Mobile Phone Scratch Classification using Deep Learning

Objective

This project aims to develop a robust image classification model capable of categorizing images into one of three distinct classes based on the presented view of a mobile phone:

- Major_Scratch
- Minor_Scratch
- No_Scratch

This classification system serves as a foundational component for subsequent applications such as product tagging, automated cataloging, and quality control within e-commerce and electronics sectors.

Frameworks & Tools

- **Programming Language:** Python
- **Deep Learning Framework:** PyTorch
- **Development Environment:** Jupyter Notebook

Models Utilized

To fully leverage the benefits of transfer learning, a strategic selection of pre-trained convolutional neural network (CNN) models was undertaken. Each chosen model offers distinct architectural advantages that contribute to robust image classification performance across varying computational and deployment environments.Employed Models:

- **ResNet50:**
    - Deep residual architecture with "skip connections" to address vanishing gradient problem.
    - Facilitates training of very deep networks.
    - Captures intricate and hierarchical visual patterns.
    - Crucial for distinguishing subtle differences and achieving high accuracy.
    - Ideal for transfer learning due to generalizable features learned from ImageNet.

- **MobileNetV2:**
    - Lightweight and highly efficient design.
    - Based on inverted residual blocks with linear bottlenecks, reducing computational complexity and parameters.
    - Suitable for real-time applications, embedded systems, and mobile deployment (resource-constrained environments).
    - Excellent balance between performance and resource utilization.
    - Optimized for deployment on edge devices, enabling on-device inference without constant cloud connectivity.
- **WearNet**
    - Designed specifically for detecting and classifying surface scratches and wear on materials like mobile phones.
    - Captures features at multiple scales, enhancing sensitivity to both minor and major scratches.
    - Utilizes a shallower architecture with fewer parameters for faster inference and lower resource consumption.
    - Incorporates a specialized output layer optimized for distinguishing between no, minor, and major scratches.
    - Maintains high accuracy under varying lighting, backgrounds, and orientations.
    - Effectively identifies subtle and faint scratches that may be missed by generic CNNs.
    - Can be combined with powerful backbone networks (e.g., ResNet50, DenseNet121) for improved feature extraction.

Task Overview

This initiative constitutes a multi-class image classification task involving three target categories: front, back, and none.

Data Preparation

Class Definitions

- **Major Scratch:** This category encompasses images displaying a distinct frontal view of a mobile device, primarily highlighting the screen or user interface, where significant scratches are evident.
- **Minor Scratch:** This category includes images presenting a clear frontal view of a mobile device, with a focus on the screen or user interface, revealing minor scratches.
- **No Scratch:** This category comprises images depicting a clear frontal view of a mobile device, showcasing the screen or user interface without any visible defects.

Challenges Observed

- The model initially demonstrated proficiency in classifying 'Major_scratch' and No_scratch images due to their highly distinguishable features (e.g., the screen).

- However, difficulties arose in differentiating between major and minor classes, largely because some samples of the phones exhibited visual similarities.

Data Augmentation

Given the constrained dataset size, data augmentation played a pivotal role in mitigating overfitting and improving generalization. The following techniques were applied:

- Random rotations
- Horizontal flipping
- Brightness and contrast adjustments
- Cropping
- Hue and saturation shifts
- Normalization (based on ImageNet statistics for pre-trained models)

Dataset Overview

Number of Classes: 3

Classes: ['MAJORSCRACH', 'MINOR_SCRACH', 'NO_SCRACH']

Dataset Statistics:

- Total Samples: 304
- Training Samples: 243
- Validation Samples: 61

Class Distribution:

Class	Train	Validation	Ratio
MAJORSCRACH	80	20	4.00
MINOR_SCRACH	83	21	3.95
NO_SCRACH	80	20	4.00

Model Architecture & Training Strategy

Transfer Learning

Both ResNet50 and MobileNetV2 were initialized with ImageNet-pretrained weights. This methodology facilitates:

- Accelerated convergence
- Improved generalization on smaller datasets
- Leveraging pre-learned low-level features (e.g., edges, shapes)

Fine-Tuning Strategy

- **Frozen Early Layers:** The initial layers were kept frozen to preserve the pretrained visual features.
- **Unfrozen Top Layers:** The upper layers were unfrozen to allow adaptation to the specific mobile classification task.
- Ensemble Layer of the pretrained with Wearnet

Fine-Tuning Strategy

- **Frozen Early Layers:** The initial layers were maintained in a frozen state to preserve the integrity of the pretrained visual features.
- **Unfrozen Top Layers:** The upper layers were unfrozen to facilitate their adaptation to the specific mobile classification task.
- An ensemble layer was created by combining the pretrained model with Wearnet.

