iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** XI **Month of publication:** November 2021

DOI: https://doi.org/10.22214/ijraset.2021.39061

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

Convolutional Neural Network for Image Recognition

Ankith I[1], Akshaya H P[2]
[1]*UG Scholar, Dept. of CSE, BNM Institute of Technology, Bengaluru, Karnataka*
[2]*UG Scholar Dept. of CSE, BNM Institute of Technology, Bengaluru, Karnataka*

Abstract: ***Recent developments in the field of machine learning have changed the way it operates for ever, especially with the rise of Artificial Neural Networks (ANN). There is no doubt that these biologically inspired computational models are capable of performing far better than previous forms of artificial intelligence in common machine learning tasks as compared to their previous versions. There are several different forms of artificial neural networks (ANNs), but one of the most impressive is the convolutional neural network (CNN). CNN's have been extensively used for solving difficult pattern recognition tasks using images. With their simple yet precise architecture, they offer a simplified approach to getting started with ANNs. The goal of this paper is to provide a brief introduction to CNN. It discusses the latest papers and newly formed techniques in order to develop these absolutely brilliant models of image recognition. This introduction assumes that you already have a basic understanding of ANNs and machine learning.***

Keywords: Pattern recognition, artificial neural networks, machine learning, image analysing.

I. INTRODUCTION

The concept of artificial neural networks (ANNs) is a computational processing system heavily inspired by biological nervous systems (such as the human brain), especially in how they function. This term is used loosely to refer to an artificial neural network in which a great number of computational nodes (called neurons) are interconnected in a distributed manner, all of which work together to collectively learn what the input is in order to make better decisions. In order to model the basic structure of an ANN, a diagram similar to Figure 1 is shown below. We would load the input, usually in the form of a multidimensional vector, into the input layer, which will distribute the input to the hidden layers. It is in this subsequent layer, which will make decisions from the previous layer. It will assess how a stochastic change within itself negatively or positively affects the final output. This is where the process of learning takes place. There is a concept known as deep learning, which refers to the process of having multiple hidden layers stacked upon each other.

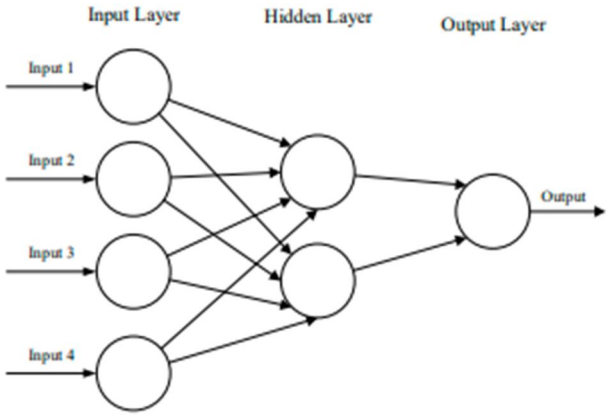


Fig 1. Basic structure of ANN

Image processing tasks can be divided into two key learning paradigms: supervised and unsupervised learning.
In the supervised learning mode, all inputs are pre-labeled, which also serve as targets to be attained. There will be a set of input values (vectors) for each training example as well as one or more output values associated with the training example. This method of training aims to reduce the model's overall classification error, by calculating the output value of each training example correctly in order to reduce the error of the overall system.

Unlike supervised learning, unsupervised learning does not include any labels in the training set. The success of a network is primarily determined by its ability to reduce or increase the associated cost functions. While supervised learning has its place in many pattern-recognition tasks involving images, it is imperative to also recognize that the majority of image-based pattern-recognition tasks rely on classification.

The Convolutional Neural Networks (CNNs) are similar to traditional artificial neural networks (ANN) in the way that they consist of neurons with self-optimizing behavior via training. It is still the case that each neuron receives an input and performs a task (such as a scalar product followed by a non-linear function) - the basis of countless Artificial Neural Networks. It is still evident that the entire network will still use a single perceptive score function (the weight) regardless of whether it takes input raw image vectors or a final output class score. Each class will be represented as a layer in this layer, and all the tips and tricks that we have already employed for traditional ANNs will naturally be applicable here as well.

