International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-3, February 2019

Optimization of Siamese Neural Networks Using Genetic Algorithm

Vishal Prem, Mark Sheridan Nonghuloo, Nagaraja Rao A

Abstract: ***The power of Deep Learning Networks has allowed us to build applications far beyond what was thought possible during our time. But the basic forms of these architectures still have their limitations in terms of the data needed and difficulty in understanding and tuning the parameters of these networks. Our project aims to deal with these limitations as effectively as possible. For this reason we have chosen to implement a Siamese Neural Network in order to overcome the requirements of classical Deep Learning based image classification. And in order to further increase the efficiency of the network, we will process it using a genetic algorithm and harness it to improve the architecture and training efficiency by letting the algorithm fine-tune the parameters to get the best possible configuration for the neural network.***

Keywords: ***Siamese Neural Network, Genetic Algorithm, Neural Network Optimization.***

I.INTRODUCTION

Image recognition has fast become a very essential part of major technological and business related companies around the world. The ease of access for cameras in the present world and the trends of social networking have led to an exponential rise in the number of images being recorded every day. Thus it has become essential to find solutions to warehousing, indexing and storage of this massive data dump. Image recognition plays a vital role in this process. When it comes to businesses like logistics, efficient pipelines to recognize and segregate goods and materials is a 300 billion dollar industry in India alone [1]. Image recognition plays a huge role in automating this process. As such there is always research being done to find better and faster ways of achieving these goals.

Neural networks have been around since almost the beginning of computational science. But at that point of time we lacked the processing power needed to run these networks particularly in a data intensive field like image processing. We also haven't found an architecture that is able to map the spatial data regarding the pixels into an effective network. That came with the advent of Convolutional Neural Networks**.** The first proposed CNN was in 1994 and what became the foundation of deep learning. This innovation by Yann LeCun was named LeNet5 [2]. Previous networks would take each pixel separately in order to process them.

Manuscript published on 28 February 2019.
***** Correspondence Author (s)
Vishal Prem*, SCOPE, Department of Computer Science and Engineering, Vellore Institute of Technology,Vellore, India.
Mark Sheridan, Nonghuloo, SCOPE, Department of Computer Science and Engineering,Vellore Institute of Technology, Vellore, India.
Nagaraja Rao A, SCOPE, Department of Computer Science and Engineering, Vellore Institute of Technology,Vellore, India**.**

CNNs allowed the spatial location of the pixel with respect to other pixels to also be used which is a vital part of an image as they have very strong correlation. The years since from 1994 to 2010, this approach to image processing did not have much progress. In 2010 the first proposal appeared for a GPU based architectures. The popularity of gaming had let to a huge increase in the processing power of graphical processing units. These units while not as effective as CPU in application processing, were extremely efficient at mathematical computations and parallel processing. Thus in 2010 researchers Dan Claudiu Ciresan and Jurgen Schmidhuber [3] implemented a nine layer neural network that was trained on an NVIDIA GTX 280.But the true rise of the CNN began in 2012 when Alex Krizhevsky released AlexNet. AlexNet [4]is a much deeper network than any previous and it was trained on a GPU. The combination allowed it to win the prestigious and challenging ImageNet competition by an extremely huge margin. He redesigned LeNet into a much deeper architecture that could understand object hierarchies. His success started a small revolution in the field and CNNs became the industry standard for image recognition. Since then multiple different varieties of CNN have been developed like GoogleNet [6], DenseNet [7] and ResNet [8] which have used progressively deeper architectures. These have achieved extremely high accuracy in general image recognition over the years as seen in Fig. 1.

.

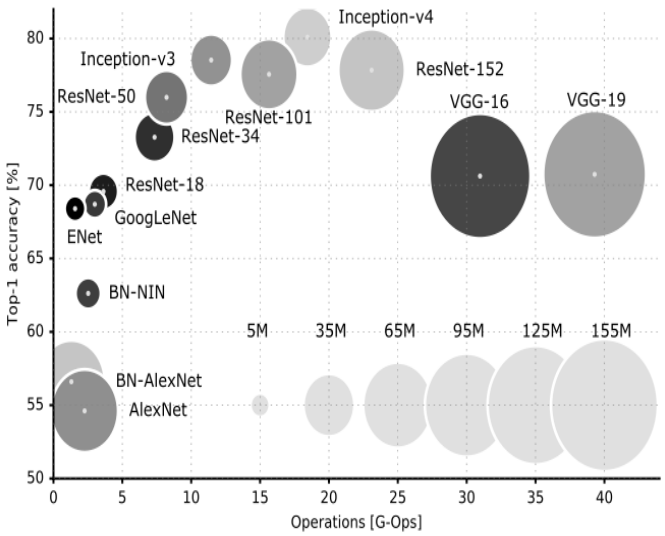

Fig. 1. No of operations in state-of-the-art CNNs vs Accuracy[5].Higher number of operations indicate more number of network layers.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

