Image Preprocessing & Classification Using CNN

1. Introduction

The project aims to classify different types of footwear using Transfer Learning with MobileNetV2. Transfer learning leverages a pretrained model, which accelerates training and improves accuracy, especially with limited datasets. Strong data augmentation is applied to make the model more robust to variations in orientation, lighting, and scale.

2. Dataset Overview

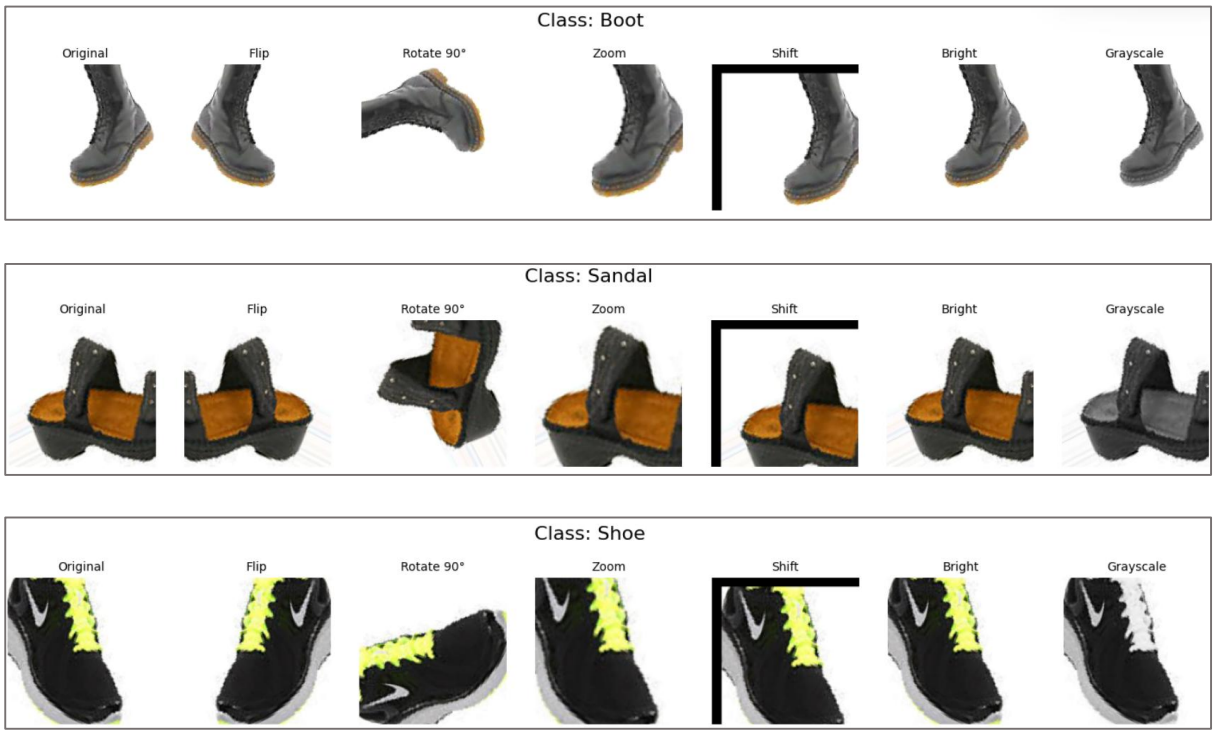
- Total images: ~6,000
- Classes: Shoe, Sandal, Boot.
- Dataset split: 80% training, 20% validation
- Organized in class-wise directories

3. Preprocessing and Data Augmentation

```
img_size = (160, 160)
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.3,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2],
    validation_split=0.2
)
```

4. Original vs Augmented Images

Visualize original image vs augmented images for each class.

