

# **IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK**

**By,  
Sarikaa S,  
2018103058,  
19-04-2021.**

## Problem Statement:

To classify the CIFAR10 dataset into 10 different categories using convolutional neural networks.

## Dataset:

- CIFAR10 dataset contains 10 different categories of images .
- It consists of 50,000 32×32 color training images, labeled over 10 categories, and 10,000 test images.
- The dataset is divided into five training batches , each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain exactly 5000 images from each class.
- The classes are completely mutually exclusive. There is no overlap between automobiles and trucks.
- The class labels are:

Label	Description
0	airplane
1	automobile
2	bird
3	cat

4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

### Modules:

- Loading the cifar10 dataset into Keras.
- Reshaping the image array into [0,1] interval
- One hot encoding the labels
- Defining the model
- Feed the data to the network ,compile and fit the model
- Test the performance

### CNN model Summary :

```
In [10]: model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 30, 30, 32)	896
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	36928
conv2d_3 (Conv2D)	(None, 10, 10, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 128)	0
flatten (Flatten)	(None, 3200)	0
dense (Dense)	(None, 128)	409728
dense_1 (Dense)	(None, 10)	1290
=====		
Total params: 541,194		
Trainable params: 541,194		
Non-trainable params: 0		

## Loading the cifar10 dataset into Keras:

First Keras is installed and imported. The cifar10 dataset is loaded from the keras.datasets module.

```
pip install keras
```

```
import keras
```

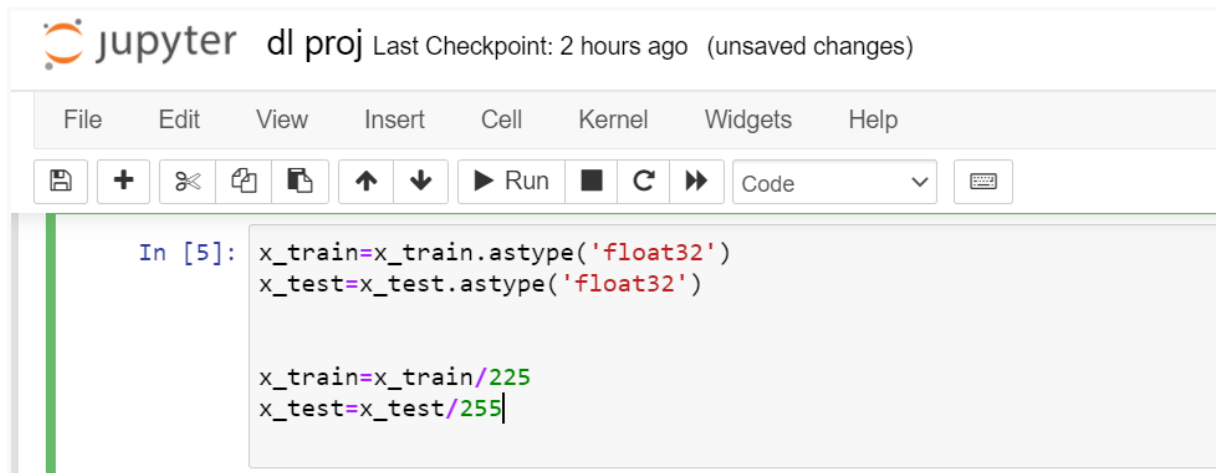
```
from keras.datasets import cifar10  
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
x_train.shape
```

