This code implements a Convolutional Neural Network (CNN) model for image classification using the Keras library in Google Colab. The purpose of this script is to train a model that can classify images into two categories: cars and planes.

The code utilizes the Keras library for building and training the CNN model. It includes data preprocessing, model definition, training, and prediction steps. The image data is preprocessed and augmented using techniques such as scaling, shearing, zooming, and horizontal flipping.

The CNN architecture consists of convolutional layers for feature extraction, max pooling layers for spatial downsampling, batch normalization layers for normalization, activation layers for introducing non-linearity, dropout layers for regularization, and dense layers for classification.

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import BatchNormalization,Activation, Dropout, Flatten, Dense
from keras import backend as K


img_width, img_height = 224, 224

train_data_dir = '/content/drive/MyDrive/v_data/train'
validation_data_dir = '/content/drive/MyDrive/v_data/test'
nb_train_samples =400
nb_validation_samples = 100
epochs = 10
batch_size = 16


if K.image_data_format() == 'channels_first':
  input_shape = (3, img_width, img_height)
else:
  input_shape = (img_width, img_height, 3)


model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=input_shape))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(1))
model.add(Activation('sigmoid'))


model.compile(loss='binary_crossentropy',
            optimizer='rmsprop',
            metrics=['accuracy'])


train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
```

```
    class_mode='binary')

model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size)
```

```
Found 400 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
<ipython-input-28-e400c965663a>:21: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Pleas
  model.fit_generator(
Epoch 1/10
25/25 [==============================] - 12s 308ms/step - loss: 0.5617 - accuracy: 0.7750 - val_loss: 1.3682 - val_accuracy: 0.5000
Epoch 2/10
25/25 [==============================] - 6s 234ms/step - loss: 0.3569 - accuracy: 0.8325 - val_loss: 2.1127 - val_accuracy: 0.4792
Epoch 3/10
25/25 [==============================] - 6s 235ms/step - loss: 0.3036 - accuracy: 0.8800 - val_loss: 2.1126 - val_accuracy: 0.5104
Epoch 4/10
25/25 [==============================] - 8s 299ms/step - loss: 0.2500 - accuracy: 0.8975 - val_loss: 2.9917 - val_accuracy: 0.4896
Epoch 5/10
25/25 [==============================] - 7s 293ms/step - loss: 0.2304 - accuracy: 0.9125 - val_loss: 2.8600 - val_accuracy: 0.5104
Epoch 6/10
25/25 [==============================] - 6s 246ms/step - loss: 0.1569 - accuracy: 0.9450 - val_loss: 3.3552 - val_accuracy: 0.5104
Epoch 7/10
25/25 [==============================] - 7s 281ms/step - loss: 0.1818 - accuracy: 0.9200 - val_loss: 2.9283 - val_accuracy: 0.5104
Epoch 8/10
25/25 [==============================] - 6s 237ms/step - loss: 0.1661 - accuracy: 0.9400 - val_loss: 3.3366 - val_accuracy: 0.5104
Epoch 9/10
25/25 [==============================] - 7s 294ms/step - loss: 0.1298 - accuracy: 0.9500 - val_loss: 3.6237 - val_accuracy: 0.4896
Epoch 10/10
25/25 [==============================] - 6s 245ms/step - loss: 0.1293 - accuracy: 0.9650 - val_loss: 4.1880 - val_accuracy: 0.4896
<keras.callbacks.History at 0x790e046109d0>
```

```
import tensorflow as tf
model.save('model_saved.h5')
```

```
from keras.models import load_model
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
import numpy as np

from keras.models import load_model

model = load_model('/content/model_saved.h5')

image = load_img('/content/drive/MyDrive/v_data/test/planes/10.jpg', target_size=(224, 224))
img = np.array(image)
img = img / 255.0
img = img.reshape(1,224,224,3)
label = model.predict(img)
print("Predicted Class (0 - Cars , 1- Planes): ", round(label[0][0]))
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x790e0465feb0>
1/1 [==============================] - 0s 307ms/step
Predicted Class (0 - Cars , 1- Planes):  1
```

✓ 0s completed at 14:22