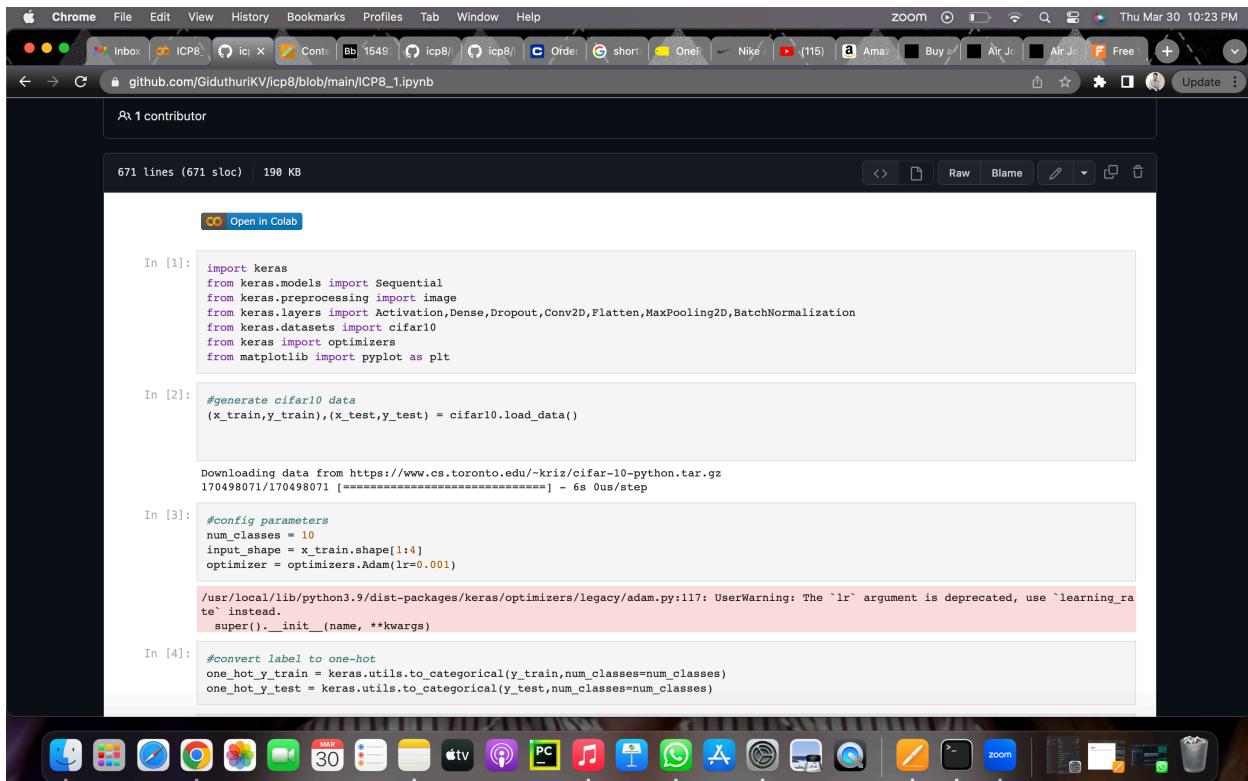


ICP 8

Krishna Vamsi G

700743211

Loaded the cifar10 dataset and split into train and test data with 80% train data and 20% test data



A screenshot of a Mac desktop showing a browser window with a Jupyter Notebook cell for loading the cifar10 dataset. The browser window title is "github.com/GiduthuriKV/icp8/blob/main/ICP8_1.ipynb". The notebook cell contains the following code:

```
import keras
from keras.models import Sequential
from keras.preprocessing import image
from keras.layers import Activation,Dense,Dropout,Conv2D,Flatten,MaxPooling2D,BatchNormalization
from keras.datasets import cifar10
from keras import optimizers
from matplotlib import pyplot as plt
```

The cell is labeled "In [1]".

```
#generate cifar10 data
(x_train,y_train),(x_test,y_test) = cifar10.load_data()
```

The cell is labeled "In [2]". A status message indicates the data is being downloaded from <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz>.

```
#config parameters
num_classes = 10
input_shape = x_train.shape[1:4]
optimizer = optimizers.Adam(lr=0.001)
```

The cell is labeled "In [3]". A warning message from the Adam optimizer is displayed: "/usr/local/lib/python3.9/dist-packages/keras/optimizers/legacy/adam.py:117: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead." The code continues with:

```
super().__init__(name, **kwargs)
```

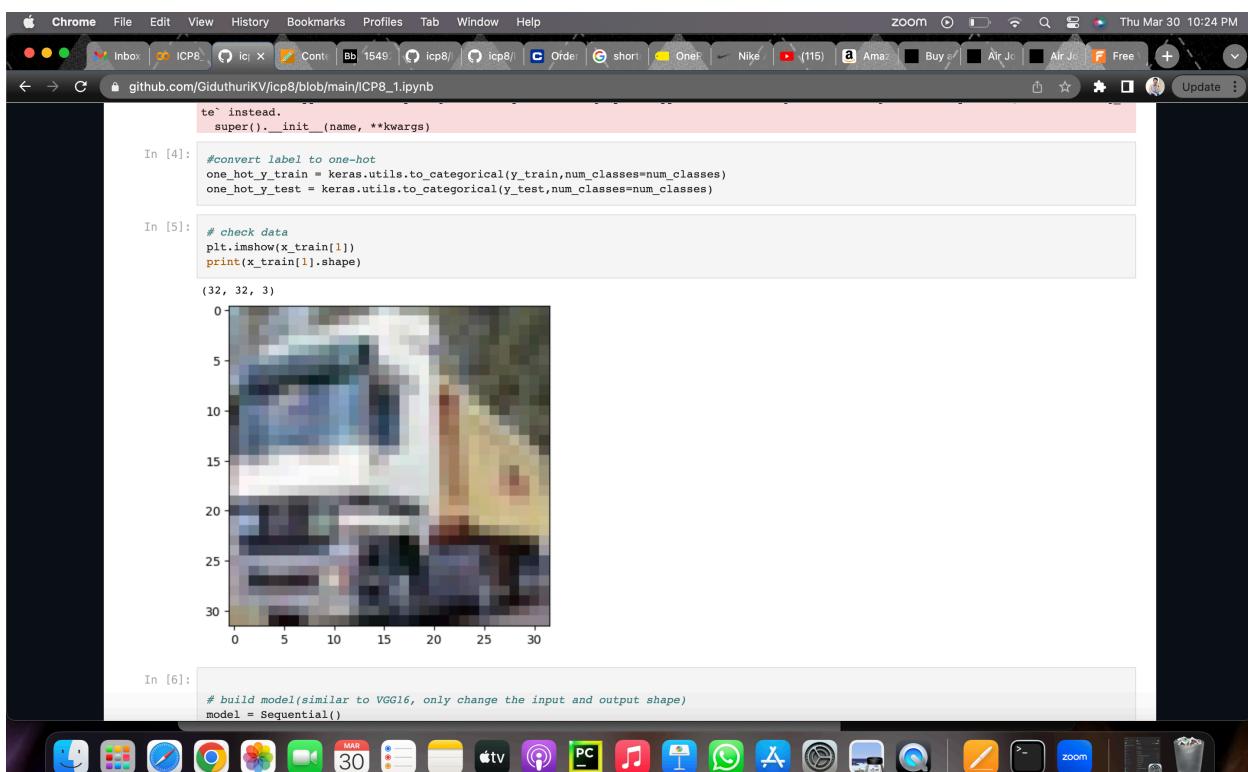
The cell is labeled "In [4]". The code converts the labels to one-hot encoding:

```
#convert label to one-hot
one_hot_y_train = keras.utils.to_categorical(y_train,num_classes=num_classes)
one_hot_y_test = keras.utils.to_categorical(y_test,num_classes=num_classes)
```

The cell is labeled "In [5]". The code checks the data shape and displays a plot of a single image:

```
# check data
plt.imshow(x_train[1])
print(x_train[1].shape)
```

The image plot shows a 32x32 pixel image of a car.



A screenshot of a Mac desktop showing a browser window with a Jupyter Notebook cell for displaying a cifar10 image and building a model. The browser window title is "github.com/GiduthuriKV/icp8/blob/main/ICP8_1.ipynb". The notebook cell contains the following code:

```
super().__init__(name, **kwargs)
```

The cell is labeled "In [4]". The code converts the labels to one-hot encoding:

```
#convert label to one-hot
one_hot_y_train = keras.utils.to_categorical(y_train,num_classes=num_classes)
one_hot_y_test = keras.utils.to_categorical(y_test,num_classes=num_classes)
```

The cell is labeled "In [5]". The code checks the data shape and displays a plot of a single image:

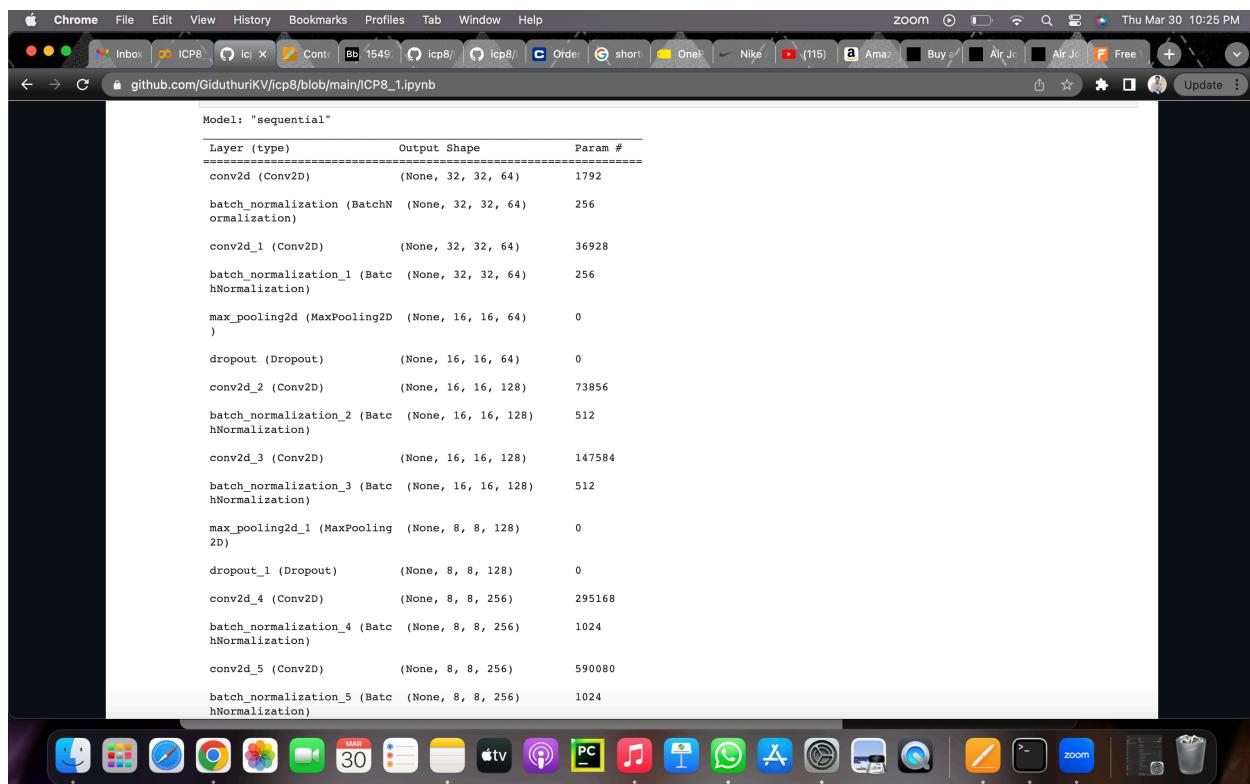
```
# check data
plt.imshow(x_train[1])
print(x_train[1].shape)
```

The image plot shows a 32x32 pixel image of a car.

The cell is labeled "In [6]". The code builds a Sequential model:

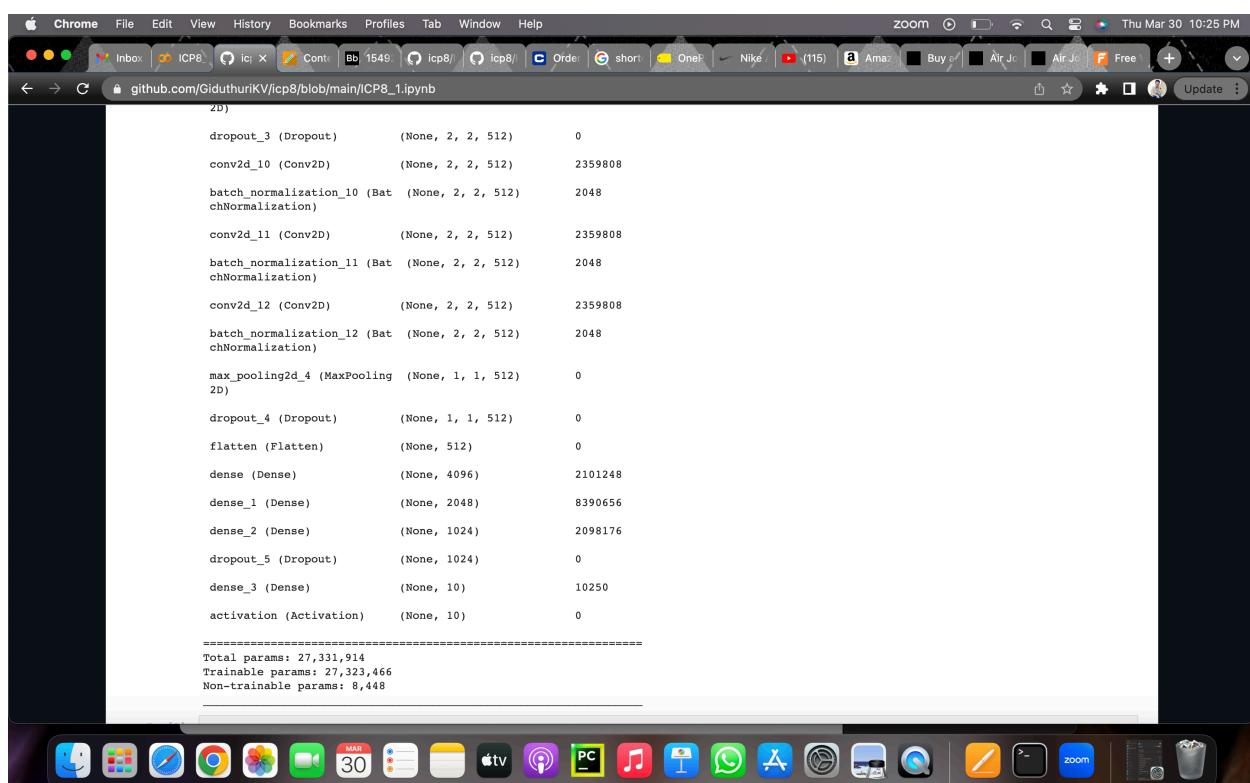
```
# build model(similar to VGG16, only change the input and output shape)
model = Sequential()
```

Now with to_categorical we are differentiating it into multiple categories.



Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	1792
batch_normalization (BatchN ormalization)	(None, 32, 32, 64)	256
conv2d_1 (Conv2D)	(None, 32, 32, 64)	36928
batch_normalization_1 (BatchN ormalization)	(None, 32, 32, 64)	256
max_pooling2d (MaxPooling2D)	(None, 16, 16, 64)	0
dropout (Dropout)	(None, 16, 16, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_2 (BatchN ormalization)	(None, 16, 16, 128)	512
conv2d_3 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_3 (BatchN ormalization)	(None, 16, 16, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_1 (Dropout)	(None, 8, 8, 128)	0
conv2d_4 (Conv2D)	(None, 8, 8, 256)	295168
batch_normalization_4 (BatchN ormalization)	(None, 8, 8, 256)	1024
conv2d_5 (Conv2D)	(None, 8, 8, 256)	590080
batch_normalization_5 (BatchN ormalization)	(None, 8, 8, 256)	1024



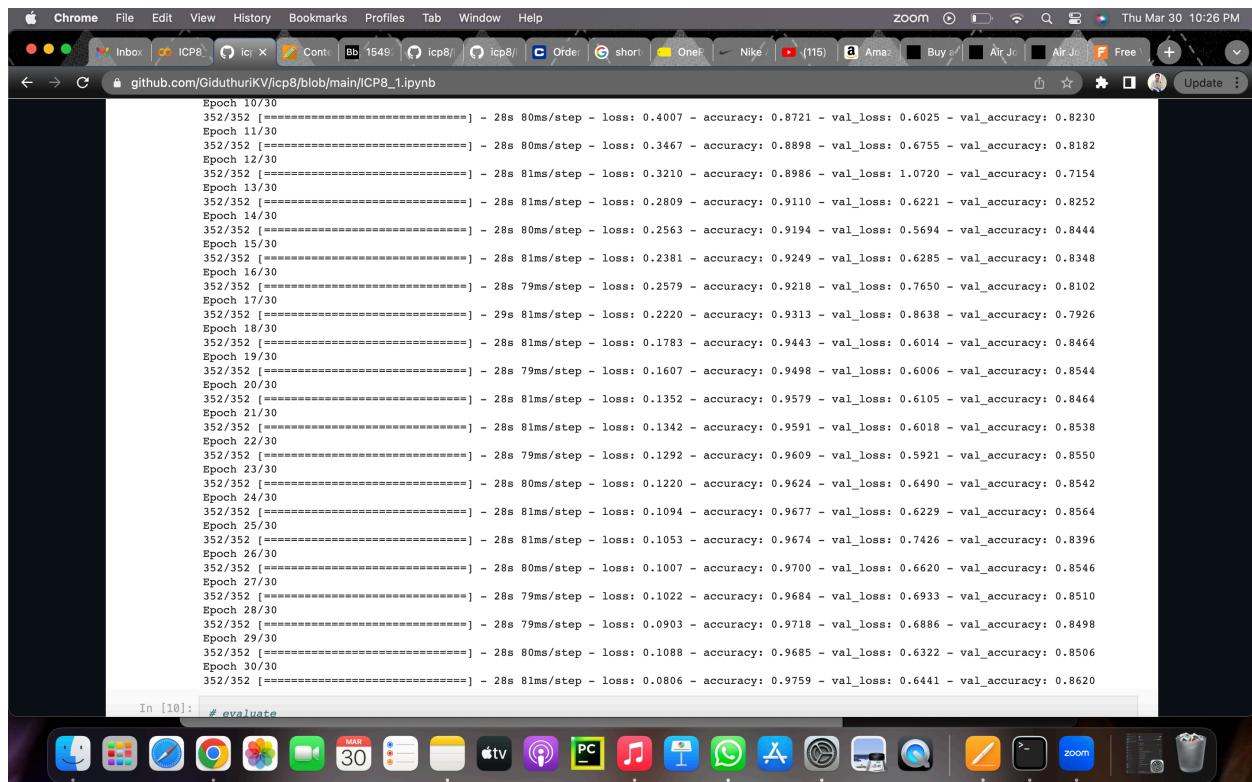
2D)

dropout_3 (Dropout)	(None, 2, 2, 512)	0
conv2d_10 (Conv2D)	(None, 2, 2, 512)	2359808
batch_normalization_10 (BatchN ormalization)	(None, 2, 2, 512)	2048
conv2d_11 (Conv2D)	(None, 2, 2, 512)	2359808
batch_normalization_11 (BatchN ormalization)	(None, 2, 2, 512)	2048
conv2d_12 (Conv2D)	(None, 2, 2, 512)	2359808
batch_normalization_12 (BatchN ormalization)	(None, 2, 2, 512)	2048
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 512)	0
dropout_4 (Dropout)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 4096)	2101248
dense_1 (Dense)	(None, 2048)	8390656
dense_2 (Dense)	(None, 1024)	2098176
dropout_5 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 10)	10250
activation (Activation)	(None, 10)	0

=====

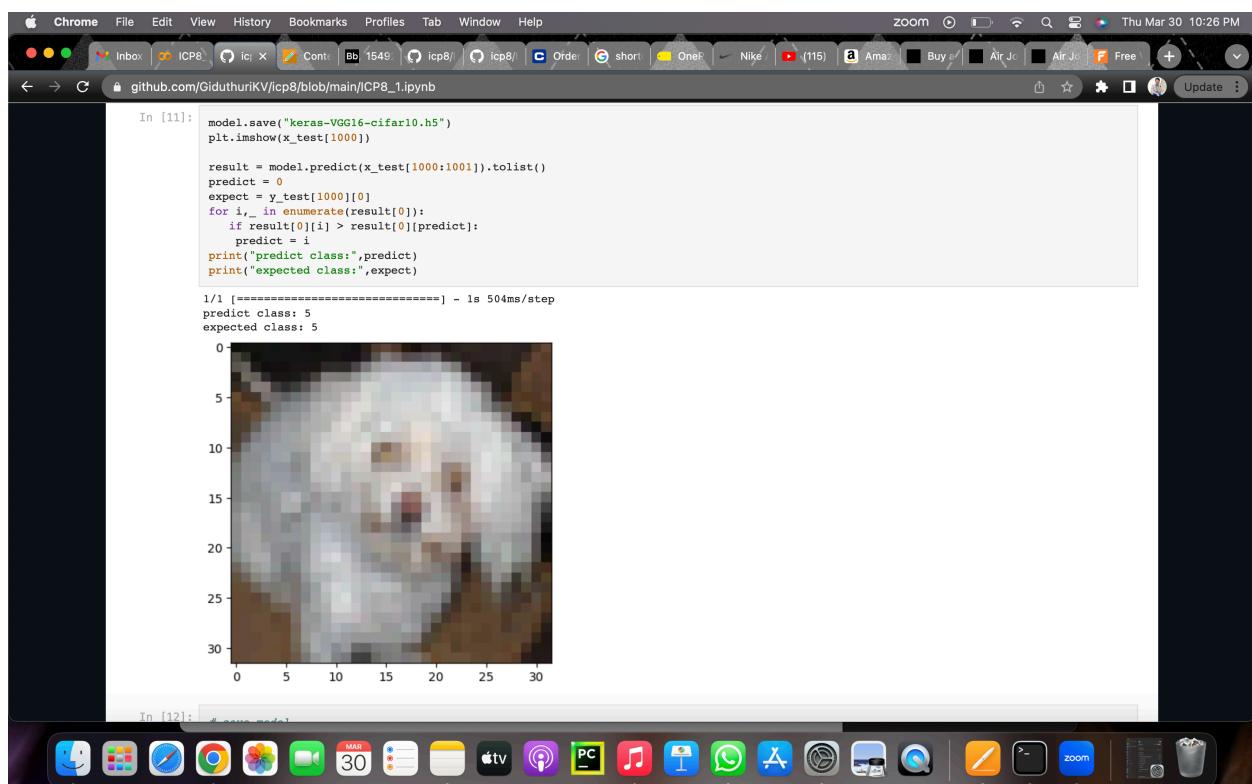
Total params: 27,331,914
Trainable params: 27,323,466
Non-trainable params: 8,448

Using ImageDataGenerator we are fit and train the obtained image data then created the Sequential model using the optimizers we just optimized the model to train fast and fit and trained



```
Epoch 10/30
352/352 [=====] - 28s 80ms/step - loss: 0.4007 - accuracy: 0.8721 - val_loss: 0.6025 - val_accuracy: 0.8230
Epoch 11/30
352/352 [=====] - 28s 80ms/step - loss: 0.3467 - accuracy: 0.8898 - val_loss: 0.6755 - val_accuracy: 0.8182
Epoch 12/30
352/352 [=====] - 28s 81ms/step - loss: 0.3210 - accuracy: 0.8986 - val_loss: 1.0720 - val_accuracy: 0.7154
Epoch 13/30
352/352 [=====] - 28s 81ms/step - loss: 0.2809 - accuracy: 0.9110 - val_loss: 0.6221 - val_accuracy: 0.8252
Epoch 14/30
352/352 [=====] - 28s 80ms/step - loss: 0.2563 - accuracy: 0.9194 - val_loss: 0.5694 - val_accuracy: 0.8444
Epoch 15/30
352/352 [=====] - 28s 81ms/step - loss: 0.2381 - accuracy: 0.9249 - val_loss: 0.6285 - val_accuracy: 0.8348
Epoch 16/30
352/352 [=====] - 28s 79ms/step - loss: 0.2579 - accuracy: 0.9218 - val_loss: 0.7650 - val_accuracy: 0.8102
Epoch 17/30
352/352 [=====] - 29s 81ms/step - loss: 0.2220 - accuracy: 0.9313 - val_loss: 0.8638 - val_accuracy: 0.7926
Epoch 18/30
352/352 [=====] - 28s 81ms/step - loss: 0.1783 - accuracy: 0.9443 - val_loss: 0.6014 - val_accuracy: 0.8464
Epoch 19/30
352/352 [=====] - 28s 79ms/step - loss: 0.1607 - accuracy: 0.9498 - val_loss: 0.6006 - val_accuracy: 0.8544
Epoch 20/30
352/352 [=====] - 28s 81ms/step - loss: 0.1352 - accuracy: 0.9579 - val_loss: 0.6105 - val_accuracy: 0.8464
Epoch 21/30
352/352 [=====] - 28s 81ms/step - loss: 0.1342 - accuracy: 0.9591 - val_loss: 0.6018 - val_accuracy: 0.8538
Epoch 22/30
352/352 [=====] - 28s 79ms/step - loss: 0.1292 - accuracy: 0.9609 - val_loss: 0.5921 - val_accuracy: 0.8550
Epoch 23/30
352/352 [=====] - 28s 80ms/step - loss: 0.1220 - accuracy: 0.9624 - val_loss: 0.6490 - val_accuracy: 0.8542
Epoch 24/30
352/352 [=====] - 28s 81ms/step - loss: 0.1094 - accuracy: 0.9677 - val_loss: 0.6229 - val_accuracy: 0.8564
Epoch 25/30
352/352 [=====] - 28s 81ms/step - loss: 0.1053 - accuracy: 0.9674 - val_loss: 0.7426 - val_accuracy: 0.8396
Epoch 26/30
352/352 [=====] - 28s 80ms/step - loss: 0.1007 - accuracy: 0.9700 - val_loss: 0.6620 - val_accuracy: 0.8546
Epoch 27/30
352/352 [=====] - 28s 79ms/step - loss: 0.1022 - accuracy: 0.9684 - val_loss: 0.6933 - val_accuracy: 0.8510
Epoch 28/30
352/352 [=====] - 28s 79ms/step - loss: 0.0903 - accuracy: 0.9718 - val_loss: 0.6886 - val_accuracy: 0.8498
Epoch 29/30
352/352 [=====] - 28s 80ms/step - loss: 0.1088 - accuracy: 0.9685 - val_loss: 0.6322 - val_accuracy: 0.8506
Epoch 30/30
352/352 [=====] - 28s 81ms/step - loss: 0.0806 - accuracy: 0.9759 - val_loss: 0.6441 - val_accuracy: 0.8620
```

In [10]: # evaluate



```
In [11]: model.save("keras-VGG16-cifar10.h5")
plt.imshow(x_test[1000])

result = model.predict(x_test[1000:1001]).tolist()
predict = 0
expect = y_test[1000][0]
for i, _ in enumerate(result[0]):
    if result[0][i] > result[0][predict]:
        predict = i
print("predict class:", predict)
print("expected class:", expect)

1/1 [=====] - 1s 504ms/step
predict class: 5
expected class: 5
```



In [12]: # model

And got the required results and plot the required graphs with the obtained results.