CNNs differ primarily from traditional Artificial Neural Networks in that their primary use is in pattern recognition within images. This is the only notable difference between CNNs and traditional ANNs. With this technique we can encode specific image-specific characteristics into the network, making it better suited to image-specific tasks - and further reducing the parameters required for the setup of the network.

There are several large limitations to conventional forms of artificial neural networks, one of which is that conventional forms of ANN tend to struggle with the computational complexity required to compute image data. A common benchmarking dataset for machine learning is the MNIST database of handwritten digits, which is suitable for most types of Artificial Neural Networks. The reason for this is the relatively small size of the image which is just 28 x 28 pixels. If one neural network is constructed with this dataset, a single neuron in the first hidden layer will contain 784 weights (28*28*1 where 1 bear in mind that MNIST uses only black and white values), which are manageable for most forms of artificial neural networks.

When you consider that a better coloured image input would be, say, 64 * 64, then the number of weights on just one neuron of the first layer would increase substantially to 12,288. Consider, as well, that to handle this scale of input, a network would also need to be considerably larger than the one used for classifying colour-normalised MNIST digits, and then you will see the drawbacks in using models such as these.

A. Overfitting

However, why does it matter in the first place? Isn't it possible that we could increase the number of hidden layers in our network, as well as the number of neurons that are contained within them, if possible? If you asked me that question, I would simply reply, "No". I believe there are two main reasons for this. One is the fact that we do not have unlimited computing power and time to train such a huge amount of ANN's.

The second reason is that overfitting must be stopped or minimized in order to eradicate it. Essentially, overfitting occurs when a network is not able to learn properly due to a variety of reasons. Machine learning algorithms include this concept as a key component, and it is important to take every precaution to ensure that the effects are reduced as much as possible. In the event that our models were to show signs of overfitting, we might see a reduced ability to pinpoint the generalised features of not only our training dataset, but also our test and prediction datasets as well.

As a result, we decided to reduce the complexity of our ANNs as much as possible. If fewer parameters are required to train, then there is less likelihood that the network will overfit - and this, of course, improves the predictive ability of the model.

II. CNN ARCHITECTURE

As mentioned earlier, CNNs are primarily based on the assumption that the input will consist of images as part of the algorithm. This requires the architecture to be in a position to be designed in a way that is best suited to the needs of dealing with a specific type of data.

The most significant difference is that the cells within the layers of the CNN are comprised of a variety of neurons that are organized into three dimensions, the spatial dimension of the input (height and width) and the depth. The depth does not refer to the total number of layers of a neural network, but rather to the third dimension of an activation volume. The neurons within any one layer of an ANN will only connect to a small region of the layer preceding it. This contrasts with the neurons within a standard ANN. In practice this would mean that for the example given earlier, the input 'volume' will have a dimensionality of $64 \times 64 \times 3$ (height, width, and depth), leading to a final output layer comprised of a dimensionality of $1 \times 1 \times n$ (where n represents the possible number of classes) as we would have condensed the full input dimensionality into a smaller volume of class scores filed across the depth dimension.

A. Overall Architecture

There are three different types of layers in a CNN. Among them are convolutional layers, pooling layers, and fully connected layers. A CNN architecture is produced as a result of stacking these layers. An example of a simplified CNN architecture for MNIST classification can be observed in Figure 2.

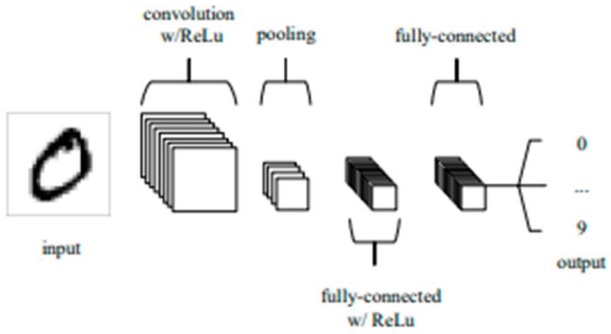


Fig. 2: A simple CNN architecture, comprised of just five layers

To breakdown the functionality of the example CNN above, four key areas can be identified.

