1)what is the COVARIATE SHIFT issue,and how does it affect you?

Ans : Covariate shift is a specific type of dataset shift often encountered in machine learning. It is when the distribution of input data shifts between the training environment and live environment. Although the input distribution may change, the output distribution or labels remain the same. Covariate shift is also known as covariate drift, and is a very common issue encountered in machine learning. Models are usually trained in offline or local environments on a sample of labeled training data. It’s not unusual for the distribution of inputs in a live and dynamic environment to be different from the controlled training environment. Covariate shift is a common problem faced within the supervised type of machine learning methodology. It will occur when a model has been trained on a dataset with a distribution which is much different to new datasets. Because the distribution of input variables has shifted, the model may misclassify data points in a live environment.common approach to correcting covariate shift is impor- tance reweighting: each individual training point is assigned a positive weight intended to diminish the discrepancy between training and test marginals by some criterion

2)what is the process of batch normalization ?

Ans : Batch normalization is a technique for training very deep neural networks that normalizes the contributions to a layer for every mini-batch. This has the impact of settling the learning process and drastically decreasing the number of training epochs required to train deep neural networks.Using batch normalization allows us to use much higher learning rates, which further increases the speed at which networks train. Makes weights easier to initialize Weight initialization can be difficult, and it's even more difficult when creating deeper networks.

3)using our own terms and diagrams,explain LENET architecture.

Ans : The LeNet architecture is straightforward and small, (in terms of memory footprint), making it perfect for teaching the basics of CNNs — it can even run on the CPU making it a great “first CNN”.The LeNet architecture is an excellent “first architecture” for Convolutional Neural Networks (especially when trained on the MNIST dataset, an image dataset for handwritten digit recognition).

LeNet is small and easy to understand — yet large enough to provide interesting results. Furthermore, the combination of LeNet + MNIST is able to run on the CPU, making it easy for beginners to take their first step in Deep Learning and Convolutional Neural Networks.

As a representative of the early convolutional neural network, LeNet possesses the basic units of convolutional neural network, such as convolutional layer, pooling layer and full connection layer, laying a foundation for the future development of convolutional neural network. As shown in the figure (input image data with 32\*32 pixels) : LeNet-5 consists of seven layers. In addition to input, every other layer can train parameters. In the figure, Cx represents convolution layer, Sx represents sub-sampling layer, Fx represents complete connection layer, and x represents layer index

Layer C1 is a convolution layer with six convolution kernels of 5x5 and the size of feature mapping is 28x28, which can prevent the information of the input image from falling out of the boundary of convolution kernel.

Layer S2 is the subsampling/pooling layer that outputs 6 feature graphs of size 14x14. Each cell in each feature map is connected to 2x2 neighborhoods in the corresponding feature map in C1.

Layer C3 is a convolution layer with 16 5-5 convolution kernels. The input of the first six C3 feature maps is each continuous subset of the three feature maps in S2, the input of the next six feature maps comes from the input of the four continuous subsets, and the input of the next three feature maps comes from the four discontinuous subsets. Finally, the input for the last feature graph comes from all feature graphs of S2.

Layer S4 is similar to S2, with size of 2x2 and output of 16 5x5 feature graphs.

Layer C5 is a convolution layer with 120 convolution kernels of size 5x5. Each cell is connected to the 5\*5 neighborhood on all 16 feature graphs of S4. Here, since the feature graph size of S4 is also 5x5, the output size of C5 is 1\*1. So S4 and C5 are completely connected. C5 is labeled as a convolutional layer instead of a fully connected layer, because if LeNet-5 input becomes larger and its structure remains unchanged, its output size will be greater than 1x1, i.e. not a fully connected layer.

F6 layer is fully connected to C5, and 84 feature graphs are output.

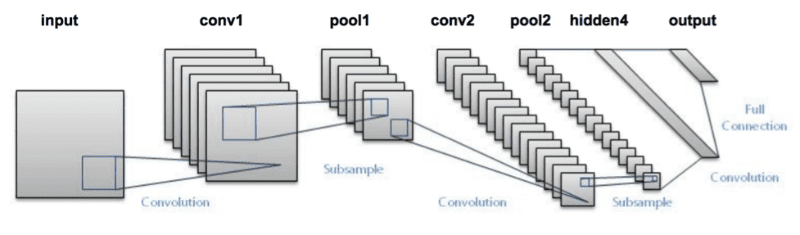


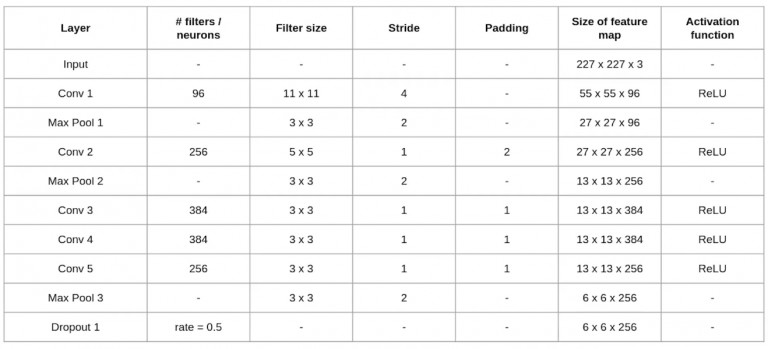
Fig : LeNet Architecture.

4)using our own terms & diagrams explain the ALEXNET architecture .

Ans : The Alexnet has eight layers with learnable parameters. The model consists of five layers with a combination of max pooling followed by 3 fully connected layers and they use Relu activation in each of these layers except the output layer.

They found out that using the relu as an activation function accelerated the speed of the training process by almost six times. They also used the dropout layers, that prevented their model from overfitting. Further, the model is trained on the Imagenet dataset. The Imagenet dataset has almost 14 million images across a thousand classes. Alexnet is a deep architecture, the authors introduced padding to prevent the size of the feature maps from reducing drastically. The input to this model is the images of size 227X227X3.

Convolution and Max Pooling Layers:



Then we apply the first convolution layer with 96 filters of size 11X11 with stride 4. The activation function used in this layer is relu. The output feature map is 55X55X96.

how to calculate the output size of a convolution layer output= ((Input-filter size)/ stride)+1

Also, the number of filters becomes the channel in the output feature map.

Next, we have the first Max Pooling layer, of size 3X3 and stride 2. Then we get the resulting feature map with the size 27X27X96.

After this, we apply the second convolution operation. This time the filter size is reduced to 5X5 and we have 256 such filters. The stride is 1 and padding 2. The activation function used is again relu. Now the output size we get is 27X27X256.

Again we applied a max-pooling layer of size 3X3 with stride 2. The resulting feature map is of shape 13X13X256.

Now we apply the third convolution operation with 384 filters of size 3X3 stride 1 and also padding 1. Again the activation function used is relu. The output feature map is of shape 13X13X384.

Then we have the fourth convolution operation with 384 filters of size 3X3. The stride along with the padding is 1. On top of that activation function used is relu. Now the output size remains unchanged i.e 13X13X384.After this, we have the final convolution layer of size 3X3 with 256 such filters. The stride and padding are set to one also the activation function is relu. The resulting feature map is of shape 13X13X256.So if we look at the architecture till now, the number of filters is increasing as we are going deeper. Hence it is extracting more features as we move deeper into the architecture. Also, the filter size is reducing, which means the initial filter was larger and as we go ahead the filter size is decreasing, resulting in a decrease in the feature map shape.

Next, we apply the third max-pooling layer of size 3X3 and stride 2. Resulting in the feature map of the shape 6X6X256.

Fully Connected and Dropout Layers :

After this, we have our first dropout layer. The drop-out rate is set to be 0.5.

Then we have the first fully connected layer with a relu activation function. The size of the output is 4096. Next comes another dropout layer with the dropout rate fixed at 0.5. This followed by a second fully connected layer with 4096 neurons and relu activation.Finally, we have the last fully connected layer or output layer with 1000 neurons as we have 10000 classes in the data set. The activation function used at this layer is Softmax.

This is the architecture of the Alexnet model. It has a total of 62.3 million learnable parameters.



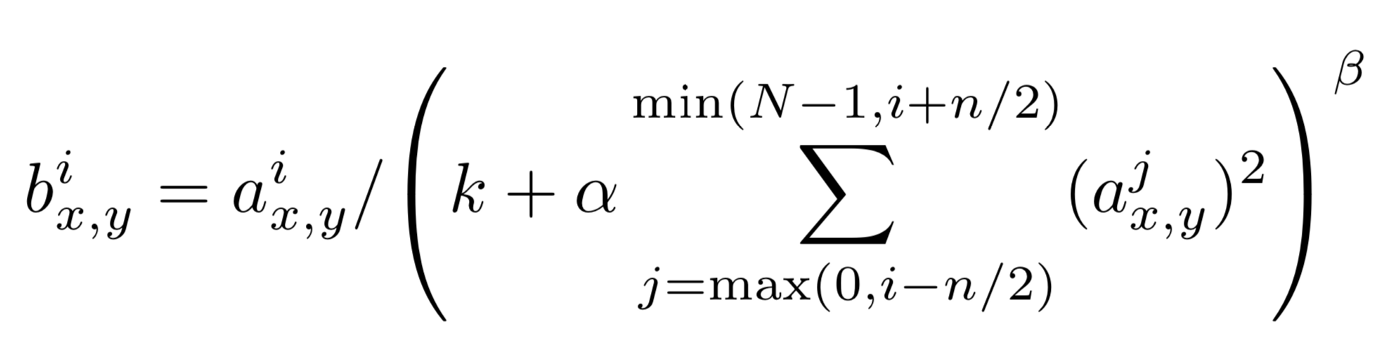
5)describe the vanishing gradient problem.

Ans : when we are taking the derivative and derivative of the sigmoid is always below 0.25 and hence when we multiply a lot of derivatives together according to the chain rule, we end up with a vanishing value such that we cant use them for error calculation.

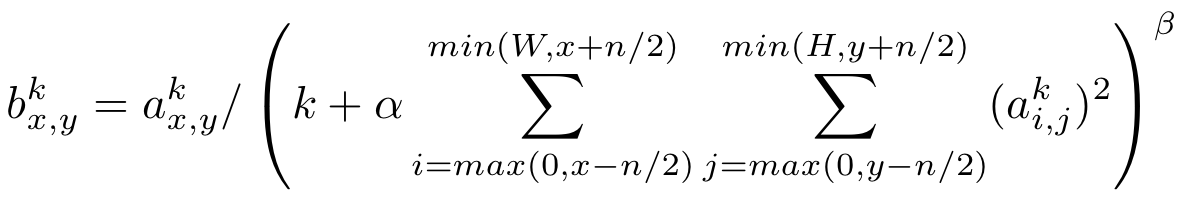
6)what is Normalization of local response

.

Ans : Local Response Normalization (LRN) was first introduced in AlexNet architecture where the activation function used was ReLU as opposed to the more common tanh and sigmoid at that time. Apart from the reason mentioned above, the reason for using LRN was to encourage lateral inhibition.In DNNs, the purpose of this lateral inhibition is to carry out local contrast enhancement so that locally maximum pixel values are used as excitation for the next layers.LRN is a non-trainable layer that square-normalizes the pixel values in a feature map within a local neighborhood. There are two types of LRN based on the neighborhood.

Inter-Channel LRN: This is originally what the AlexNet paper used. The neighborhood defined is across the channel. For each (x,y) position, the normalization is carried out in the depth dimension and is given by the following formula. 

Intra-Channel LRN: In Intra-channel LRN, the neighborhood is extended within the same channel only as can be seen in the figure above. The formula is given by



7)In alexnet,what weight regularization was used ?

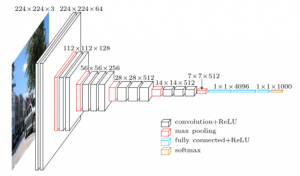
Ans : Weight regularization is a generic approach. It can be used with most, perhaps all, types of neural network models, not least the most common network types of Multilayer Perceptrons, Convolutional Neural Networks, and Long Short-Term Memory Recurrent Neural Networks.Many are familiar with batch normalization, but the AlexNet architecture used a different method of normalization within the network: Local Response Normalization (LRN). LRN is a technique that maximizes the activation of neighbouring neurons.The regularization term Ω is defined as the Euclidean Norm (or L2 norm) of the weight matrices, which is the sum over all squared weight values of a weight matrix. The regularization term is weighted by the scalar alpha divided by two and added to the regular loss function that is chosen for the current task.

8)using our own terms & diagrams,explain vggnet Architecture.

Ans : VGG Stand for visual geometry group.The VGG architecture consists of blocks, where each block is composed of 2D Convolution and Max Pooling layers.

VGG is a classical convolutional neural network architecture. It was based on an analysis of how to increase the depth of such networks. The network utilizes small 3 x 3 filters. Otherwise the network is characterized by its simplicity: the only other components being pooling layers and a fully connected layer.

VGG Architecture :The input to VGG based convNet is a 224\*224 RGB image. Preprocessing layer takes the RGB image with pixel values in the range of 0–255 and subtracts the mean image values which is calculated over the entire ImageNet training set.



The input images after preprocessing are passed through these weight layers. The training images are passed through a stack of convolution layers. There are total of 13 convolutional layers and 3 fully connected layers in VGG16 architecture. VGG has smaller filters (3\*3) with more depth instead of having large filters. It has ended up having the same effective receptive field as if you only have one 7 x 7 convolutional layers.Another variation of VGGNet has 19 weight layers consisting of 16 convolutional layers with 3 fully connected layers and same 5 pooling layers. In both variation of VGGNet there consists of two Fully Connected layers with 4096 channels each which is followed by another fully connected layer with 1000 channels to predict 1000 labels. Last fully connected layer uses softmax layer for classification purpose.

The first two layers are convolutional layers with 3\*3 filters, and first two layers use 64 filters that results in 224\*224\*64 volume as same convolutions are used. The filters are always 3\*3 with stride of 1

After this, pooling layer was used with max-pool of 2\*2 size and stride 2 which reduces height and width of a volume from 224\*224\*64 to 112\*112\*64.

This is followed by 2 more convolution layers with 128 filters. This results in the new dimension of 112\*112\*128.

After pooling layer is used, volume is reduced to 56\*56\*128.

Two more convolution layers are added with 256 filters each followed by down sampling layer that reduces the size to 28\*28\*256.

Two more stack each with 3 convolution layer is separated by a max-pool layer.

After the final pooling layer, 7\*7\*512 volume is flattened into Fully Connected (FC) layer with 4096 channels and softmax output of 1000 classes.

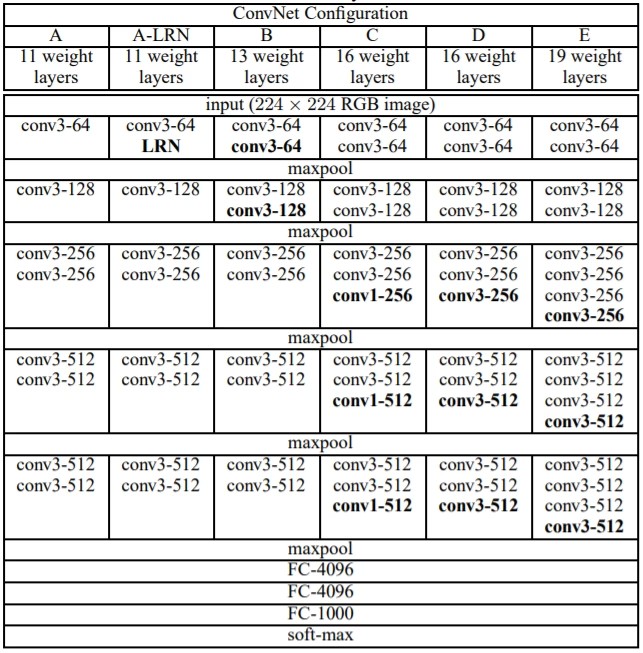
9)describe vggnet configurations.

Ans : vggnet configurations A stack of multiple (usually 1, 2, or 3) convolution layers of filter size 3 x 3, stride one, and padding 1, followed by a max-pooling layer of size 2 x 2, is the basic building block for all of these configurations. Different configurations of this stack were repeated in the network configurations to achieve different depths. The number associated with each of the configurations is the number of layers with weight parameters in them.

The convolution stacks are followed by three fully connected layers, two with size 4,096 and the last one with size 1,000. The last one is the output layer with Softmax activation. The size of 1,000 refers to the total number of possible classes in ImageNet.

VGG16 refers to the configuration “D” in the table listed below. The configuration “C” also has 16 weight layers. However, it uses a 1 x 1 filter as the last convolution layer in stacks 3, 4, and 5. This layer was used to increase the non-linearity of the decision functions without affecting the receptive field of the layer.

The left-most “A” configuration is called VGG11, as it has 11 layers with weights – primarily the convolution layers and fully connected layers. As we go right from left, more and more convolutional layers are added, making them deeper and deeper.



10)what regularization methods are used in vggnet to prevent overfitting ?

Ans : One of the ways is to apply Regularization to the model. Regularization is a better technique than Reducing the number of features to overcome the overfitting problem as in Regularization we do not discard the features of the model. Regularization is a technique that penalizes the coefficient.

One of the most powerful features to avoid/prevent overfitting is cross-validation. The idea behind this is to use the initial training data to generate mini train-test-splits, and then use these splits to tune your model. In a standard k-fold validation, the data is partitioned into k-subsets also known as folds.Regularization comes into play and shrinks the learned estimates towards zero. In other words, it tunes the loss function by adding a penalty term, that prevents excessive fluctuation of the coefficients. Thereby, reducing the chances of overfitting.