SKIN CANCER CLASSIFICATION USING MOBILENET ARCHITECTURE

CS F425 **Deep Learning**

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

14th April 2025

Skin Cancer Classification Using MobileNet Architecture

SUBMITTED TO

Bharat Richariya

(Instructor-in-charge CS F425)

(BITS Pilani)

Prepared by

Atharva Agarwal **2022B3A70537P**

Shivansh Sharma **2022B3A70922P**

Birla Institute of Technology and Science

Pilani, Rajasthan

TABLE OF CONTENTS

INTRODUCTION

The objective of this project is to build a skin cancer classification system capable of distinguishing between the following seven classes:

1. AKIEC: Actinic keratoses and intraepithelial carcinoma/Bowen's disease
2. BCC: Basal cell carcinoma
3. BKL: Benign keratosis-like lesions
4. DF: Dermatofibroma
5. MEL: Melanoma
6. NV: Melanocytic nevi
7. VASC: Vascular lesions

The models are trained on a 3000 grayscale image dataset of size 300x300. By leveraging transfer learning and fine-tuning techniques, the objective is to build a robust classifier capable of distinguishing between different types of skin cancer with high accuracy. The report outlines the methodology, model architecture, training procedures, and performance evaluation of the implemented system.

PREPROCESSING AND DATA AUGMENTATION

PREPROCESSING:

In this section, we outline the preprocessing steps employed to prepare the dataset for our model. The primary goal of preprocessing was to enhance the dataset by increasing the number of training samples and to ensure uniformity in the input format.

Before feeding the images into the network, normalization is applied to scale pixel values to the range [0, 1]. This helps accelerate the training process and stabilizes gradient updates. Normalization also mitigates the effects of varying lighting conditions and contrast across images.

To ensure the model generalizes well and does not overfit the training data, additional preprocessing steps are applied, including:

- **Resizing and Rescaling**: Each image is resized to 300×300 pixels and rescaled by dividing pixel values by 255.
- **Channel Stacking**: Grayscale images are converted to three-channel format by repeating the single channel across the depth dimension.
- **Label Encoding**: Class names are mapped to integer labels and subsequently one-hot encoded for compatibility with categorical cross-entropy loss.

DATA AUGMENTATION:

Given the limited size and diversity of medical datasets, data augmentation is vital for improving the model's ability to learn invariant features and prevent overfitting. Real-time augmentation is performed using TensorFlow's `ImageDataGenerator` or `Sequential` preprocessing layers. The following augmentation techniques are applied during training:

- **Random Rotations**: Small rotations are introduced to simulate different lesion orientations.
- **Horizontal and Vertical Flips**: Images are randomly flipped to account for symmetrical variations in lesion appearance.
- **Zoom and Shift Transformations**: Random zooms and translations help the model become invariant to scale and positioning.
- **Brightness and Contrast Adjustments**: These modifications improve robustness to lighting variations in dermoscopic imaging.

To ensure a fair assessment of the model's actual performance, these augmentation approaches are only used on the training dataset. This prevents false noise from being introduced into the validation and test sets.

DATA VISUALIZATION:

The visualization process involves first fetching a batch of images from the training data loader. Since the images were previously normalized for model input, they are unnormalized before display to restore their original appearance. A helper function is used to convert the tensor

images into a format suitable for display with `matplotlib`. The `make_grid` function from the `torchvision.utils` module is used to arrange multiple images into a single composite image grid, making it easier to visually inspect multiple samples at once.

- A batch of images is retrieved using `next()` on the `train_loader` iterator
- The first 16 images from this batch are selected and arranged in a 4×4 grid.
- The images are unnormalized and displayed using `matplotlib`, with axes removed for clarity.

MODEL ARCHITECTURE 1: MOBILENETV3 - LARGE

MobileNetV3 is a state-of-the-art lightweight convolutional neural network designed for high performance on mobile and edge devices. It balances accuracy and efficiency by combining advanced architectural components like depthwise separable convolutions, Squeeze-and-Excitation (SE) modules, and non-linear activation functions.