1) It is also found that in other forms of ANN, the input layer contains the pixel values of the input image.

2) As a result of the calculation of the scalar product between the weights of the input neurons and the region connected to the input volume in the convolutional layer, we find the output of neurons whose output is connected to regions of the input. The rectified linear unit (commonly known as ReLu) is a method of applying an 'elementwise' activation function such as sigmoid to the output of the previous layer that produces the activation.

3) Pooling layers will then simply perform a down sampling operation along the spatial dimensionality of the given input, further reducing the number of parameters within the activation.

4) The fully connected layers will then perform the same tasks as found in standard ANNs and will attempt to generate class scores based on the activations to be used for classification. In addition, ReLU may be used by layer in order to enhance performance.

This simple technique allows for CNNs to not only transform the input layer by layer but also to down sample it is using convolution techniques using a down-sampling algorithm to produce class scores for both classification and regression applications.

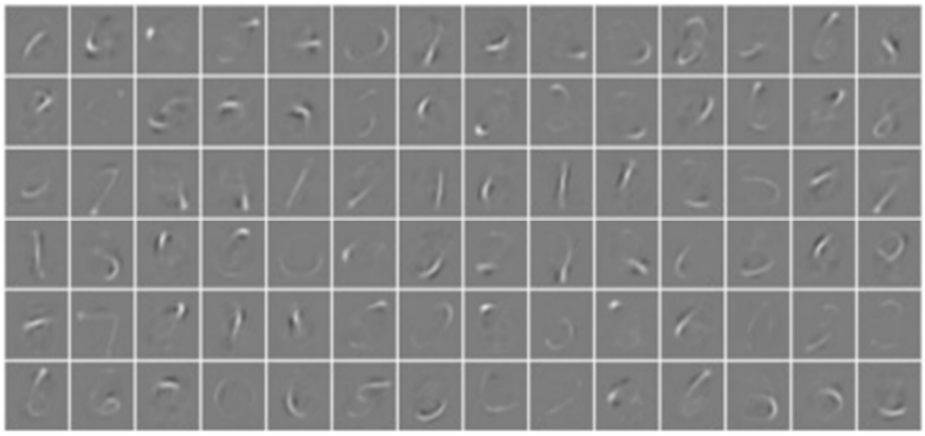


Fig 3. Activations from first CNN layer, after training on the MNIST database of handwritten digits

Having said that, it is vital to keep in mind that simply understanding the overall architecture of a CNN architecture will not suffice. It can take quite a while for these models to be created and optimized as well as be somewhat confusing to do so. The next step in our discussion is to elaborate on each layer in more detail, by describing the hyperparameters and connectivity between them.

B. Convolutional Layer

In order to understand how the CNN operates, it is important to understand the function of the convolutional layer. This layer makes use of learnable kernels as part of its parameters.

It is usually the case that these kernels are the smallest in spatial dimension, however, they cover the whole depth of the input. It is important to remember that when your data is hit by a convolutional layer, the layer combines each filter across the spatial dimension of the input, producing a 2D activation map. The activation maps can be visually viewed in Figure 3, which illustrates how they can be visualized.

In order to calculate the scalar product, each value in the kernel is calculated individually as we pass through the input. In the case of Figure 4, the network is learning kernels that 'fire' when they see a specific feature in the input at the given spatial position. Such kernels are often referred to as activation kernels.

