Classification of Images using Transfer Learning

Utsav Jain
Computer Science and Engineering
Meerut Institute of Engg. And Technology,
Meerut , India
utsav.jain.cse.2017@miet.ac.in

Suchi Aeron
Computer Science and Engineering
Meerut Institute of Engg. And Technology,
Meerut , India
suchi.r.cse.2017@miet.ac.in

Muskan Singh
Computer Science and Engineering
Meerut Institute of Engg. And Technology,
km.muskan.cs.2017@miet.ac.in

Aditi Joshi
Computer Science and Engineering
Meerut Institute of Engg. And Technology,
aditi.joshi@miet.ac.in

Abstract -- **For better accuracy and faster convergence, image classification using Transfer Learning and deep feed-forward neural networks (DFNN) is important. We have used existing models in Transfer Learning that save training time, improve neural network performance, and do not require a lot of data. A number of image classification models have been developed so far that articulates the major issue concerned with recognition accuracy. Image recognition is the most critical problem encountered in the areas applying practical applications of Computer Vision. The attribute of Object Recognition governed by autonomous vehicles with robotic influences, obstacle or pedestrian detection system, etc are few of the practical examples dealing with recognition accuracy. Machine learning, especially neural networks like the (DFNN) that won image classification competitions, has received a lot of attention. Such a DFNN architecture model should be researched and investigated, according to this article. Using new image datasets to see if Transfer Learning will perform better in terms of precision and productivity.**

There are some similarities to state-of-the-art approaches. The DFNN can be tweaked by changing the total number of hidden layers, hidden neurons in each hidden layer, and the number of connections made in between layers.

Keywords: Transfer Learning (TL) · Hidden neurons (HN) · Hidden layer (HL) · Tabu search (TS) · Deep feed-forward neural network (DFNN)

I. INTRODUCTION

In this project we use Transfer Leaning to create a model which take image as an input and classify the class of an object in the given image. The ability of a method to identify and apply information and skills gained in a previous task that has certain parallels. To train a model, Transfer Learning needs less data and time.

The Deep feed-forward neural network (DFNN) is a concept that is based on artificial neural networks (ANNs). The ANNs are composed of an input layer, a hidden layer, and then an output layer. It has a straightforward interface, excellent learning capabilities, and the ability to solve complex problems .

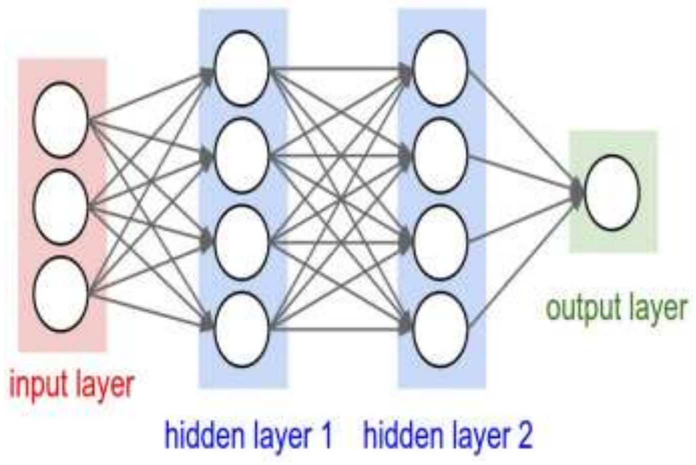
The Deep Feed-forward Neural Network (DFNN) model relies on ANNs`. It is a mathematical model that mimics the processing of human brain in interpreting the stimuli (knowledge). ANN is comprised of minimum three layers named as an input layer, a hidden layer and an output layer. It has a fundamental essence, a high learning capacity, and the ability to solve complex problems .It has a basic nature, a strong capacity for learning, and is able to solve complex problems .The problem occurs when an application demands that the dataset be divided into 100 or more levels. With simple ANN, learning functions for 100 or 1000 classes are not possible; therefore, deeper architecture is needed.

In a DFNN, the number of processing units i.e. neurons varies at the input layers and output layers varies depending on the complexity of the problem. There are no obvious set of rules or parameters that can support in determining the number of hidden layers in DFNN. This means that DFNN's architectural design is crucial and can be thought of as an optimization challenge. The solution to this challenge is the populated architectures in DFNN optimization with testing error as a cost function. The sole aim of this optimization technique is to detect the optimum structure attributed with the lowest testing error.

