**PC Parts Image Classification using CNN**

**Overview**

This document describes the process of building and training a Convolutional Neural Network (CNN) for image classification of PC parts. The model is designed to categorize images into different classes representing various PC components. The dataset is split into training and testing sets, and the model is trained using the TensorFlow and Keras frameworks.

Dataset link : https://www.kaggle.com/datasets/asaniczka/pc-parts-images-dataset-classification

**Data Preparation**

1. **Dataset Splitting:**
   * The original dataset containing PC part images is split into training and testing sets using the **split-folders** library.
   * An 80-20 ratio is maintained for training and testing data.
2. **Data Augmentation:**
   * Data augmentation is applied to the training dataset using the **ImageDataGenerator** to increase dataset diversity.
   * Augmentation techniques include rotation, width and height shifts, shear, zoom, and horizontal flips.
3. **Data Loading:**
   * Training and testing datasets are loaded using **flow\_from\_directory** with specified batch sizes and categorical class mode.

**CNN Model Architecture**

1. **Convolutional Neural Network (CNN):**
   * A sequential model is created with convolutional layers followed by max-pooling layers.
   * The architecture consists of four convolutional layers with increasing filters (32, 64, 128, 256) and max-pooling layers.
   * Flattening is performed to transition from convolutional layers to fully connected layers.
   * Two dense layers with 512 and 256 neurons, respectively, are added with ReLU activation.
   * The output layer has 14 neurons representing the classes with softmax activation.
2. **Model Compilation:**
   * The model is compiled with the Adam optimizer and categorical cross-entropy loss function.
   * Accuracy is chosen as the metric to monitor during training.

**Model Training**

1. **Training Process:**
   * The model is trained for 40 epochs on the training dataset with validation on the testing dataset.
   * Training history is stored for later analysis.
2. **Training Visualization:**
   * Training and validation accuracy curves are plotted to visualize the model's learning progress over epochs.

**Model Evaluation**

1. **Class Mapping:**
   * Class labels are mapped using the class indices obtained from the training dataset.
2. **Image Prediction:**
   * A function **predict\_img** is defined to predict the class of an input image.
   * The function loads, preprocesses, and predicts the class using the trained model.
3. **Example Image Prediction:**
   * An example image is provided to the **predict\_img** function, and the predicted class label is returned.

**Conclusion**

* + The CNN model is successfully trained for PC parts image classification.
  + Data augmentation is employed to enhance model generalization.
  + The model's accuracy is monitored and visualized during training.
  + Class mapping and image prediction functions are defined for practical use.