**Questions**

**1.When we know whether its overfit model? How to overcome it?**

Ans: Overfitting or high variance in machine learning models occurs when the accuracy of your training dataset, the dataset used to “teach” the model, is greater than your testing accuracy. In terms of ‘loss’, overfitting reveals itself when your model has a low error in the training set and a higher error in the testing set.

Following methods to be followed to overcome it:

1. **Regularization -** optimizes a model by penalizing complex models, therefore minimizing loss and complexity. Thus this forces our neural network to be simpler.
2. **Weight Initialization -** Weight initialization sets up the weights vector for all neurons for the first time before the training process begins. Choosing the correct weights is crucial because we want to get as close as possible to the global minimum of our cost function in an adequate amount of time. However, in our code we are using pretrained weights of imagenet dataset
3. **Dropout Regularization -** Dropout regularization ignores a random subset of units in a layer while setting their weights to zero during that phase of training.
4. **Weight Constraints -** A weight constraint checks the size of the network weights and rescales them if the size exceeds a predefined limit. Not possible to implement in our case as we are using pretrained weights.
5. **Data Augmentation – More images from one image but having different flips. Already performed in our code.**

**2.Can we use other models like logistic regression or random forest model?**

Ans: Limitation of Random Forest: Due to the way regression trees are constructed it is not possible to predict beyond the range of the response values in the training data. For example, if the training data for a biomass model contain low and moderate biomass values but no high values it will not be possible to accurately predict high biomass values when the model is applied to the full data set. It is extremely important that the training data include samples that cover the entire range of response data values. Another issue related to regression is that random forests tends to overestimate the low values and underestimate the high values. This is because the response from random forests in the case of regression is the average (mean) of all of the trees. Also, large number of trees can make the **algorithm** too slow and ineffective for real-time predictions.

**Random forest works better with tabular data**

Logistic Regression: The “classic” application of logistic regression model is binary classification. However, we can also use “flavors” of logistic to tackle multi-class classification problems, e.g., using the One-vs-All or One-vs-One approaches, via the related softmax regression / multinomial logistic regression. Although there are kernelized variants of logistic regression exist, the standard “model” is a linear classifier. Thus, logistic regression is useful if we are working with a dataset where the classes are more or less “linearly separable.”

SVM: The Deep Learning method gives significantly higher accuracy than the "classical" methods (such as SVM, Logistic Regression, etc.).

**Q3: Benefits of CNN for image classification over other models:**

Ans: CNNs are used for image classification and recognition because of its high accuracy. The CNN follows a hierarchical model which works on building a network, like a funnel, and finally gives out a fully-connected layer where all the neurons are connected to each other and the output is processed. More generally, CNNs work well with data that has a partial relationship. Therefore CNNs are go-to method for any type of prediction problem involving image data as an input.  
The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. The weights are smaller and shared — less wasteful, easier to train than ANN model and more effective too. They can also go deeper. Layers are sparsely connected rather than fully connected.  
It takes matrices as well as vectors as inputs.  
*The layers are*sparsely connected *or partially connected rather than*fully connected*.*Every node does not connect to every other node.

**Q4: Why do we train the last layer of our model?**

Ans: For our base models we are considering pre-trained models like VGG16, Vgg19, exception etc. However, these models have been trained on imagenet dataset and has 1000 categories. Inorder to train our model on the 4 categories of our choice we train the final dense layer.

**Q5: Why Xception model is better?**

Ans: The following reasons:

1. Model is light weighted
2. Better accuracy
3. Better computational as number of parameters are way less than other models.

**Q6: Number of layers in a neural network.**

Ans: A network may have three types of layers: input layers that take raw input from the domain, **hidden layers** that take input from another layer and pass output to another layer, and **output layers** that make a prediction.

**Q7: What is activation function? How to select an activation function?**

**Ans:** An [activation function](https://en.wikipedia.org/wiki/Activation_function) in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model. The [rectified linear activation function](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/), or ReLU activation function, is perhaps the most common function used for hidden layers.

It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh. Specifically, it is less susceptible to [vanishing gradients](https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/) that prevent deep models from being trained, although it can suffer from other problems like saturated or “dead” units.

The sigmoid activation function is also called the logistic function.It is the same function used in the logistic regression classification algorithm.The function takes any real value as input and outputs values in the range 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0.

The hyperbolic tangent activation function is also referred to simply as the Tanh (also “tanh” and “TanH“) function.It is very similar to the sigmoid activation function and even has the same S-shape.The function takes any real value as input and outputs values in the range -1 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

Activation for Output Layers

The output layer is the layer in a neural network model that directly outputs a prediction.All feed-forward neural network models have an output layer.There are perhaps three activation functions you may want to consider for use in the output layer; they are:

* Linear
* Logistic (Sigmoid)
* Softmax

This is not an exhaustive list of activation functions used for output layers, but they are the most commonly used.

The [softmax function](https://machinelearningmastery.com/softmax-activation-function-with-python/) outputs a vector of values that sum to 1.0 that can be interpreted as probabilities of class membership.

It is related to the [argmax function](https://machinelearningmastery.com/argmax-in-machine-learning/) that outputs a 0 for all options and 1 for the chosen option. Softmax is a “softer” version of argmax that allows a probability-like output of a winner-take-all function.

As such, the input to the function is a vector of real values and the output is a vector of the same length with values that sum to 1.0 like probabilities.

* Binary Classification: One node, sigmoid activation.
* Multiclass Classification: One node per class, softmax activation.
* Multilabel Classification: One node per class, sigmoid activation.

**Q8: What is loss function? How to select one?**

Ans: The**Loss Function** is one of the important components of Neural Networks. **Loss** is nothing but a prediction error of Neural Net. And the method to calculate the loss is called Loss Function.In simple words, the Loss is used to calculate the gradients. And gradients are used to update the weights of the Neural Net. This is how a Neural Net is trained .**Keras** and **Tensorflow** have various inbuilt loss functions for different objectives. Find the following essential loss functions, which could be used for most of the objectives.

* **Mean Squared Error (MSE)**
* **Binary Crossentropy (BCE)**
* **Categorical Crossentropy (CC)**

# **Binary Crossentropy**

**BCE** loss is used for the binary classification tasks. If you are using BCE loss function, you just need one output node to classify the data into two classes. The output value should be passed through a sigmoid activation function and the range of output is (0 – 1).

For example, we have a neural network that takes atmosphere data and predicts whether it will rain or not. If the output is greater than 0.5, the network classifies it as rain and if the output is less than 0.5, the network classifies it as not rain. (it could be opposite depending upon how you train the network). More the probability score value, the more the chance of raining.

# **Categorical Crossentropy**

When we have a multi-class classification task, one of the loss function you can go ahead is this one. If you are using CCE loss function, there must be the same number of output nodes as the classes. And the final layer output should be passed through a softmax activation so that each node output a probability value between (0–1).

## Example

For example, we have a neural network that takes an image and classifies it into a cat or dog. If the cat node has a high probability score then the image is classified into a cat otherwise dog. Basically, whichever class node has the highest probability score, the image is classified into that class.

**Q9: What is optimizer?**

Ans: **Optimizers** are algorithms or methods used to minimize an error function(loss function)or to maximize the efficiency of production. Optimizers are mathematical functions which are dependent on model’s learnable parameters i.e Weights & Biases. Optimizers help to know how to change weights and learning rate of neural network to reduce the losses.

Some of the optimizers:

# **Gradient Descent**

Gradient descent is an optimization algorithm based on a convex function and tweaks its parameters iteratively to minimize a given function to its local minimum. Gradient Descentiteratively reduces a loss function by moving in the direction opposite to that of steepest ascent. It is dependent on the derivatives of the loss function for finding minima. uses the data of the entire training set to calculate the gradient of the cost function to the parameters which requires large amount of memory and slows down the process.

**Advantages of Gradient Descent**

1. Easy to understand
2. Easy to implement
3. **Disadvantages of Gradient Descent**
4. Because this method calculates the gradient for the entire data set in one update, the calculation is very slow.
5. It requires large memory and it is computationally expensive.

# **Learning Rate**

How big/small the steps are gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights.

# **Stochastic Gradient Descent**

It is a variant of Gradient Descent. It update the model parameters one by one. If the model has 10K dataset SGD will update the model parameters 10k times.

**Advantages of Stochastic Gradient Descent**

1. Frequent updates of model parameter
2. Requires less Memory.
3. Allows the use of large data sets as it has to update only one example at a time.

**Disadvantages of Stochastic Gradient Descent**

1. The frequent can also result in noisy gradients which may cause the error to increase instead of decreasing it.
2. High Variance.
3. Frequent updates are computationally expensive.

# **Mini-Batch Gradient Descent**

It is a combination of the concepts of SGD and batch gradient descent. It simply splits the training dataset into small batches and performs an update for each of those batches. This creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. it can reduce the variance when the parameters are updated, and the convergence is more stable. It splits the data set in batches in between 50 to 256 examples, chosen at random.

**Advantages of Mini Batch Gradient Descent:**

1. It leads to more stable convergence.
2. more efficient gradient calculations.
3. Requires less amount of memory.

**Disadvantages of Mini Batch Gradient Descent**

1. Mini-batch gradient descent does not guarantee good convergence,
2. If the learning rate is too small, the convergence rate will be slow. If it is too large, the loss function will oscillate or even deviate at the minimum value.

# **SGD with Momentum**

**SGD with Momentum** is a stochastic optimization method that adds a momentum term to regular stochastic gradient descent. Momentum simulates the inertia of an object when it is moving, that is, the direction of the previous update is retained to a certain extent during the update, while the current update gradient is used to fine-tune the final update direction. In this way, you can increase the stability to a certain extent, so that you can learn faster, and also have the ability to get rid of local optimization.

**Advantages of SGD with momentum**

1. Momentum helps to reduce the noise.
2. Exponential Weighted Average is used to smoothen the curve.

**Disadvantage of SGD with momentum**:Extra hyperparameter is added.

# **AdaGrad(Adaptive Gradient Descent)**

In all the algorithms that we discussed previously the learning rate remains constant. The intuition behind AdaGrad is can we use different Learning Rates for each and every neuron for each and every hidden layer based on different iterations.

**Advantages of AdaGrad**

1. Learning Rate changes adaptively with iterations.
2. It is able to train sparse data as well.

**Disadvantage of AdaGrad**

1. If the neural network is deep the learning rate becomes very small number which will cause dead neuron problem.

# **RMS-Prop (Root Mean Square Propagation)**

RMS-Prop is a special version of Adagrad in which the learning rate is an exponential average of the gradients instead of the cumulative sum of squared gradients. RMS-Prop basically combines momentum with AdaGrad.

**Advantages of RMS-Prop**

1. In RMS-Prop learning rate gets adjusted automatically and it chooses a different learning rate for each parameter.

**Disadvantages of RMS-Prop**:Slow Learning

# **AdaDelta**

Adadelta is an extension of Adagrad and it also tries to reduce Adagrad’s aggressive, monotonically reducing the learning rate and remove decaying learning rate problem. In Adadelta we do not need to set the default learning rate as we take the ratio of the running average of the previous time steps to the current gradient.

**Advantages of Adadelta**

1. The main advantage of AdaDelta is that we do not need to set a default learning rate.

**Disadvantages of Adadelta**

1. Computationally expensive

# **Adam(Adaptive Moment Estimation)**

Adam optimizer is one of the most popular and famous gradient descent optimization algorithms. It is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients , similar to momentum and also the decaying average of the past squared gradients , similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both the methods.

**Advantages of Adam**

1. Easy to implement
2. Computationally efficient.
3. Little memory requirements.

# **How to choose optimizers?**

* If the data is sparse, use the self-applicable methods, namely Adagrad, Adadelta, RMSprop, Adam.
* RMSprop, Adadelta, Adam have similar effects in many cases.
* Adam just added bias-correction and momentum on the basis of RMSprop,
* As the gradient becomes sparse, Adam will perform better than RMSprop.

**Q: What are the obstacles faced while implementation of project?**

**Ans:** The different issues faced -

1**.**Problem statement had multi label but the metadata file shared had one target mapped to each image. Hence, realized multi class classification.

2. Selection of best transferring models. First tried with SVM machine learning model which gave 74% accuracy. CNN model like RESNet also gave less accuracy.

3.Execution of code for more than 24 hours blocked the further development process.

Q: Difference between train,validation and test data?

Ans: **Training Set**: this data set is used to adjust the weights on the neural network.

**Validation Set**: this data set is used to minimize overfitting. You're not adjusting the weights of the network with this data set, you're just verifying that any increase in accuracy over the training data set actually yields an increase in accuracy over a data set that has not been shown to the network before, or at least the network hasn't trained on it (i.e. validation data set). If the accuracy over the training data set increases, but the accuracy over the validation data set stays the same or decreases, then you're overfitting your neural network and you should stop training.

**Testing Set**: this data set is used only for testing the final solution in order to confirm the actual predictive power of the network.