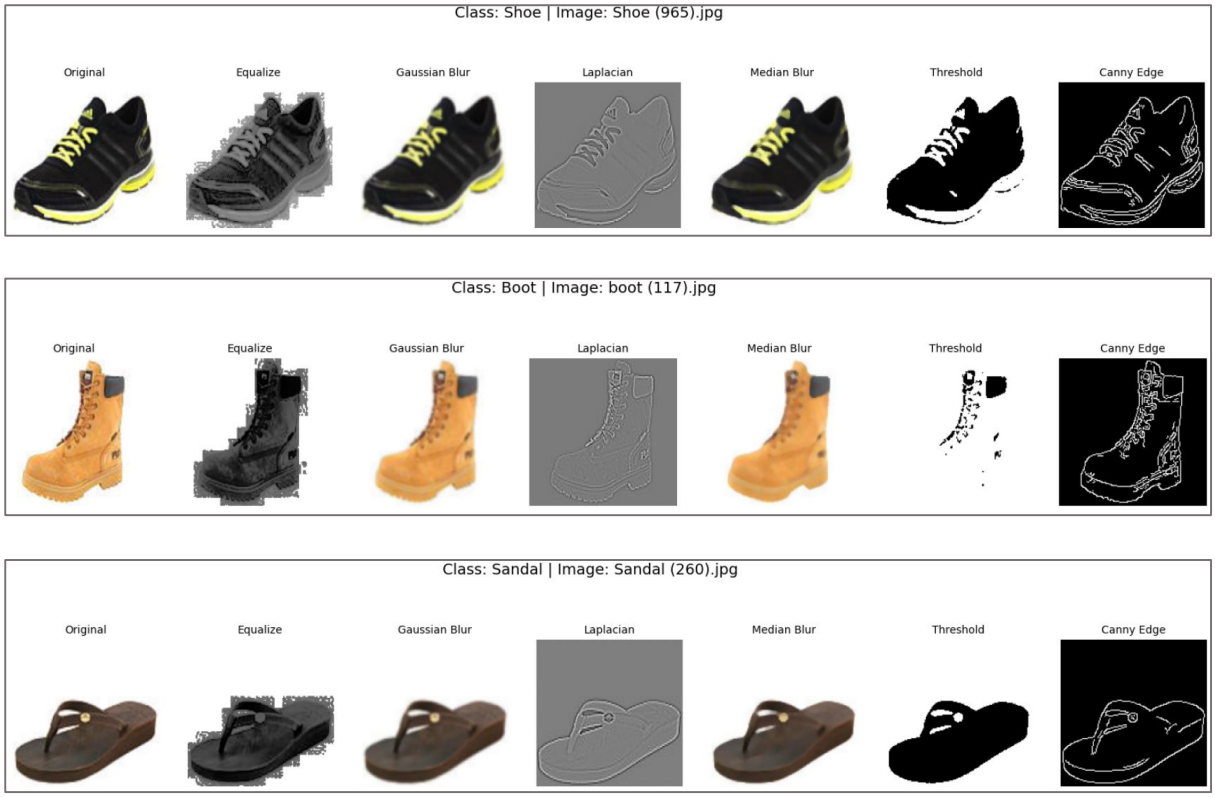


5. Image Filtering & Feature Extraction

```python
def apply_filters(img):
    filters = {
        "Original": img,
        "Equalize": cv2.equalizeHist(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)),
        "Gaussian Blur": cv2.GaussianBlur(img, (5,5), 0),
        "Laplacian": cv2.Laplacian(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), cv2.CV_64F),
        "Median Blur": cv2.medianBlur(img, 5),
        "Threshold": cv2.threshold(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), 127, 255, cv2.THRESH_BINARY)[1],
        "Canny Edge": cv2.Canny(img, 100, 200)
    }
    return filters
```

In this step, we applied multiple classical image processing filters on randomly selected images from each class in the dataset.This helps us analyze how different filters enhance features like edges, textures, brightness, and contrast, which may be useful for better feature extraction.

Output Example:



