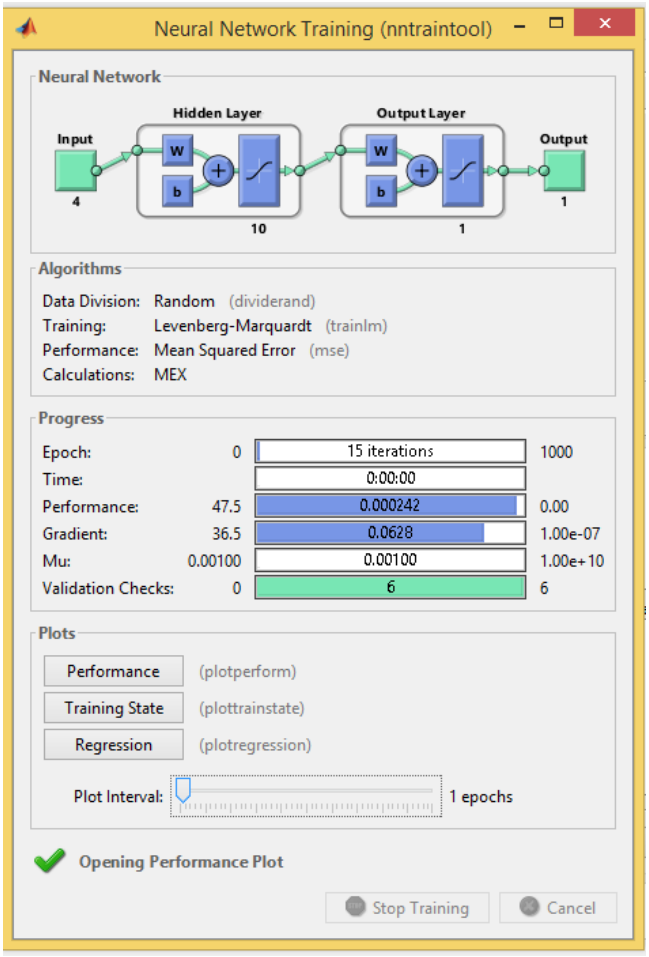

Fig.17. Classic Neural Network Training Interface

6. CONCLUSIONS

This paper presented a computational model for creating training and testing datasets to be used in AI platforms that include neural networks and neuro-fuzzy systems. Even though the simulation of the model was in MATLAB, it could be realized in other programming and numerical computing environments such as Java, C/C++, Python etc. Image processing using Python requires libraries such as PIL, Pillow, Mahotas, scikit-image, scipy, PythonMagick, pycairo, OpenCV-Python, SimpleITK. The common libraries for image processing in C/C++ are VIPS, GD2, CImg, Simd, OpenCV, Boost GIL and Magick++. Java DIP Open Source Libraries are commonly used in Java programming language. One of the most challenging aspects of pattern or character recognition is how to successfully distinguish similar patterns/characters, such as (8 vs. B), (0 vs. D vs. O), (5 vs. S) etc [23]. Singh et al. [24] have proposed a hybrid method for differentiating similar patterns/characters by comparing the angle of two pixels. In this paper however, it is expected that these types of challenges could be intelligently tackled by feeding the dataset into the neuro-fuzzy system (that creates the fuzzy inference system for model training and validation). Automatic recognition of patterns/characters from images captured by multimedia security devices is very imperative owing to the fact that the human expert has his inherent natural limitations. Therefore, artificial intelligent systems and models play a vital role in

complementing human efforts in recognizing patterns/character with speed and accuracy.

REFERENCES

[1] D. Trier, A.K. Jain and T. Taxt, "Feature Extraction Method for Character Recognition-A Survey", *Pattern Recognition*, Vol. 29, No. 4, pp. 641-662, 1996.

[2] S.A. Nirve and G.S. Sable, "Optical Character Recognition for Printed Text in Devanagari using ANFIS", *International Journal of Scientific and Engineering Research*, Vol. 4, No. 10, pp. 23-34, 2013.

[3] D. Deb, P. De and N. Debbarma, "Modified Method of Texture Feature Extraction and Analysis using Daubechies Wavelet and SVM", *International Journal of Computer Applications*, Vol. 110, No. 16, pp. 1-8, 2015.

[4] Z. Hussain, "*Digital Image Processing: Practical Applications of Parallel Processing Techniques*", Redwood Press, 1991.

[5] S.V. Rajashekararadhya and P. Vanajaranjan, "Efficient Zone Based Feature Extraction Algorithm for Handwritten Numeral Recognition of Four Popular South-Indian Scripts", *Journal of Theoretical and Applied Information Technology*, Vol. 4, No. 12, pp. 1171-1181, 2008.

