We at The Data Monk hold the vision to make sure everyone in the IT industry has an equal stand to work in an open domain such as analytics. Analytics is one domain where there is no formal under-graduation degree and which is achievable to anyone and everyone in the World.

We are a team of 30+ mentors who have worked in various product-based companies in India and abroad, and we have come up with this idea to provide study materials directed to help you crack any analytics interview.

Every one of us has been interviewing for at least the last 6 to 8 years for different positions like Data Scientist, Data Analysts, Business Analysts, Product Analysts, Data Engineers, and other senior roles. We understand the gap between having good knowledge and converting an interview to a top product-based company.

Rest assured that if you follow our different mediums like our blog cum questions-answer portal www.TheDataMonk.com , our youtube channel - The Data Monk, and our e-books, then you will have a very strong candidature in whichever interview you participate in.

There are many blogs that provide free study materials or questions on different analytical tools and technologies, but we concentrate mostly on the questions which are asked in an interview. We have a set of 100+ books which are available both on Amazon and on The Data Monk e-shop page

We would recommend you to explore our website, youtube channel, and e-books to understand the type of questions covered in our articles. We went for the question-answer approach both on our website as well as our e-books just because we feel that the best way to go from beginner to advance level is by practicing a lot of questions on the topic.

We have launched a series of 50 e-books on our website on all the popular as well as niche topics. Our range of material ranges from SQL, Python, and Machine Learning algorithms to ANN, CNN, PCA, etc.

We are constantly working on our product and will keep on updating it. It is very necessary to go through all the questions present in this book.

Give a rating to the book on Amazon, do provide your feedback and if you want to help us grow then please subscribe to our Youtube channel.

CONVOLUTION NEURAL NETWORKS

Q1. What is the idea behind neural network?

A1. The idea of neural network in machine learning is taken from our brain cell. The basic idea behind it is to copy our brain cell .To act like our brain cells and solve problems.

Q2. What is the basic concept of Neural Network?

A2. In neural network the basic process is forward propagation than calculating the loss function and backward propagation.

Q3. What is forward propagation?

A3. It starts with image given as an input. That image is a matrix of pixels. So we pass them to Conv2D layer where filter is applied to it , we also apply a padding around the input matrix. When we have the output after Conv2D layer, then we pass it pooling layer where try to reduce the dimensions of the matrix such as we get all the important features regarding the matrix and remove the noise. Then we will flatten the layer such that there is only one column. After this we will fully connect the network and output is produces. (Remember: at each step we apply an activation function link artificial neural network).

Q4. What is activation function?

A4. Activation functions in neural networks are used to make the input (y) in the range according to the type of activation function applied. For ex. If we apply sigmoid the range is 0 to 1. There are different types of activation function. There are three layers in the neural network: input layer, hidden layer, output layer. Mostly, same activation function is applied to every hidden layer. But in output layer a different activation function is applied and it is selected on the basis of what type of prediction is required by the model. For example, if someone puts a hot object on your hand. So that neurons will get activated and send the signals to the brain to react on that.

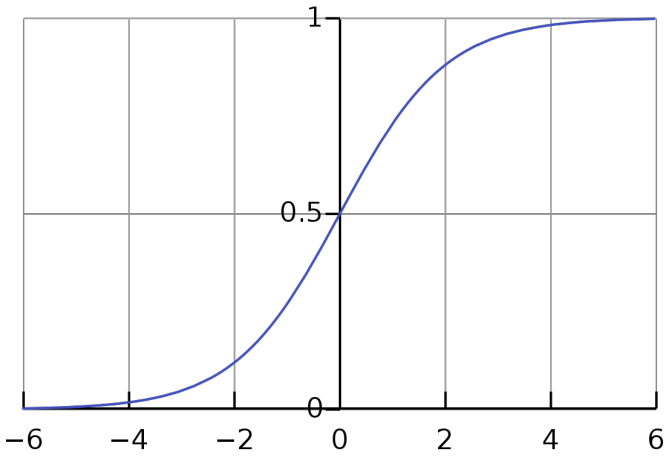
Q5. Different types of activation functions?

A5. There are many types of activation functions:

- Sigmoid activation function
- ReLU activation function
- Leaky ReLU activation function
- PReLU activation function
- ELU activation function
- Softmax activation function
- Tanh activation function

Q6. What is a Sigmoid activation function?

A6. You have studied about this activation function earlier in logistic regression algorithm. Sigmoid is basically an “S” liked curve, in which we set a threshold value. So if the input value is greater than threshold value than only the activation function will get activated otherwise it will get deactivated.



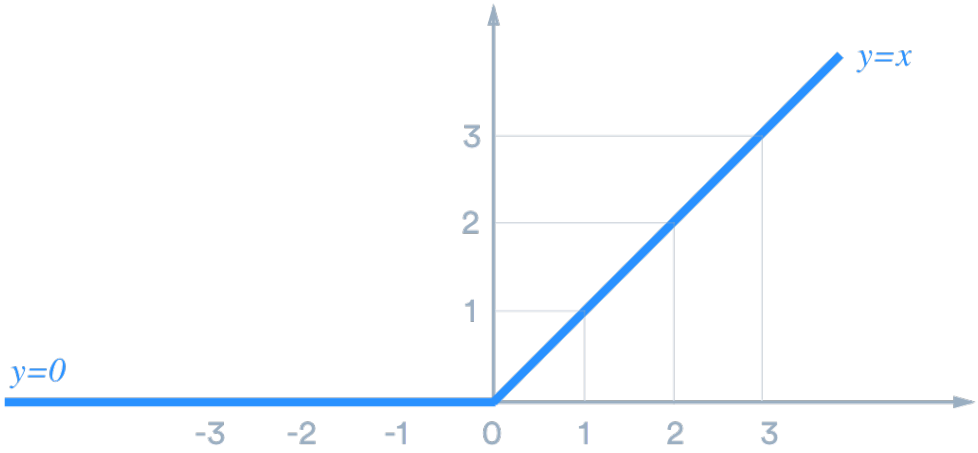
It is represented by the formula:

Sigmoid = 1 / (1 + e^-z)

Sigmoid activation function is basically used in the output layer.

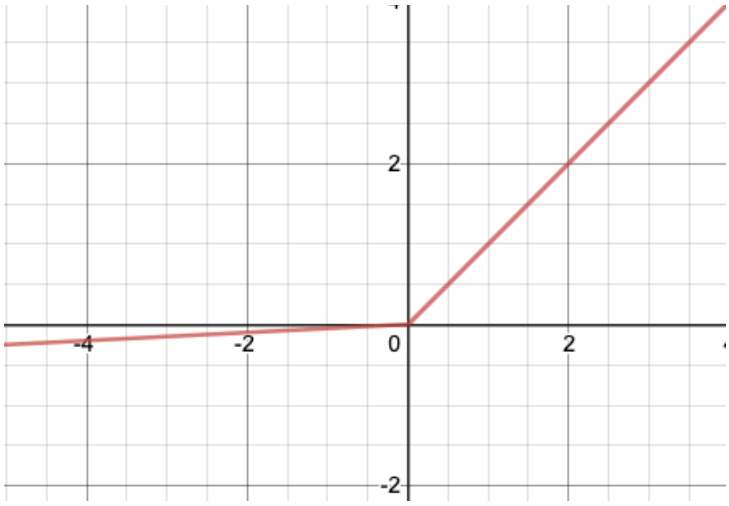
Q7. What is a ReLU activation function?