MobileNetV3 comes in two variants—**Small** and **Large**—tailored for different resource constraints. The **Large** version is optimized for tasks requiring higher accuracy and provides a more expressive feature extractor. It includes:

- **Depthwise Separable Convolutions** to reduce computational cost.
- **Squeeze-and-Excitation (SE) blocks** for adaptive feature recalibration.
- **Hard-Swish (h-swish)** and **ReLU** activation functions.
- **Inverted Residual blocks** that allow gradients to propagate more efficiently in deep networks.

The final architecture for this task has been modified to suit the unique characteristics of the dataset—namely, grayscale dermoscopic images and a 7-class classification problem.

- **Input Layer:** The original MobileNetV3-Large model expects 3-channel RGB images therefore each grayscale image was converted to a 3-channel format by duplicating the

channel, resulting in an input size of 3×224×224. This allowed the use of pretrained ImageNet weights without modifying the model's initial convolutional layers.

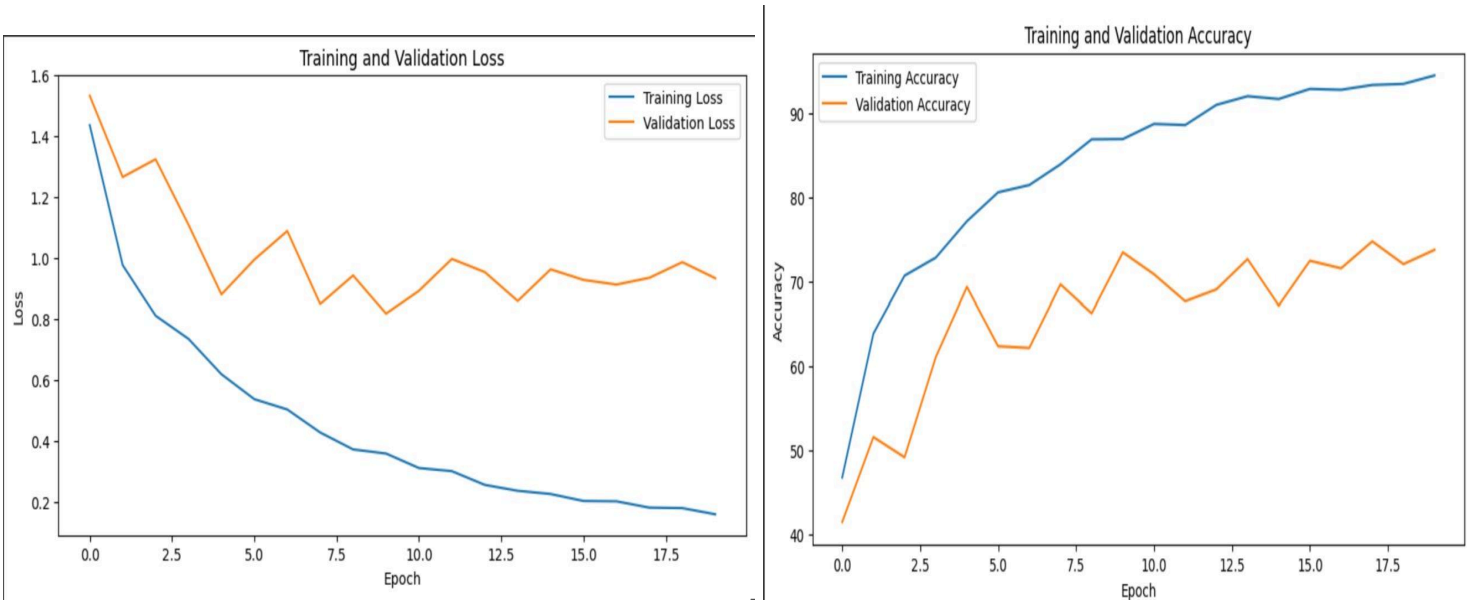
- The **feature extraction backbone** of the model consists of the standard MobileNetV3-Large layers, which include depth wise separable convolutions, inverted residual blocks, and squeeze-and-excitation (SE) modules. These components help reduce computational complexity while maintaining high representational power.
- **Final Classifier:** The classifier head of the model was modified by replacing the original 1000-class output layer with a new fully connected layer that outputs predictions for 7 classes. It now contains:
    - **Dropout Layer** (with p=0.2): Helps prevent overfitting by randomly disabling a fraction of neurons during training.
    - **Linear Layer (1280 → 7)**: Projects the extracted features into a 7-dimensional output vector, where each element corresponds to a class logit.
    - **Softmax Activation (applied during evaluation)**: Converts the output logits into class probabilities for prediction.
- **Loss Function and Optimization:** For training, CrossEntropyLoss was used as the loss function, appropriate for multi-class classification. The model was optimized using AdamW, and a WeightedRandomSampler was incorporated in the training data loader to handle class imbalance by giving underrepresented classes higher sampling probabilities.