```
(50000, 32, 32, 3)
```

## Reshaping the image array into [0,1] interval:

The image is a matrix of pixel values with RGB code for each pixel ranging from 0 to 255. First the image is converted to the integer values of the pixel to float and then divide the pixel values by the 255 .



The image shows a Jupyter Notebook interface. At the top, the title bar says "jupyter dl proj" followed by "Last Checkpoint: 2 hours ago (unsaved changes)". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Under the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and running code. The main area contains a code cell with the following Python code:

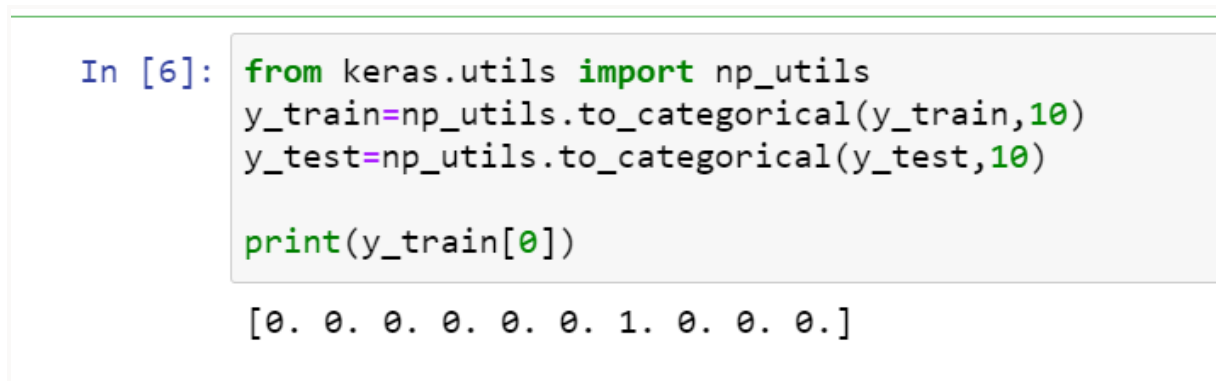
```
In [5]: x_train=x_train.astype('float32')
x_test=x_test.astype('float32')

x_train=x_train/225
x_test=x_test/255
```

### One hot encoding the labels:

One hot encoding is used to convert the integer value into a 10 channel one hot vector using `to_categorical()` function from `keras.utils`.

0: airplane,1: automobile,2: bird,3: cat,4: deer,5: dog,6: frog,7: horse,8: ship,9: truck



The image shows a Jupyter Notebook interface. The code cell contains the following Python code:

```
In [6]: from keras.utils import np_utils
y_train=np_utils.to_categorical(y_train,10)
y_test=np_utils.to_categorical(y_test,10)

print(y_train[0])
```

The output of the code is:

```
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```

The above output indicates frog.

### Defining the model:

```

In [8]: from keras.models import Sequential
        from keras.layers.convolutional import Conv2D, MaxPooling2D
        from keras.layers import Dense, Flatten, Dropout
        from keras.optimizers import Adam

In [9]: model=Sequential()
        model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(32,32,3), activation='relu'))
        model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
        model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        Dropout(0.4)
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        Dropout(0.5)
        model.add(Dense(10,activation='softmax'))

```

- First there are two conv layers ,one with 32 and other with 64 ,3x3 filters with activation function as relu.Followed by a maxpool of size 2x2.
- Then there are two conv layers, one with 64 and other with 128 filters of size 3x3 and activation function as relu.Followed by a maxpool of size 2x2 and a dropout of 0.4.
- Next in the flatten layer the data is converted to 1 dimension and fed to the dense layer with 128 neurons and activation function as relu.Followed by a dropout of 0.5.The last dense layer has 10 neurons because there are 10 different classes in the dataset with softmax as the activation function.

```

In [10]: model.summary()

Model: "sequential"

```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	36928
conv2d_3 (Conv2D)	(None, 10, 10, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 128)	0
flatten (Flatten)	(None, 3200)	0
dense (Dense)	(None, 128)	409728
dense_1 (Dense)	(None, 10)	1290