A7. ReLU stands for Rectified Linear Unit. ReLU activation function works on the formula of “max(z,0)”.We take the max number out of the two. If the input value is smaller than 0 than output will be “0”. If the input value is greater than “0” than the output will be considered as that value.



Q8.What is Leaky ReLU activation function?

A8. There is a limitation in the ReLU activation function. The limitation is that when we calculate the derivative of the ReLU activation function than the derivative of values less than “0” is “0”.Now suppose we have a large weight and we are calculating the derivative to adjust the weights through chain rule and one of the derivative comes out to be “0” in the chain rule. Than the whole derivative will become “0” and it will considered as a dead neuron. So to overcome this problem we use Leaky ReLU activation function. Leaky ReLu adds a small value to the derivative for value less than “0”, the value will be small like “0.01”. Due to which it will not let the derivative value be “0”.



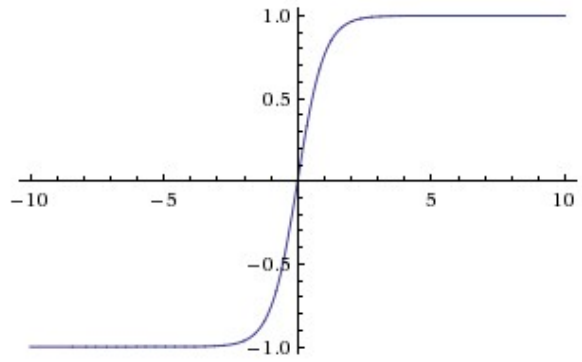
Q9. What is softmax activation function?

A9. Softmax activation function is mostly used in the time when there is multiple classification problem. It is represented by the formula given below. It is mostly used in the output layer.

$$\phi(z) = \frac{e^i}{\sum_{j=0}^{k} e^j} \quad \text{where } i=0,1,\ldots.k$$

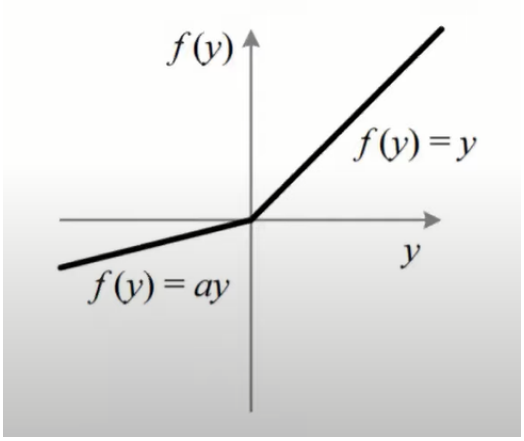
Q10. What is tanh activation function?

A10. Tanh activation function is similar to the sigmoid activation function. The only difference is that tanh is a zero-centric activation function. Due to which it is better than sigmoid activation function.



Q11. What is PReLu activation function?

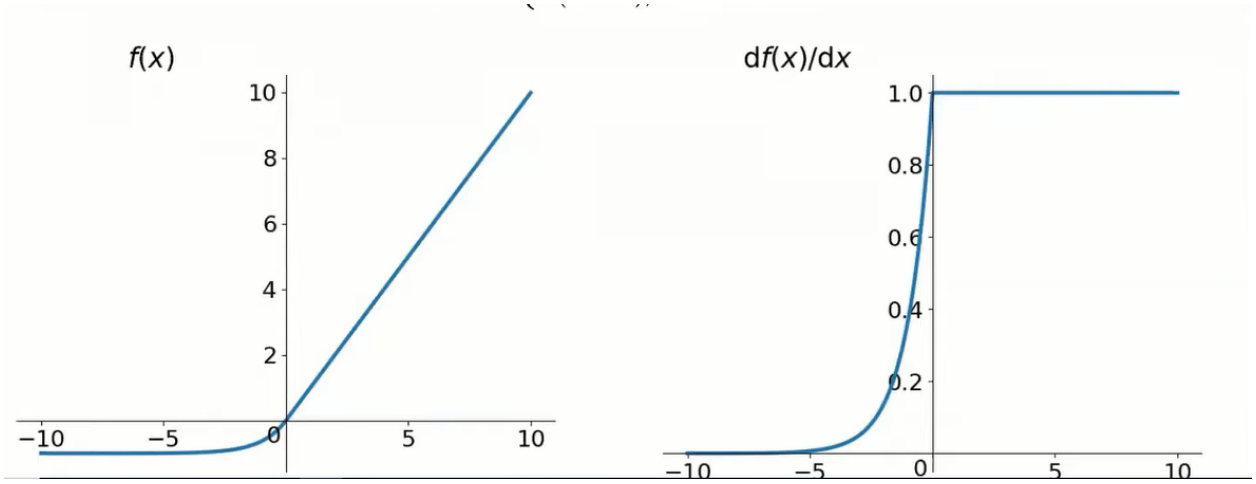
A11. PReLU is basically parametric ReLU activation function. It is kind of an improved version of ReLU. Basically, it has a small slope in the negative region which prevents the problem of dead neuron.



$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

Q12. What is ELU activation function?

A12. ELU stands for Exponential Linear Unit. It is an updated kind of ReLU . In this we add a hyper parameter to the values less than “0”, whose derivative comes out to be “0”. But after applying hyper parameter to it than the derivative will not be “0”. But it has a disadvantage that it is computationally expensive.



$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$

Q13. What would happen without activation function?

A13. If we do not apply any activation function to the inputs then the value of (y) will be a large value and it will be becoming a larger value after every layer. It will not be easy to calculate that values, it will be requiring very high computational power. So, we use an activation function to sum down the values.

Q14. What is loss function?

A14. When we reach the output layer with a predicted output, now we need to compare it with the actual output. So here we use the loss function we calculate the difference between the actual output and predicted value. If the difference is large then we will adjust the weights and try to reduce it as much as possible.

Q15. What are different types of loss functions?

A15. There are different types of loss functions to handle different types of variables

- Regression
- Single class classification
- Multi class classification

Q16. What is the loss function used in regression problem?

A16. Different types of loss function in regression problem are:

- Mean Square Error Loss :

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Where yi is actual output and yi^ is predicted output.