In case of a single hidden layered feed-forward neural network (FNN), only hidden neurons need to be specified for that layer, but the layout is quite complicated in the case of DFNN. This paper aims to contribute a new approach for searching the best configuration for a DNN that unifies the advantages of Tabu search as well as Gradient-descent approach with the momentum backpropagation training algorithm. It solves the issues dealing with identification of hidden layers and their associated neurons automatically, which was formerly a manual operation.

We use various activation functions in this project, such as Relu, Sigmoid, Tanh, etc.

A neuron uses the Relu activation function to weight all of its inputs, add a value knownas "bias," and put the result through a "activation function." The weights and biases were initially unknown. The neural network will be initialised at random and "learned" after being trained on a large amount of known data.



II. METHODOLOGY

A popular meta-heuristic approach for optimizing model parameters is Tabu Search. A meta-heuristic is a general approach to directing and manipulating real heuristics. Memory systems are also thought to be built into Tabu Search's local search strategies. Tabu Search was created to correct some of the shortcomings that exist in local search.The basic principle of Tabu Search is that gestures that bring the solution into previously visited search spaces are penalised (also known as tabu). Tabu Search, on the other hand, takes non-improving options into account deterministically to prevent getting stuck in local minimums.

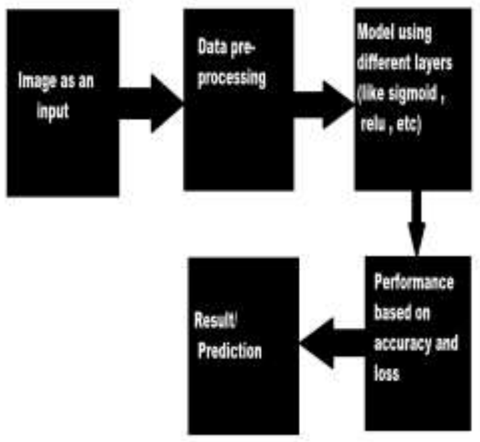
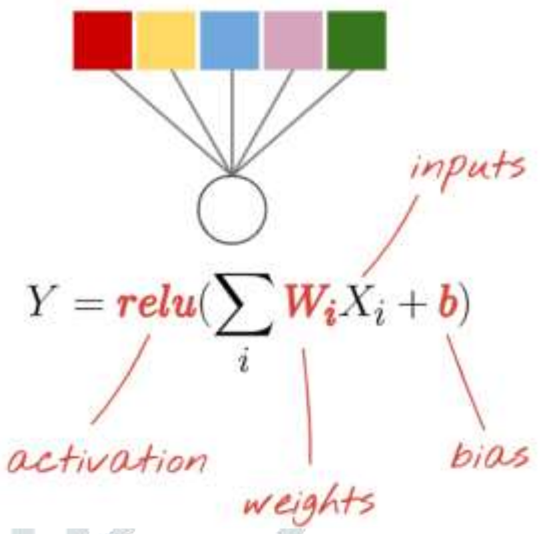


Image as an input
Data pre-processing
Model using different layers (like sigmoid , relu , etc)
Performance based on accuracy and loss
Result/ Prediction

A structured process is illustrated in the diagram that shows the flow of the research performed in the model building.We use various activation functions in this, such as relu, sigmoid, softmax,etc

RELU ACTIVATION:

The rectified linear activation function (ReLU) is a linear function that will directly provide an output if the input is positive, else the output will be zero. The disappearing gradient problem is solved by the rectified linear activation function, which allows models to adaptmore quickly andbetter results are achieved.The greatest advantage of using the ReLU feature over other stimulation functions is that all of the neurons are not activated simultaneously. Due to this, during the backpropagation process, the weights and biases for certain neurons are not altered. This can create neurons that are dead that are never activated.



