CODE TO CLASSIFY IMAGES (CIFAR-10) USING CNNs

STEP 0: PROBLEM STATEMENT

- CIFAR-10 is a dataset that consists of several images divided into the following 10 classes:
    - Airplanes
    - Cars
    - Birds
    - Cats
    - Deer
    - Dogs
    - Frogs
    - Horses
    - Ships
    - Trucks
- The dataset stands for the Canadian Institute For Advanced Research (CIFAR)
- CIFAR-10 is widely used for machine learning and computer vision applications.
- The dataset consists of 60,000 32x32 color images and 6,000 images of each class.
- Images have low resolution (32x32).

STEP #1: IMPORT LIBRARIES/DATASETS

In [14]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn
```

In [15]:
```
from keras.datasets import cifar10
(X_train, y_train) , (X_test, y_test) = cifar10.load_data()
```

In [16]:
```
X_train.shape
```

Out[16]: (50000, 32, 32, 3)

In [17]:
```
X_test.shape
```

Out[17]: (10000, 32, 32, 3)

In [18]:
```
y_train.shape
```

Out[18]: (50000, 1)

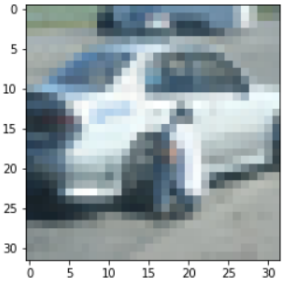
In [19]:
```
y_test.shape
```

Out[19]: (10000, 1)

STEP #2: VISUALIZE DATA

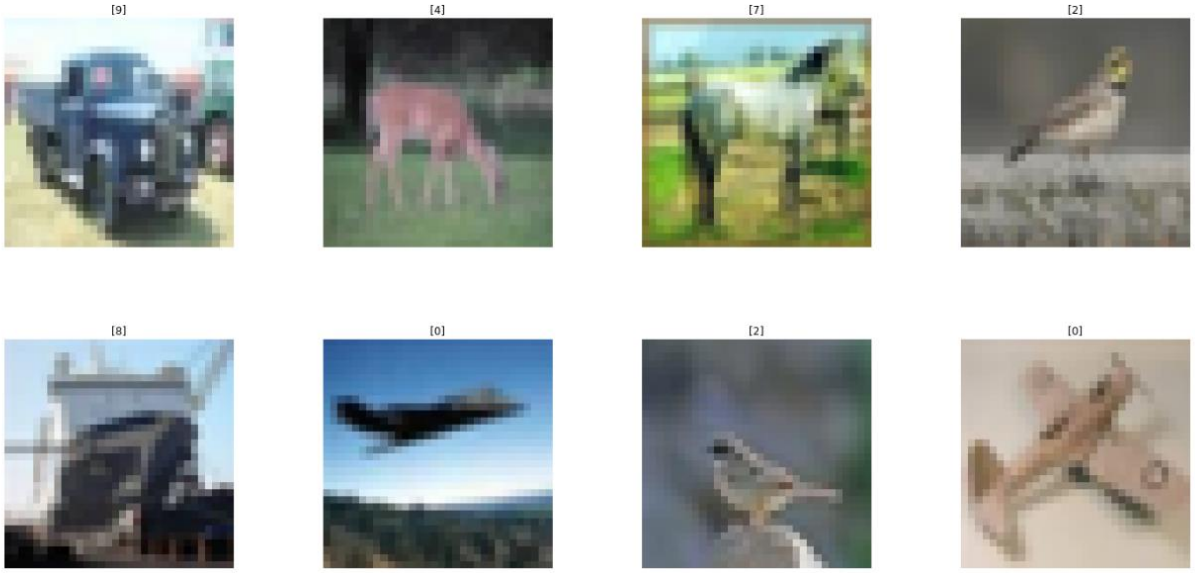
In [20]:
```
i = 30009
plt.imshow(X_train[i])
print(y_train[i])
```

```
[1]
```



In [21]:
```
W_grid = 4
L_grid = 4

fig, axes = plt.subplots(L_grid, W_grid, figsize = (25, 25))
axes = axes.ravel()

n_training = len(X_train)

for i in np.arange(0, L_grid * W_grid):
    index = np.random.randint(0, n_training) # pick a random number
    axes[i].imshow(X_train[index])
    axes[i].set_title(y_train[index])
    axes[i].axis('off')

plt.subplots_adjust(hspace = 0.4)
```

Output:



In [22]:
```
n_training
```

Out[22]:
```
50000
```

STEP #3: DATA PREPARATION

In [23]:
```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
```

In [24]:
```
number_cat = 10
```

In [25]:
```
y_train
```

Out[25]:
```
array([[6],
       [9],
       [9],
       ...,
       [9],
       [1],
       [1]], dtype=uint8)
```

In [26]:
```
import keras
y_train = keras.utils.to_categorical(y_train, number_cat)
```

In [27]:
```
y_train
```

Out[27]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [28]:
```
y_test = keras.utils.to_categorical(y_test, number_cat)
```

In [29]:
```
y_test
```

Out[29]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)
```

In [30]:
```
X_train = X_train/255
X_test = X_test/255
```

In [31]:
```
X_train
```

Out[31]:
```
array([[[[0.23137255, 0.24313726, 0.24705882],
         [0.16862746, 0.18039216, 0.1764706 ],
         [0.19607843, 0.1882353 , 0.16862746],
         ...,
         [0.61960787, 0.5176471 , 0.42352942],
         [0.59607846, 0.49019608, 0.4       ],
         [0.5803922 , 0.4862745 , 0.40392157]],

        [[0.0627451 , 0.07843138, 0.07843138],
         [0.        , 0.        , 0.        ],
         [0.07058824, 0.03137255, 0.        ],
         ...,
         [0.48235294, 0.34509805, 0.21568628],
         [0.46666667, 0.3254902 , 0.19607843],
         [0.47843137, 0.34117648, 0.22352941]],

        [[0.09803922, 0.09411765, 0.08235294],
         [0.0627451 , 0.02745098, 0.        ],
         [0.19215687, 0.10588235, 0.03137255],
         ...,
         [0.4627451 , 0.32941177, 0.19607843],
         [0.47058824, 0.32941177, 0.19607843],
         [0.42745098, 0.28627452, 0.16470589]],

        ...,

        [[0.8156863 , 0.6666667 , 0.3764706 ],
         [0.7882353 , 0.6       , 0.13333334],
         [0.7764706 , 0.6313726 , 0.10196079],
         ...,
         [0.627451  , 0.52156866, 0.27450982],
         [0.21960784, 0.12156863, 0.02745098],
         [0.20784314, 0.13333334, 0.07843138]],

        [[0.7058824 , 0.54509807, 0.3764706 ],
         [0.6784314 , 0.48235294, 0.16470589],
         [0.7294118 , 0.5647059 , 0.11764706],
         ...,
         [0.72156864, 0.5803922 , 0.36862746],
         [0.38039216, 0.24313726, 0.13333334],
         [0.3254902 , 0.20784314, 0.13333334]],

        [[0.69411767, 0.5647059 , 0.45490196],
         [0.65882355, 0.5058824 , 0.36862746],
         [0.7019608 , 0.5568628 , 0.34117648],
         ...,
         [0.84705883, 0.72156864, 0.54901963],
         [0.5921569 , 0.4627451 , 0.32941177],
         [0.48235294, 0.36078432, 0.28235295]]],


       [[[0.6039216 , 0.69411767, 0.73333335],
         [0.49411765, 0.5372549 , 0.53333336],
         [0.4117647 , 0.40784314, 0.37254903],
         ...,
         [0.35686275, 0.37254903, 0.2784314 ],
         [0.34117648, 0.3529412 , 0.2784314 ],
         [0.30980393, 0.31764707, 0.27450982]],

        [[0.54901963, 0.627451  , 0.6627451 ],
         [0.5686275 , 0.6       , 0.6039216 ],
         [0.49019608, 0.49019608, 0.4627451 ],
```

```
   [0.3764706 , 0.3882353 , 0.30588236],
   [0.3019608 , 0.3137255 , 0.24313726],
   [0.2784314 , 0.28627452, 0.23921569]],

  [[0.54901963, 0.60784316, 0.6431373 ],
   [0.54509807, 0.57254905, 0.58431375],
   [0.4509804 , 0.4509804 , 0.4392157 ],
   ...,
   [0.30980393, 0.32156864, 0.2509804 ],
   [0.26666668, 0.27450982, 0.21568628],
   [0.2627451 , 0.27058825, 0.21568628]],

  ...,

  [[0.6862745 , 0.654902  , 0.6509804 ],
   [0.6117647 , 0.6039216 , 0.627451  ],
   [0.6039216 , 0.627451  , 0.6666667 ],
   ...,
   [0.16470589, 0.13333334, 0.14117648],
   [0.23921569, 0.20784314, 0.22352941],
   [0.3647059 , 0.3254902 , 0.35686275]],

  [[0.64705884, 0.6039216 , 0.5019608 ],
   [0.6117647 , 0.59607846, 0.50980395],
   [0.62352943, 0.6313726 , 0.5568628 ],
   ...,
   [0.40392157, 0.3647059 , 0.3764706 ],
   [0.48235294, 0.44705883, 0.47058824],
   [0.5137255 , 0.4745098 , 0.5137255 ]],

  [[0.6392157 , 0.5803922 , 0.47058824],
   [0.61960787, 0.5803922 , 0.47843137],
   [0.6392157 , 0.6117647 , 0.52156866],
   ...,
   [0.56078434, 0.52156866, 0.54509807],
   [0.56078434, 0.5254902 , 0.5568628 ],
   [0.56078434, 0.52156866, 0.5647059 ]]],


[[[1.        , 1.        , 1.        ],
  [0.99215686, 0.99215686, 0.99215686],
  [0.99215686, 0.99215686, 0.99215686],
  ...,
  [0.99215686, 0.99215686, 0.99215686],
  [0.99215686, 0.99215686, 0.99215686],
  [0.99215686, 0.99215686, 0.99215686]],

 [[1.        , 1.        , 1.        ],
  [1.        , 1.        , 1.        ],
  [1.        , 1.        , 1.        ],
  ...,
  [1.        , 1.        , 1.        ],
  [1.        , 1.        , 1.        ],
  [1.        , 1.        , 1.        ]],

 [[1.        , 1.        , 1.        ],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  ...,
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843],
  [0.99607843, 0.99607843, 0.99607843]],

 ...,

 [[0.44313726, 0.47058824, 0.4392157 ],
  [0.43529412, 0.4627451 , 0.43529412],
  [0.4117647 , 0.4392157 , 0.41568628],
  ...,
  [0.28235295, 0.31764707, 0.3137255 ],
  [0.28235295, 0.3137255 , 0.30980393],
  [0.28235295, 0.3137255 , 0.30980393]],

 [[0.43529412, 0.4627451 , 0.43137255],
  [0.40784314, 0.43529412, 0.40784314],
  [0.3882353 , 0.41568628, 0.38431373],
```

```
  [0.26666668, 0.29411766, 0.28627452],
  [0.27450982, 0.29803923, 0.29411766],
  [0.30588236, 0.32941177, 0.32156864]],

 [[0.41568628, 0.44313726, 0.4117647 ],
  [0.3882353 , 0.41568628, 0.38431373],
  [0.37254903, 0.4       , 0.36862746],
  ...,
  [0.30588236, 0.33333334, 0.3254902 ],
  [0.30980393, 0.33333334, 0.3254902 ],
  [0.3137255 , 0.3372549 , 0.32941177]]],


...,


[[[0.13725491, 0.69803923, 0.92156863],
  [0.15686275, 0.6901961 , 0.9372549 ],
  [0.16470589, 0.6901961 , 0.94509804],
  ...,
  [0.3882353 , 0.69411767, 0.85882354],
  [0.30980393, 0.5764706 , 0.77254903],
  [0.34901962, 0.5803922 , 0.7411765 ]],

 [[0.22352941, 0.7137255 , 0.91764706],
  [0.17254902, 0.72156864, 0.98039216],
  [0.19607843, 0.7176471 , 0.9411765 ],
  ...,
  [0.6117647 , 0.7137255 , 0.78431374],
  [0.5529412 , 0.69411767, 0.80784315],
  [0.45490196, 0.58431375, 0.6862745 ]],

 [[0.38431373, 0.77254903, 0.92941177],
  [0.2509804 , 0.7411765 , 0.9882353 ],
  [0.27058825, 0.7529412 , 0.9607843 ],
  ...,
  [0.7372549 , 0.7647059 , 0.80784315],
  [0.46666667, 0.5294118 , 0.5764706 ],
  [0.23921569, 0.30980393, 0.3529412 ]],


  [[0.28627452, 0.30980393, 0.3019608 ],
   [0.20784314, 0.24705882, 0.26666668],
   [0.21176471, 0.26666668, 0.3137255 ],
   ...,
   [0.06666667, 0.15686275, 0.2509804 ],
   [0.08235294, 0.14117648, 0.2       ],
   [0.12941177, 0.1882353 , 0.19215687]],

  [[0.23921569, 0.26666668, 0.29411766],
   [0.21568628, 0.27450982, 0.3372549 ],
   [0.22352941, 0.30980393, 0.40392157],
   ...,
   [0.09411765, 0.1882353 , 0.28235295],
   [0.06666667, 0.13725491, 0.20784314],
   [0.02745098, 0.09019608, 0.1254902 ]],

  [[0.17254902, 0.21960784, 0.28627452],
   [0.18039216, 0.25882354, 0.34509805],
   [0.19215687, 0.3019608 , 0.4117647 ],
   ...,
   [0.10588235, 0.20392157, 0.3019608 ],
   [0.08235294, 0.16862746, 0.25882354],
   [0.04705882, 0.12156863, 0.19607843]]],


 [[[0.7411765 , 0.827451  , 0.9411765 ],
   [0.7294118 , 0.8156863 , 0.9254902 ],
   [0.7254902 , 0.8117647 , 0.92156863],
   ...,
   [0.6862745 , 0.7647059 , 0.8784314 ],
   [0.6745098 , 0.7607843 , 0.87058824],
   [0.6627451 , 0.7607843 , 0.8627451 ]],

  [[0.7607843 , 0.8235294 , 0.9372549 ],
   [0.7490196 , 0.8117647 , 0.9254902 ],
   [0.74509805, 0.80784315, 0.92156863],
```

```
 [0.6784314 , 0.7529412 , 0.8627451 ],
 [0.67058825, 0.7490196 , 0.85490197],
 [0.654902  , 0.74509805, 0.84705883]],

[[0.8156863 , 0.85882354, 0.95686275],
 [0.8039216 , 0.84705883, 0.9411765 ],
 [0.8       , 0.84313726, 0.9372549 ],
 ...,
 [0.6862745 , 0.7490196 , 0.8509804 ],
 [0.6745098 , 0.74509805, 0.84705883],
 [0.6627451 , 0.7490196 , 0.84313726]],

...,

[[0.8117647 , 0.78039217, 0.70980394],
 [0.79607844, 0.7647059 , 0.6862745 ],
 [0.79607844, 0.76862746, 0.6784314 ],
 ...,
 [0.5294118 , 0.5176471 , 0.49803922],
 [0.63529414, 0.61960787, 0.5882353 ],
 [0.65882355, 0.6392157 , 0.5921569 ]],

[[0.7764706 , 0.74509805, 0.6666667 ],
 [0.7411765 , 0.70980394, 0.62352943],
 [0.7058824 , 0.6745098 , 0.5764706 ],
 ...,
 [0.69803923, 0.67058825, 0.627451  ],
 [0.6862745 , 0.6627451 , 0.6117647 ],
 [0.6862745 , 0.6627451 , 0.6039216 ]],

[[0.7764706 , 0.7411765 , 0.6784314 ],
 [0.7411765 , 0.70980394, 0.63529414],
 [0.69803923, 0.6666667 , 0.58431375],
 ...,
 [0.7647059 , 0.72156864, 0.6627451 ],
 [0.76862746, 0.7411765 , 0.67058825],
 [0.7647059 , 0.74509805, 0.67058825]]],


[[[0.8980392 , 0.8980392 , 0.9372549 ],
  [0.9254902 , 0.92941177, 0.96862745],
  [0.91764706, 0.9254902 , 0.96862745],
  ...,
  [0.8509804 , 0.85882354, 0.9137255 ],
  [0.8666667 , 0.8745098 , 0.91764706],
  [0.87058824, 0.8745098 , 0.9137255 ]],

 [[0.87058824, 0.8666667 , 0.8980392 ],
  [0.9372549 , 0.9372549 , 0.9764706 ],
  [0.9137255 , 0.91764706, 0.9647059 ],
  ...,
  [0.8745098 , 0.8745098 , 0.9254902 ],
  [0.8901961 , 0.89411765, 0.93333334],
  [0.8235294 , 0.827451  , 0.8627451 ]],

 [[0.8352941 , 0.80784315, 0.827451  ],
  [0.91764706, 0.9098039 , 0.9372549 ],
  [0.90588236, 0.9137255 , 0.95686275],
  ...,
  [0.8627451 , 0.8627451 , 0.9098039 ],
  [0.8627451 , 0.85882354, 0.9098039 ],
  [0.7921569 , 0.79607844, 0.84313726]],

 ...,

 [[0.5882353 , 0.56078434, 0.5294118 ],
  [0.54901963, 0.5294118 , 0.49803922],
  [0.5176471 , 0.49803922, 0.47058824],
  ...,
  [0.8784314 , 0.87058824, 0.85490197],
  [0.9019608 , 0.89411765, 0.88235295],
  [0.94509804, 0.94509804, 0.93333334]],

 [[0.5372549 , 0.5176471 , 0.49411765],
  [0.50980395, 0.49803922, 0.47058824],
  [0.49019608, 0.4745098 , 0.4509804 ],
```

```
 [0.70980394, 0.7058824 , 0.69803923],
 [0.7921569 , 0.7882353 , 0.7764706 ],
 [0.83137256, 0.827451  , 0.8117647 ]],

[[0.47843137, 0.46666667, 0.44705883],
 [0.4627451 , 0.45490196, 0.43137255],
 [0.47058824, 0.45490196, 0.43529412],
 ...,
 [0.7019608 , 0.69411767, 0.6784314 ],
 [0.6431373 , 0.6431373 , 0.63529414],
 [0.6392157 , 0.6392157 , 0.6313726 ]]], dtype=float32)
```

In [32]:
```
X_train.shape
```

Out[32]:
```
(50000, 32, 32, 3)
```

In [33]:
```
Input_shape = X_train.shape[1:]
```

In [34]:
```
Input_shape
```

Out[34]:
```
(32, 32, 3)
```

STEP #4: TRAIN THE MODEL

In [35]:
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D, Dense, Flatten, Dropout
from keras.optimizers import Adam
from keras.callbacks import TensorBoard
```

In [36]:
```
cnn_model = Sequential()
cnn_model.add(Conv2D(filters = 64, kernel_size = (3,3), activation = 'relu', input_shape = Input_shape))
cnn_model.add(Conv2D(filters = 64, kernel_size = (3,3), activation = 'relu'))
cnn_model.add(MaxPooling2D(2,2))
cnn_model.add(Dropout(0.4))


cnn_model.add(Conv2D(filters = 128, kernel_size = (3,3), activation = 'relu'))
cnn_model.add(Conv2D(filters = 128, kernel_size = (3,3), activation = 'relu'))
cnn_model.add(MaxPooling2D(2,2))
cnn_model.add(Dropout(0.4))

cnn_model.add(Flatten())

cnn_model.add(Dense(units = 1024, activation = 'relu'))

cnn_model.add(Dense(units = 1024, activation = 'relu'))

cnn_model.add(Dense(units = 10, activation = 'softmax'))
```

In [37]:
```
cnn_model.compile(loss = 'categorical_crossentropy', optimizer = keras.optimizers.rmsprop(lr = 0.001), metrics = ['accuracy'])
```

In [ ]:
```
history = cnn_model.fit(X_train, y_train, batch_size = 32, epochs = 1, shuffle = True)
```

```
Epoch 1/1
10912/50000 [=====>........................] - ETA: 5:32 - loss: 2.0551 - acc: 0.2424
```

STEP #5: EVALUATE THE MODEL

In [ ]:
```
evaluation = cnn_model.evaluate(X_test, y_test)
print('Test Accuracy: {}'.format(evaluation[1]))
```

In [ ]:
```
predicted_classes = cnn_model.predict_classes(X_test)
predicted_classes
```

In [ ]:
```
y_test
```

In [ ]:
```
y_test = y_test.argmax(1)
```

In [ ]:
```
y_test
```

In [ ]:
```
L = 7
W = 7
fig, axes = plt.subplots(L, W, figsize = (12, 12))
axes = axes.ravel()

for i in np.arange(0, L*W):
    axes[i].imshow(X_test[i])
    axes[i].set_title('Prediction = {}\n True = {}'.format(predicted_classes[i], y_test[i]))
    axes[i].axis('off')

plt.subplots_adjust(wspace = 1)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_test, predicted_classes)
cm
plt.figure(figsize = (10, 10))
sns.heatmap(cm, annot = True)
```

STEP #6: SAVING THE MODEL

In [ ]:
```
import os
directory = os.path.join(os.getcwd(), 'saved_models')

if not os.path.isdir(directory):
    os.makedirs(directory)
model_path = os.path.join(directory, 'keras_cifar10_trained_model.h5')
cnn_model.save(model_path)
```

STEP #7: IMPROVING THE MODEL WITH DATA AUGMENTATION

- Image Augmentation is the process of artificially increasing the variations of the images in the datasets by flipping, enlarging, rotating the original images.
- Augmentations also include shifting and changing the brightness of the images.

STEP 7.1 DATA AUGMENTATION FOR THE CIFAR-10 DATASET

In [ ]:
```
import keras
from keras.datasets import cifar10
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

In [ ]:
```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
```

In [ ]:
```
X_train.shape
```

In [ ]:
```
n = 8
X_train_sample = X_train[:n]
```

In [ ]:
```
X_train_sample.shape
```

In [ ]:
```
from keras.preprocessing.image import ImageDataGenerator

# dataget_train = ImageDataGenerator(rotation_range = 90)
# dataget_train = ImageDataGenerator(vertical_flip=True)
# dataget_train = ImageDataGenerator(height_shift_range=0.5)
dataget_train = ImageDataGenerator(brightness_range=(1,3))


dataget_train.fit(X_train_sample)
```

In [ ]:
```
from scipy.misc import toimage

fig = plt.figure(figsize = (20,2))
for x_batch in dataget_train.flow(X_train_sample, batch_size = n):
    for i in range(0,n):
        ax = fig.add_subplot(1, n, i+1)
        ax.imshow(toimage(x_batch[i]))
    fig.suptitle('Augmented images (rotated 90 degrees)')
    plt.show()
    break;
```

STEP 7.2 MODEL TRAINING USING AUGEMENTED DATASET

In [ ]:
```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
                            rotation_range = 90,
                            width_shift_range = 0.1,
                            horizontal_flip = True,
                            vertical_flip = True
                             )
```

In [ ]:
```
datagen.fit(X_train)
```

In [ ]:
```
cnn_model.fit_generator(datagen.flow(X_train, y_train, batch_size = 32), epochs = 2)
```

In [ ]:
```
score = cnn_model.evaluate(X_test, y_test)
print('Test accuracy', score[1])
```

In [ ]:
```
# save the model
directory = os.path.join(os.getcwd(), 'saved_models')

if not os.path.isdir(directory):
    os.makedirs(directory)
model_path = os.path.join(directory, 'keras_cifar10_trained_model_Augmentation.h5')
cnn_model.save(model_path)
```