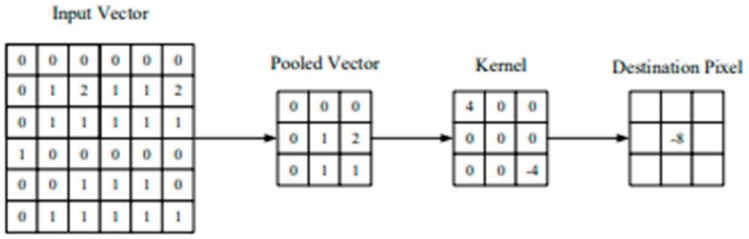


Fig. 4: A visual representation of a convolutional layer

In the convolutional layer, every kernel will have a corresponding activation map that will be stacked along the depth dimension to form the full output volume of the layer.

Training ANNs on inputs such as images causes them to produce models which are too large in size to train efficiently. As we mentioned earlier, this can increase training time. As a result of the fully connected manner in which standard ANN neurons are configured, all the neurons in a convolutional layer are only connected to a small portion of the input volume, in order to mitigate this effect. The size of the receptive field of the neuron is commonly referred to as the dimension of the receptive field of this region. Generally speaking, as a rule of thumb, the magnitude of the connection through the depth is equal to the depth of the input.

For example, if the input to the network is an image of size 64 × 64 × 3 (a RGB coloured image with a dimensionality of 64 × 64) and we set the receptive field size as 6 × 6, we would have a total of 108 weights on each neuron within the convolutional layer. This equation is (6 * 6 * 3 where 3 denotes the quantity of connectivity across the depth of the volume). To put this into perspective, a normal neuron in any other form of ANN would contain 12, 288 weights each.

Additionally, convolutional layers can be used to significantly reduce the complexity of a model through the optimization of its output. There are three hyperparameters that are optimised in this case, the depth, the stride, and setting zero padding.

Using the number of neurons inside each layer of the convolutional layer, one can be able to manually adjust the depth of the output volume produced by the convolutional layer. It is possible to do this as long as the input region is in the same position. Another example of this can be seen in other types of ANNs. In these, all of the neurons connected to every neuron within the hidden layer are directly connected to each other. As a result of this hyperparameter reduction, the total number of neurons in the network can be drastically reduced, however the pattern recognition capabilities of the model can also be drastically lowered.

Additionally, we may define the stride around which we set the depth of our receptive field in order to place the spatial dimensionality of the inputs. The stimulation field would remain heavily overlapped even if the stride was set to 1, which would result in very large activations of the receptive field. Additionally, changing the stride from a small number to a significant number will reduce the amount of overlap and thus result in a smaller spatial dimension.

An effective method of giving further command over the dimensionality of the output volumes is to pad the borders of the input by zero. This is a simple way of modifying the input border by zero.

Through the use of these techniques, we can alter the spatial dimension of the output of convolutional layers. In order to calculate this, you can make use of the following formula:

$$(V - R) + 2Z / (S + 1)$$

The input volume size (height*width*depth) is represented by V, the receptive field size is represented by R, the amount of zero padding is represented by Z and the stride size is represented by S. The stride has been set incorrectly if the calculated result is not a whole integer. As a result, the neurons will not fit neatly across the input.

Despite all our best efforts so far, we will still find that our models are still enormous if we use an input image that has any degree of dimensionality to it. The use of convolutional layers, however, has allowed considerable reduction in the number of parameters within the overall layer.

The parameter sharing works on the assumption that if a region feature is useful in computing one spatial region in a particular way, then it is likely to be useful in computing another spatial region in the same way. Constrained to the same weights and bias for each of the activation maps within the output volume, we will see a significant reduction in the number of parameters being created by the convolutional layer as a result.

Due to the way the backpropagation stage operates (as opposed to all neurons in the output being updated at once), as each neuron represents the overall gradient that can be totaled across the depth, only updating a single set of weights, instead of updating every single one.

C. Pooling Layer

By pooling layers together, we are gradually reducing the number of parameters and the computational complexity of the model, which in turn will also reduce the number of parameters and the computational complexity.