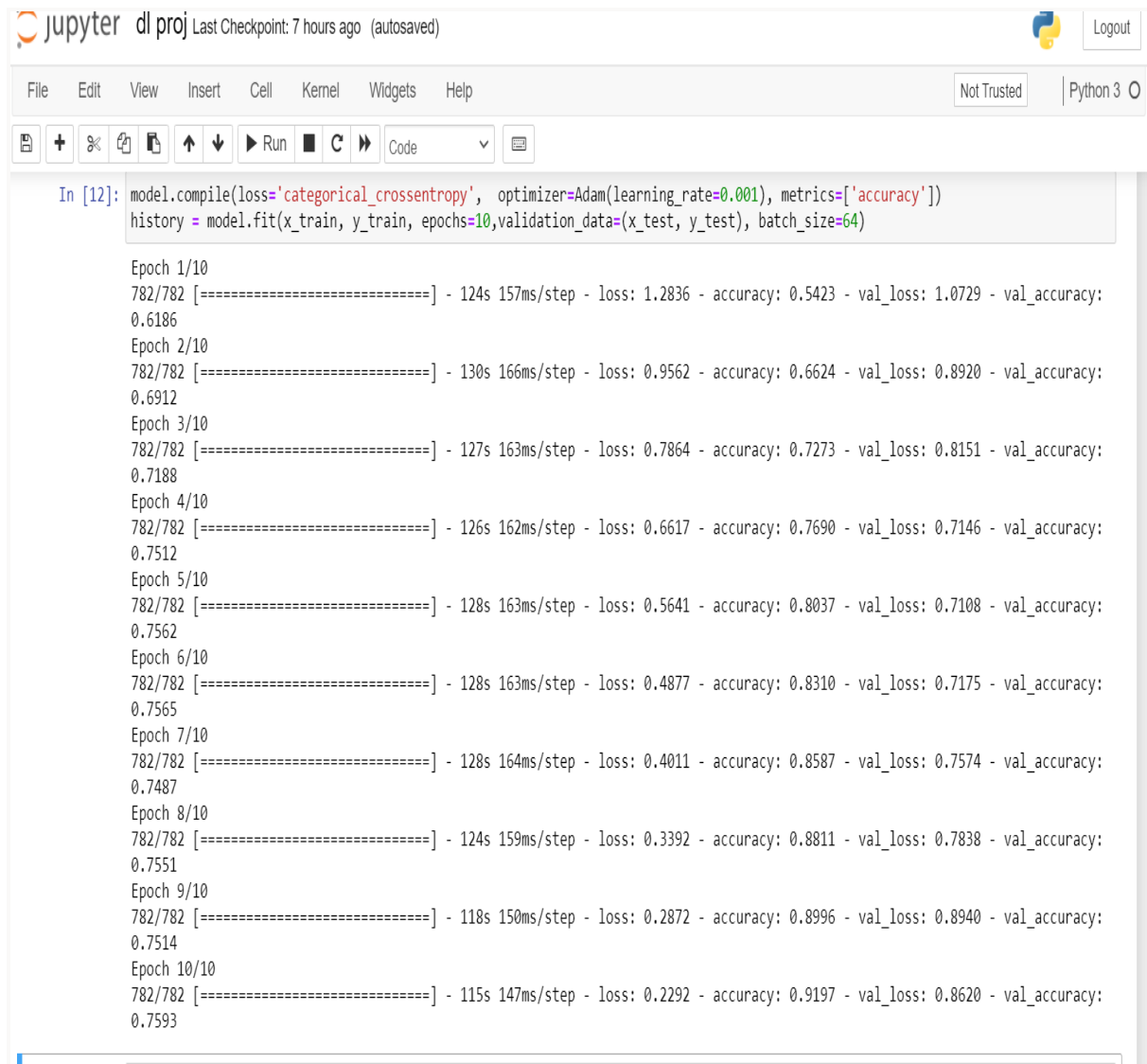
```

Total params: 541,194
Trainable params: 541,194
Non-trainable params: 0

```

## Feed the data to the network ,compile and fit the model:

Categorical cross entropy is used as the loss function for multiclass classification.Adam optimizer with learning rate 0.001 is used and the model is compiled.The model is fitted with batch size 64 and epochs 10.