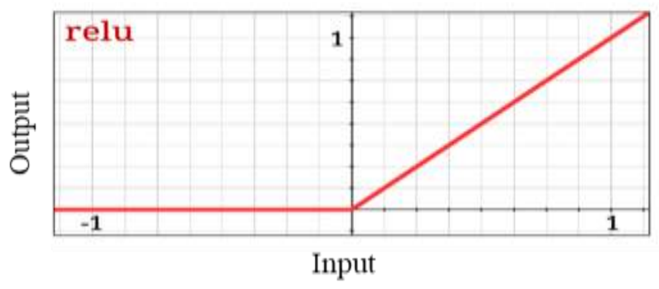


Fig. RELU FUNCTION

SIGMOID ACTIVATION:

Traditionally, the sigmoid activation feature is a very common activation function for neural networks, also called the logistic feature. The function's input is converted to a value between 0.0 and 1.0.0. Since anything is only likely in the 0 and 1 range, the correct option is a sigmoid. The behaviours are distinguishable.

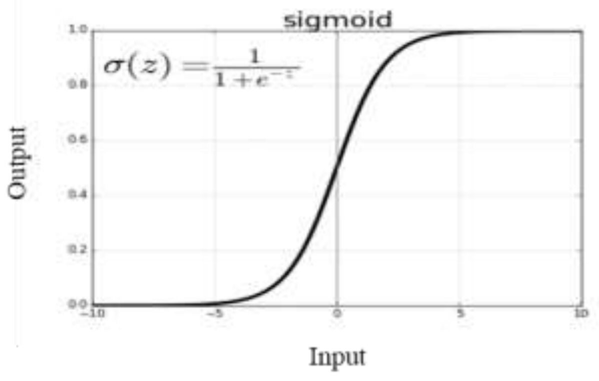


Fig. SIGMOID ACTIVATION FUNCTION

SOFTMAX ACTIVATION

SoftMax is a property of activation. SoftMax is exponential and raises variations - moving one outcome closer to 1 while another closer to 0. It transforms scores into probabilities aka logits. The output of SoftMax and true labels is also calculated for cross entropy (cost function) (encoded in one hot encoding). SoftMax is applied to a vector by taking the exponential of each variable and then normalising the vector using the L1 standard such that the values can be represented as probabilities and add up to 1.

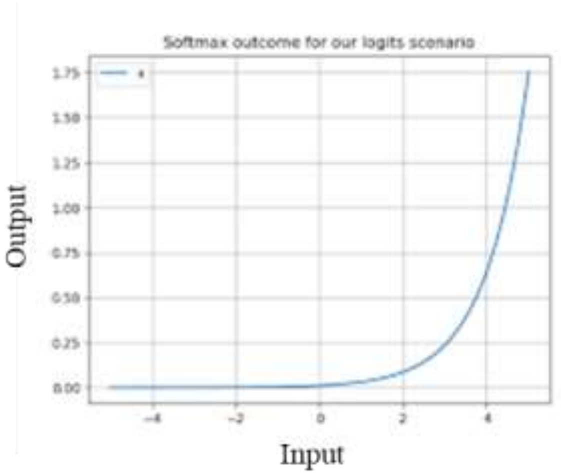
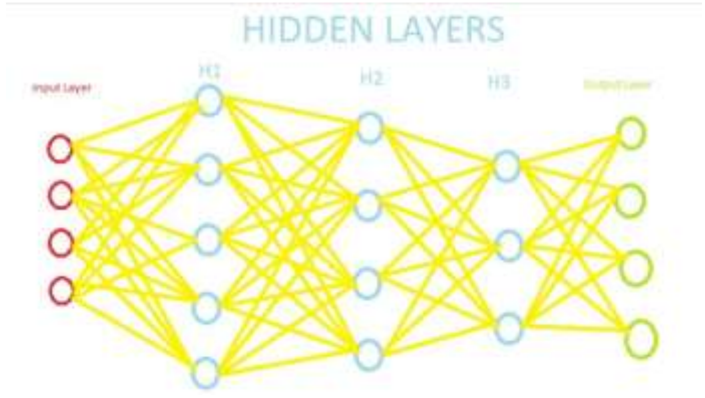


Fig.SOFTMAX ACTIVATION FUNCTION

This article combines Tabu search with Gradient descent utilizing the momentum backpropagation (GDM) learning algorithm. This optimization aims to use GDM for finding best solutions from a set of solutions S, such that f (s') ≤f (s), where f represents a real-value of cost function, and for all sES. The technique begins with a hidden layer of FNN and initializes hidden neurons randomly.Tabu search employs GDM to create a neighbourhood around each solution and accept the best of it. The flow map for the proposed approach is shown in Figure 1. It functions with the maximum number of hidden layers that can be varied. The maximum number is set to 5 here. A solution iterates several times; with each iteration, a population is generated of size Pmax which is centred on the solution that is selected and build on some the fitness function. If the best of I ter called s is less than sbest, the sbest get upgraded and move to the next iteration and if not then update the Tabu list and continue exploring s until no stopping criteria are reached. The following concepts have to be considered for its implementation: (1) solution representation, (2) population generation.



