Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 10, No. 4, November 2019), Pages: 75-85
www.jacr.iausari.ac.ir

MCSM-DEEP: A Multi-Class Soft-Max Deep Learning Classifier for Image Recognition

Aref Safari✉
Department of Computer Engineering, Rasht Branch, Islamic Azad University, Rasht, Iran
safari.aref@qodsiau.ac.ir

Received: 2020/06/22; Accepted: 2020/09/03

Abstract

Convolutional neural networks show outstanding performance in many image processing problems, such as image recognition, object detection and image segmentation. Semantic segmentation is a very challenging task that requires recognizing, understanding what's in the image in pixel level. The goal of this research is to develop on the known mathematical properties of the soft-max function and demonstrate how they can be exploited to conclude the convergence of learning algorithm in a simple application of image recognition in supervised learning. So, we utilize results from convex analysis theory which associated with hierarchical architecture to derive additional properties of the soft-max function not yet covered in the existing literature for Multi-Class Classification problems. The proposed MC-DEEP model represents an average accuracy of 90.25% in different layers setting with 95% confidence interval in best initial settings in deep convolutional layers which applied on MNIST dataset. The results show that the regularized networks not only could provide better segmentation results with regularization effect than the original ones but also have certain robustness to noise.

Keywords: *Deep Learning, Image Recognition, Soft-Max Activation Function*

1. Introduction

Though the state of the art has been greatly improved by CNNs, there is no explicit connections between prediction of neighboring pixels. In the other hand, the problem of training of the multi-layer neural network was finally solved in 1986 when the back-propagation algorithm was introduced. The neural network was on stage again. However, it was soon met with another problem. Its performance on practical problems did not meet expectations. Of course, there were various attempts to overcome the limitations, including the addition of hidden layers and addition of nodes in the hidden layer. However, none of them worked. Many of them yielded even poorer performances. In an effort to overcome the practical limitations of the single-layer, the neural network evolved into a multi-layer architecture. However, it has taken approximately 30 years to just add on the hidden layer to the single-layer neural network. [1] As the training process is the only method for the neural network to store

information, untrainable neural networks are useless. A proper learning rule for the multi-layer neural network took quite some time to develop. The previously introduced delta rule is ineffective for training of the multi-layer neural network. This is because the error, the essential element for applying the delta rule for training, is not defined in the hidden layers. [2] The error of the output node is defined as the difference between the correct output and the output of the neural network. However, the training data does not provide correct outputs for the hidden layer nodes, and hence the error cannot be calculated using the same approach for the output nodes. The input data of the neural network travels through the input layer, hidden layer, and output layer. In contrast, in the back-propagation algorithm, the output error starts from the output layer and moves backward until it reaches the right next hidden layer to the input layer. This process is called backpropagation, as it resembles an output error propagating backward. Even in back-propagation, the signal still flows through the connecting lines and the weights are multiplied. The only difference is that the input and output signals flow in opposite directions. While increasing depth often reduces the number of parameters of the network, it leads to different types of practical issues [3]. Propagating backwards using the chain rule has its drawbacks in networks with a large number of layers in terms of the stability of the updates. In particular, the updates in earlier layers can either be negligibly small (vanishing gradient) or they can be increasingly large (exploding gradient) in certain types of neural network architectures [4].

2. Research Background

Convolutional Neural Networks (CNNs) have achieved prominent performance in a series of image processing tasks, such as image classification [12], object detection [13] and image segmentation [14]. CNNs which dwarf systems relying on hand-crafted features use millions of parameters to learn latent features from large scale training datasets. It is intractable to design a hand-crafted feature which allows to learn increasingly abstract data representations. However, CNNs could do it well [15]. Image segmentation is a process of segmenting a digital image into different regions. It aims to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Semantic image segmentation is a much more challenging segmentation task. It requires understanding the image in pixel level. More precisely, it is a process of assigning a label to every pixel in an image such that pixels with some label share some common features [16].