The image shows a Jupyter Notebook interface with the following components:

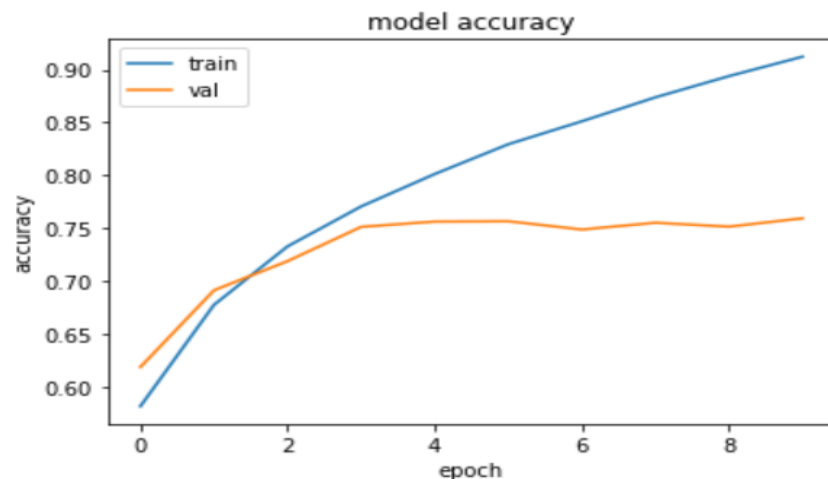
- Header:** "jupyter dl proj" with a "Last Checkpoint: 7 hours ago (autosaved)" status and a "Logout" button.
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Toolbar:** Includes icons for saving, running, and other notebook functions, along with a "Code" dropdown menu.
- Code Cell:** Contains the following Python code:

```
In [12]: model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test), batch_size=64)
```
- Output:** The output shows the training progress for 10 epochs. Each epoch displays the time taken (782/782), steps per second, loss, accuracy, validation loss, and validation accuracy. The metrics generally improve over the epochs.

Epoch	Time	Steps	Loss	Accuracy	Val Loss	Val Accuracy
Epoch 1/10	782/782	124s 157ms/step	1.2836	0.5423	1.0729	0.6186
Epoch 2/10	782/782	130s 166ms/step	0.9562	0.6624	0.8920	0.6912
Epoch 3/10	782/782	127s 163ms/step	0.7864	0.7273	0.8151	0.7188
Epoch 4/10	782/782	126s 162ms/step	0.6617	0.7690	0.7146	0.7512
Epoch 5/10	782/782	128s 163ms/step	0.5641	0.8037	0.7108	0.7562
Epoch 6/10	782/782	128s 163ms/step	0.4877	0.8310	0.7175	0.7565
Epoch 7/10	782/782	128s 164ms/step	0.4011	0.8587	0.7574	0.7487
Epoch 8/10	782/782	124s 159ms/step	0.3392	0.8811	0.7838	0.7551
Epoch 9/10	782/782	118s 150ms/step	0.2872	0.8996	0.8940	0.7514
Epoch 10/10	782/782	115s 147ms/step	0.2292	0.9197	0.8620	0.7593

## Test the performance:

```
In [13]: import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='best')
plt.show()
```



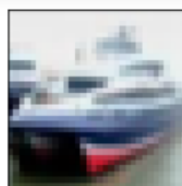
## Predictions:

For each image in the test set, the true class and predicted class is displayed.

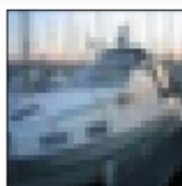
jupyter dl proj Last Checkpoint: 6 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code



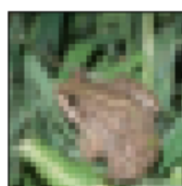
True: ship, Pred: ship



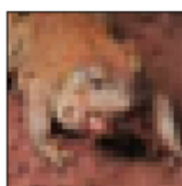
True: ship, Pred: ship



True: aeroplane, Pred: aeroplane



True: frog, Pred: frog



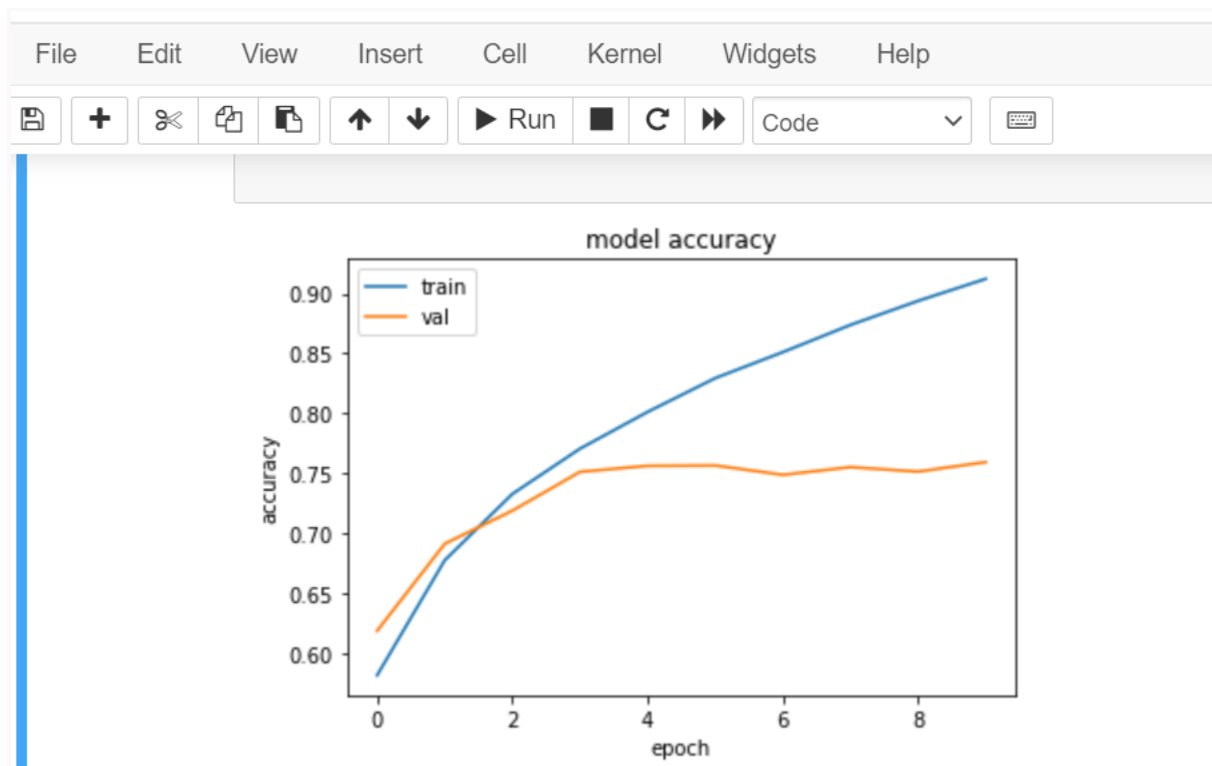
True: frog, Pred: frog



True: automobile, Pred: automobile



## Results:



First I trained the model with 20 epochs , the training accuracy was 95% and validation accuracy was 70%.The generalisation gap was large. Then I trained for 10 epochs the training accuracy was 91% and validation accuracy was 76%,now the generalisation gap is reduced compared to the 20 epochs.

Conclusion:

Thus the Convolutional neural network is built to classify the cifar10 image dataset.The model is built ,fitted and evaluated.

References:

- 1)[Datasets in Keras - GeeksforGeeks](#)
- 2)Deep Learning- Convolution Neural Network (CNN)-[Deep Learning- Convolution Neural Network \(CNN\) in Python | RP's Blog on Data Science \(datafai.com\)](#)
- 3)[Convolutional Neural Networks in Python - DataCamp](#)