6. Model Architecture

```
base_model = MobileNetV2(weights="imagenet", include_top=False, input_shape=(160,160,3))

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(len(class_names), activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

The model uses MobileNetV2 pretrained on ImageNet with added GlobalAveragePooling2D, Dense(256, ReLU), Dropout(0.5), and Dense softmax layers. It is trained with Adam optimizer (1e-4 frozen, 1e-5 fine-tune) and categorical crossentropy loss.

7. Training & Fine-Tuning

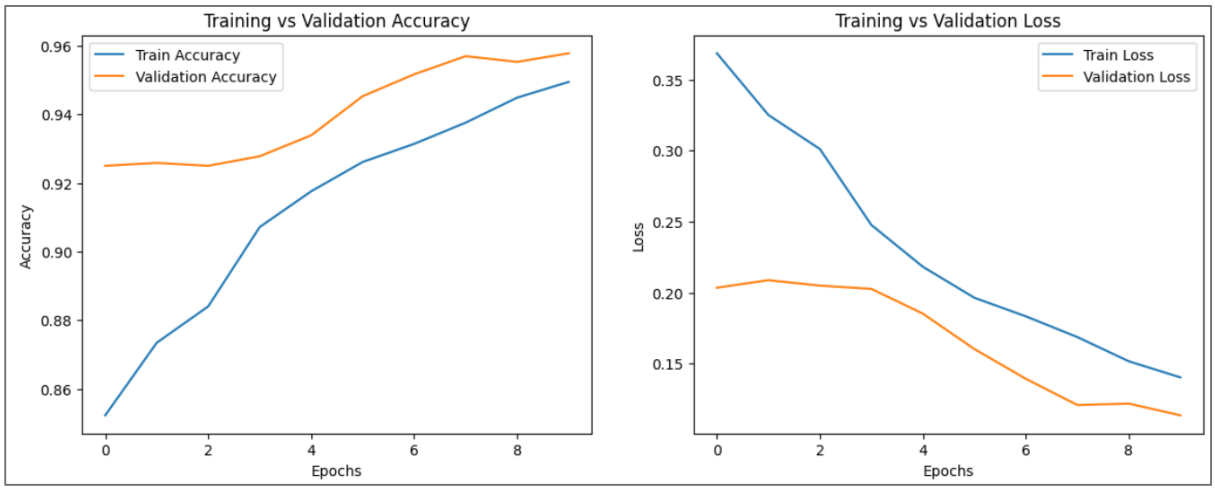
- **Initial Training (Frozen Base):** 10 epochs
- **Fine-Tuning (Unfreeze last layers):** 10 epochs

Final Accuracy:

```
print(f"Training Accuracy: {train_acc_percent:.2f}%")
print(f"Validation Accuracy: {val_acc_percent:.2f}%")
```

```
Training Accuracy: 95.13%
Validation Accuracy: 96.00%
```

8. Accuracy & Loss Curves



Observation:

- Both training and validation accuracy increase steadily.
- Loss decreases gradually, showing good convergence.
- No severe overfitting observed.

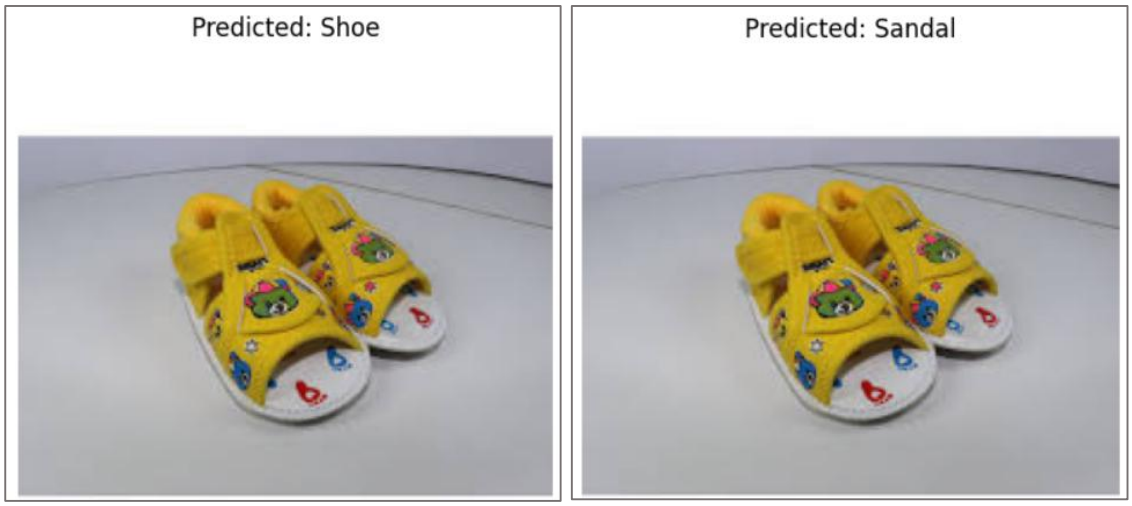
9. External Image Classification

```
test_img_path = "/content/5.jpg"

img = image.load_img(test_img_path, target_size=(128,128))
img_array = image.img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)

pred_prob = model.predict(img_array)
pred_class = np.argmax(pred_prob, axis=1)[0]
```

Before **After**



10. Conclusion

The MobileNetV2-based model successfully classifies different types of footwear with high accuracy. Strong data augmentation improved its robustness, and the training/validation curves show stable convergence without severe overfitting. Overall, the model generalizes well and can accurately predict unseen external images.