# Assignment 3

1. Name: Vaibhav Bichave
2. Batch: P-10
3. Roll No.: 43209

## Problem Statement :

Build the Image classification model

#### Importing the libraries

In [1]:

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
```

#### a. Loading and preprocessing the image data

#### Grabbing CIFAR10 dataset

In [2]:

```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_da
train_images, test_images = train_images / 255.0, test_images / 255.0
```

In [4]:

```
type(train_images)
```

Out[4]:

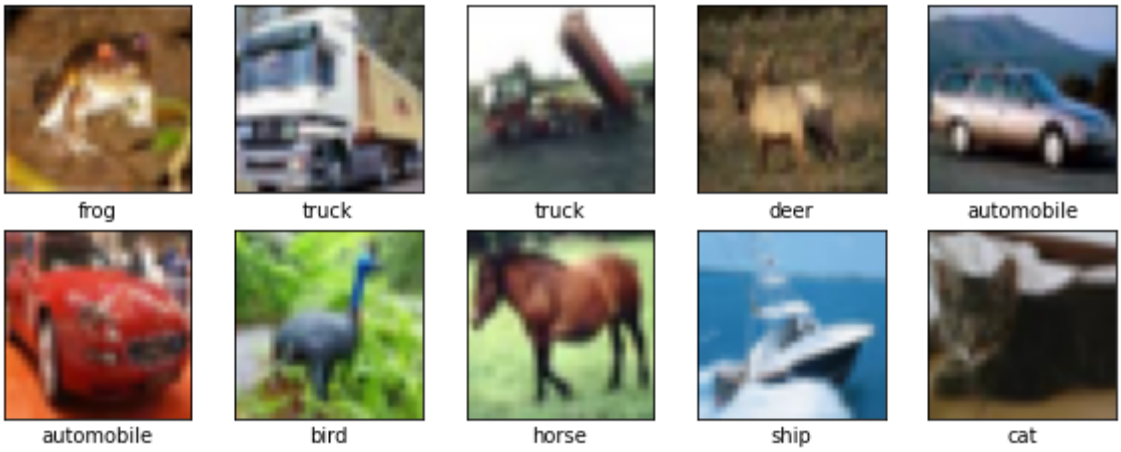
```
numpy.ndarray
```

#### Showing images of mentioned categories

In [5]:

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer','dog', 'frog', 'hors

plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```




**b. Defining the model's architecture**

**Building CNN model**

In [6]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

**c. Training the model**

**Model compilation**

In [7]:

```
model.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy(f
```

**d. Estimating the model's performance**

In [9]:

```
epochs = 10
h = model.fit(train_images, train_labels, epochs=epochs, validation_data=(test_imag
```

```
Epoch 1/10
1563/1563 [==============================] - 66s 42ms/step - loss: 1.1
581 - accuracy: 0.5890 - val_loss: 1.0580 - val_accuracy: 0.6277
Epoch 2/10
1563/1563 [==============================] - 69s 44ms/step - loss: 1.0
107 - accuracy: 0.6472 - val_loss: 0.9943 - val_accuracy: 0.6511
Epoch 3/10
1563/1563 [==============================] - 66s 43ms/step - loss: 0.9
166 - accuracy: 0.6772 - val_loss: 0.9544 - val_accuracy: 0.6693
Epoch 4/10
1563/1563 [==============================] - 65s 41ms/step - loss: 0.8
453 - accuracy: 0.7075 - val_loss: 0.9113 - val_accuracy: 0.6789
Epoch 5/10
1563/1563 [==============================] - 66s 42ms/step - loss: 0.7
916 - accuracy: 0.7241 - val_loss: 0.9141 - val_accuracy: 0.6884
Epoch 6/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.7
393 - accuracy: 0.7418 - val_loss: 0.8678 - val_accuracy: 0.7065
Epoch 7/10
1563/1563 [==============================] - 66s 42ms/step - loss: 0.6
958 - accuracy: 0.7573 - val_loss: 0.9028 - val_accuracy: 0.6945
Epoch 8/10
1563/1563 [==============================] - 66s 42ms/step - loss: 0.6
597 - accuracy: 0.7692 - val_loss: 0.8688 - val_accuracy: 0.7020
Epoch 9/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.6
243 - accuracy: 0.7818 - val_loss: 0.8780 - val_accuracy: 0.7071
Epoch 10/10
1563/1563 [==============================] - 65s 42ms/step - loss: 0.5
905 - accuracy: 0.7920 - val_loss: 0.8929 - val_accuracy: 0.7151
```

In [ ]:

```

```