Above figure is the architecture of our model, which contains fully connected network under which there are three hidden layers three of them contains Relu activation functions. First layer is input layer which have 3072 neurons and the last layer is output layer whichhave 10 output neurons. This architecture is fully connected architecture in which each and every neurons is connected to every neurons of next layer.

In this model there are three hidden layers first hidden layer (i.e. H1) contain 598 neurons with Relu activation function , second HL (i.e. H2) contains 379 neurons with Relu activation function and the third HL i.e.H3 contains 253 neurons with Relu activation function and the output layer have 10 output neurons with softmax activation function.

Representation of Solution

The DFNN comprising of multiple hidden layers is considered for the evaluation. Each of the network comprises of I number of input nodes, Hi hidden nodes at the hidden layer, and O output nodes. Generally, the number of input and output nodes determined by the complexity of the problem that aims to determine the optimal number of hidden layers and nodes. The DFNN architecture can be represented as:

A=(I*H1+B*H1)+(H1*H2+B*H2)+…….+(Hmax ***O+B*O)** (1)

where ‘B' stands for prejudice. A solution has three attributes; HL which contains a number of hidden layers, HN which contains the number of layer-wise hidden neurons in a vector. TE denotes the testing error of the solution is denoted by TE.

S= (HL,HN,TE) (2)

HN = (H1,H2,….Hmax**), H**i∈ **N** (3)

HL=(1,2,….max) , & TE∈ _ (4)

The first approach implements a complete connected FNN with a set number of input and output neurons and a single input and output neuron.a sheet that is secret. A random selection method with a range between.[(I+O)/2, (I+O)2/3] determines the layers' hidden neurons. The starting weights are uniformly distributed between the intervals of [1.0, +1.0], and the chosen set is dependent on two thumb laws.

Generating Population

The evaluation of initial solution i.e. s = (HL, HN, TE), the suggested approach populates an entire generation of solutions via implementing algorithm 2 before the stopping requirements are satisfied. The following measures are used to build each new population solution: At a special layer, split the population size into two equal sections, one for growth and the other for neuron decline.

If (W>=p)

H_N=H_N+K

Else

No_change

Where K+ denotes a K percent increment in the number of neurons, till the value does not exceed the upper boundary. The Case 2 produces a random number in the similar way:

If (W>=p)

H_N=H_N-K

Else

No_change

where K- denotes a K percent decrease in the number of neurons before a lower limit is reached. This method of finding novel ideas helps the hunt for the next global approach to go forward as well as backward.
The algorithm will halt after optimising max secret layers up to I terterations in this mechanism. However, in some cases, it stops nerves from being altered. It automatically goes on to the next step when the algorithm tries to add on the number of neurons and hits its threshold mark; or when the algorithm tries to diminish the number of neurons. It is now at its lowest stage.

DATASET

The dataset taken here is CIFAR-10 which consists of around 60000 32x32 colour images in 10 classes, with an average of 6000 images per class. Each with 10000 images, is split into five training batches and one test set. There are 10000 exam images and 50000 preparation images. There are precisely 1000 randomly-selected photographs from each class in the test batch. The training batches randomly hold the left-out images with some training batches holding relatively more images. The training batches between them contain precisely 5000 images from each class.
Classes are mutually exclusive entirely. Automobiles and trucks do not overlap. "Automobile" involves SUVs, Sedans, etc while "Truck" only encompasses huge truck vehicles itself. It doesn't include even pickup trucks either.