While these work very effectively, there are still some major drawbacks to implementing and using standard convolutional neural networks which are as follows.

1. In traditional networks, let us say we need to find if an image has Person A or Person B. We will need to get a large number of labeled images of both Person A or Person B and train the network. The network can now classify between them. But we would need that initial large dataset of labeled data for the individuals. Also if the net needs to learn to classify a third person, it would need to be remodeled and retrained with all three individual's images. We address this issue using Siamese neural networks.
2. The neural network architecture needed for effective recognition is still a very much researched process as we still haven't found a way to design the architecture based on the use case it is needed. The components of the architecture are decided by hyper parameters. CNNs work the most effectively if the architecture is modeled on the problem at hand, but there is a huge limitation as our understanding of how to implement this is still very limited. Our proposed algorithm automates this process of finding the best architecture.

The rest of this paper is set as follows. Firstly, the background is presented in the next section. Then, the details of the proposed algorithm are documented in the third section. Next, the designs and results are shown in the fourth and fifth sections, respectively. Lastly, conclusions and future works are outlined in the last section

II.BACKGROUND

Here we give the background on how Siamese neural networks and genetic algorithms work and their need to help readers understand the algorithm used and the related work and set the foundations of our research.

A. Siamese Neural Networks

Like it was stated above, one of the primary problems with standard CNNs is that if a new class is to be added then the whole network would have to be trained again. Thus we need a solution wherein the network learns the general pattern and then can make predictions for any class. This forms the essence of One Shot Learning. Siamese Neural nets are designed to accomplish this. They were first implemented by Gregory Koch, Richard Zemel and Ruslan Salakhutdinov from the University of Toronto in 2015[9]. Instead of classifying if an image belongs to a particular class, they are designed to map the similarity between two images. So essentially they take two input images which will be the known image of the class and the unknown image and give the output as whether they are from the same class or not. The architecture of the Network is as follows.

The Network consists of two sister networks that share the same weights. These are both Convolutional Neural networks that map both the images to vectors that contain the image characteristics. Specifically, the CNN uses convolutional layer that employ filters to perform convolutional operations on the images. One filter can be viewed as a matrix. The filters scan the image pixels using a sliding window and output a feature map. This is repeated multiple times until a thorough feature map of the image is created. The number of layers or feature maps is decided using a hyper parameter. Then a set of fully connected layers added at the end of the CNN to give the final feature vector. Once the feature vectors of the two images are available we run them through a *contrastive loss function* [10] during training that processes the final loss with respect to how close we are to classifying them correctly. If the loss is high then we are wrongly classifying them as the same or different class. The training of the network is to optimize this loss. The function is defined as follows:

$$(1-Y)\frac{1}{2}(D_w)^2 + (\mathrm{Y})\frac{1}{2}\{\max(0, m - D_w)\}^2$$

Where Y is 0 if the inputs are form the same class, else 1 and *m* is a margin value that makes sure that dissimilar pairs that are above this will not be part of the final loss. *Dw* is the Euclidean distance between the two vectors which is computed as:

$$\sqrt{\{G_w(X1) - G_w(X2)\}^2}$$

During training this loss value is minimized until it is able to correctly predict if the two images belong to the same class or a different class. Once optimized, a SNN can predict any class as long as one known image exists of the respective class. We simply give one image input as the known image and the other as the image to classify and the output will be the dissimilarity between them. A low dissimilarity value indicates they belong to the same class. Thus this network does not have to be retrained to classify a new class. It only needs a single image and hence is called One Shot learning.