2.1 Multiclass Classification Problem

This section introduces how to utilize the neural network to deal with the classification of three or more classes. Consider a classification of the given inputs of coordinates (x, y) into one of three classes. In order to calculate the error, we switch the class names into numeric codes, as we did in the previous section. Considering that we have three output nodes from the neural network, we create the classes as the following vectors [5]:

$$\text{Class } 1 \rightarrow [1\ 0\ 0],\ \text{Class } 2 \rightarrow [0\ 1\ 0] \text{ and Class } 3 \rightarrow [1\ 0\ 0]$$

This transformation implies that each output node is mapped to an element of the class vector, which only yields 1 for the corresponding node. For example, if the data belongs

to Class 2, the output only yields 1 for the second node and 0 for the others. This expression technique is called one-hot encoding or 1-of-N encoding. The reason that we match the number of output nodes to the number of classes is to apply this encoding technique. the activation function of the output node should be defined. Since the correct outputs of the transformed training data range from zero to one, we are not able to just use the sigmoid function as we did for the binary classification. In general, multiclass classifiers employ the soft-max function as the activation function of the output node [6]. The activation functions that we have discussed so far, including the sigmoid function, account only for the weighted sum of inputs. They do not consider the output from the other output nodes. However, the soft-max function accounts not only for the weighted sum of the inputs, but also for the inputs to the other output nodes. For example, when the weighted sum of the inputs for the three output nodes are 2, 1, and 0.1, respectively, the soft-max function calculates the outputs shown in Equation (1). All of the weighted sums of the inputs are required in the denominator [7]:

$$v = \begin{bmatrix} 2 \\ 1 \\ 0.1 \end{bmatrix} \rightarrow \varphi(v) = \begin{bmatrix} \frac{e^2}{e^2 + e^1 + e^{0.1}} \\ \frac{e^1}{e^2 + e^1 + e^{0.1}} \\ \frac{e^{0.1}}{e^2 + e^1 + e^{0.1}} \end{bmatrix} = \begin{bmatrix} 0.659 \\ 0.242 \\ 0..098 \end{bmatrix} \quad (1)$$

The soft-max function maintains the sum of the output values to be one and also limits the individual outputs to be within the values of 0-1. As it accounts for the relative magnitudes of all the outputs, the soft-max function is a suitable choice for the multiclass classification neural networks. The output from the i-th output node of the soft-max function is calculated as follows Equation (2):

$$y_i = \varphi(v_i) = \frac{e^{v_i}}{e^{v_1} + e^{v2_2} + e^{v_3} + \cdots . + e^{v_M}} = \frac{e^{v_i}}{\sum_{k=1}^{M} e^{v_k}} \quad (2)$$

where, V_i is the weighted sum of the i-th output node, and M is the number of output nodes. Following this definition, the soft-max function satisfies the following Equation (3) [8]:

$$\varphi(v_1) + \varphi(v_2) + \varphi(v_3) + \cdots + \varphi(v_M) = 1 \quad (3)$$

Finally, the learning rule should be determined. The multiclass classification neural network usually employs the cross entropy-driven learning rules just like the binary classification network does. This is due to the high learning performance and simplicity that the cross-entropy function provides. We should note that carefully, Soft-max is used for multi-classification in the Logistic Regression model, whereas Sigmoid is used for binary classification in the Logistic Regression model. So, The Soft-Max activation layer is similar to the Sigmoid function. The difference is that, in the denominator, we sum together all of the values.

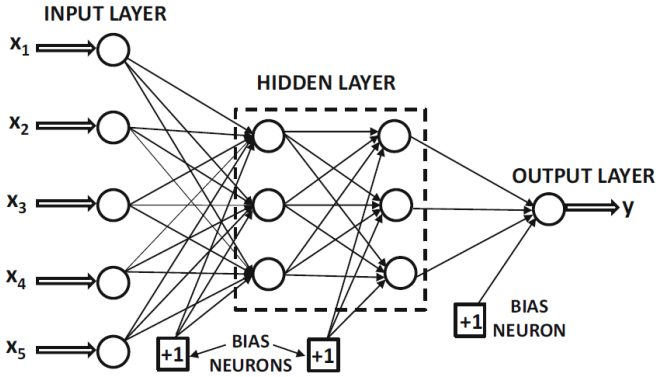

Fig.1 A multiple-class model for categorical classification with a Soft-max layer