Pooling is achieved by scaling the dimensionality of each activation map in the input using the "MAX" function, which operates over each activation map in the input. There are several most common ways of creating CNNs using max-pooling layers with kernels of a dimensionality of 2 * 2 applied with a step of 2 along the spatial dimensions of the input data. It is possible to scale the activation map down to 25% of its original size in this manner while keeping the depth volume at the standard size.

Due to the destructive nature of the pooling layer, the maximum pooling technique is only observed in two general ways. There is a standard method of setting the stride and the filters of pooling layers to be two by two. This will ensure that the pooling layer extends to encompass the full spatial dimensionality of the input. In addition, it may be possible to implement overlapping pooling by setting the stride to 2 and the kernel size to 3. Having a kernel size much larger than 3 usually leads to an adverse impact on the model's performance due to the destructive nature of pooling.

Further, it is imperative to keep in mind that CNN architectures may also contain general-pooling functionality beyond max-pooling. There are several general pooling layers, consisting of pooling neurons that are capable of performing a multitude of operations including normalisation, pooling of averages, and multivariate normalization. Nevertheless, this tutorial will primarily be concerned with the implementation of max-pooling.

D. Fully Connected Layer

There are neurons at the fully connected layer that are directly connected to those neurons at the fully connected layer of the two adjacent layers, without being connected to any of the layers within the two adjacent layers. An analogy can be drawn between how neurons are organized in traditional ANNs and the way they are arranged here. (Figure 1).

III. CONCLUSION

A particular property of Convolutional Neural Networks that sets them apart from other types of Artificial Neural Networks is that instead of focusing on the entire domain of the problem, they exploit information about the specific type of input. The result of this is that a network architecture can be set up in a much simpler way.

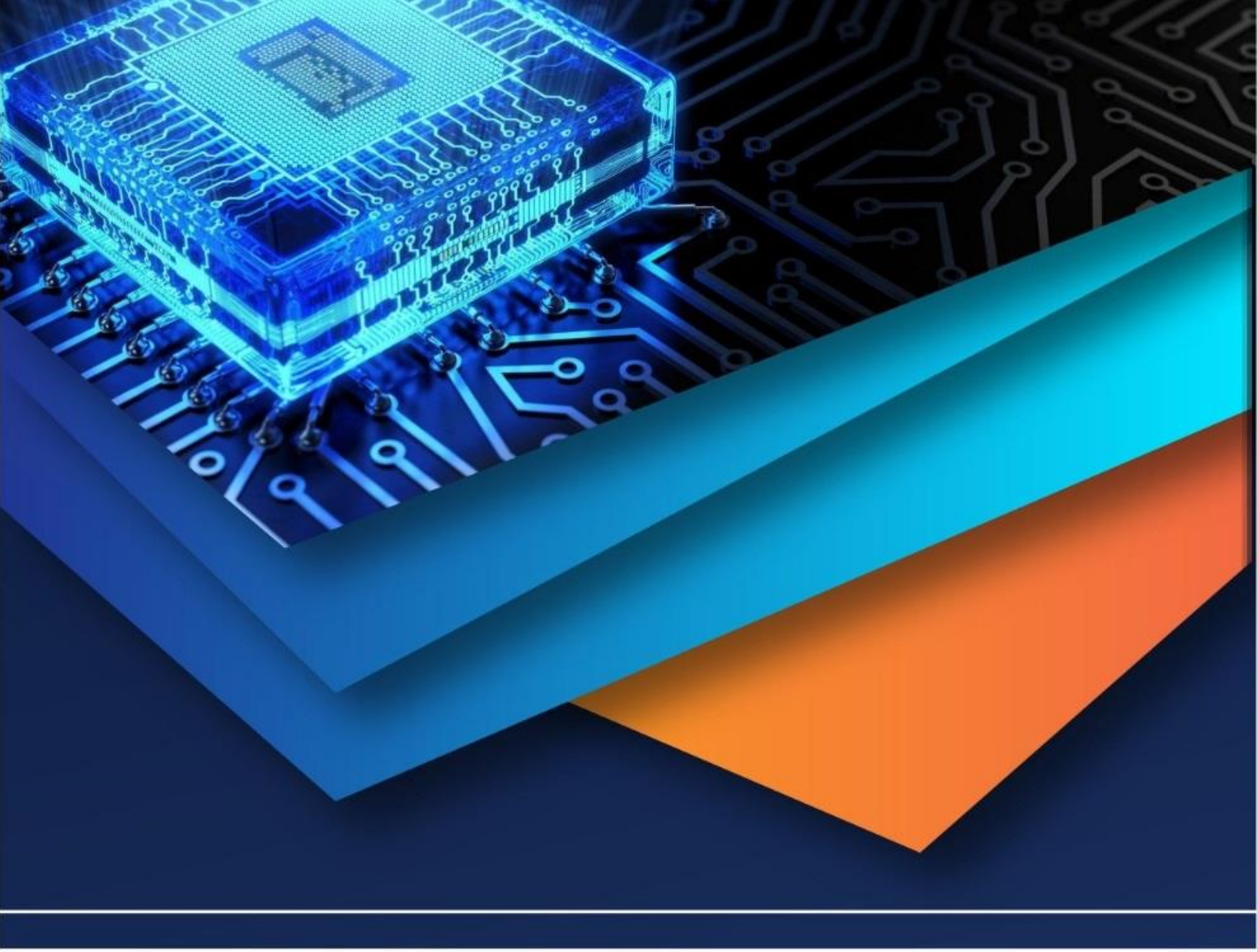
In this paper, we have presented the basic concepts of Convolutional Neural Networks, outlined the layers required to construct one and described the best structure to use for most image analysis tasks.