[6] R.G. Casey and E. Lecolinet, "A Survey of Methods and Strategies in Character Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 7, pp. 690-706, 1996.

[7] O.B. Ali and A. Shaout, "Hybrid Arabic Handwritten Character Recognition using PCA and ANFIS", *Proceedings of International Arab Conference on Information Technology*, pp. 14-18, 2016.

[8] E. Ismayilov and U. Bakhishoff, "Hand-Printed Character/Digit Recognition by ANFIS System", *Journal of Contemporary Applied Mathematics*, Vol. 3, No 2, pp. 23-28, 2013.

[9] J. Pradeep, E Shrinivasan and S. Himavathi, "Diagonal Based Feature Extraction for Handwritten Alphabets Recognition System using Neural Network", *International Journal of Computer Science and Information Technology*, Vol. 3, No. 1, pp. 1-12, 2011.

[10] O.P. Sharma, M.K. Ghose, K.B. Shah and B.K. Thakur, "Recent Trends and Tools for Feature Extraction in OCR Technology", *International Journal of Soft Computing and Engineering*, Vol. 2, No. 6, pp. 110-118, 2013.

[11] G. Kumar and P.K. Bhatia, "A Detailed Review of Feature Extraction in Image Processing Systems", *Proceedings of 4th International Conference on Advanced Computing and Communication Technologies*, pp. 112-118, 2014.

[12] M. Billah, S. Waheed and A. Hanifa, "An Optical Character Recognition System from Printed Text and Text Image using Adaptive Neuro Fuzzy Inference System". *International Journal of Computer Applications*, Vol. 130, No. 16, pp. 62-74, 2015.

[13] T. Devi, P.N. Swathi, J. Prabaharan and J. Manigandan, "Offline Handwriting Identification using Adaptive Neural Fuzzy Inference System", *Proceedings of 1st International Conference on Innovations in Computing and Networking*, pp. 12-16, 2016.

[14] G.M.N. Thilakarathna and W.K.I.L. Wanniarachchi, "English Character Recognition of an Image and Voicing System", *Proceedings of International Conference on Business, Law and Technology*, pp. 14-20, 2017.

[15] B.P.K. Sandhya and L. Pavithira, "Adaptive Neuro Fuzzy Inference System based Optical Character Recognition", *International Journal of Advance Research in Science and Engineering*, Vol. 6, No. 8, pp. 1-7, 2017.

[16] V. Vaidhehi, "The Role of Dataset in Training ANFIS System for Course Advisor", *International Journal of Innovative Research in Advanced Engineering*, Vol. 1, No. 6, pp. 249-256, 2014.

[17] F. Yang, M. Hamit, C.B. Yan, J. Yao, A. Kutluk, X.M. Kong and S.X. Zhang, "Feature Extraction and Classification on Esophageal X-Ray Images of Xinjiang Kazak Nationality", *Journal of Healthcare Engineering*, Vol. 2017, pp. 1-11, 2017.

[18] R.M. Haralick, K. Shanmugam and I. Distenin, "Textural Features for Image Classification", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 3, No. 6, pp. 610-621, 1973.

[19] A.A. Hashem, M.Y.I. Idris and M.T. El-Melegy, "Extraction Characters from Scene Image based on Shape Properties and Geometric Features", *International Journal of Computer Applications*, Vol. 169, No. 3, pp. 30-35, 2017.

[20] R.M.C. Croda, D.E. Gibaja Romero and S.O.C. Morales. "Sales Prediction through Neural Networks for a Small Dataset", *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 5, No. 4, pp. 35-41, 2018.

[21] S. Karsoliya, "Approximating Number of Hidden Layer Neurons in Multiple Hidden Layer BPNN Architecture", *International Journal of Engineering Trends and Technology*, Vol. 3, No. 6, pp. 714-717, 2012.

[22] F.S. Panchal and M. Panchal, "Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network", *International Journal of Computer Science and Mobile Computing*, Vol. 3, No. 11, pp. 455-464, 2014.

[23] Divya Gilly and Kumudha Raimond, "A Survey on License Plate Recognition Systems", *International Journal of Computer Applications*, Vol. 61, No. 6, pp. 34-40, 2013.

[24] B. Singh, S. Kataria, K. Singh and N.S. Shekhawat, "A Hybrid Approach for Characters Recognition in License plate Recognition System", *Proceedings of International Conference on Advances in Computer Science*, pp. 1-6, 2013.