3. Proposed MCSM-DEEP Model

Briefly, Deep Learning is a Machine Learning technique that employs the deep neural network. As you know, the deep neural network is the multi-layer neural network that contains two or more hidden layers. Although this may be disappointingly simple, this is the true essence of Deep Learning. The deep neural network lies in the place of the final product of Machine Learning, and the learning rule becomes the algorithm that generates the model (the deep neural network) from the training data. The reason that the neural network with deeper layers yielded poorer performance was that the network was not properly trained. The backpropagation algorithm experiences the following three primary difficulties in the training process of the deep neural network: Vanishing gradient, Overfitting and the Computational load. In the other hand, Convolutional Neural Network (ConvNet) is not just a deep neural network that has many hidden layers. It is a deep network that imitates how the visual cortex of the brain processes and recognizes images. Therefore, even the experts of neural networks often have a hard time understanding this concept on their first encounter. That is how much ConvNet differs in concept and operation from the previous neural networks.

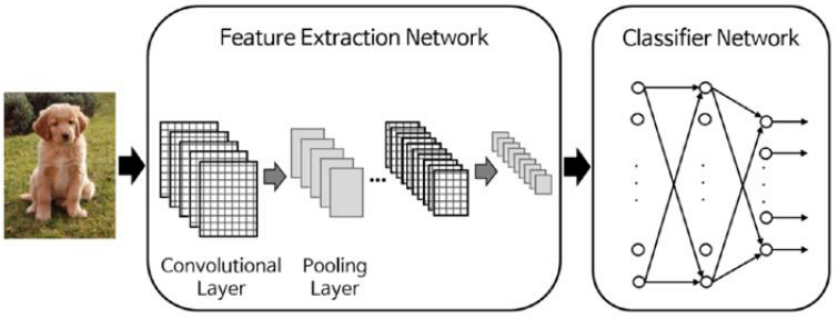

Fig.2 Typical architecture of ConvNet

The convolution layer generates new images called feature maps. The feature map accentuates the unique features of the original image. The convolution layer operates in a very different way compared to the other neural network layers. This layer does not employ connection weights and a weighted sum. Instead, it contains filters that convert images. We will call these filters convolution filters. The process of the inputting the image through the convolution filters yields the feature map. The pooling layer reduces the size of the image, as it combines neighboring pixels of a certain area of the image

into a single representative value. Pooling is a typical technique that many other image processing schemes have already been employing.

For two-fold classification problems, the activation function of the output unit is chosen to be the logistic sigmoid function in Equation (4):

$$\sigma_x = \frac{1}{1 + e^{-x}} \tag{4}$$

where the two classes are denoted as {0,1} If y = 1 is defined as label for the class C_1 and y = 0 denotes class C_2, then the output h_W, $b_{(x)}$ can be interpreted as the class membership probability for C_1. The class membership probability for C_2 is then given as 1-h_W, $b_{(x)}$. But, deliberate a classification problem in which we have an extremely large number of classes. In such a case, learning becomes too slow, because of the large number of separators that need to be updated for each training instance. This situation can occur in applications like text mining, where the prediction is a target word. Predicting target words is particularly common in neural language models, which try to predict the next word given the immediate history of previous words. The cardinality of the number of classes will typically be larger than 105 in such cases. In the other hand, hierarchical soft-max is a way of improving learning efficiency by decomposing the classification problem hierarchically. The idea is to group the classes hierarchically into a binary tree-like structure, and then perform *log2(k)* binary classifications from the root to the leaf for k-way classification. Although the hierarchical classification can compromise the accuracy to some extent, the efficiency improvements can be significant. The native approach for obtaining the hierarchy of classes is to create a random hierarchy. However, the specific grouping of classes has an effect on performance. Grouping similar classes tends to improve performance. It is possible to use domain-specific insights to improve the quality of the hierarchy. Consider a setting with k different classes. Each training instances: $(\bar{x}_i, c(i))$ Contains a *d*-dimensional feature vector Xi and the index $c(i) \in \{1 \ldots \mathrm{k}\}$ of its observed class. In such a case, we would like to find *k* different linear separators $\overline{w_1}, \ldots \overline{w_k}$ simultaneously so that the value of $\bar{W}c(i)\ .\ Xi$ is larger than $Wr \cdot Xi$ for each $r \neq c(i)$. This is because one always predicts a data instance Xi to the class r with the largest value of $Wr \cdot Xi$. Therefore, the loss function for the *i-th* training instance in the case of the multiclass perceptron is defined as follows:

$$L_i = max_{r:r \neq c(i)} \max(\bar{W}r.\bar{X}i - \bar{W}c(i).\bar{X}i, 0) \tag{5}$$

As in all neural network models, one can use gradient-descent in order to determine the updates. For a correctly classified instance, the gradient is always 0, and there are no updates.

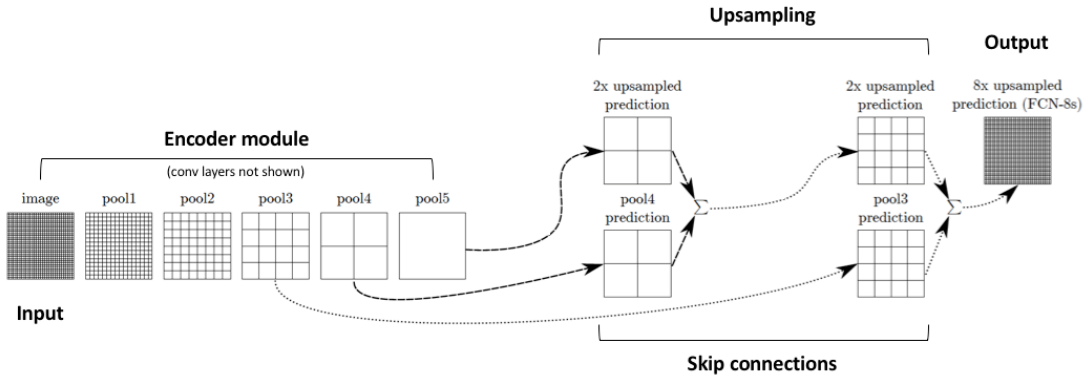

Fig.3 Proposed Structure: Our proposed model learns to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are omitted. Solid line: Our single-stream net and up samples stride 32 predictions back to pixels in a single step. Dashed line: Combining predictions from both the final layer and the pool 4 layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information

4. Experimental Results and Performance Evaluation

The MNIST database, which stands for Modified National Institute of Standards and Technology database, is a large database of handwritten digits. As the name suggests, this data set was created by modifying an original database of handwritten digits provided by NIST. MNIST data is used for testing all types of neural network algorithms beyond the domain of computer vision. We implement a neural network that takes the input image and recognizes the digit that it represents. The training data is the MNIST database, which contains 70,000 images of handwritten numbers. In general, 60,000 images are used for training, and the remaining 10,000 images are used for the validation test. Each digit image is a 28-by-28-pixel black-and-white image, as shown in Figure 3. Considering the training time, this example employs only 10,000 images with the training data and verification data in an 8:2 ratio. Therefore, we have 8,000 MNIST images for training and 2,000 images for validation of the performance of the neural network. As you may know well by now, the MNIST problem is caused by the multiclass classification of the 28×28-pixel image into one of the ten-digit classes of 0-9. Let's consider a ConvNet that recognizes the MNIST images. As the input is a 28×28-pixel black-and-white image, we allow 784(=28×28) input nodes.

