**DEEP LEARNING**

**ACTIVITY 2**

**PROJECT REPORT**

**JAIN UNIVERSITY**

**School of Sciences**

**Course:** B.Sc.(Honours) Data Science and Analytics

**Semester:** V

**Team members:**

Utsav Mondal(21BSR18018)

Sahana Kaluve (21BSR18019)

Isika Sen (21BSR18020)

Adithya Tej (21BSR18021)

**Title: Detailed Report on the Convolutional Neural Network (CNN) Model for Image Classification**

**1. Introduction:**

**The provided code implements a Convolutional Neural Network (CNN) model for image classification using TensorFlow and Keras. This report aims to provide a detailed overview of the key components and steps involved in this CNN model.**

**2. Data Preparation:**

**The model uses image data for training, validation, and testing.**

**The data is organized into separate directories for training and testing, each containing subdirectories for different classes.**

**TensorFlow's `image\_dataset\_from\_directory` function is used to create data pipelines for training, validation, and testing datasets.**

**Data augmentation techniques like random flips, rotations, and zooms are applied to enhance the dataset's diversity.**

**3. Model Architecture:**

**The model is built upon the Xception architecture, a pretrained model on the ImageNet dataset, which serves as a feature extractor.**

**Input images are resized to 224x224 pixels and preprocessed using the Xceptionspecific preprocessing function.**

**Data augmentation layers are added to the input pipeline to increase the model's robustness.**

**The Xception base model is followed by a global average pooling layer and a dropout layer to regularize the model.**

**The final dense layer with softmax activation is used for multiclass classification.**

**4. Transfer Learning:**

**The Xception base model's layers are frozen to retain its pretrained knowledge.**

**Transfer learning allows the model to leverage the feature extraction capabilities of Xception while finetuning the top layers for the specific task of image classification.**

**This approach is useful for improving training efficiency and achieving good performance with a smaller dataset.**

**5. Model Training:**

**The model is compiled with the Adam optimizer and sparse categorical crossentropy loss.**

**Training is performed for 10 epochs using the training dataset, with validation on a subset.**

**Training progress is monitored using metrics such as loss and accuracy.**

**After training, the model's performance is evaluated on the test dataset.**

**6. Evaluation:**

**The `model.evaluate` method is used to compute the model's accuracy and loss on the test dataset.**

**Predictions are generated for a batch of test images using `model.predict\_on\_batch`.**

**The predictions are converted into class labels by selecting the class with the highest probability.**

**The model's predictions are compared to the true labels, and the results are printed.**

**7. Gradio Interface:**

**The Gradio library is used to create a simple web interface for making predictions with the trained model.**

**Users can upload an image, and the model provides the top three predicted class labels along with their probabilities.**

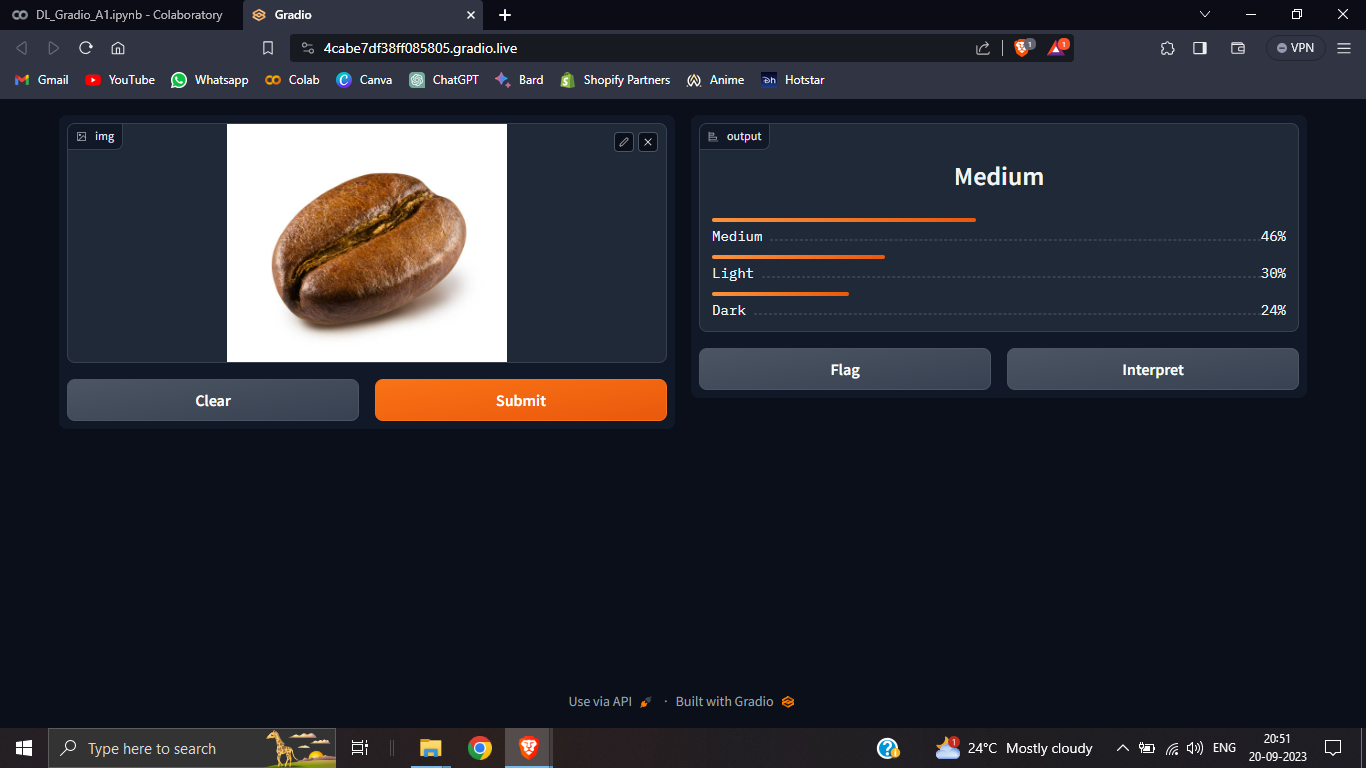
**8. Conclusion:**

**The provided CNN model demonstrates the process of building an image classification model using transfer learning.**

**Transfer learning with Xception allows the model to achieve good performance even with limited data.**

**Data augmentation techniques are applied to improve the model's ability to generalize.**

**The Gradio interface makes the model accessible to users for realtime predictions.**

****

**Project File:** **https://github.com/UTSAVMON/Gradio-DL.git**

**Overall, this CNN model serves as a practical example of image classification using deep learning techniques and showcases the use of transfer learning to leverage pretrained models for improved performance.**