B. Genetic Algorithm

The genetic algorithm [11] is designed based on how evolution works. In the theory of evolution the best combination of genes survives while the others quickly die out. Implemented as an algorithm, the combination of genes is called a *population* or a *gene pool* and it is the set of parameters to be optimized. Firstly, we run a fitness function which is a heuristic evaluation of the *gene pool*. The result of the evaluation decides which combination of gens gives the best results. Then from these the best are selected as *parents*. We then run a cross over function over the *parents* to simulate mating and gene transfer. This results in a set of *offspring*. These offspring become our new *gene pool* and the process is repeated a number of times until we get an optimized set of *offspring*. The algorithm can be depicted as follows.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

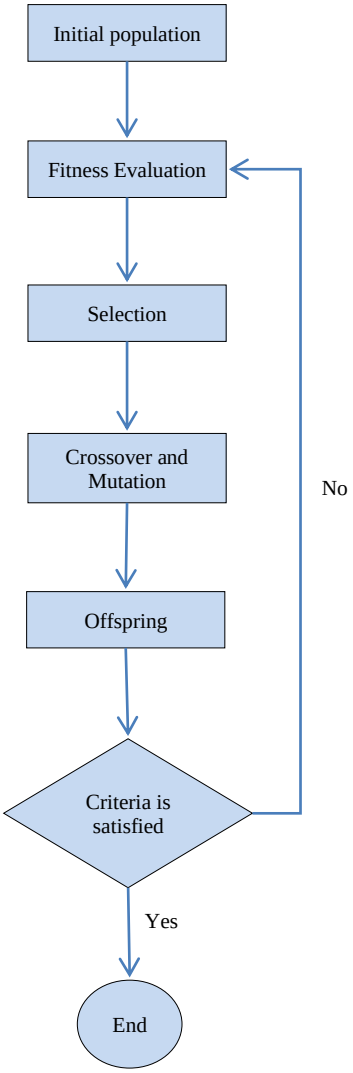

Fig. 2. The flow of a Genetic Algorithm

III. THE PROPOSED ALGORITHM

In this section we present the algorithm to be used. It will consist of the base framework followed by the steps followed by every component of the genetic algorithm

C. Algorithm Overview

Algorithm 1 presents the base framework that will run by combining all the genetic components. We give an initial population as the input to the algorithm.

Algorithm 1: *Base Framework of the Algorithm*

Input:
Set of initial population
Number of generations to run
Output:
Optimized population set.
Foreach No of generation **do**
Foreach Population **do**
F Evaluate the fitness function
End For
Select parents based on fitness
Generate Offspring using the crossover
Set the offspring as the new population
End for
Return The final population set

The output will be the optimized set of parameters. Then for the number of generations specified we run the complete gene cycle. We first compute the fitness of each element in the population which is the combination of parameters. The fitness function will return how well the particular set of parameters performs. Thus we get a list of accuracies for the respective parameter set. We then select the best out of these as parents for the next step. In the next step we generate offspring from the parent set that we have created. We would end up with twice the number of offspring as that of the parents. This offspring set is initialized as the new population over which the algorithm is run again. This is repeated until we have reached the number of generations needed. We then return the final population set. The combination among them that gives the best fitness value is chosen as the most optimized set of parameters.

D. Fitness Function

Algorithm 2 presents the fitness function that is to be used. The fitness function is a heuristic or approximate estimation of the performance of the network. Our fitness function essentially sets the parameters in the Siamese neural network and runs it. The parameters essentially change the architecture and training parameters of the neural network. Thus they have a high level correlation with the accuracy of the network. This accuracy is defined by the loss that that the network returns during testing. A high loss means that the network is not performing well at all. A low loss depicts higher levels of accuracy. The output of the fitness function will be the loss that the Siamese Neural Network returns after training the network and testing it. This gives us an approximation of the performance of the Neural Network using the hyper parameters specified.

Algorithm 2: *Fitness Function*

Input:
Combination of parameters
Output:
Final Loss Of the Network
Train the Siamese Neural Network using the combination of parameters specified by the input. These are:
Size of each batch
No of Epochs
Learning Rate
No of Layers in the SNN
Return The final loss

E. Training the Siamese Neural Network

Algorithm 3 presents the training involved in every individual run of the network. The input is the parameters that the fitness function sends to the training module and the training dataset. The network will be trained and tuned for a set number of iterations called epochs. For every epoch a certain set of image pairs are trained on which is called the batch size. Both of these are parameters we set. In the training module we set these parameters and also initialize the network with the parameters needed. Then for the number of epochs needed, we train the network on the set batch of randomly sampled image pairs. We then optimize the network using the contrastive loss function.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Algorithm 3: *Training the Siamese Neural Net*

