1. Libraries and Modules Import

To begin the project, several essential libraries and modules are imported:

- os: For operating system functionalities.

- numpy: For numerical operations.

- tensorflow: For building and training the neural network.

- matplotlib: For plotting and visualizing data.

- sklearn: For preprocessing and evaluation.

- cv2: For image processing.

2. Dataset Directories and Parameters

- Train Directory: Specifies the path to the directory containing the training dataset.

- Path: `C:\Users\acer\OneDrive\Desktop\CAPSTONE\TRAIN_CAPSTONE`

- Test Directory: Specifies the path to the directory containing the testing dataset.

- Path: `C:\Users\acer\OneDrive\Desktop\CAPSTONE\TEST_CAPSTONE`

- Image Size (IMG_SIZE): Defines the dimensions to which all images will be resized (224x224 pixels).

- Batch Size (BATCH_SIZE): Number of images processed in one iteration during training.

- Number of Classes (NUM_CLASSES): Indicates the number of output categories, in this case, 'fire' and 'non-fire'.

3. Label Encoding

- Classes: The two categories of images:

- `fire_images_pre_done`

- `non_fire_images_pre_done`

- Label Encoder: Transforms categorical class labels (strings) into numerical labels for model training.

4. Data Augmentation

Data augmentation is used to artificially expand the training dataset by applying random transformations:

- Training Data Augmentation: Includes rescaling, rotations, shifts, shearing, zooming, and horizontal flips. This helps in creating a more robust model by exposing it to various transformations of the training images.

- Test Data Augmentation: Only involves rescaling to normalize the pixel values.

5. Data Generators

- Training Data Generator: Loads and preprocesses images in batches from the training directory, applying the specified augmentations.

- Test Data Generator: Loads and preprocesses images from the test directory without augmentation but with rescaling.

6. Model Definition and Compilation

- Base Model: EfficientNetB0, a state-of-the-art convolutional neural network architecture, is used here without pre-trained weights to train from scratch.

- Compile Model: The model is compiled with the Adam optimizer, categorical cross-entropy loss function, and accuracy as a metric. This prepares the model for training.

7. Training the Model

- Model Training: The model is trained on the augmented training dataset for 40 epochs, with the validation dataset used to monitor performance and avoid overfitting.

- Plot Training History: The accuracy and loss for both training and validation sets are plotted over the training epochs to visualize the model's learning progress.

8. Model Saving

- Save Model: After training, the model is saved in the specified directory. This allows for later use without the need to retrain.

9. Model Loading and Visualization

- Load Model: The saved model is loaded for inference.

- Prediction and Visualization: A function is defined to predict the class of a single image and visualize it with the predicted class label. This involves preprocessing the image, making a prediction, and displaying the image with its predicted label.

10. Confusion Matrix and Feature Maps

- Confusion Matrix: This is used to evaluate the model's performance by comparing predicted labels with true labels across the test dataset. It helps in understanding the model's accuracy and the types of errors it makes.

- Feature Maps Visualization: Feature maps from different layers of the model are visualized to gain insights into what features the model is learning at each stage. This helps in understanding the internal workings and representations formed by the neural network.

11. Sample Prediction

- Example Usage: The function for predicting single images is demonstrated with a sample image from the test dataset. This involves loading the image, preprocessing it, and using the loaded model to predict and visualize the result.