**Chart Image Classification using CNN**

(Develop a CNN based classification architecture for classifying a given chart image to one of five chart classes, namely “Line”,“Dot Line”,“HorizontalBar”,“Vertical Bar”, and “Pie” chart.)

**Aim:**

Convolutional neural network (CNN) modelling for image categorization is the goal of the code. Line charts, dot line charts, horizontal bar charts, vertical bar charts, and pie charts are among the types of charts included in the dataset used to train the model. A different test set of chart images is used to evaluate the model, and metrics like the classification report and confusion matrix are used to assess how well it performed. Additionally, use transfer learning to fine-tune a pretrained network and a pre-trained VGG16 model on a unique image dataset to achieve high accuracy in image classification.

**Introduction:**

To create a convolutional neural network (CNN), the code's goal is Importing the necessary libraries, such as NumPy, TensorFlow, Keras, Pandas, Matplotlib, OpenCV, and others, is the first step in the code. The locations of the training and test image folders as well as the CSV file containing the picture labels are then specified. Using data from chart images that are initially loaded from a folder directory and their related labels that are then loaded from a CSV file, the model is trained. The model is then tested using distinct chart images, labels, and their corresponding CSV files from distinct folder directories. To achieve high accuracy in picture categorization, a pre-trained model using the VGG16 convolutional neural network (CNN) architecture was used. Pre-trained weights are loaded into the pre-trained VGG16 model.

**The methodology of the code is as follows:**

* The relevant libraries, including TensorFlow, Keras, pandas, NumPy, Matplotlib, scikit-learn, and OpenCV, are imported into the code.
* The picture and CSV file directories' paths are specified.
* Using the os and PIL libraries, images and label data are loaded from the directory.
* The images are then resized to (224, 224) and transformed to a NumPy array, while the scikit-learn Label Encoder is used to turn the string labels into numerical labels.
* Using the np.save function from NumPy, the NumPy array and numerical labels are saved to a file in.npy format.
* Then, using NumPy's np.load function, the saved files are loaded into the array.
* The testing data is subjected to the same process again.
* The Matplotlib package is then used to visualise the supplied images.
* Using a dictionary, the string label categories are converted to number labels.
* The NumPy arrays are divided by 255 to normalise the photos.
* The training data's indices are divided into training and validation sets after being randomised.
* With convolutional layers, max pooling layers, dropout layers, and dense layers, a sequential model is produced using Keras.
* The model is then put together using an accuracy metric, an Adam optimizer, and a categorical cross-entropy loss function.
* With a batch size of 32 and 10 epochs, the model is then trained using the training data and validated using the validation data.
* The testing data are then used to evaluate the trained model, and a classification report and confusion matrix are printed.
* Using Matplotlib, the model's accuracy and loss during training and validation are shown.
* The layers of the loaded VGG16 model are frozen to prevent updates during training.
* The dataset is augmented with data to make it larger.
* The ImageDataGenerator class is used to generate the augmented data in batches.
* The training and testing data are used to fit the generators.
* To avoid overfitting, the model is trained using the fit approach with early halting.
* Plots of the training and validation accuracy show how well the model performs.

**Conclusion:**

The code shows how to create a CNN model for image categorization using the Keras library in conclusion. It demonstrates how to load and prepare picture data, establish the model architecture, train and assess the model, and assess the model's performance using several metrics. The model performs well in terms of accuracy when applied to the test set of chart images, demonstrating how well it can categorise various charts. Using transfer learning, the pre-trained VGG16 model successfully classified images on the unique dataset with high accuracy. By updating the final few layers of the model on the new dataset while maintaining the learnt weights in the first layers, the fine-tuning procedure boosts the model's performance. By extending the amount of the data set, the data augmentation strategy enhances the model's generalization skills.