The training process involved 25 epochs, during which the model's performance was monitored on a validation set to detect overfitting or underfitting. Learning rate was set to 0.0001. The batch size was fixed to 8 samples per batch.

RESULTS:

Our model MobileNetV3-Large achieved an accuracy of 94.53% on the training data with a training loss of 0.1584. In contrast, on the validation data, the model yielded an accuracy of 73.80% and a test loss of 0.9331.

```
Training started!
Epoch [1/20], Train Loss: 1.4351, Train Accuracy: 46.80%, Val Loss: 1.5319, Val Accuracy: 41.50%
Epoch [2/20], Train Loss: 0.9761, Train Accuracy: 63.90%, Val Loss: 1.2652, Val Accuracy: 51.60%
Epoch [3/20], Train Loss: 0.8099, Train Accuracy: 70.73%, Val Loss: 1.3234, Val Accuracy: 49.20%
Epoch [4/20], Train Loss: 0.7339, Train Accuracy: 72.87%, Val Loss: 1.1085, Val Accuracy: 61.10%
Epoch [5/20], Train Loss: 0.6179, Train Accuracy: 77.20%, Val Loss: 0.8801, Val Accuracy: 69.40%
Epoch [6/20], Train Loss: 0.5361, Train Accuracy: 80.63%, Val Loss: 0.9942, Val Accuracy: 62.40%
Epoch [7/20], Train Loss: 0.5022, Train Accuracy: 81.50%, Val Loss: 1.0881, Val Accuracy: 62.20%
Epoch [8/20], Train Loss: 0.4267, Train Accuracy: 83.97%, Val Loss: 0.8487, Val Accuracy: 69.70%
Epoch [9/20], Train Loss: 0.3711, Train Accuracy: 86.93%, Val Loss: 0.9423, Val Accuracy: 66.30%
Epoch [10/20], Train Loss: 0.3577, Train Accuracy: 86.97%, Val Loss: 0.8165, Val Accuracy: 73.50%
Epoch [11/20], Train Loss: 0.3101, Train Accuracy: 88.77%, Val Loss: 0.8920, Val Accuracy: 70.90%
Epoch [12/20], Train Loss: 0.2998, Train Accuracy: 88.63%, Val Loss: 0.9964, Val Accuracy: 67.70%
Epoch [13/20], Train Loss: 0.2548, Train Accuracy: 91.03%, Val Loss: 0.9532, Val Accuracy: 69.10%
Epoch [14/20], Train Loss: 0.2354, Train Accuracy: 92.07%, Val Loss: 0.8586, Val Accuracy: 72.70%
Epoch [15/20], Train Loss: 0.2249, Train Accuracy: 91.73%, Val Loss: 0.9622, Val Accuracy: 67.20%
Epoch [16/20], Train Loss: 0.2019, Train Accuracy: 92.93%, Val Loss: 0.9275, Val Accuracy: 72.50%
Epoch [17/20], Train Loss: 0.2009, Train Accuracy: 92.83%, Val Loss: 0.9122, Val Accuracy: 71.60%
Epoch [18/20], Train Loss: 0.1801, Train Accuracy: 93.40%, Val Loss: 0.9345, Val Accuracy: 74.80%
Epoch [19/20], Train Loss: 0.1785, Train Accuracy: 93.53%, Val Loss: 0.9857, Val Accuracy: 72.10%
Epoch [20/20], Train Loss: 0.1584, Train Accuracy: 94.53%, Val Loss: 0.9331, Val Accuracy: 73.80%
```