Input:
Combination of parameters
Input training images
Output:
Trained Model
Loss and Training History
Load the training dataset
Set sampling configuration based on batch size
Initialize the network architecture using the parameters
Initialize the optimizer function with the learning rate
Foreach in epoch **do**
Sample random image pairs for training on
Train the SNN using the GPU
Calculate the contrastive loss function
Optimize the network based on function output
Save loss and iteration information
End For
Write the model to disk
Write loss history to disk
Return The final loss

F. Parent Selection and Crossover

When the fitness function returns the performance information for the population set, the network architectures with the highest accuracies are chosen as the selected parents for the mating pool. They will go on to the next stage which is the crossover.

Parent 1	1010	10010
Parent 2	1011	10110
Offspring 1	1010	10110
Offspring 2	1011	10010

Table 1. One point crossover to create two new offspring

Crossover [12] is when we take two set of parameters or genes and combine them in different combinations. This is demonstrated in Fig. 3. For every set of parents we will have two 1-Point Crossovers which will result in two offspring. So a certain portion of the parameters from Parent 1 will be merged with the complementary set from Parent 2 to form a new offspring. The crossover point is decided randomly.

Algorithm 4: *Generating Offspring*

Input:
Parent Gene Pool
Output:
Offspring Set
Foreach Parent Pair **do**
Select a random crossover point
Split the first parent from 0 to crossover pt
Split the second parent from crossover pt to end
Merge both splits to form an offspring
Repeat steps 4 to 7 for second offspring
End For
Return Offspring Set

Once the offspring are generated then they are considered as the new parent pool and the next generation run of the entire algorithm will take place. This will continue until the set numbers of generations are completed.

IV. IMPLEMENTATION

In order to implement our approach we needed a strong dataset and an efficient deep learning framework. The dataset used is the AT&T Laboratories face dataset. It contains a set of faces captured between April 1992 and April 1994 at the AT&T Laboratories [13]. Since these were captured in various poses, facial expressions, facial details and lighting conditions, they are a strong feature set for our training and testing purpose. The size of each image is 92x112 pixels. They have 256 grey levels per pixel. The images are organized in different directories (one for each subject), which have names of the form sX, where X indicates the subject number. In each of these directories, there are ten different images of that subject, which have names of the form Y.pgm, where Y is the image number for that subject (between 1 and 10).There are a total of 40 such sets of images.

Fig. 3. AT&T Laboratories face dataset

We set aside three sets of faces for testing that will be unseen data for the models. The rest of the images are samples and image pairs are created wherein some image pairs will be of the same person or class and the other pairs will be of two different people or different classes.
In order to implement the deep learning architecture, we used the pytorch [14] framework written in python. It allows for fast prototyping of Deep Learning Networks and also allows customization of the networks layers which is essential for us. It also support running on GPU using NVIDIA CUDA [11] for parallel computation as its backend is on C++ libraries.
We run the genetic algorithm on the network for five generations with an initial randomized population pool.

II. RESULTS AND OBSERVATIONS

The algorithm ran for five generations, training and selecting parameters based on the accuracy of the neural networks. Every model was trained on samples from the training population based on the given hyper parameters. The total time taken was 11hrs 32 min. The time taken per network run depended on the number of layers and the training epoch given. Higher number of epochs led to more number of training iterations. More number of layers resulted in longer training time per training iteration. The batch size also was directly proportional to the trainig time.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

```
Loading Training Data...
Building Model Architecure...
Parameters:
Batch_Size:32 No_of_Epochs:100 Learning_Rate:0.0001 Convolutional_Layers:3
Final Loss:1.0763094425201416,Total Time Taken:128.433425
```

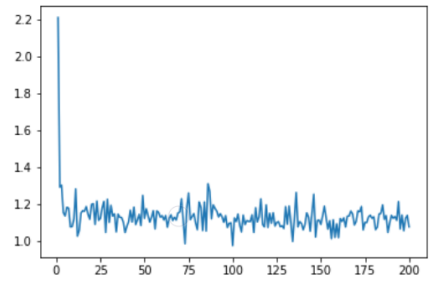

Fig. 4. Output from a single training run of one of the models. The graph represents the loss vs. number of iterations.(X=[1.0-200], Epochs; Y=[0-2.2], Losses)