Fig.4 A 28-by-28-pixel black-and-white image from the MNIST database

The feature extraction network contains a single convolution layer with 20 9×9 convolution filters. The output from the convolution layer passes through the Rectified Linear Unit (ReLU) function, followed by the pooling layer. The pooling layer employs the mean pooling process of two by two submatrices. The classification neural network consists of the single hidden layer and output layer. This hidden layer has 1000 nodes that use the ReLU activation function. Since we have 10 classes to classify, the output layer is constructed with 10 nodes. The input nodes between the pooling layer and the hidden layer, which are the square nodes left of the W_5 block, are the transformations of the two-dimensional image into a vector. As this layer does not involve any operations, these nodes are denoted as squares. The function MnistConv, which trains the network using the back-propagation algorithm, takes the neural network's weights and training data and returns the trained weights. We use the soft-max activation function for the output nodes. The table-1 summarizes the implementation of MC-DEEP model and the architecture of this neural network has been shown in figure 4.

Table-1 summarizes the implementation of proposed MC-DEEP model.

Layer	Remark	Activation Function
Input	28×28 nodes	-
Convolution	20 convolution filters (9×9)	ReLU
Pooling	1 mean pooling (2×2)	-
Hidden	100 nodes	ReLU
Output	10 nodes	Soft-Max

Lastly, let's investigate how the image is processed while it passes through the convolution layer and pooling layer. The original dimension of the MNIST image is 28´28. Once the image is processed with the 9×9 convolution filter, it becomes a 20×20 feature map.11 As we have 20 convolution filters, the layer produces 20 feature maps. Through the 2×2 mean pooling process; the pooling layer shrinks each feature map to a 10´10 map. The process is illustrated in Figure 4. The final result after passing the convolution and pooling layers is as many smaller images as the number of the convolution filters; ConvNet converts the input image into the many small feature maps.

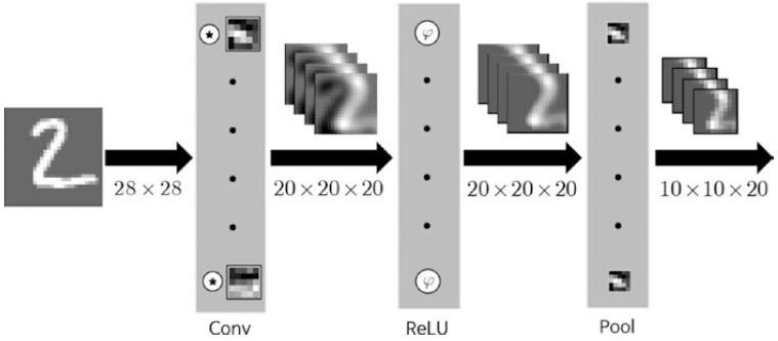

Fig. 5 How the image is processed while it passes through the convolution and pooling layers

Figure 6 a) is the first image of the screen, which consists of the 20 trained convolution filters. Each filter is pixel image and shows the element values as grayscale shades. The greater the value is, the brighter the shade becomes. These filters are what ConvNet determined to be the best features that could be extracted from the MNIST image, the b) is the second image from the screen, which provides the results (y1) of the image processing of the convolution layer. This feature map consists of 20 20×20-pixel images. The various alterations of the input image due to the convolution filter can be noticeable from this figure. The c) in figure 5 represent the third result, which provides the images after the mean pooling process in which the ReLU layer produces. Each image inherits the shape of the previous image in a 10×10-pixel space, which is half the previous size. The d), is the final result of the feature extraction neural network. These images are transformed into a one-dimensional vector and stored in the classification neural network. Statistical evaluation of analytic performance in general and specifically ROC curve analysis was conducted for calculating the performance of predictive proposed model. The confusion matrix was calculated to define the performance of the suggested approaches. The confusion matrix describes all possible results of forecasting results in the table structure.