The mark knowledge is a list of 10,000 items from 0 to 9, one for each of the ten CIFAR-10 classes

- **airplane : 0**
- **automobile : 1**
- **bird : 2**
- **cat : 3**
- **deer : 4**
- **dog : 5**
- **frog : 6**
- **horse : 7**
- **ship : 8**
- **truck : 9**

RESULTS

The proposed algorithm will be implemented using Python in the experiment.The min-max approach normalizes the datasets.For classification datasets measure the efficiency of methodology. The paper aims to discover the optimum DFNN architecture. When the dataset of interest holds large number of functions, using more than one hidden layer becomes necessary.

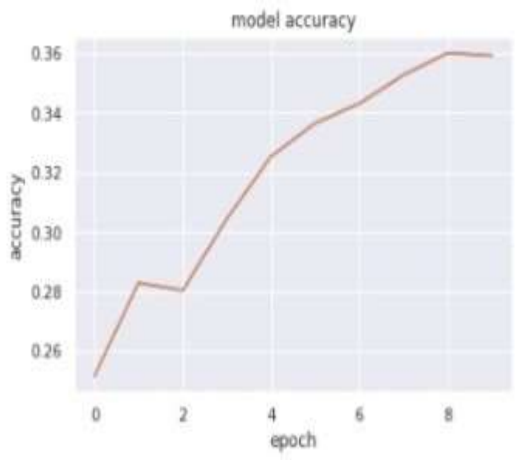


Fig. Training Accuracy Graph

(This graph contains training accuracy of model after 10 epochs but after 150epochs we get 85% accuracy.)

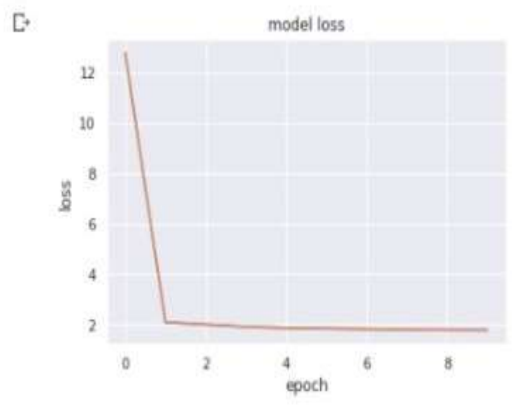


Fig. Training loss Graph

(This figure contains graph to show training loss of model after 10 epochs).

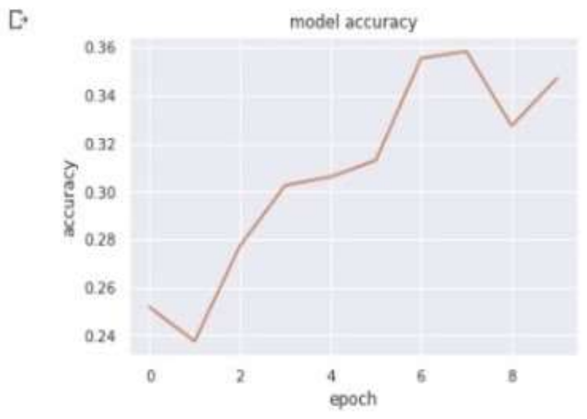


Fig.Validation Accuracy

(This figure contains validation accuracy of model after 10 epochs)

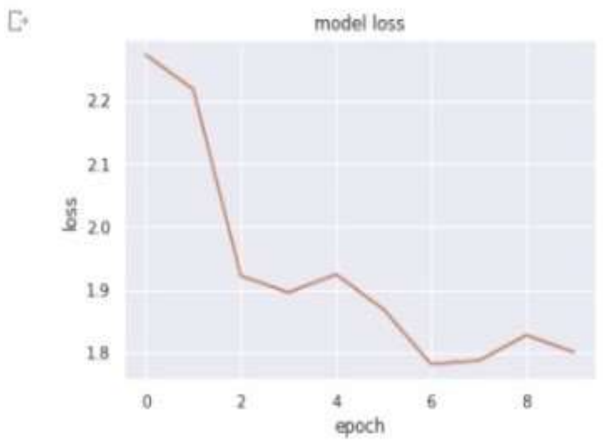
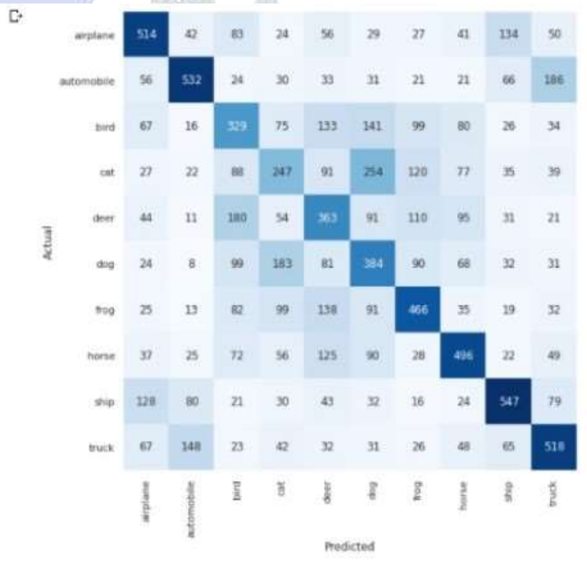