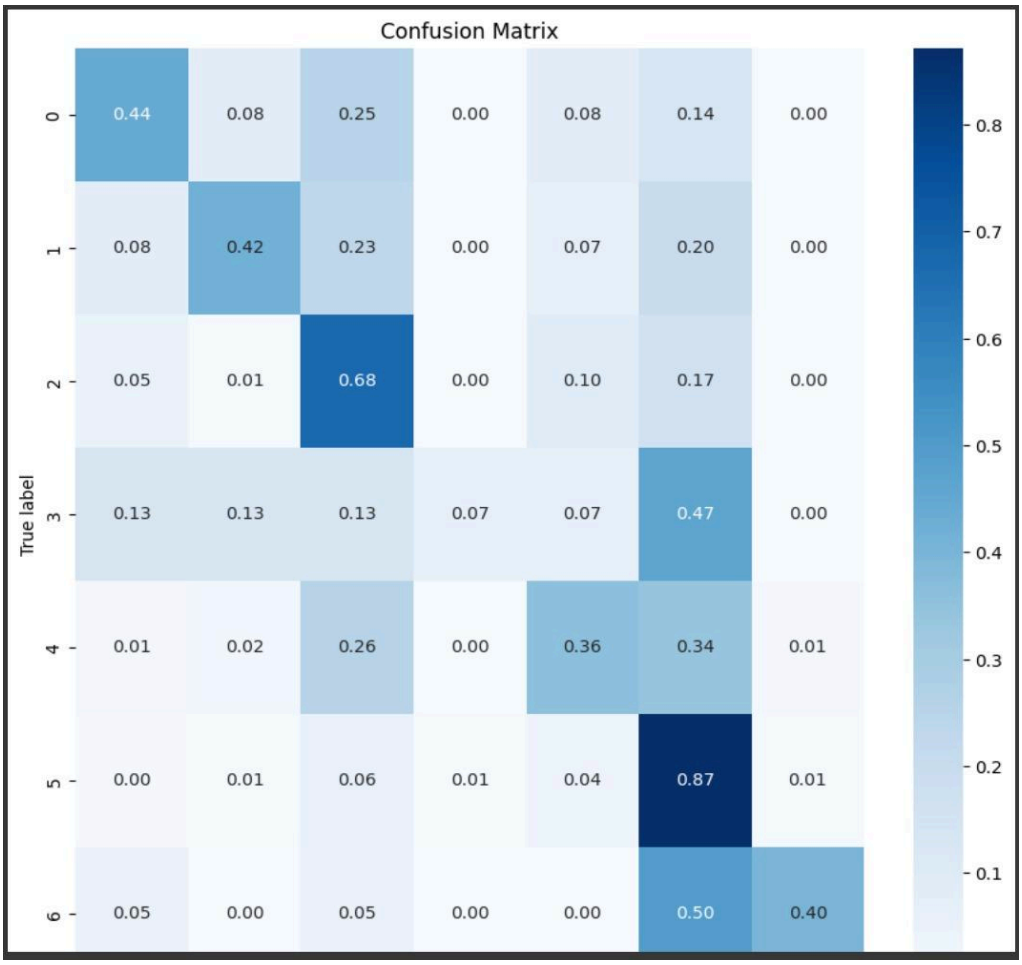


Training and Validation loss and Accuracies plots vs Epochs

```
Validation Accuracy: 73.80%
              precision    recall  f1-score   support

           0       0.48      0.44      0.46        36
           1       0.64      0.42      0.51        60
           2       0.44      0.68      0.53       102
           3       0.14      0.07      0.09        15
           4       0.43      0.36      0.39        97
           5       0.87      0.87      0.87       670
           6       0.53      0.40      0.46        20

    accuracy                           0.74      1000
   macro avg       0.51      0.46      0.47      1000
weighted avg       0.74      0.74      0.73      1000
```