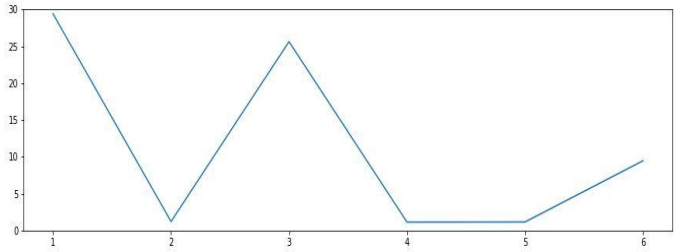

Fig. 5. Output of the final losses from the first generation (X=[0-6], Models; Y=[0-30], Losses)

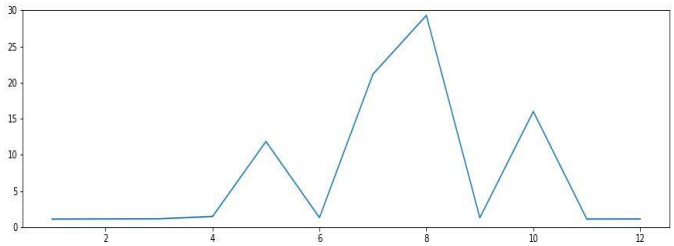

Fig. 6. Output of the final losses from the second generation (X=[0-12], Models; Y=[0-30], Losses)

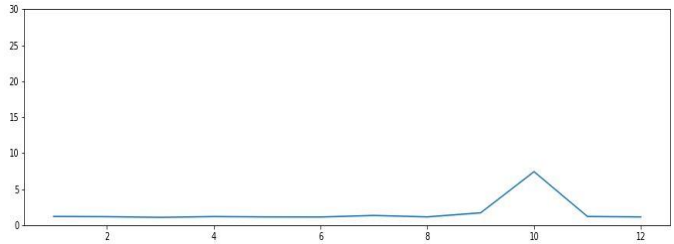

Fig. 7. Ouput of the final losses from the third generation (X=[0-12], Models; Y=[0-30], Losses)

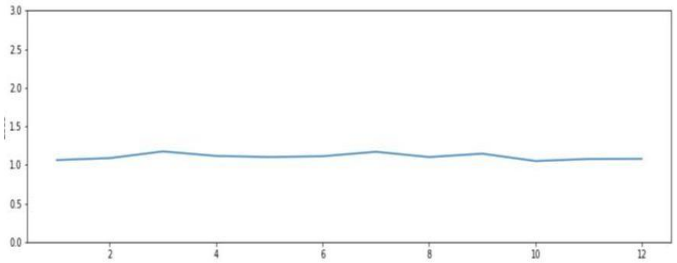

Fig. 8. Output of the final losses from the fourth generation (X=[0-12], Models; Y=[0-30], Losses)

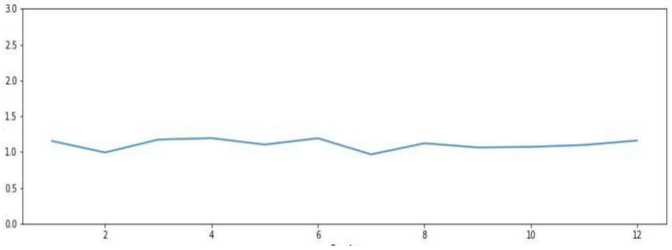

Fig. 9. Ouput of the final losses from the fifth generation (X=[0-12], Models; Y=[0-30], Losses)

In the initial generation the network losses fluctuated rapidly betweeen models. But as the crossover operations and fitness based selections were performed, the algorithm began to detect the optimum parameters or genes that gave the higher accuracies and lower losses. Lower learning rates were given more priority over higher rates. More number of layers combined with more number of epochs gave better performance. Intermediate batch sizes that were neither too small nor large became predominant in the later generations.As can be seen from Fig. 6. The first generation losses for the networks were very varied and on average very high. In the second generation the network losses fluctuation still exist. In the fourth and fifth generation the network losses has reduced drastically which implies a higher accuracy.

The benchmark Siamese model with standard contrastive loss function has lots fluctuations as shown in Fig. 11. Compared to it, the model that is fine tuned by the genetic algorithm, Fig. 12, showed lesser fluctuations and better loss stability.