In recent years, research in the field of image analysis using neural networks has somewhat slowed down. I believe that this is largely due to the incorrect belief that there is a level of complexity and knowledge needed before you can begin attempting to model these highly sophisticated machine learning algorithms. The authors hope that this paper has helped to alleviate some of this confusion and to make the subject more accessible for beginners.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, Geoffery Hinton, “Deep Learning”, Nature, Volume 521, pp. 436-444, Macmillan Publishers, May 2015.
[2] Norhidayu binti Abdul Hamid, Nilam Nur Binti Amir Sjarif, “Handwritten Recognition Using SVM, KNN and Neural Network”, www.arxiv.org/ftp/arxiv/papers/1702/1702.00723
[3] Cheng-Lin Liuכ ,Kazuki Nakashima, Hiroshi Sako, Hiromichi Fujisawa, “Handwritten digit recognition: benchmarking of state of-the-art techniques”, ELSEVIER, Pattern Recognition 36 (2003) 2271 – 2285).
[4] Ping kuang, Wei-na cao and Qiao wu, “Preview on Structures and Algorithms of Deep Learning”, 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP), IEEE, 2014.

[5] Mahmoud M. Abu Ghosh; Ashraf Y. Maghari, "A Comparative Study on Handwriting Digit Recognition Using Neural Networks", IEEE, 2017.
[6] Youssouf Chherawala, Partha Pratim Roy and Mohamed Cheriet, "Feature Set Evaluation for Offline Handwriting Recognition Systems: Application to the Recurrent Neural Network," IEEE Transactions on Cybernetics, VOL. 46, NO. 12, DECEMBER 2016.
[7] Alex Krizhevsky, "Convolutional Deep belief Networks on CIFAR-10". Available: https://www.cs.toronto.edu/~kriz/convcifar10-aug2010.pdf.
[8] Yehya Abouelnaga , Ola S. Ali , Hager Rady , Mohamed Moustafa, " CIFAR-10: KNN-based ensemble of classifiers", IEEE, March 2017.
[9] Caifeng Shan, Shaogang Gong, Peter W. McOwan, "Facial expression recogniton based on Local binary patterns: A comprehensive study", ELSEVIER, Image and Vision Computing 27, pp. 803-816, 2009.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY