**Topic: Convolution Neural Network (CNN)**

**Instructions:**

**1. Business Problem**

* 1. **Objective**
  2. **Constraints (if any)**

**Using Python perform:**

**2. Data Pre-processing (if applicable)**

**2.1 Data cleaning, Feature Engineering etc.**

**3. Exploratory Data Analysis (EDA): (if applicable)**

**3.1. Summary**

**3.2. Univariate analysis**

**3.3. Bivariate analysis**

**4. Model Building**

**4.3 Using Python libraries perform the below tasks**

**5. Result Share the benefits/impact of the solution - how or in what way the business (client) gets benefit from the solution provided. (If applicable)**

**Note:**

The assignment should be submitted in the following format:

* Python code
* Code Modularization should be maintained
* Documentation of the modules (elaborating on steps mentioned above).

1. **Build a CNN model on CIFAR-10 dataset by applying few regularization techniques like drop out and data augmentation**

**If the input of an image is 64x64x3 which has been convolved by 10 5x5 filters with stride 1 and padding 2.**

**a. How many activation maps are obtained?**

**b. What is the size of the activation maps?**

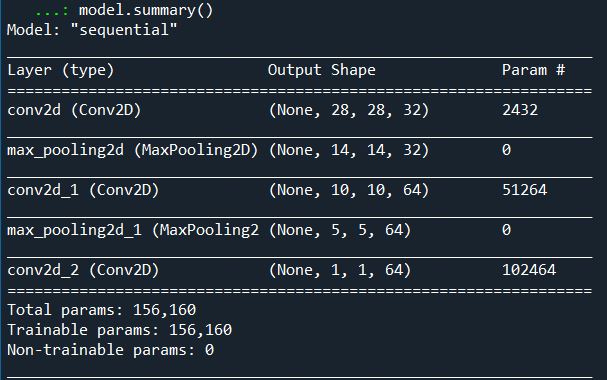
**c. How many parameters are calculated?**

**During training, I get into overfitting issues. What are the different techniques will you apply to overcome this issue and why?**

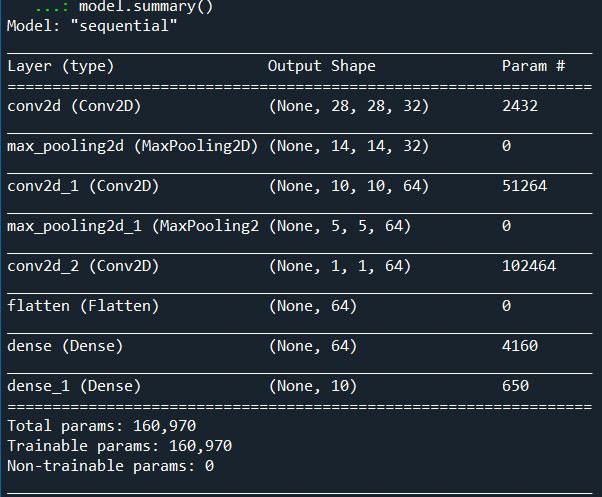
**Solution:**

The dataset is comprised of 60,000 32×32-pixel color photographs of objects from 10 classes, such as frogs, birds, cats, ships, etc.

1. By the taking the input of a image as 64x64x3 and convolution by 10 5x5 filters the model is built.



1. From the above image, the total parameters are 1,56,160.
2. Now the model is flattened (converting the data into a 1-dimensional array for inputting it to the next layer) by using “relu” and “softmax” function to check the parameters.
3. Total formed parameters are 1,60,970 as shown in below image.



**Model Building:**

1. Firstly, using the keras library, the dataset of cifar10 is imported
2. As we know that there are 10 classes and these classes are represented as unique integers.
3. Then using  [one hot encoding](https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/) for the class element of each sample, transforming the integer into a 10 element binary vector with a 1 for the index of the class value. We can achieve this using ***to\_categorical()*** utility function.
4. Here, we have 50,000 observations in the training dataset and 10,000 in the test dataset and these images are indeed squared with 32×32 pixels and color, with three channels. Then, these pixel values for each image in the dataset are unsigned integers in the range between no color and full color, or 0 and 255.
5. Finally, we check Fitting the model with the help of no of training epochs and batch size which needs to be specified. Here, We have used 15 training epochs after checking for 5 & 10 epochs respectively which showed a very low accuracy along with batch size of 64.
6. So, We have achieved the accuracy of 86% at epoch 15 after considering the highest of all epochs.

2. **Find out the differences between Convnet filter and the Maxpool layers**

1. There is no learning done in max pooling layers as no weights / parameters to update, it will just down sample the data. Whereas, in convolutional layers there are weights that are learned so it down samples the data (if no padding is used) but it also extracts learned features.
2. So, the major difference is a **ConvNet** filter is extracting features from the matrix of data, whereas the **Max** **pooling layer** is only down sampling the matrix of data i.e., reducing the size of the data.

3. If the input of an image is 64x64x3 which has been convolved by 10 5x5 filters with stride 1 and padding 2.

a. How many activation maps are obtained? **Answer) 10**

b. What is the size of the activation maps? **Answer) 64x64x10**

**Formula: N – F / Stride + 1 = 64 + 4 (padding 2) – 5 (Filter) / 1 (Stride) + 1 = 64 is the answer.**

c. How many parameters are calculated? **Answer) 760**

**So, here 5\*5\*3 + 1 ( for bias) = 76 params 🡪 76 \* 10 = 760 parameters.**

4. During training, I get into overfitting issues. What are the different techniques will you apply to overcome this issue and why?

Overfitting happens when your model fits too well to the training set. It then becomes difficult for the model to generalize to new examples that were not in the training set. For example, your model recognizes specific images in your training set instead of general patterns. Your training accuracy will be higher than the accuracy on the validation/test set.

* Steps to reduce the Overfitting of the model.

1. **Add more data:**

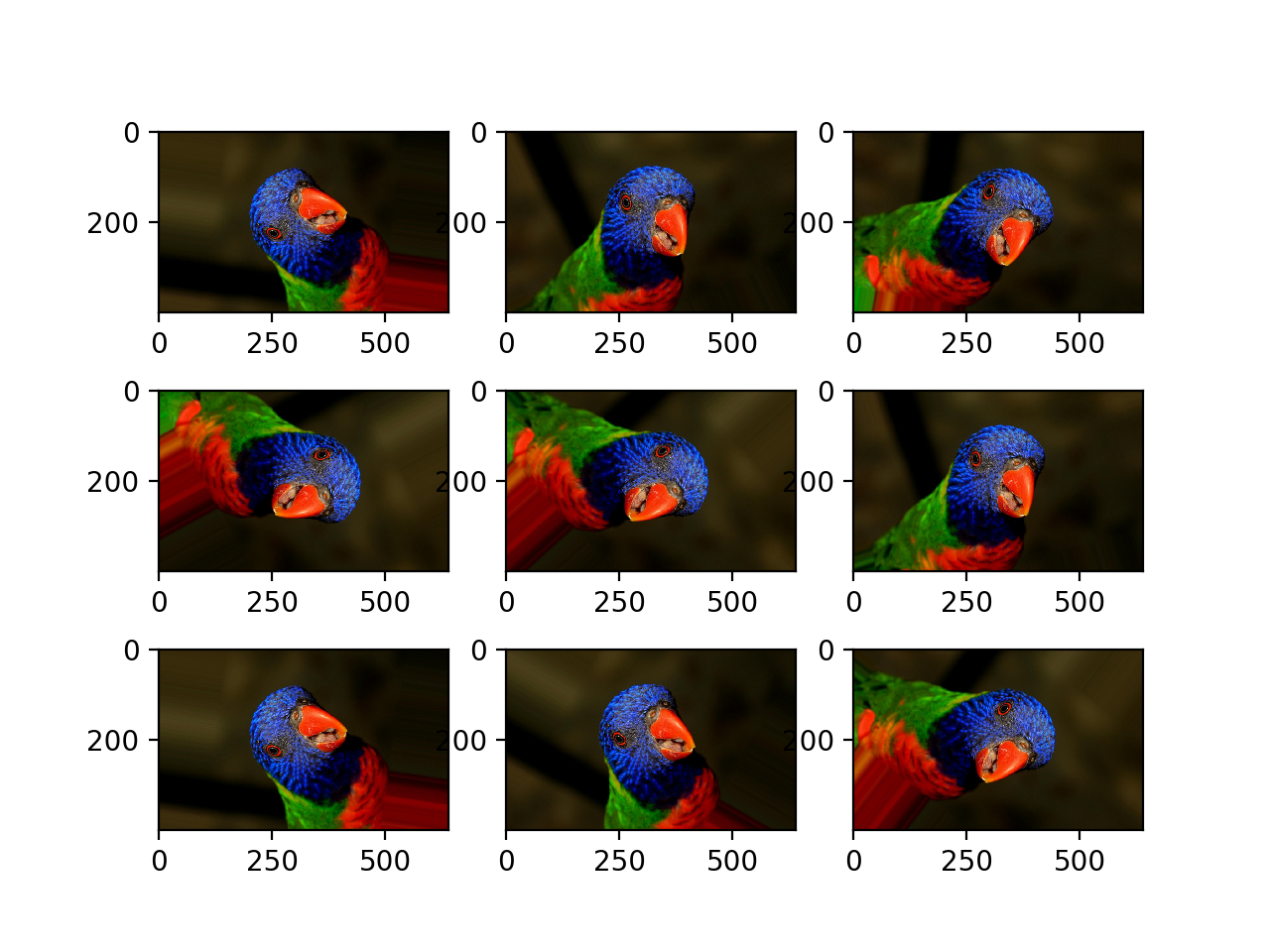
Here, we can add more data which is generally not possible but lets assume for instance we collected additional data and then we need to do data augmentation as next step.

1. **Data augmentation:**

Data augmentation means changing the image to various ways like rotating, Scaling, zoom in or adding colour filer which will be done only on training dataset but not on test data. Now even after zooming in or adding colour filters to maximum level wherein the image gets invisible then we proceed to architecture level.



**Original Image**



**After Augmentation**

1. **Use architectures that generalize well. However, the most important is the next step of regularisation.**
2. **Add regularization like L1/L2 regularization can be applied:**

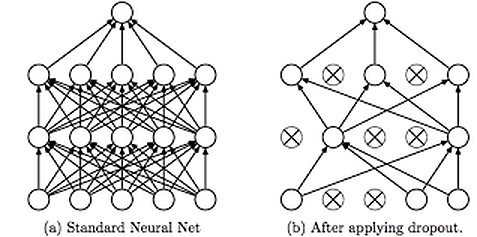
L1 regularisation penalises sum of absolute value of weights whereas, L2 penalises the sum of square values of weights.

L1 generates model which is simple & interpretable whereas, L2 regularisation is able to learn complex data patterns.

L1 is robust to outliers whereas, L2 is not robust to outliers.

1. **Reduce architecture complexity.**

Dropout is a regularization technique that prevents neural networks from overfitting. Regularization methods like L1 and L2 reduce overfitting by modifying the cost function. Dropout on the other hand, modify the network itself. It randomly drops neurons from the neural network during training in each iteration. When we drop different sets of neurons, it’s equivalent to training different neural networks. The different networks will overfit in different ways, so the net effect of dropout will be to reduce overfitting.



This technique is shown in the above diagram. As we can see, dropouts are used to randomly remove neurons while training of the neural network. This technique has proven to reduce overfitting to a variety of problems involving image classification, image segmentation, word embeddings, semantic matching etc.