MAJOR PROJECT

Image Classification using CNN

DONE BY -V S DHARINEESH(vz7116@srmist.edu.in)
-PREETHI V(pg7541@srmist.edu.in)

Code Documentation for Python Code

Python
```
# import libraries
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
```

This code imports the necessary libraries for building and training a convolutional neural network (CNN) for image classification.

Python
```
# loading data set
datasets.cifar10.load_data()
```

This code loads the CIFAR-10 dataset, which is a popular dataset for image classification. The dataset contains 60,000 images of 10 different classes (airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks).

Python
```
# splitting the dataset
(X_train, y_train), (X_test, y_test) =
datasets.cifar10.load_data()

# check shape of test and train
X_train.shape, X_test.shape
```

This code splits the dataset into training and test sets. The training set contains 50,000 images, and the test set contains 10,000 images. The shape of each image is (32, 32,

3), where the first two dimensions are the height and width of the image, and the third dimension is the number of channels (RGB).

Python
```
# plot the figure
plt.figure(figsize=(15,2))
plt.imshow(X_train[1])
```

This code plots one of the training images.

Python
```
# check shape and content of y_train
y_train.shape, y_train[:5]
```

This code checks the shape and content of the training labels. The shape of the labels is (50000,), and the first five labels are [6, 9, 9, 4, 1].

Python
```
# reshape y_train
y_train = y_train.reshape(-1,)
y_train[:5]
```

This code reshapes the training labels to a one-dimensional vector.

Python
```
# label classes
classes = ["airplane", "automobile", "bird", "cat", "deer",
"dog", "frog", "horse", "ship", "truck"]
```

This code defines a list of the 10 class labels.

Python
```
# function to plot sample
def plot_sample(X, y, index):
 plt.figure(figsize=(15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[y[index]])
```

This function plots a sample image and its label.

Python
```
plot_sample(X_train, y_train, 3)
```

This code plots the training image at index 3 and its label.

Python
```
# normalise dataset
X_train = X_train/255
X_test = X_test/255
```

This code normalizes the training and test data by dividing each pixel by 255. This is a common practice when training neural networks, as it helps to improve the performance of the model.

Python
```
# Build simple artificial neural network for image
classification
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)
```

This code builds and trains a simple artificial neural network (ANN) for image classification. The ANN has three dense layers, with 3000, 1000, and 10 neurons, respectively. The input layer is flattened, so that the images are converted into one-dimensional vectors. The output layer has 10 neurons, one for each class. The ANN is trained using the stochastic gradient descent (SGD) optimizer and the sparse categorical crossentropy loss function. The accuracy metric is used to evaluate the performance of the model.

Python

```
# We can see that at the end of 5 epochs, accuracy is at around
49%
```

Project Source Code Jupyter Notebook Link

https://drive.google.com/file/d/1wTeU2gAF0BdE3PcL1A1gYk5jYMSYasQ6/view?usp=sharing

CONCLUSION

The Project has been implemented using all the instructions as given. The Project has been implemented successfully using the jupyter notebook platform and the results are shown appropriately for the provided dataset.