In [30]:

```
import glob
import random
import numpy as np
import cv2
```

In [31]:

```
base_dir = []
for i in glob.glob(r'C:\Users\User\Anaconda3\Lib\Neural networks\Dataset\cars_*'):
    base_dir.append(i)
```

In [32]:

```
class_name = []
for i in base_dir:
    class_name.append(i.split('\\')[7])
class_name
```

Out[32]:

```
['cars_audi', 'cars_benz', 'cars_bmw', 'cars_hcity']
```

In [33]:

```
list_of_tuples = []
for i in base_dir:
    paths = glob.glob(i+'/*.jpg')
    current_class = i.split('\\')[7]
    for path in paths:
        path_dictionary = {}
        path_dictionary['path'] = path
        list_of_tuples.append((current_class,path_dictionary))
```

In [34]:

```
random.shuffle(list_of_tuples)
```

In [35]:

```
epochs = 30
alpha = 0.1
```

In [36]:

```
def ReLu(x):
    return max(0,x)
```

In [37]:

```
input_neurons = 4900
train_test_ratio = 0.6
random.shuffle(list_of_tuples)
training_list = list_of_tuples[:int(len(list_of_tuples)*train_test_ratio)]
testing_list = list_of_tuples[int(len(list_of_tuples)*train_test_ratio):]
weights = {c:np.array([0.0 for _ in range(input_neurons)]) for c in class_name}
```

In [38]:

```
def training_dataset():

        for _ in range(epochs):
            for clas,path_dictionary in training_list:
                image = cv2.imread(path_dictionary['path'])
```

```
                dim = (70,70)
                resized_image = cv2.resize(image,dim,cv2.INTER_AREA)
                input_array = []

                for i in range(resized_image.shape[0]):
                    for j in range(resized_image.shape[1]):
                        input_array.append(np.mean(resized_image[i][j]/255))

                inputs = np.array(input_array)
                maxi=0
                t=1
                predicted_class = class_name[0]

                for i in class_name:
                    yin = np.dot(inputs,weights[i])
                    y = ReLu(yin)
                    if y >= maxi:
                        maxi = y
                        predicted_class = i

                if not(clas == predicted_class):
                    weights[clas] += [alpha*t*x for x in inputs]
                else:
                    weights[clas]=weights[clas]
```

In [39]:

```
training_dataset()
```

In [40]:

```
def prediction(path_dictionary):

        img = cv2.imread(path_dictionary['path'])
        dim = (70,70)
        resized_image = cv2.resize(img,dim,cv2.INTER_AREA)
        input_array = []
        for i in range(resized_image.shape[0]):
            for j in range(resized_image.shape[1]):
                input_array.append(np.mean(resized_image[i][j]/255))

        inputs = np.array(input_array)
        maxi = 0
        predicted_class = class_name[0]
        for i in class_name:
            yin = np.dot(inputs,weights[i])
            y = ReLu(yin)
            if y >= maxi:
                maxi = y
                predicted_class = i

        return predicted_class
```

In [41]:

```
def accuracy():
        correct = 0
        incorrect = 0
        random.shuffle(list_of_tuples)
        testing_list = list_of_tuples[int(len(list_of_tuples)*train_test_ratio):]
        for path_dictionary in testing_list:
            actual_class = path_dictionary[0]
            predicted_class = prediction(path_dictionary[1])

            if actual_class == predicted_class:
                correct = correct+1
            else:
                incorrect = incorrect+1

        print("Accuracy is ",(correct * 1.0) / ((correct + incorrect) * 1.0))
```

In [42]:

In [42]:

```
def random_testing():
        s = glob.glob(r'C:\Users\User\Anaconda3\Lib\Neural networks\Testing_Sample\*')
        img = cv2.imread(s[0])
        dim = (70,70)
        resized_image = cv2.resize(img,dim,cv2.INTER_AREA)
        input_array = []
        for i in range(resized_image.shape[0]):
            for j in range(resized_image.shape[1]):
                input_array.append(np.mean(resized_image[i][j]/255))
        inputs = np.array(input_array)
        maxi = 0
        predicted_class = class_name[0]
        for i in class_name:
            yin = np.dot(inputs,weights[i])
            y = ReLu(yin)
            if y >= maxi:
                maxi = y
                predicted_class = i

        print("The Predicted Class for the Sample Image is")
        print(predicted_class)
```

In [43]:

```
accuracy()
```

```
Accuracy is  0.71875
```

In [44]:

```
random_testing()
```

```
The Predicted Class for the Sample Image is
cars_benz
```

In [ ]:

```

```