***Specificity*:** The prospect of the test finding the correct class among all classes [10,11]:

$$\frac{TN}{TN + FP} \tag{6}$$

***Accuracy*:** The fraction of test results those are correct:

$$\frac{TP + TN}{TP + FN + TN + FP} \tag{7}$$

***Precision*:** Precision or positive predictive value:

$$\frac{TP}{TP + FP} \tag{8}$$

***Sensitivity* (*Recall*):** Hit rate

$$\frac{TP}{Total\ Positive} \tag{9}$$

Table.2 Applied confusion matrix for proposed model with different layers

Depth / Performance	10 Layer	15 Layer	30 Layer	50 Layer
Precision	84%	89%	90%	94%
Sensitivity	82%	88%	92%	93%
Specificity	83%	89%	91%	95%
Accuracy	85%	89%	93%	94%
Confidence Interval	[82-87]	[87-91]	[89-94]	[90-95]

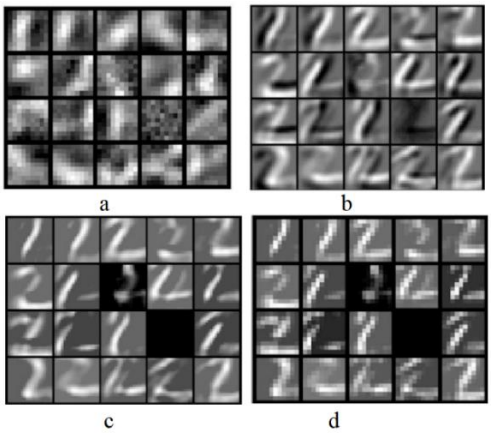

Fig. 6* a) *Image showing 20 trained convolution filters, b) The results (y1) processing of the convolution layer, c) ReLU function processed on the feature map d) The images after the mean pooling process.

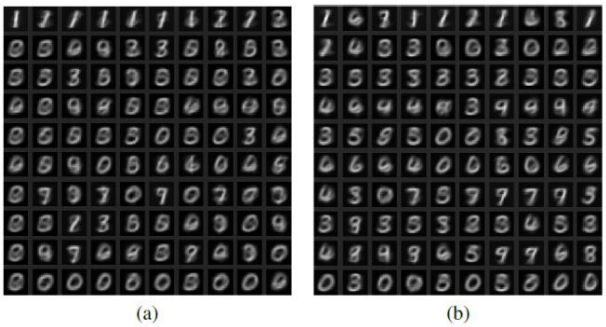

Fig.7 (a) The weighted average image of first hidden-layer features. (b) The visualization of the third hidden-layer.