Advantages:

1. The Mean Squared Error penalizes the model for making large errors by squaring them.
2. When we solve the above equation we will get a quadratic equation:
   (i). When we plot the graph for quadratic equation , we will get a Gradient Descent with only 1 global minima.
   (ii). We do not get any local minima

Disadvantages:

1. It is not robust to the outliers. Meaning, whenever we have an outlier it will not function properly and leave errors.

- Mean Absolute Error Loss :

$$\text{MAE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} \underbrace{|y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i|}_{\text{actual value}}$$

When we solve the above equation, we will get a linear equation.

Advantages:

1. It is more robust to outliers as compared to the mean square error loss.

- Huber Loss :

$$Hubber\ Loss = \begin{cases} \frac{1}{2}(y - y_p)^2, & |y - y_p| \le \delta \\ \delta|y - y_p| - \frac{1}{2}\delta^2, & |y - y_p| > \delta \end{cases},$$

where , y is actual output , yp is predicted output and dell symbol is the hyper parameter.

Huber Loss is basically a combination of mean squared error and mean absolute error. As we know if we have outliers than the quadratic equation will not work nicely. So we splitted the cases in Huber loss , if we have outliers in the dataset than we will use linear equation i.e. mean absolute error loss and if we do not have outliers in the dataset than we will use quadratic equation i.e. mean square error loss.

Q17. What is the loss function used in classification problem?

A17. The widely used loss function in the classification problem is Cross Entropy. It is basically makes use of sigmoid function as we studied in logistic regression.

It is represented by the formula:

loss = -y * log(yp) – (1-y) * log(1 – yp)

Where, y is actual output and yp is predicted output.

We calculate the yp by using sigmoid function:

yp = sigmoid = 1/ (1 + e^(-z))

Q18.What is the loss function used in multi-class classification problem?

A18. In multi class classification problem we use Multi Class Cross Entropy Loss .

It is represented by the formula:

$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \hat{y}^{(k)}$$

Where , y^ is predicted output , y is actual output.

So, when we have multiple classes in dependant feature than we will apply one hot encoding to it. So, the value of “k” in the above equation is 0 or 1. “1” if the element is present in that class and “0” if the element is not present in that class.

Q19. How do we classify which kind of classification problem is it?

A19. If the output features are having only two categories then it will be considered as single variable classification problem, because after the implementation of the code we will apply label encoder to the output variable so it will be converted into 0 or 1. Now if we have more than two categories than it is considered as multi-class classification problem. This happens because when we apply label encoder to the output feature then we will get the different values of the different categories.

Q20. What is cost function?

A20. It is synonymous to the loss function. The only difference is that it cost function is basically the average loss for the whole dataset.

Q21. What is Optimizer?

A21. When we calculate the loss, then we need to calculate by how much we are going to adjust the weights or how the weights must be adjusted so that our loss is minimum. Optimizer are algorithm or model used to change the attributes of the neural network such as weights and learning rate in such a way that the loss in minimized as much as possible.

Q22. What are different types of optimizers?

Q22. There are various types of optimizers:

- Gradient Descent
- Stochastic Gradient Descent(sgd)
- Mini-Batch stochastic Gradient Descent
- Stochastic Gradient Descent with momentum
- Adagrad optimizer
- Adadelta and RMS prop optimizer
- Adam optimizer

Q23.What is Gradient Descent optimizer?

A23. It is used to update the weights in the back propagation of neural network. In gradient descent we take all the inputs at a single time to perform the propagations.

It is represented by the formula:

$$Wn = Wo – n * (dL / dW)$$

Where , Wn is new/updated weight , Wo is old weight , n is learning rate.

The value of dL/dW is negative if the slope is negative or downward going and positive if the slope is positive or upward going .

Q24. What is Stochastic Gradient Descent?

A24. It is an upgraded / improved version of gradient descent. In gradient descent, suppose we have a dataset of 1000 data points we will take all the 1000 data points at a single time to compute the derivative which will take very much computational power. But in case of stochastic gradient descent , we take a single data point out of 1000 at the time of propagation.

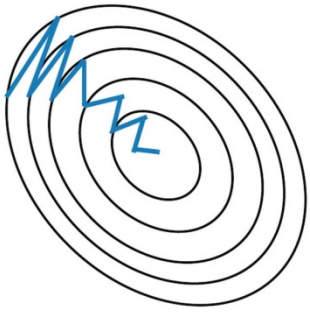
Q25. What is Mini Batch Stochastic Gradient Descent?

A25. It is and upgraded/ improved version of stochastic gradient descent. In stochastic gradient descent, we take only one data point in propagation which will use many resources to get computed and will be slow in computation. So to overcome this issue we introduced Mini Batch Stochastic Gradient Descent, in this we take “k” number of data points in the propagation which will use less resources and computational power.

Q26. What is Stochastic Gradient Descent with Momentum?

A26. It is an updated/improved version of mini batch stochastic gradient descent. When we use mini batch stochastic gradient descent we will have some noisy data. So in order to remove the noisy data we will use momentum. We will assign momentum Y=0.5(exponentially moving average). When we calculate the derivatives with the chain rule , suppose we are calculating for the W3 so with the help of momentum we will give it more importance , less importance to W2 and least importance to W2, due to which we will be able to remove some noise from the dataset.

Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

Q27. What is Adagrad optimizer?

A27. Till now, we have seen that learning rate is fixed in gradient descent , stochastic gradient descent , mini batch stochastic gradient descent. So, idea behind adagrad optimizer is that it changes learning rate for every layer and neuron for every iteration. It has only one problem i.e. when number of iterations are very large than the coefficients's value will become very large . So , learning rate will become very small and it will lead to slow convergence .

Q28. What is Adadelta and RMS prop optimizer?

A28. As we know about the disadvantage of the adagrad optimizer we use Adadelta optimizer.

Wt=W(t-1) - Nt (dL/dW(t-1))

Nt=N/ sqrt(Wavg +E)

Wavg = Y* Wavg(t-1) + (1-Y) (dL/dW(t))^2

Now , also the value of Wavg will become large , but it has Y to restrict it . So, the value of Nt will be decreasing very slowly.

Q29. What is Adam optimizer?

A29. It is the most widely used and the best till date optimizer as it has the property of both the stochastic gradient descent with momentum and changing learning rate efficiently as in RMS prop.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

The first equation is with the effect of momentum and the second equation is for the change in learning rate efficiently.

So , our final formula for the adam optimizer is given below:

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Q30. What is backward propagation?

A30. After the forward propagation, we received a predicted output. Now we will compare the predicted output with the actual output .If the difference between the predicted output and actual output is a large value than we will try to minimize. To minimize the difference we use optimizers, the weights are now adjusted in such a manner that loss in minimum.

Q31. How are weights adjusted in backward propagation?

A31. The weights are adjusted with the help of optimizer function. We compare the predicted output with the actual output. We get a difference and according to this difference we adjust the weights such that this difference becomes minimum.

Q32.What is batch size?

A31. Batch size basically means that how much input values are taken in a single propagation. For ex. Before lockdown we go to any place in a bulk but after lockdown we stand in a queue and we go in small batches.

Q33. What is the library used to perform neural networks?

A33. Libraries used to perform neural network:

- Tensorflow
- Sklearn
- keras

Q34.What is epochs?

A34. When we are training the neural network for one repetition with the training data .One propagation means forward propagation + backward propagation.

Q35. How many layers are there in neural networks?

A35. There are basically three types of layers:

-Input layer

-Hidden layer

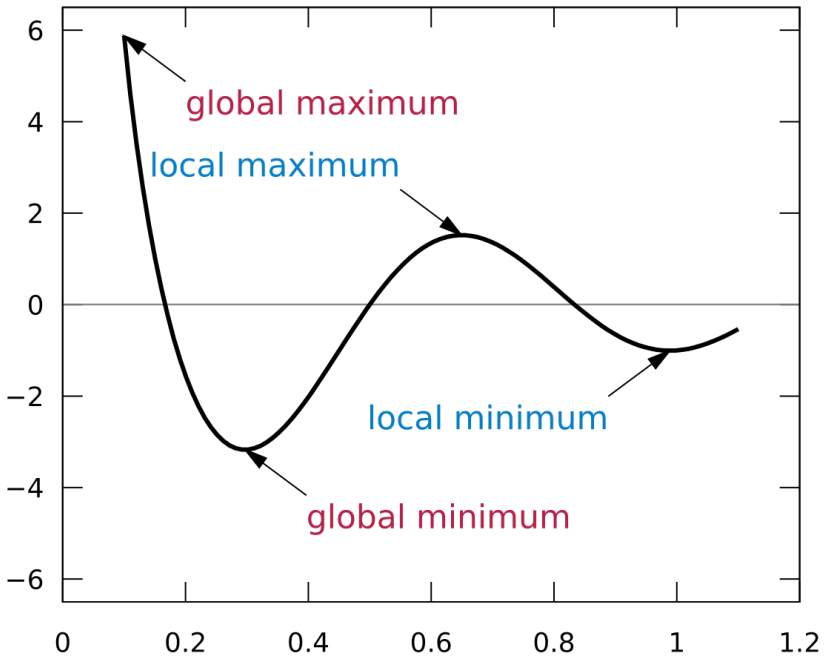
-Output layer

Q36. Which is the best optimizer?

A36. The best and most efficient optimizer till date is “adam” optimizer .As it got the smoothness from the momentum of stochastic gradient descent and efficient changing of learning rate from the RMS prop.

Q37. What are local minima and maxima?

A37. When we reach global minima than the slope becomes “0” and it is the lowest point in the curve, but in the case of local minima, we reach a point where slope becomes “0” but it is not the lowest point in the curve. If we move forward we will reach a point which will be the lowest point in the curve (global minima).



When we reach global maxima our slope becomes “0” and it is the highest point in the curve. But in the case of local maxima we reach appoint where slope is “0” but it is not the highest point in the curve. If we move forward in the curve we will reach a higher point tan it (global maxima).

Q38. What will happen if the learning rate is set too high?

A38. If we set the learning rate of the neural network too high then it will never reach the global minima point. Because at the weight adjustment time if the learning rate is set too high then it will make the new adjusted weights a high value. (You can think of it as a hyper active scenario).

Q39. What will happen if the learning rate is set too low?

A39. If we set the learning rate of the neural network too small then it will take much more time than the usual time to reach the global minima point. Because if we set the learning rate too small then the new adjusted will the a bit different

from the old value and it will take much more time to it to reach the global minima.

Q40. When does over fitting and under fitting happens in neural network?

A40. When we have a deep neural network i.e. neural network with many layers than the problem of over fitting arises. Because neural networks repeats the propagations until it gets the same predicted output as the actual output.

When we have a neural network with only one hidden layer than only the under fitting can happen.

Q41. How are weights initialized in neural network?

A41. When we initialize weights, weights should not be initialized very small , weights should not be same for every neuron , weights must haver a good variance. There are different types of weight initialization technique:

- Uniform distribution -> weight = Uniform [-1/ sqrt(ni) , 1/ sqrt(ni)]
- Xavier/ Gorat Normal -> weight = normal (0 , sqrt(2/ ni +no)]
- Xavier Uniform -> weight = uniform [-sqrt(6) / sqrt(ni+no) , sqrt(6) / sqrt(ni+ no)]
- He Uniform -> weight = uniform [-sqrt(6 / ni) , sqrt(6 / ni)]
- He Normal -> weight = normal (0 , sqrt(2 / ni))

Where, ni = number of incoming weights at a neuron .

no = number of outgoing weights from a neuron.

Q42. What will happen if we initialize all neurons with same weight?

A42. If we initialize all the neurons with the same weight than we will be sending the same inputs to all the neurons in the hidden layers and the same info will be

passed on and the neural network will propagate with the same values again and again. This type of neural network will be useless.

Q43. What are the common data structures used in neural network?

A43. Common data structures used in neural network are:

- Linked List
- Binary Search Tree or Binary Tree
- Heap
- Set
- Graphs
- Hashing

Q44. What are advantages of using neural network?

A44. In neural network the information gets stored in the whole network. It learns from itself and does not require extra input data. They are well known for performing multiple tasks in parallel without affecting the performance of the neural network.

Q45. What are disadvantages of neural network?

A45. When we have a deep neural network i.e. a neural network with many layers, we also have many weights and bias values. So, our cnn model in that case tries to overfit.

Q46. What are prerequisite for learning neural network?

A46. To learn and understand how a neural network works you first need to cover the topics listed below:

- Calculus - As there are various formulas and graphs which you need to understand in order to implement neural network.
- Logistic Regression – As most of the neural networks are simultaneously working logistic regressions.

- Coding – Your coding area must be clear in order to implement neural network.

Q47. Most used activation Function?

A47. The most used activation function is the leaky Relu function and softmax function. As they deal with mostly all the limitations or disadvantages in the other activation function.

Q48. Difference between artificial intelligence, machine learning, neural network?

A48. Artificial intelligence-It was originated in 1940s. It is basically a process which enables machine to behave like a human being.

Machine learning - It was originated in 1950s. It is basically a study where we use statistical data to make machine work better as it gains experience. It is a part of artificial intelligence.

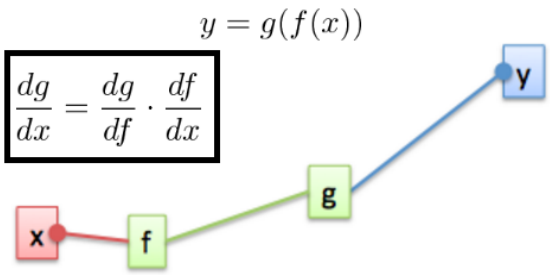
Neural networks -It was originated in 1960s. It is basically imitates the neural network of a human being. It is a part of machine learning.

Q49. What is Vanishing Gradient Problem?

A49. This problem basically came forward in the early time of the neural network and is basically due to the sigmoid activation function. As in earlier times, there was only sigmoid activation function discovered. So, what basically happened is sigmoid converts all the values from 0 to 1. But the derivative value of the sigmoid function ranges between “0” to “0.25”. As the layers will be increasing, the value of derivative gets on getting smaller. And a point will come when the new weight will be almost equal to the old weight.

Q50. How chain rule helps in backward propagation?

A50. You can think of a chain rule as a system of traders. From industry to the market.



Q51. Why do we use dropout in neural network?

A51. When we have a deep neural network i.e. neural network with many layers. We will also be having many weights and bias values and due to which our ann model tries to overfit. So, in order to remove/avoid overfitting we use the techniques like dropout.

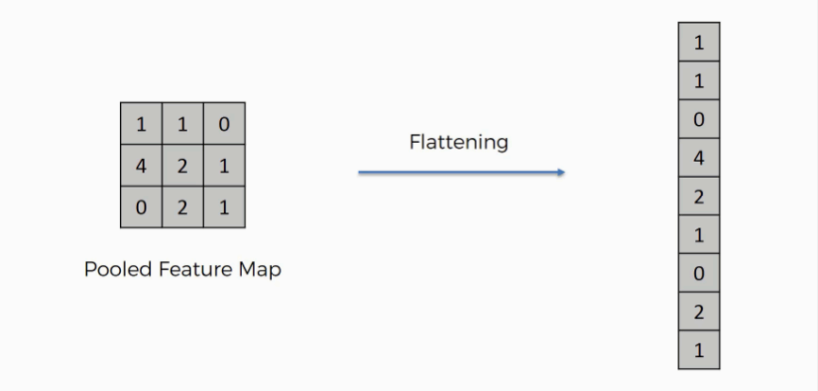
Q52.What is Dropout in neural network?

A52. So when we have a deep neural network, we assign every layer a dropout value. A dropout value is like a probability of neurons we are choosing. It lies between 0 and 1. So we assign a dropout value for each and every layer in the neural network. When we assign the dropout value than that much neurons will be deactivated and will not be considered in neural network in that very propagation. It chooses random neurons from every layer. It completes its one propagation with the activated neurons and the weights are adjusted. Now, in next propagation it will randomly choose neurons.

(Suppose our first layer has 10 neurons and we chose a dropout value (P) as “0.3”. So, 3 neurons randomly will be deactivated.)

Q53. What is Flattening in convolution neural network?

A53. When we reach near to the final step , we have out input after max pooling. So we will flatten that matrix as a single column matrix. For ex, if we have a matrix of 2x2 we will flatten it. After flattening it will we have output matrix as 4x1.



Q54. What is Max Pooling in convolution neural network?

A54. When the input matrix passes through the Conv2D layer after applying the filter to it. We try to reduce the size of the input matrix by keeping all the important features and removing the noise from the matrix. We do it by choosing small matrix of maybe 2x2, we iterate over the matrix with this matrix and pick all the important features from it and remove all the noise. So, by doing this we will reduce the computation power and less resource will be used. In max pooling we pick the important features by taking max value from the small matrix we have created. To understand this scenario, let's say you se a person and sometimes by seeing their eyes you are able to detect who he is

Q55. What is padding in cnn?

A55. When we pass the input image matrix through a filter the size of the input matrix gets reduced due to which there is loss in the input data. So we add a padding layer around the input matrix. So that after the matrix is passed through the filter the size of the input matrix remains the same and there will be no data loss.

Formula before adding the padding:

- N-f+1

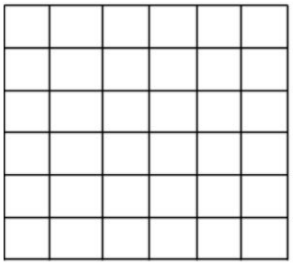
Formula after adding the padding:

- N+2p-f+1

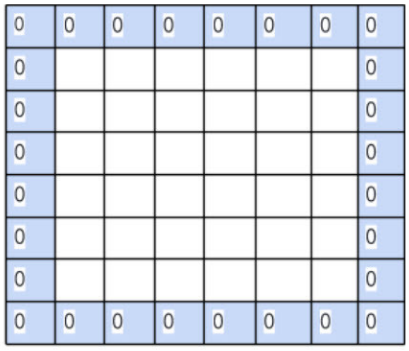
Where, N is the size of input matrix.

F, is the size of filter

P, is the size of padding



6x6 image



6x6 image with 1 layer of zero padding

Q56. What is the size of the pixels?

A56. The size of the pixel for and colored image is “255”. As these bits are binary so and we have 8 for every byte. So, (2^8 = “256”) but we consider “0” as default so it becomes “255”. In cnn, “0” is pitch black and “255” is bright white and all the other shades are in between “0” to “255”.

Q57. Why are pixel normalized ?

A57. Normalization ensures that each and every pixel has a similar data distribution. It is done by subtracting mean from every pixel and divide that by standard deviation.

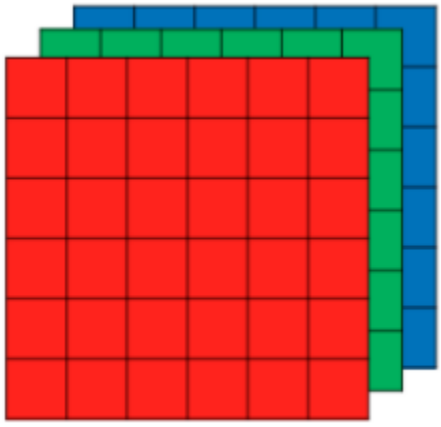
Q58. What is the difference between gray scale images and rgb images?

A58. Each pixel contains 1 byte (8 bits)

In grayscale images there is only one channel.

-1	0	1
-2	0	2
-1	0	1

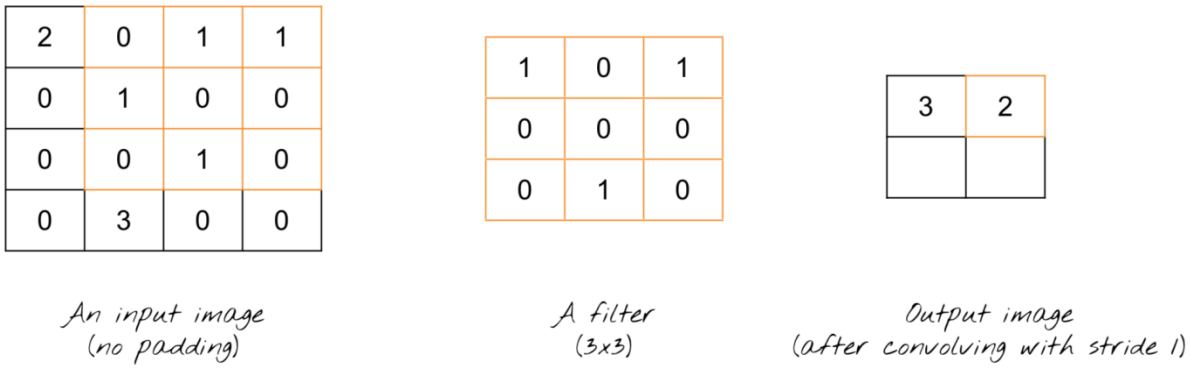
But, in rgb images, there are three channels.



Q59. What is data augmentation?

A59. Data augmentation is viewing the image in as many ways as possible. For example, you have an image of a cat. So we will try to view it by zooming in, horizontally flipping it and other ways and output will remain same. By doing data augmentation we can increase the data, model can become robust and there will be invariance in the images.

Q60. What is Conv2D layer?



Q61. What are different filters?

A61. There are different types of filters in convolution neural network. There are layers like edge detection, face detection , eyes detection and many more.

Q62. How filter helps in convolution neural network ?

A62. Filters in convolution neural network are used to detect different features in the images and get a better understanding of the image. Suppose a filter is used to detect the edges in the images. Like the edge detection there are different kind of filters present in keras Conv2D layer. When we write the code we have parameter as “filter” , where we provide number of filters.

Q63. What are different types of pooling techniques?

A63. It is a kind of down sampling technique used to reduce the height and width of the input. There are mainly two types of pooling layer:

- MAX POOLING- In the output matrix from Conv2D we take the max pool pixels from every small matrix stride by stride.
- AVERAGE POOLING- In the output matrix from Conv2D we take the average of pixels from every small matrix stride by stride.

Pooling helps us to take the important information and reduce the noise in the data.

Q64. What is a fully connected layer?

A64. It means that every layer of "n" layer will be connected to all the nodes present in the "n+1" layer. The probabilities of different class are outputted in a 3D array.

Code for initializing neural network:

About the data: We have a file named "Dog vs Cat" . This dataset contains different images of cats and dogs in different scenarios and situations possible. So we have to train out model such that we are able to identify whether it is a dog or a cat.

Code:

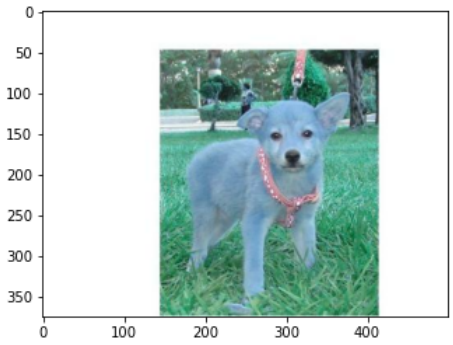
#Importing Libraries

```
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LeakyReLU,PReLU,ELU
from keras.layers import Dropout
```

 # Loading Some Images

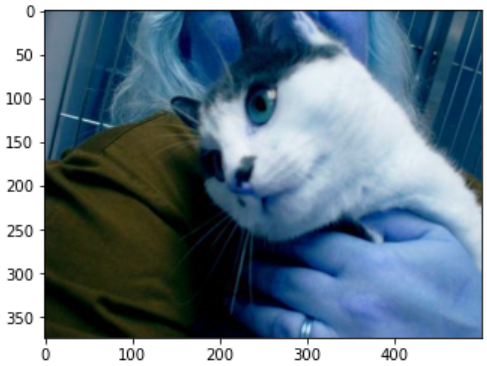
```
plt.imshow(cv2.imread("/kaggle/input/cat-and
dog/training_set/training_set/dogs/dog.1599.jpg"))
```

```
<matplotlib.image.AxesImage at 0x7f9239080a90>
```



plt.imshow(cv2.imread("/kaggle/input/cat-and-dog/training_set/training_set/cats/cat.2245.jpg"))

```
<matplotlib.image.AxesImage at 0x7f9238ff7a10>
```



 # Part 1 - Data Preprocessing

 # Preprocessing the Training set

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

```
Found 8005 images belonging to 2 classes.
```

 # Part 2 - Building the CNN

```
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 2023 images belonging to 2 classes.
```

 # Initialising the CNN

```
cnn = tf.keras.models.Sequential()
```

 # Step-1 Convolution

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu',
input_shape=[64, 64, 3]))
```

#Step-2 Pooling

```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

 # Adding a second Convolution Layer

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

#Step-3 Flattening

```
cnn.add(tf.keras.layers.Flatten())
```

#Step-4 Full Connection

```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

 # Step-5 Output Layer

```
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

 # Part-3 Training the CNN Model

Compiling the CNN

```
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])
```

 # Training the CNN on the Training set and evaluating it on the Test set

```
cnn.fit(x = training_set, validation_data = test_set, epochs = 25)
```

```
Epoch 19/25
251/251 [==============================] - 35s 139ms/step - loss: 0.3656 - accuracy: 0.8367
- val_loss: 0.4465 - val_accuracy: 0.7939
Epoch 20/25
251/251 [==============================] - 36s 142ms/step - loss: 0.3553 - accuracy: 0.8377
- val_loss: 0.4290 - val_accuracy: 0.7993
Epoch 21/25
251/251 [==============================] - 36s 143ms/step - loss: 0.3386 - accuracy: 0.8509
- val_loss: 0.4418 - val_accuracy: 0.8008
Epoch 22/25
251/251 [==============================] - 35s 140ms/step - loss: 0.3406 - accuracy: 0.8506
- val_loss: 0.4596 - val_accuracy: 0.7963
Epoch 23/25
251/251 [==============================] - 36s 141ms/step - loss: 0.3329 - accuracy: 0.8562
- val_loss: 0.4765 - val_accuracy: 0.8003
Epoch 24/25
251/251 [==============================] - 35s 140ms/step - loss: 0.3366 - accuracy: 0.8561
- val_loss: 0.4573 - val_accuracy: 0.8057
Epoch 25/25
251/251 [==============================] - 36s 143ms/step - loss: 0.3204 - accuracy: 0.8555
- val_loss: 0.4556 - val_accuracy: 0.8018

<tensorflow.python.keras.callbacks.History at 0x7f923825d410>
```

 # Part-4 Making a Single Prediction

```
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('dataset/single_prediction/cat_or_dog_1.jpg',
target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
  prediction = 'dog'
else:
  prediction = 'cat'
print(prediction)
```

```
dog
```

You can also check the above code from my github.

Now suppose you a problem where you want to detect different types of images but you have more than two types of images.

THIS IS CALLED AS TRANSFER LEARNING WITH KERAS.

For this a competition named as "imagenet" is held where different teams participate to solve the multi-class classification of images. So there are many techniques came up till now like:

- Resnet
- VGG16
- VGG19
- AlexNet

Now let us take an example of different of images of mountain, glaciers , buildings and other.

Now let us code the problem out.

#IMPORTING LIBRARIES

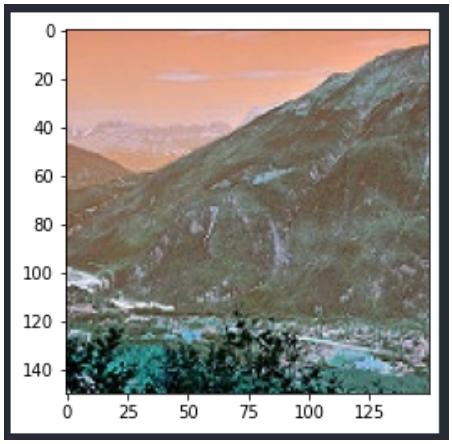
```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
```

```
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
import cv2
```

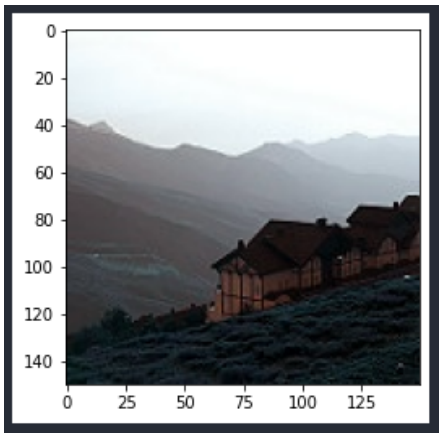
#IMPORTING DATASET

#DISPLAYING THE DIFFERENT TYPES OF IMAGES

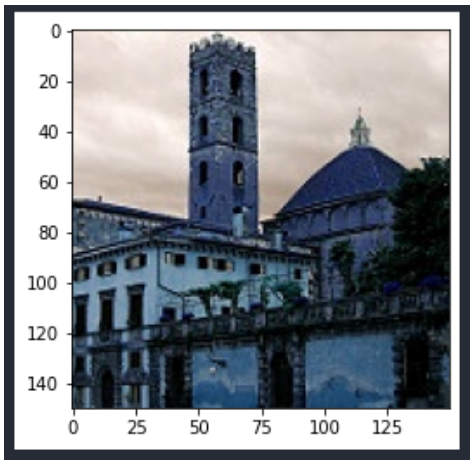
```
plt.imshow(cv2.imread("/kaggle/input/intel-image-classification/seg_test/seg_test/glacier/23154.jpg"))
```



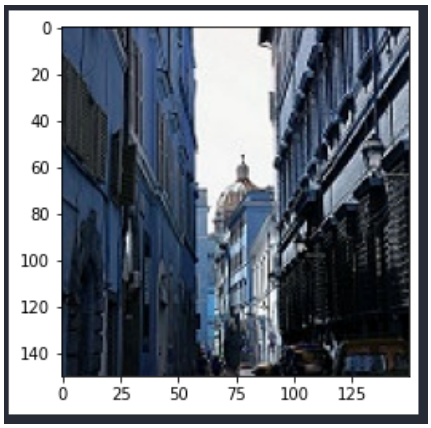

```
plt.imshow(cv2.imread("../input/intel-image-classification/seg_test/seg_test/mountain/20120.jpg"))
```



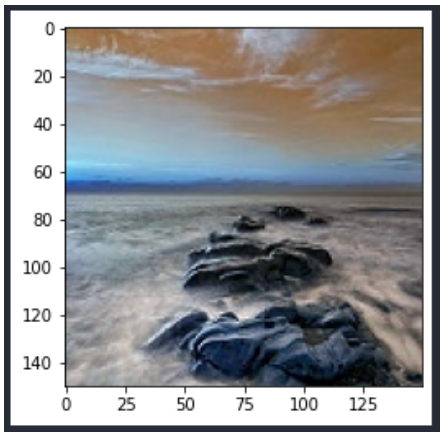
```
plt.imshow(cv2.imread("../input/intel-image-classification/seg_test/seg_test/buildings/20061.jpg"))
```



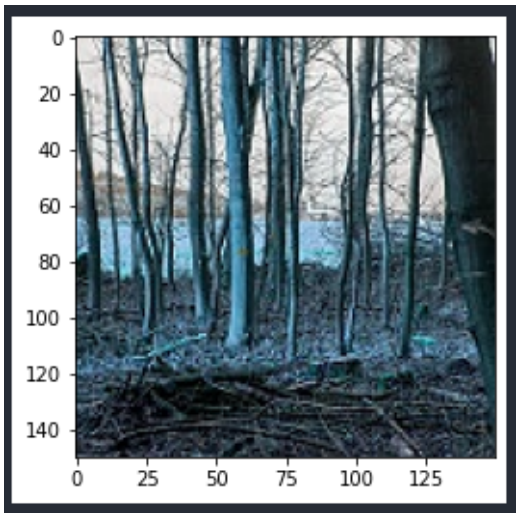

```
plt.imshow(cv2.imread("../input/intel-image-classification/seg_test/seg_test/street/20070.jpg"))
```




```
plt.imshow(cv2.imread("../input/intel-image-classification/seg_test/seg_test/sea/20099.jpg"))
```



plt.imshow(cv2.imread("../input/intel-image-classification/seg_test/seg_test/forest/20117.jpg"))



#RE – SCALING THE IMAGES

IMAGE_SIZE = [224,224]

#SETTING TRAIN AND TEST PATH

train_path = '/kaggle/input/intel-image-classification/seg_train/seg_train'
valid_path = '/kaggle/input/intel-image-classification/seg_test/seg_test'

#ADD PREPROCESSIN LAYER TO FRONT OF VGG

```
vgg=VGG16(input_shape=IMAGE_SIZE + [3] , weights="imagenet" ,
include_top=False)
```

#DON'T TRAIN EXISTING WEIGHTS

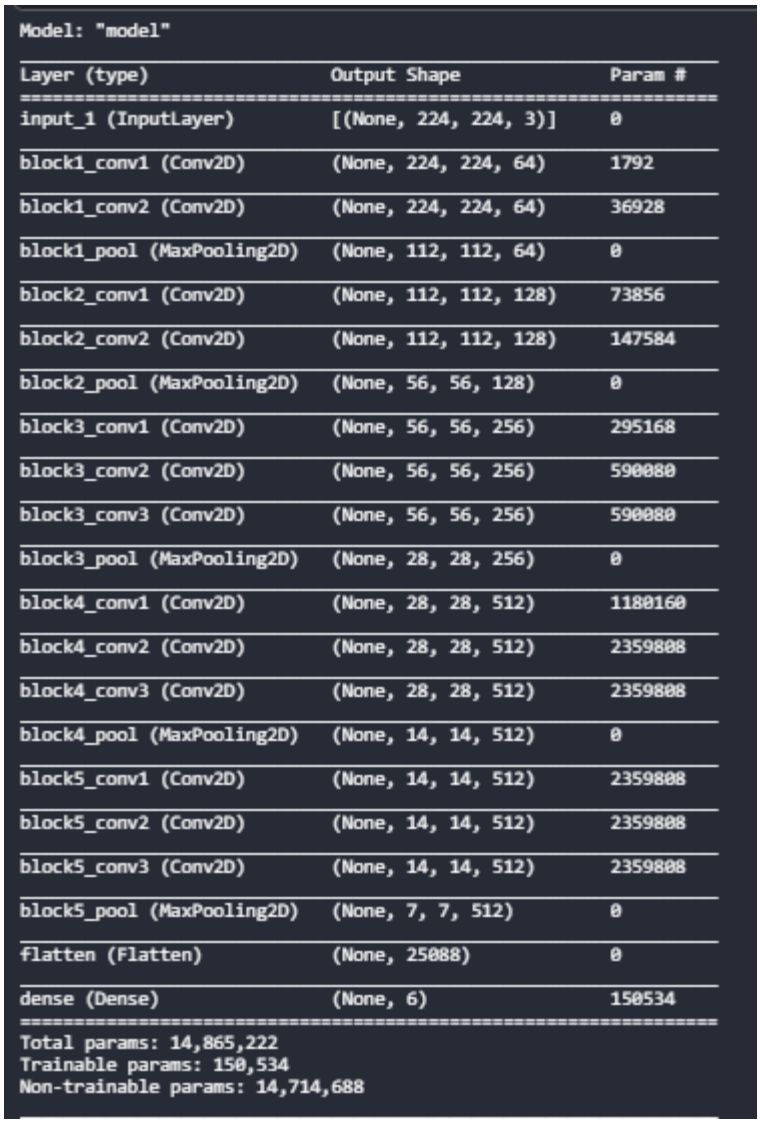
```
for layer in vgg.layers:
   layer.trainable=False
```

#USEFUL FOR GETTING NUMBER OF CLASSES

```
folders = glob('/kaggle/input/intel-image-classification/seg_train/seg_train/*')
```

#LAYERS TAT WE WILL BUILD

```
x=Flatten()(vgg.output)
prediction = Dense(len(folders) , activation="softmax")(x)
model=Model(inputs=vgg.input , outputs=prediction)
model.summary()
```



```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 6)                 150534
=================================================================
Total params: 14,865,222
Trainable params: 150,534
Non-trainable params: 14,714,688
_________________________________________________________________
```

#COMPILING THE MODEL

model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])

#PREPROCESSING THE TRAINING SET

from keras.preprocessing.image import ImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1./255,

```
                    shear_range=0.2,
                    zoom_range=0.2,
                    horizontal_flip=True)

training_set = train_datagen.flow_from_directory("/kaggle/input/intel-image-classification/seg_train/seg_train",
                                                 target_size=(224,224),
                                                 batch_size=32,
                                                 class_mode="categorical")
```

```
Found 3000 images belonging to 6 classes.
```

#PREPROCESSING THE TEST SET

```
test_datagen=ImageDataGenerator(rescale=1./225)

test_set=test_datagen.flow_from_directory("/kaggle/input/intel-image-classification/seg_test/seg_test",
                                          target_size=(224,224),
                                          batch_size=32,
                                          class_mode="categorical")
```

```
Found 3000 images belonging to 6 classes.
```

#FITTING THE MODEL

```
r=model.fit(training_set,
            validation_data=test_set,
            epochs=5,
            steps_per_epoch=len(training_set),
            validation_steps=len(test_set))
```

```
Epoch 1/5
110/110 [==============================] - 851s 7s/step - loss: 0.9155 - accuracy: 0.6559 -
val_loss: 0.4227 - val_accuracy: 0.8420
Epoch 2/5
110/110 [==============================] - 787s 7s/step - loss: 0.4248 - accuracy: 0.8537 -
val_loss: 0.3711 - val_accuracy: 0.8627
Epoch 3/5
110/110 [==============================] - 787s 7s/step - loss: 0.3487 - accuracy: 0.8742 -
val_loss: 0.3442 - val_accuracy: 0.8750
Epoch 4/5
110/110 [==============================] - 788s 7s/step - loss: 0.3186 - accuracy: 0.8896 -
val_loss: 0.3234 - val_accuracy: 0.8817
Epoch 5/5
110/110 [==============================] - 787s 7s/step - loss: 0.3001 - accuracy: 0.8963 -
val_loss: 0.3517 - val_accuracy: 0.8633
```

#GETTING INDICES OF TRAINING SET

```
from keras.preprocessing import image
training_set.class_indices
names=np.array(["Building","Forest","Glacier","Mountain","Sea","Street"])
```

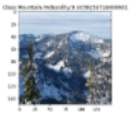
#CREATING A LIST OF 5 RANDOM IMAGES

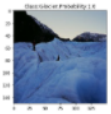
```
import os
pred = os.listdir("../input/intel-image-classification/seg_pred/seg_pred/")
import random
images = []
for i in range(5):
   images.append("../input/intel-image-
classification/seg_pred/seg_pred/"+random.choice(pred))
```

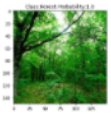
```
{'buildings': 0,
 'forest': 1,
 'glacier': 2,
 'mountain': 3,
 'sea': 4,
 'street': 5}
```

#PREDICTING CLASSES OF 5 DIFFERENT IMAGES

```
for img in images:
    test_img=image.load_img(img,target_size=(224,224))
    test_img=image.img_to_array(test_img)
    test_img=np.expand_dims(test_img,axis=0)
    result=model.predict(test_img)
    plt.title('Class:{},Probability:{}'.format(names[result.argmax()],result.max()))
    plt.imshow(image.img_to_array(image.load_img(img,
target_size=(150,150)))/255.)
    plt.show()
```





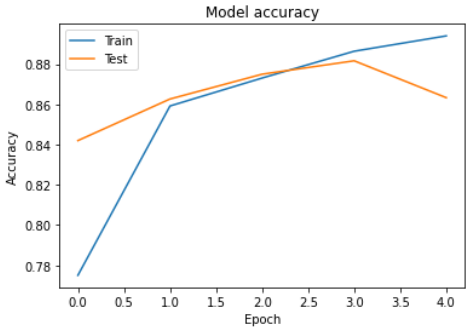


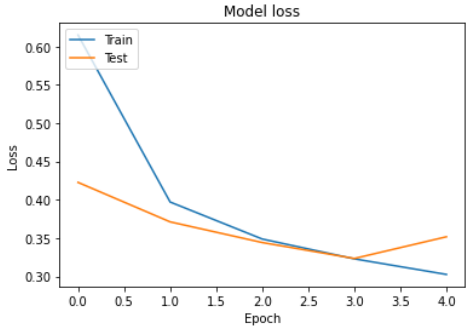
#VISUALIZING THE LOSS AND ACCURACY

```
plt.plot(r.history['accuracy'])
plt.plot(r.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
```

```
plt.show()

plt.plot(r.history['loss'])
plt.plot(r.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```





WE USED THE VGG IN THE ABOVE CODE AND GOT AN AMAZING ACCURACY OF ALMOST 90%.