**MAJOR PROJECT**

• **Project Name:**

Artificial Intelligence August Major Project

**• Project Description:**

Problem statement: Create a Deep learning model to predict the different CIFAR10 images.

The CIFAR-10 data consists of 60,000 32x32 color images in 10 classes, with 6000 images per class.

There are 50,000 training images and 10,000 test images in the official data. We have preserved the train/test split from the original dataset. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The classes are completely mutually exclusive. There is no overlap between automobiles and trucks.

"Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

**Dataset** – Dataset can be imported from tensorflow as follows

from tensorflow.keras.datasets import cifar10

(x\_train, x\_test),(y\_train,y\_test) = cifar10.load\_data()

**Details of datasets:**

Image size: 32\*32\*3

Color space: Colored

Number of classes: 10

Each training and test example is assigned to one of the following labels:

• airplane

• automobile

• bird

• cat

• deer

• dog

• frog

• horse

• ship

• truck

**Steps to consider:**

1. Normalize images by dividing pixels by 255 (if required)

2. Convert labels to categories (if required)

3. Reshape images so as to fit them to convolution

4. Build a CNN Architecture

5. Execute the model for appropriate number of epochs

6. Depict loss vs. val\_loss on line chart.

7. Depict accuracy vs. val\_accuracy on line chart.

8. Generate predictions on test\_data.

9. Compute Confusion matrix and classification report.

**SOLUTION:**

The first step towards writing any code is to import all the required libraries and modules. This includes importing tensorflow and other modules like numpy. If the module is not present then you can download it using pip install tensorflow on the command prompt (for windows) or if you are using a jupyter notebook then simply type !pip install tensorflow in the cell and run it in order to download the module. Other modules can be imported similarly.

Now we have the required module support so let’s load in our data. The dataset of CIFAR-10 is available on tensorflow keras API, and we can download it on our local machine using tensorflow.keras.datasets.cifar10 and then distribute it to train and test set using load\_data() function.

<https://drive.google.com/file/d/1jW3SoNYJRAZ6S-flcZhWIDhDzFcTaPdJ/view?usp=sharing>

**import** tensorflow **as** tf

**from** tensorflow.keras **import** datasets, layers, models

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

(X\_train, y\_train), (X\_test,y\_test) **=** datasets**.**cifar10**.**load\_data()

X\_train**.**shape

X\_test**.**shape

y\_train**.**shape

y\_train[:5]

y\_train **=** y\_train**.**reshape(**-**1,)

y\_train[:5]

y\_test **=** y\_test**.**reshape(**-**1,)

classes **=** ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]

**def** plot\_sample(X, y, index):

plt**.**figure(figsize **=** (15,2))

plt**.**imshow(X[index])

plt**.**xlabel(classes[y[index]])

plot\_sample(X\_train, y\_train, 0)

plot\_sample(X\_train, y\_train, 1)

X\_train **=** X\_train **/** 255.0

X\_test **=** X\_test **/** 255.0

ann **=** models**.**Sequential([

layers**.**Flatten(input\_shape**=**(32,32,3)),

layers**.**Dense(3000, activation**=**'relu'),

layers**.**Dense(1000, activation**=**'relu'),

layers**.**Dense(10, activation**=**'softmax')

])

ann**.**compile(optimizer**=**'SGD',

loss**=**'sparse\_categorical\_crossentropy',

metrics**=**['accuracy'])

ann**.**fit(X\_train, y\_train, epochs**=**5)

**from** sklearn.metrics **import** confusion\_matrix , classification\_report

**import** numpy **as** np

y\_pred **=** ann**.**predict(X\_test)

y\_pred\_classes **=** [np**.**argmax(element) **for** element **in** y\_pred]

print("Classification Report: \n", classification\_report(y\_test, y\_pred\_classes))

cnn **=** models**.**Sequential([

layers**.**Conv2D(filters**=**32, kernel\_size**=**(3, 3), activation**=**'relu', input\_shape**=**(32, 32, 3)),

layers**.**MaxPooling2D((2, 2)),

layers**.**Conv2D(filters**=**64, kernel\_size**=**(3, 3), activation**=**'relu'),

layers**.**MaxPooling2D((2, 2)),

layers**.**Flatten(),

layers**.**Dense(64, activation**=**'relu'),

layers**.**Dense(10, activation**=**'softmax')

])

cnn**.**compile(optimizer**=**'adam',

loss**=**'sparse\_categorical\_crossentropy',

metrics**=**['accuracy'])

cnn**.**fit(X\_train, y\_train, epochs**=**10)

cnn**.**evaluate(X\_test,y\_test)

y\_pred **=** cnn**.**predict(X\_test)

y\_pred[:5]

y\_classes **=** [np**.**argmax(element) **for** element **in** y\_pred]

y\_classes[:5]

y\_test[:5]

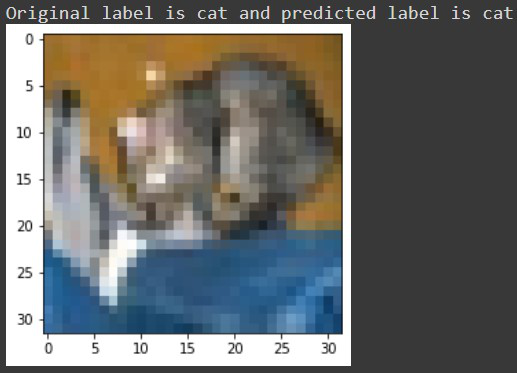
plot\_sample(X\_test, y\_test,3)

classes[y\_classes[3]]

classes[y\_classes[1]]

Let’s make a prediction over an image from our model using model.predict() function. Before sending the image to our model we need to again reduce the pixel values between 0 and 1 and change its shape to (1,32,32,3) as our model expects the input to be in this form only. To make things easy let us take an image from the dataset itself. It is already in reduced pixels format still we have to reshape it (1,32,32,3) using reshape() function. Since we are using data from the dataset we can compare the predicted output and original output.

**Output:**





**Deer**

**Presented**

**by**

**M KARISMA SATYA SAI LAKSHMI**

**JUJJUVARAPU PRATHIMA**

**TEAM-**[**AI-08-BSP6**](https://classroom.google.com/u/1/c/NTQ4NDU1MTg4Nzg5)