IC 272 - Data Science III

Assignment 5: FCNN and CNN

Deadline: November 10, 2024: 23.59 Hr.

Dataset Description:

You are given a zipped file "**cifar-3class-data.zip**" containing the Train and Test samples of three classes:

<u>Train - **1500 images** of the following classes:</u>

i. "`Class_0`": 500 RGB images of size 32 X 32. Each image contains a single aeroplane.

ii. "`Class_1`": 500 RGB images of size 32 X 32. Each image contains a single car.

iii. "`Class_2`": 500 RGB images of size 32 X 32. Each image contains a single bird.

<u>Test - **300 images** of the following classes :</u>

i. "`Class_0`": 100 RGB images of size 32 X 32. Each image contains a single aeroplane.

ii. "`Class_1`": 100 RGB images of size 32 X 32. Each image contains a single car.

iii. "`Class_2`": 100 RGB images of size 32 X 32. Each image contains a single bird.

* Use Keras for the implementation.

** The problems of this assignment can be solved in CPU mode of Google Colaboratory. Training the CNN may take at most one hour in this mode. FCNN training takes much less time than CNN.

Problem Statements:

A. FCNN for classifying an image as an aeroplane, car or bird -

I. Train an FCNN model using the **Train** samples and the following steps:

a. Write your **own** function to load the images from the **Train** dataset (Do not use any in-built function).
b. Print the size of the train set.
c. Visualise **any four** of the loaded train images using **matplotlib**.
d. Create a validation set containing only 10% images of the train set. Use the appropriate function from the **sci-kit learn** library.
e. Print the size of the new train set and the validation set.
f. Convert each of the 32 X 32 images (both in train and validation sets) into a vector of length 32*32 = 1024. Write your **own** lines of code and do not use any in-built function.
g. Normalise the vectorised images.
h. The architecture of the FCNN model should be as follows:
   - **Three** hidden layers having **256, 128,** and **64** neurons, respectively (The activation function for the hidden layers must be "**relu**").
   - The output layer that generates probabilities of a test image to belong to each of the given classes.

i. Train the model for 500 epochs with a batch size of 200. Use the loss function suitable for the classification task and "adam" as the optimization function.
j. Plot the epoch-wise training and validation accuracies in the same plot. The graph must have proper labels and legends.

II. Test the above model using the **Test** samples and the following steps:

a. Use your function, defined above to load the images from the **Test** dataset.
b. Convert each of the 32 X 32 images into a vector of length 32*32 = 1024. Reuse the codes you already have written during training..
c. Normalise the vectorised images (like as above).
d. Use these preprocessed vectorised images to evaluate the model.
e. Print the test loss and accuracy (%).

B. CNN for classifying an image as an aeroplane, car or bird -

I. Train a CNN model using the given Train samples and the following steps:

a. Load the images from the Train dataset using the function you already defined for FCNN task.

b. Create a validation set containing only 10% images of the train set as you have done for FCNN task.

c. Normalise the images (apply the same method as previos).

d. The architecture of the CNN model should be as follows:

- **Two** consecutive **convolutional** layers each having **64** filters of size **3 X 3.**
- **A maxpool** layer to **reduce the dimensions** of the feature maps (output of the second convolution layer) to **half.**
- **Two** consecutive **convolutional** layers each having **128** filters of size **3 X 3.**
- **A maxpool layer** to **reduce the dimensions** of the feature maps (output of the fourth convolution layer) to **half.**
- **A flattening** layer to vectorise the feature tensor (a matrix of more than two dimensions).
- **Two fully connected** layers having **512** and **100** neurons, respectively.
- One fully connected layer serving as the **output layer.**

NOTE:

*1. Activation functions for the layers should be "**relu**".*

*2. **Padding** for the convolutional layers must be "**same**" and "**valid**" for the maxpooling layers.*

e. Train the model for 50 epochs with a batch size of 200. Use the loss function suitable for the classification task and "adam" as the optimization function.

f. Save the model.

g. Plot the epoch-wise training and validation accuracies in the same plot. The graph must have proper labels and legends.

II. Test the above model using the Test samples and the following steps:

a. Use your function, defined above to load the images from the Test dataset.

b. Normalise the images (like as above).

c. Load the saved model.

d. Predict the class probabilities of the preprocessed images using the loaded model. The output of this step must be a 300 X 3 matrix, each row containing the three class probabilities for an test image.

f. Find the predicted class of an image as:

$$Class_{image} = \underset{class}{\arg\max} \quad Prob_{class}$$

g. Compute and display the confusion matrix using the predicted and actual class labels of the test images.

h. Print the test accuracy (%) of the model.