The Confusion Matrix shows the performance of a classification model by comparing actual versus predicted labels.

```
Accuracy for class 0: 0.44
Accuracy for class 1: 0.42
Accuracy for class 2: 0.68
Accuracy for class 3: 0.07
Accuracy for class 4: 0.36
Accuracy for class 5: 0.87
Accuracy for class 6: 0.40
```

Accuracies for each Classes

MODEL ARCHITECTURE 2: MOBILENETV2 WITH HARD DATA MINING

MobileNetV2 is a lightweight and efficient convolutional neural network developed for mobile and embedded applications, characterized by its use of **inverted residuals** and **linear bottlenecks**.

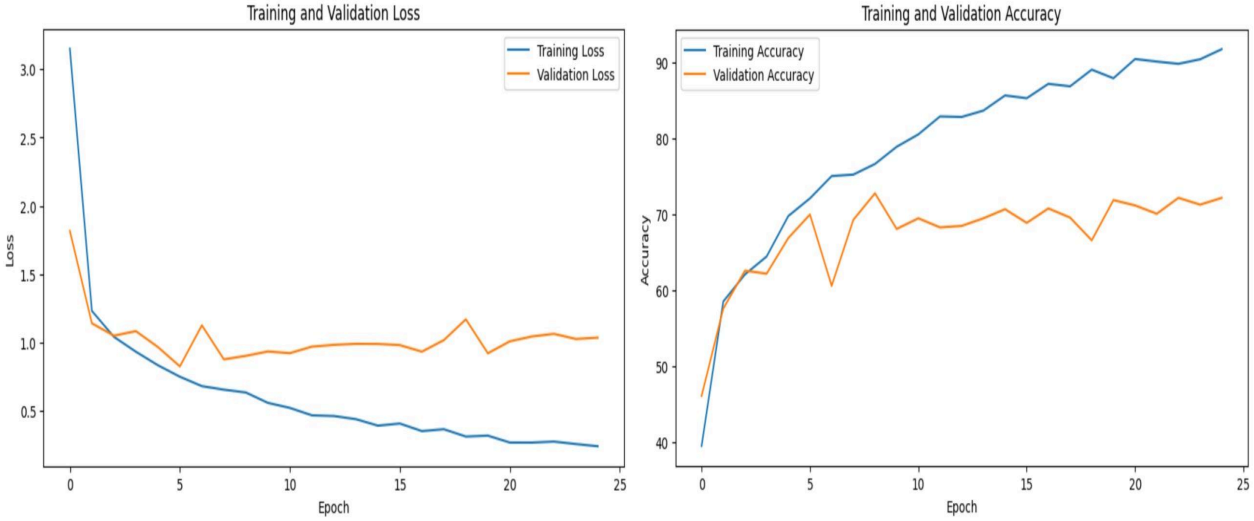
The final architecture for this task has been modified to suit the unique characteristics of the dataset—namely, grayscale dermoscopic images and a 7-class classification problem.

- **Input Layer:** Similar to MobileNetV3-Large. Input images are grayscale, but are converted to 3-channel images using replication to match the expected input format of MobileNetV2. Images are resized to 224×224 pixels.
- **Feature Extraction:** This model is composed of depthwise separable convolutions and inverted residual blocks with linear bottlenecks. During training, these outputs are interpreted as raw class logits, and a softmax function is applied during evaluation to convert them into class probabilities.
- **Loss Function and Optimization:** The model is trained using the cross-entropy loss function, which is standard for multi-class classification problems. Optimization is performed using the AdamW optimizer with a learning rate of 0.0001, which offers the benefits of adaptive learning along with decoupled weight decay for improved generalization.