Fig. Validation loss

(This figure contains graph to show validation loss of model after 10 epochs).

HEATMAP



A heatmap is a two-dimensional graphical data representation where colours are interpreted as the individual values found in a matrix. The seaborn python package enables annotated heatmaps to be generated that can be modified using Matplotlib software as required by the developer.

A heatmap is the graphical representation of data in a matrix where colours reflect the individual values that are found in a matrix. It is analogous to viewing something above at a data table. It is very helpful not to extract individual data points, but to show a generalized view of numerical data. As shown in the examples below, it is very easy to construct a heat map.

However, keep in mind the mechanisms that underpin them. Firstly, there is a need to normalise the matrix, then choose an appropriate colour palette, followed by performing cluster analysis, and lastly allowing matrix rows and columns to cluster related values together.

CONCLUSION

In order to search for an optimal hidden layers (HL) and their hidden neurons (HN), this study was initiated. For the DFNN, with minimal testing error, it was also taken into considerationfrom the outset that the dataset used for research should have a huge amount of characteristics and should be categorized into multiple grades.This research demonstrates that by merging TS and GDM, we are able to get optimal design that would have been difficult to get when using single random selection process.

A completely connected feed-forward neural network reflects the network studied here (There is still space for developmentwith several secret layers).In addition, the job can be expanded to. In order to discover the best relations in a same solution There are also several additional optimization approaches.To fix these concerns, GA, SA, PSO, etc., can be hybridized.

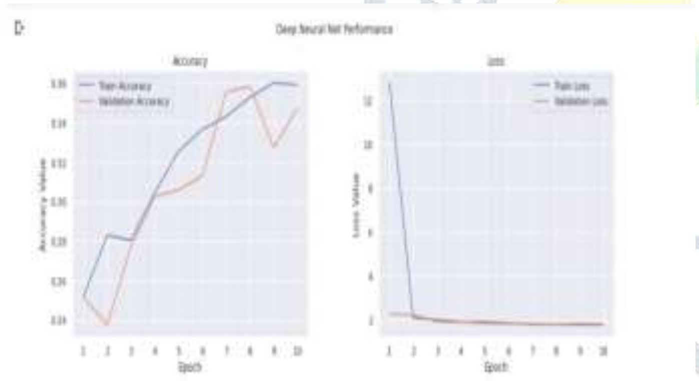


Fig. Combined Result of Accuracy and Loss

REFERENCE

[1] Tarun Kumar Gupta and Khalid Raza, *Optimizing Deep Feedforward Neural Network Architecture: A Tabu Search Based Approach* Springer Nature 2020

[2] Karl Weiss, Taghi Khoshgoftaar and DingDing Wang,*A survey of transfer learning* :Springer 2016

[3] Fred Glover,Manuel Laguna and Rafael Marti,*Principles of Tabu Search*,2008

[4] Mark Jovic A. Daday,Arnel Fajardo and Ruji Medina, *Enhancing Feed-Forward Neural Network in Image Classification: the 2nd International Conference October 2019*

[5] A Vehbi Olgac and Bekir Karlik,*Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Network, February 2011*

[6] TensorFlow (2018). Image Recognition. [ONLINE] Available:https://www.tensorflow.org/tutorials/image_recognition.

[7] Yasmine Hamdi the CIFAR-10 dataset. [ONLINE] Available:https://medium.com/@yasminehamdi/transfer-learning-experiment-on-cifar-10-dataset-41aa75f75989.