**VGG-16 ARCHITECTURE**

**VGG16** is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It can find object name of the image. It can detect and identify the fed image among thousands of images. It takes input image of size 224 \* 224 \* 3 (RGB image). The model size is 528MB It is built using

Convolutions layers (used only 3\*3 size) Max pooling layers (used only 2\*2 size)

Fully connected layers at end Total 16 layers

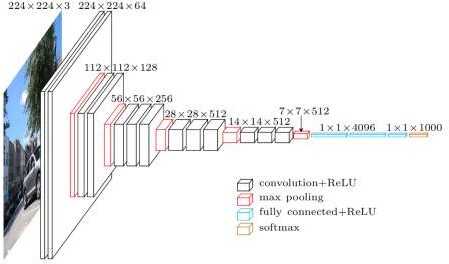
## BLOCK DIAGRAM

|  |  |  |
| --- | --- | --- |
| **Layers** | **Output shape** | **Parameters** |
| Conv2D | (None, 256, 256, 64) | 1792 |
| Conv2D | (None, 256, 256, 64) | 36928 |
| Max Pooling 2D | (None, 128,128, 64) | 0 |
| Conv2D | (None, 128,128,128) | 73856 |
| Conv2D | (None, 128,128,128) | 147584 |
| Max Pooling 2D | (None, 64, 64, 128) | 0 |
| Conv2D | (None, 64, 64, 256) | 295168 |
| Conv2D | (None, 64, 64, 256) | 590080 |
| Conv2D | (None, 64, 64, 256) | 590080 |
| Max Pooling 2D | (None, 32,32,256) | 0 |
| Conv2D | (None, 32,32,512) | 1180160 |

|  |  |  |
| --- | --- | --- |
| Conv2D | (None, 32,32,512) | 2359808 |
| Conv2D | (None, 32,32,512) | 2359808 |
| Max Pooling 2D | (None, 16,16,512) | 0 |
| Conv2D | (None, 16,16,512) | 2359808 |
| Conv2D | (None, 16,16,512) | 2359808 |
| Conv2D | (None, 16,16,512) | 2359808 |
| Max Pooling 2D | (None, 8,8, 512) | 0 |
| Flatten | (None, 32768) | 0 |
| Dense | (None, 4096) | 134221824 |
| Dense | (None, 4096) | 16781312 |
| Dense | (None, 36) | 8194 |
| Total Parameters |  | 165,726,018 |
| Trainable parameters |  | 165,726,018 |
| Non-trainable parameters |  | 0 |

**VGG 16 Architecture block diagram**

**VGG16 ARCHITECTURE LAYOUT**



**Architecture Layout**

**ARCHITECTURE EXPLANATION**

**CONVOLUTION LAYER**

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers.

## RELU ACTIVATION FUNCTION

It’s a function that you use to get the output of node. It is also known as **Transfer Function**. It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

The Activation Functions can be basically divided into 2 types-

* + - 1. Linear Activation Function
      2. Non-linear Activation Functions

## BATCH NORMALISATION

Batch Normalization is a technique to provide any layer in a Neural Network with inputs that are zero mean/unit variance. It is called “Batch” Normalization because we perform this transformation and calculate the statistics only for a subpart (a batch) of the entire trainings set. A batch normalization layer normalizes each input channel across a mini-batch. To speed up training of convolutional neural networks and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers.

## MAXPOOLING

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. Maxpooling uses the maximum value from each of a cluster of neurons at the prior layer. It reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

## FLATTEN LAYER

Flattening is the process of converting all the resultant 2 dimensional arrays into a single long continuous linear vector. The flattening step is needed so that you can make use of fully connected layers after some convolutional layers. We can combine all the found local features of the previous convolutional layers. Each feature map channel in the output of a CNN layer is a "flattened" 2D array created by adding the results of multiple 2D kernels.

## DROPOUT LAYER

Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. A Simple Way to Prevent Neural Networks from Overfitting. In this technique randomly selected neurons are ignored during training. They are “dropped-out” randomly.

## FULLY CONNECTED LAYER

In a fully connected layer, each neuron receives input from every element of the previous layer. The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network.

## SOFTMAX CLASSIFIER

The Softmax function is often used in the final layer of a neural network- based classifier. Such networks are commonly trained under a log loss (or cross- entropy) regime, giving a non-linear variant of multinomial logistic regression.

## HYPERPARAMETERS

**Hyper parameters** are the **variables which determines the network structure** and the **variables which determine how the network is trained**. **Hyper parameters** are **set before training** (before optimizing the weights and bias). Deep learning often refers to those hidden elements as hyper parameters as they are one of the most critical components of any machine learning application. Hidden layers are the layers between input layer and output layer. Many hidden units within a layer with regularization techniques can increase accuracy. Smaller number of units may cause **under fitting**.

## EPOCH

An Epoch is a complete pass through all the training data. Number of epochs is the number of times the whole training data is shown to the network while training.Increase the number of epochs until the validation accuracy starts decreasing even when training accuracy is increasing(overfitting).A Neural network is trained until the error rate is acceptable, and this will often take multiple passes through the complete data set. An iteration is when parameters are updated and is typically less than a full pass.

## BATCH SIZE

Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration. **A good default for batch size might be 32.** Also try 32, 64, 128, 256, and so on. The batch size can be one of three options:

1. Batch mode: where the batch size is equal to the total dataset thus making the iteration and epoch values equivalent
2. Mini-batch mode: where the batch size is greater than one but less than the total dataset size. Usually, a number that can be divided into the total dataset size.
3. Stochastic mode: where the batch size is equal to one. Therefore, the gradient and the neural network parameters are updated after each sample.

## OPTIMIZERS

Optimizers shape and mold model into its most accurate possible form by futzing with the weights. he loss function is the guide to the terrain, telling the optimizer when it’s moving in the right or wrong direction.

## ADAM

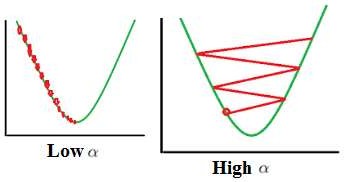
Adam stands for adaptive moment estimation, and is another way of using past gradients to calculate current gradients. Adam is a popular algorithm in the field of deep

learning because it achieves good results fast. Empirical results demonstrate that Adam works well in practice and compares favourably to other stochastic optimization methods. Adam was demonstrated empirically to show that convergence meets the expectations of the theoretical analysis. Adam was applied to the logistic regression algorithm on the MNIST character recognition and IMDB sentiment analysis datasets, a Multilayer Perceptron algorithm on the MNIST dataset and Convolutional Neural Networks on the CIFAR-10 image recognition dataset. They conclude that using large models and datasets, Adam can efficiently solve practical deep learning problems.

## LEARNING RATE

The learning rate is a hyper parameter that controls how much to change the model in response to the estimated error each time the model weights are updated. The learning rate may be the most important hyper parameter when configuring your neural network. Therefore it is vital to know how to investigate the effects of the learning rate on model performance and to build an intuition about the dynamics of the learning rate on model behaviour. The objective of any neural network is to reach the global minima which is this global minimum error of the model. Learning rate is the speed at which the model travels towards this global minima.

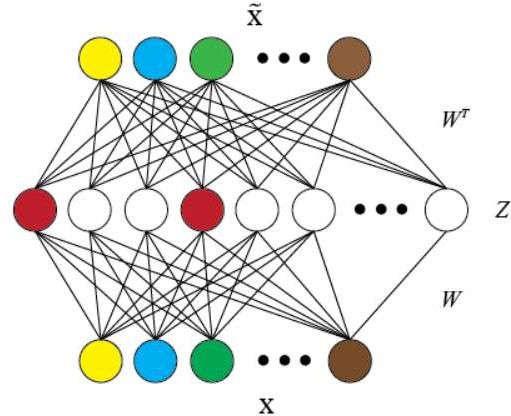
Low learning rate slows down the learning process while higher learning rate speeds up the learning process. The red line indicates that learning pattern followed by the model on the gradient descent optimizer.



## Learning Rate Graph

## SPARSE ENCODER

An autoencoder takes the input image or vector and learns code dictionary that changes the raw input from one representation to another. Where in sparse autoencoders with a sparsity enforcer that directs a single layer network to learn code dictionary which in turn minimizes the error in reproducing the input while restricting number of code words for reconstruction. The sparse autoencoder consists a single hidden layer, which is connected to the input vector by a weight matrix forming the encoding step. The hidden layer then outputs to a reconstruction vector, using a tied weight matrix to form the decoder. An advancement to sparse autoencoders is the k-sparse autoencoder. Here we choose k neurons with highest activation functions ignoring other activation functions using ReLU activation functions and adjusting the threshold to find the largest neurons. This tune the value of k to obtain sparsity level best suited for the dataset.



## SPARSE ENCODER

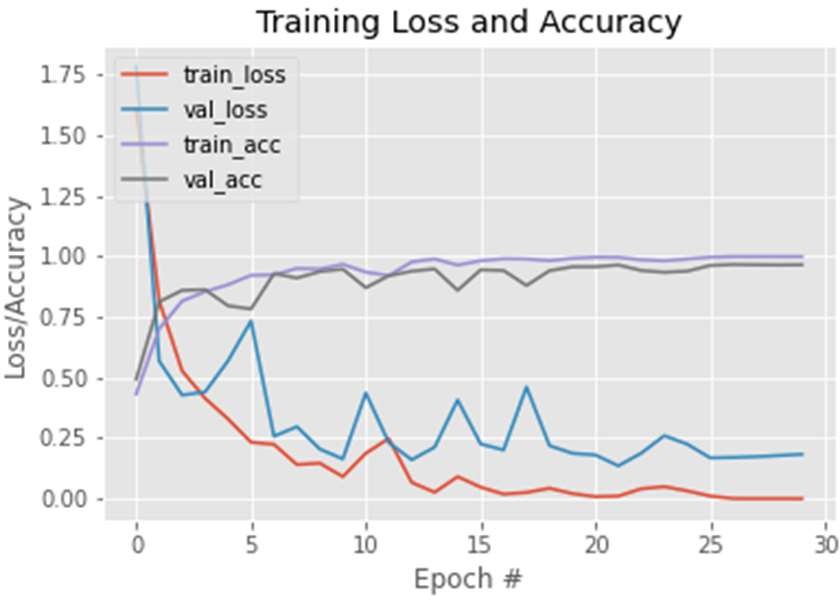
**CHAPTER 5**

**CUSTOM WEIGHT INITIALIZATION USING SPARSE ENCODER**

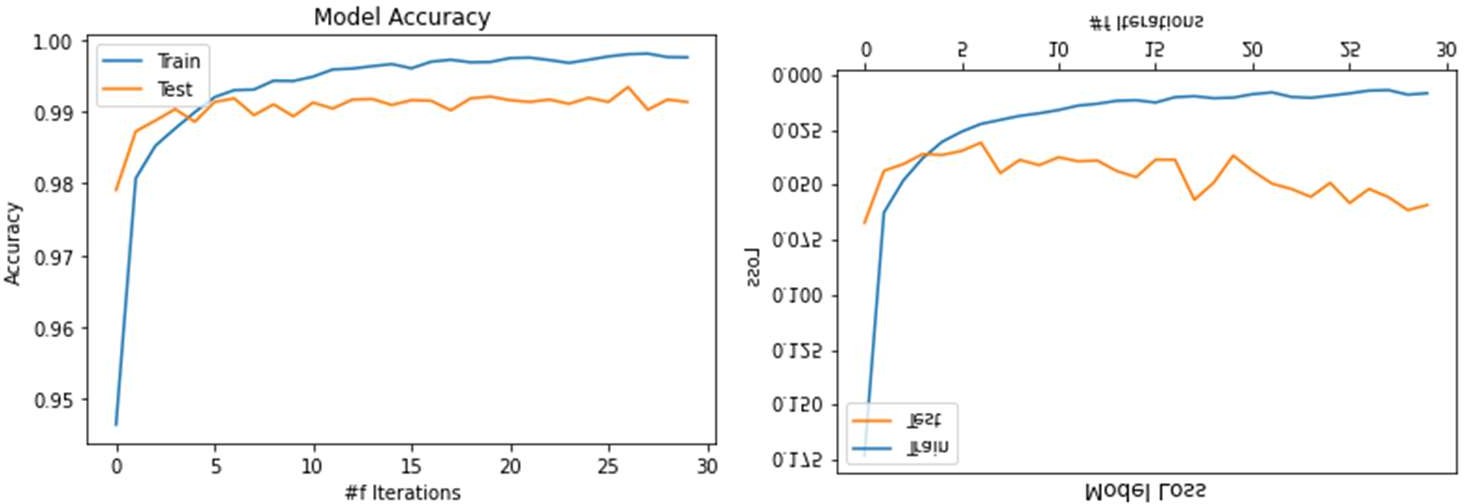
## DESCRIPTION

We have designed a sparse encoder using keras and used it as a custom weight initializer. It has improved the weight updation work of the neural network and has given better results. We have used the sparse encoder at the beginning of the VGG-16 convolution neural network so that the weight is updated.

## COMPARISION OF CNN AND CUSTOM WEIGHT UPDATED CNN



**VGG-16 TRINING LOSS AND ACCURACY**



**CUSTOM WEIGHT UPDATED VGG-16 TRAINING LOSS AND ACCURACY**