Custom Classifier Head

The final classification layer was replaced with a bespoke classifier head, comprising:

- AdaptiveAvgPool2d → Flatten
- Three Fully Connected (FC) Layers
    - FC → BatchNorm → PReLU → Dropout (0.5)
    - A final FC layer mapping to 3 output classes
- **PReLU Activation:** Enables the network to learn activation parameters, offering enhanced flexibility compared to traditional ReLU.
- **Dropout (p=0.5):** Implemented to prevent overfitting by randomly deactivating neurons during the training phase.
- **Batch Normalization:** Contributes to stable learning by normalizing feature distributions.

Class Imbalance Handling & Loss Function

Focal Loss

To address the observed class imbalance and enhance performance on challenging examples (particularly within the 'back' and 'none' classes), Focal Loss was employed:

$\text{Focal Loss} = (1 - p_t)^\gamma \cdot \text{CE}$`

Where:

- p_t represents the model's estimated probability for the true class
- γ (typically set to 2) diminishes the loss contribution from easily classified examples
- CE denotes the cross-entropy loss

Rationale for Focal Loss:

- It effectively down-weights well-classified samples.
- It directs the training focus towards more difficult, misclassified examples.

Optimization Strategy

Learning Rate Scheduler

- Utilized to decay the learning rate over time or based on a plateau in validation loss.
- Contributes to preventing overshooting and aids in escaping local minima.

Early Stopping

- Validation loss was continuously monitored.
- Training was halted when no significant improvement was observed, thereby preventing overfitting.

Summary of Improvements & Rationale

Component	Justification
Data Augmentation	Enhances generalization; crucial given the limited dataset size.
Transfer Learning	Leverages rich features from ImageNet, reducing training time and improving accuracy.
Custom Classifier	Tailored layers enable adaptation to task-specific patterns.
Focal Loss	Addresses class imbalance and focuses on challenging samples.
PReLU & BatchNorm	Improves learning flexibility and stability.
Early Stopping	Mitigates the risk of overfitting.

Learning Rate Scheduler	Promotes smoother convergence and facilitates better minima discovery.

Evaluation Strategy

Given the imbalanced nature of the dataset and the classification task, the Confusion Matrix is employed to calculate precision, recall, **F1-score,** and accuracy. This allows for a comprehensive understanding of the model's overall performance across categories, identifying both successful and failing classifications. These metrics are crucial for determining which model performs optimally on testing or unseen data.

Resnet 50:

Accuracy: 0.7705

Confusion Matrix:
[[18 2 0]
[4 13 4]
[0 4 16]]

Classification Report:

```None
              precision    recall  f1-score   support

Major_scrach       0.82      0.90      0.86        20
Minor_scrach       0.68      0.62      0.65        21
   no_scrach       0.80      0.80      0.80        20

    accuracy                           0.77        61
   macro avg       0.77      0.77      0.77        61
weighted avg       0.77      0.77      0.77        61
```

Mobilenet v2:

Accuracy: 0.8852

Confusion Matrix:
[[17 3 0]
[1 19 1]
[0 2 18]]

Classification Report:

```
None
          precision    recall  f1-score   support

Major_scrach       0.94      0.85      0.89        20
Minor_scrach       0.79      0.90      0.84        21
   no_scrach       0.95      0.90      0.92        20

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.89        61
weighted avg       0.89      0.89      0.89        61
```

WearNet and Densenet:

Accuracy: 0.8361

Confusion Matrix:

[[18 2 0]

[2 15 4]

[0 2 18]]

Classification Report:

```
None

              precision    recall  f1-score   support

Major_scrach       0.90      0.90      0.90        20
Minor_scrach       0.79      0.71      0.75        21
   no_scrach       0.82      0.90      0.86        20

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.83        61
```

WearNet and Resnet50:

Accuracy: 0.8361

Confusion Matrix:
[[17 3 0]
[0 17 4]
[0 3 17]]

Classification Report:

```
None
  precision    recall  f1-score   support

Major_scrach       1.00      0.85      0.92        20
Minor_scrach       0.74      0.81      0.77        21
   no_scrach       0.81      0.85      0.83        20

    accuracy                           0.84        61
   macro avg       0.85      0.84      0.84        61
weighted avg       0.85      0.84      0.84        61
```