1. Gerekli kütüphanelerin içe aktarılması: `numpy`, `matplotlib.pyplot`, `tensorflow`, `ImageDataGenerator` ve sınıflandırma raporu ve karmaşıklık matrisi için `classification_report` ve `confusion_matrix` modülleri.
2. Veri yollarının ve hiperparametrelerin tanımlanması: Eğitim, doğrulama ve test veri setlerinin dosya yolları (`train_data_dir`, `validation_data_dir`, `test_data_dir`), sınıf sayısı (`num_classes`), toplu boyutu (`batch_size`) ve eğitim döngüsü sayısı (`epochs`) belirtilir.
3. Veri ön işleme ve genişletme için ImageDataGenerator nesnelerinin oluşturulması: `train_datagen` ve `validation_datagen`. Bu nesneler, veriye çeşitli dönüşümler ve normalizasyon işlemleri uygulayarak veri kümesini genişletir ve verileri modelin eğitimi için uygun hale getirir.
4. Veri yükleyicilerin oluşturulması: `train_datagen` ve `validation_datagen` nesneleri kullanılarak eğitim ve doğrulama veri kümelerini yükleyen `train_generator` ve `validation_generator` oluşturulur. Bu veri yükleyiciler, verileri belirtilen boyuta yeniden boyutlandırır ve ardışık toplu halinde getirir.
5. Önceden eğitilmiş bir MobileNet modelinin yüklenmesi: `MobileNet` sınıfı kullanılarak ImageNet veri kümesi üzerinde önceden eğitilmiş bir MobileNet modeli (`base_model`) oluşturulur. Bu modelin üzerine yeni tam bağlantılı katmanlar ekleyerek transfer öğrenme yapılacaktır.
6. Yeni tam bağlantılı katmanların eklenmesi: `base_model` çıktısı üzerine `GlobalAveragePooling2D`, `Dense` ve `Dropout` katmanları eklenir. Bu katmanlar, özellik çıkarımını gerçekleştirir ve sınıflandırma için uygun bir çıktı oluşturur.
7. Yeni modelin oluşturulması: Modelin giriş ve çıkışları belirtilerek `Model` sınıfından yeni bir model (`model`) oluşturulur.
8. Önceden eğitilmiş katmanların dondurulması: `base_model` içindeki tüm katmanlar eğitilemez (`trainable = False`) olarak ayarlanır, böylece sadece yeni eklenen katmanlar eğitilir.
9. Modelin derlenmesi: `Adam` optimize edici kullanılarak modelin derlenmesi yapılır. Kayıp fonksiyonu olarak "categorical_crossentropy" ve metrik olarak doğruluk seçilir.

10. Modelin eğitilmesi: `fit` fonksiyonu kullanılarak model eğitilir. `train_generator` ve `validation_generator` kullanılarak eğitim ve doğrulama veri kümeleri üzerinde belirtilen sayıda döngü (epoch) boyunca eğitim gerçekleştirilir.
11. Modelin test verileri üzerinde değerlendirilmesi: `test_generator` kullanılarak model test verileri üzerinde değerlendirilir. Tahminler alınır (`y_pred`) ve gerçek etiketler alınır (`y_true`).
12. Sınıflandırma raporu ve karmaşıklık matrisinin hesaplanması: `classification_report` ve `confusion_matrix` kullanılarak tahminlerin doğruluğu değerlendirilir ve sınıflandırma raporu ile karmaşıklık matrisi elde edilir.
13. Modelin ince ayar yapmak için yeniden derlenmesi: Tüm katmanlar tekrar eğitilebilir hale getirilir (`trainable = True`) ve model tekrar derlenir.
14. Modelin daha uzun süre eğitilmesi: `history_finetune` değişkeni kullanılarak model daha uzun süre eğitilir. Bu aşamada önceden eğitilmiş ağırlıklar daha fazla ayarlanır.
15. Modelin test verileri üzerinde yeniden değerlendirilmesi: İnce ayar sonrası model, test verileri üzerinde tekrar değerlendirilir ve kayıp (`test_loss`) ve doğruluk (`test_accuracy`) değerleri elde edilir.
16. Eğitim sürecinin kaybının ve doğruluğunun çizdirilmesi: `matplotlib.pyplot` kullanılarak eğitim sürecinin kaybı ve doğruluğu görselleştirilir. İlk grafikte eğitim ve ince ayar sonrası kayıp değerleri karşılaştırılırken, ikinci grafikte eğitim ve ince ayar sonrası doğruluk değerleri karşılaştırılır.