Problem Statement: "Ad Creative Recognition with Computer Vision"

1. Introduction

This Python script aims to perform image classification using deep learning techniques, incorporating data augmentation and transfer learning. The model is designed to classify images into two categories: "Ad creatives" and "Non-Ad creatives." The implementation is based on the VGG16 architecture, pre-trained on the ImageNet dataset, with fine-tuning for the specific classification task.

2. Data Augmentation

The script employs the Keras library's ImageDataGenerator for data augmentation. The augmentation parameters include rotation, width and height shift, zoom, and horizontal flip. Augmented images are generated to enhance the training dataset and improve the model's robustness.

3. Model Architecture

The model architecture consists of the following components:

- Base Model: VGG16 pre-trained on ImageNet, with the top classification layers removed.
- Flatten Layer: Flattens the output from the base model.
- Dense Layer (Hidden): Fully connected layer with 512 units and ReLU activation.
- Dense Layer (Output): Fully connected layer with 1 unit and sigmoid activation, representing the binary classification output.

The base model layers are set to be non-trainable, and the model is compiled using the Adam optimizer and binary crossentropy loss.

4. Training the Model

The model is trained using the fit method on the training data generated by ImageDataGenerator. The training process is configured for 20 epochs, and validation data is used to monitor model performance during training.

5. Model Evaluation

The model's performance can be assessed using metrics such as accuracy and loss. The training history, including these metrics for each epoch, is stored in the history variable.

6. Saving and Loading Models

The trained model is saved using the save method, creating a file named 'model_0.h5.' This file can be loaded later using the load_model function from Keras.

7. Inference and Classification

The script includes a function, classify_image(img_path), to classify new images using the trained model. This function loads an image, preprocesses it, and returns the predicted label (0 or 1) and corresponding classification ("Ad creatives" or "Non-Ad creatives").

8. Conclusion

This Python script demonstrates a comprehensive approach to image classification, incorporating data augmentation for improved model generalization and transfer learning for leveraging pre-trained models. The implemented model achieves classification results and can be extended to other binary classification tasks with minimal modification.