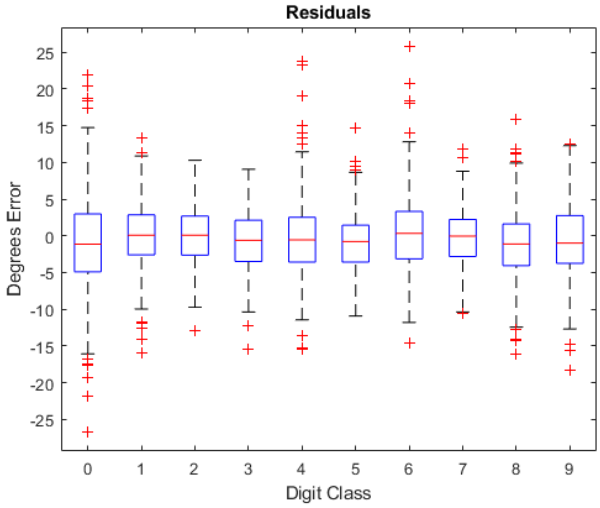

Fig. 8 Residuals and Degrees Error in all Decimal Digits

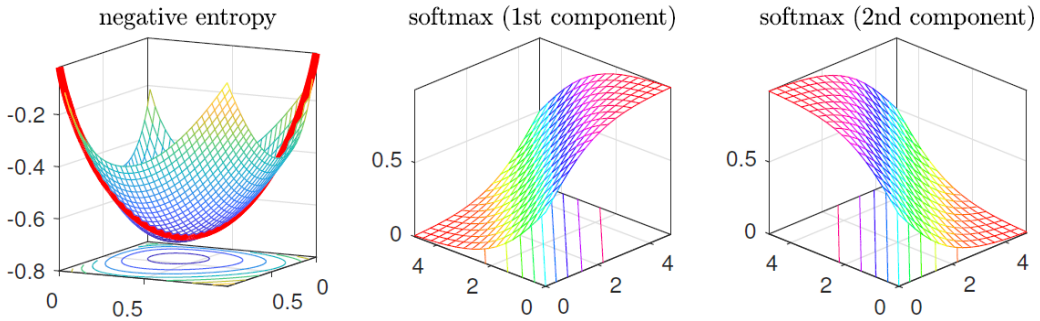

Fig.9 Plots of the log-sum-exp, negative entropy and both components of soft-max function

5. Conclusion

The proposed solution improves the classification and detection accuracy of model along with the different type of digits. Combining the modified soft-max multi-class and the modified categorical deep learning with regularization has been contributed significantly in classifying different type of digits with high accuracy and optimum processing time. The results of applied dataset indicate that the hyperparameters for the first layer is the most important parameter of this method, since Soft-Max of the first layer exhibits the strongest influence on the accuracy, the runtime, the number of group members in the first layer and the number of group members in the second layer. The model combines the modified loss function for categorical classification with the regularization. The regularization reduces the risk of overfitting of a dataset and decreases the number of neurons. Moreover, it decreases the computational time and increases the model's accuracy. This research designed for enhancing the performance of the network and increase the efficiency of the system with average accuracy of 90.25% in different layers setting with 95% confidence interval in best settings.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4):433–459, 2010.

[2] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. The Journal of Machine Learning Research, 1:113–141, 2001.

[3] Telmo Amaral, Luís M Silva, Luís A Alexandre, Chetak Kandaswamy, Jorge M Santos, and Joaquim Marques de Sá. Using different cost functions to train stacked auto-encoders. In 12th Mexican International Conference on Artificial Intelligence (MICAI), pages 114– 120. IEEE, 2013.

[4] JH Austin, NL Müller, Paul J Friedman, David M Hansell, David P Naidich, Martine Remy-Jardin, W Richard Webb, and Elias A Zerhouni. Glossary of terms for ct of the lungs: recommendations of the nomenclature committee of the fleischner society. Radiology, 200(2):327–331, 1996.

[5] Graham Bath and Judy McKay. Praxiswissen Softwaretest: Test-Analyst und technical Test-Analyst. dpunkt-Verlag, 2010.

[6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798–1828, 2013.

[7] Yoshua Bengio. Learning deep architectures for ai. Foundations and trends in Machine Learning, 2(1):1–127, 2009.

[8] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. Journal of Machine Learning Research-Proceedings Track, 27:17–36, 2012.

[9] Klemens Burg, Herbert Haf, Friedrich Wille, and Andreas Meister. Vektoranalysis, volume 2. Springer, 2012.

[10] A. Safari, R. Hosseini, M. Mazinani, A Novel Type-2 Anfis Classifier for Modelling Uncertainty in Prediction of Air Pollution Disaster, IJE Transactions B: Applications, Vol 30, No. 11, Pages 1568-1577, (2017)

[11] Aref Safari, Danial Barazandeh, Seyed Ali Khalegh Pour. A Novel Fuzzy-C Means Image Segmentation Model for MRI Brain Tumor Diagnosis. J. ADV COMP ENG TECHNOL, 6(1) Winter 2020: 19-2

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014.

[13] B. Hariharan, P. Arbel_aez, R. Girshick, and J. Malik, Simultaneous detection and segmentation in European Conference on Computer Vision, Springer, 2014, pp. 297-312.

[14] K. He, X. Zhang, S. Ren, and J. Sun, Delving deep: Surpassing human-level performance on image-net classification, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026-1034.

[15] Aref Safari, A Novel Transformation Watershed Image Segmentation Model in Digital Elevation Maps Processing, Journal of Modern Processes in Manufacturing and Production, Volume 9, No. 3, Summer 2020, pp. 13-22

[16] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks, arXiv preprint arXiv:1610.01983, (2016).