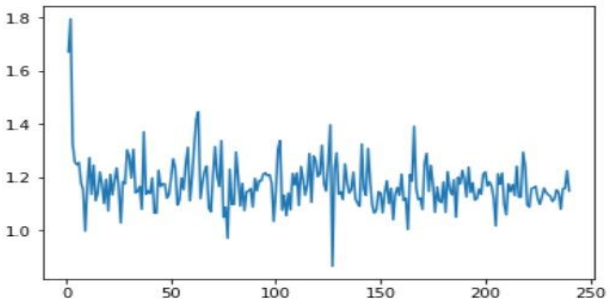

Fig. 10. Model training with standard contrastive loss function (X= Epochs; Y=Losses)

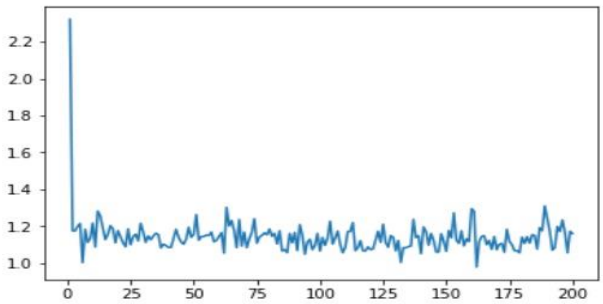

Fig. 11. Model training tuned by genetic algorithm (X= Epochs; Y=Losses)

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

V.CONCLUSIONS AND FUTURE WORK

From the above results we can conclude that the genetic algorithm is a very viable path to optimizing Deep Learning algorithms. They could also help us fins patterns of parameter combinations that work well for problem specific Deep learning models. This will give us better insight into the inner workings of neural networks which are a sort of Black Box at the moment.

In this paper we are setting the hyper parameters and the number of layers of the network using the genetic algorithm. But the number of neurons and the convolution operation parameters are kept static. This is due to the fact that the randomness of the genetic algorithm could alter the network architecture to unviable options. Further research can be expended into processing the genetic algorithm in a more controlled manner that will allow us to include these parameters as well in the gene pool and let the algorithm automate their optimization process as well.

Another area for improvement would be within the functioning of the genetic algorithm itself. Multiple forms of crossover operations can be applied to see the possible improvement they can provide. Also the fitness evaluation needs the complete training and testing of the SNN to be completed in order to give results. This is a computationally very expensive and extremely time consuming process. Future efforts will be placed on using effective evolutionary methods like dynamic programming to speed up the fitness evaluation process.

REFERENCES

1. Gyanesh Sinha, "Study of Indian Logistics Industry in Changing Global Scenario", ResearchGate, February 2016
2. Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner, "Gradient-Based Learning Applied to Document Recognition", IEEE, November 1998
3. Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, Juergen Schmidhuber, "Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition", Cornell University Library, Volume 22, Number 12, December 2010
4. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", Neural Information Processing Systems Conference, 2012.
5. Eugenio Culurciello, "The History of Neural Networks", Dataconomy, April 2017. Available: Dataconomy, https://dataconomy.com
6. K. Simonyan, A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” International Conference on Machine Learning, 2015.
7. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, “Going deeper with convolutions”, IEEE Conference on ComputerVision and Pattern Recognition, 2015
8. K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition”, IEEE Conference on Computer Vision and Pattern Recognition, 2016
9. Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition",
10. Sumit Chopra, Raia Hadsell, Yann LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification", Yann LeCun's Publications, 2005
11. K.F. Man, K.S. Tang, S. Kwong, "Genetic algorithms: concepts and applications",IEEE, Volume 43, Issue 5, October 1996.
12. A.J. Umbarkar, P.D. Sheth, "Crossover Operators in Genetic algorithms: A review", ICTACT Journal on Soft Computing, Volume 6, Issue 1, October 2015.

AUTHOR PROFILE

Vishal Prem *SCOPE, Department of Computer Science and Engineering, Vellore Institute of Technology,* Vellore, India

Mark Sheridan Nonghuloo, *SCOPE, Department of Computer Science and Engineering, Vellore Institute of Technology,* Vellore, India

Nagaraja Rao A, *SCOPE, Department of Computer Science and Engineering, Vellore Institute of Technology,* Vellore, India

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)