RESULTS FOR MOBILENETV2:

Our model MobileNetV2 achieved an accuracy of 91.80% on the training data with a training loss of 0.2408. In contrast, on the validation data, the model yielded an accuracy of 72.20% and a test loss of 1.0359.

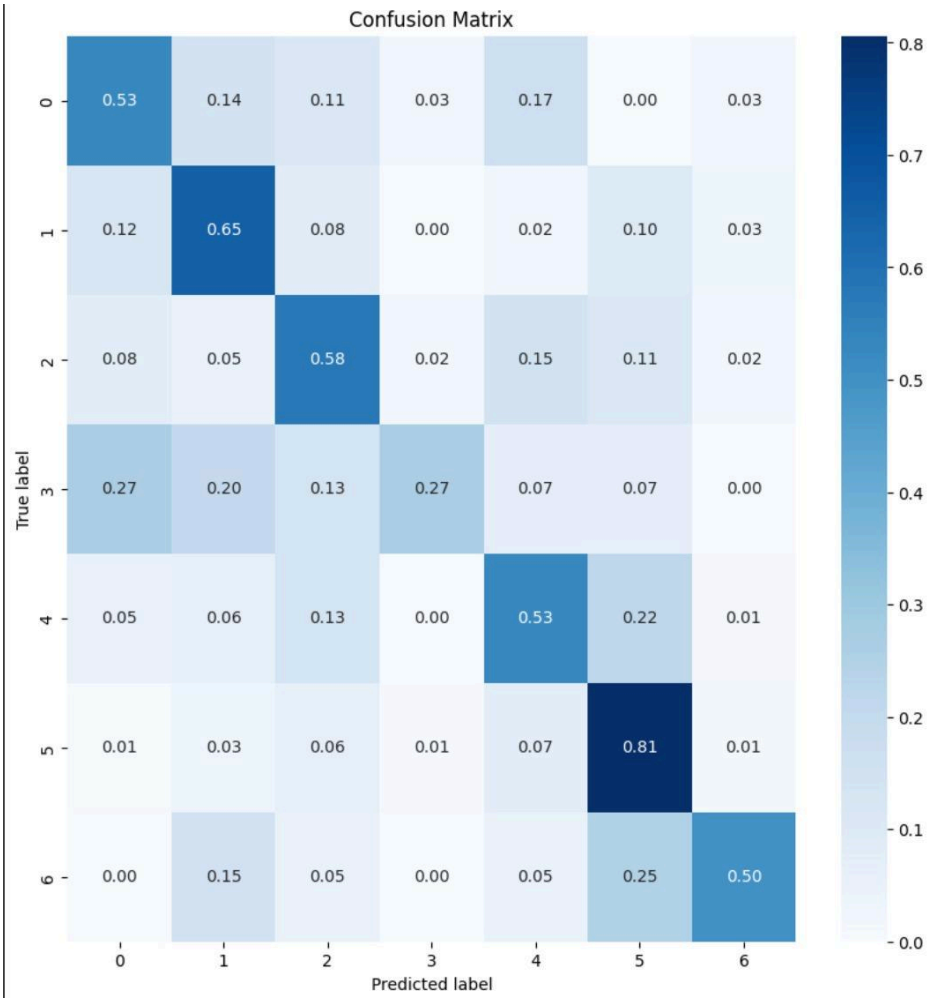
```
Epoch [1/25], Train Loss: 3.1505, Train Accuracy: 39.50%, Val Loss: 1.8169, Val Accuracy: 46.10%
Epoch [2/25], Train Loss: 1.2315, Train Accuracy: 58.53%, Val Loss: 1.1409, Val Accuracy: 57.60%
Epoch [3/25], Train Loss: 1.0420, Train Accuracy: 62.10%, Val Loss: 1.0502, Val Accuracy: 62.60%
Epoch [4/25], Train Loss: 0.9329, Train Accuracy: 64.47%, Val Loss: 1.0834, Val Accuracy: 62.20%
Epoch [5/25], Train Loss: 0.8340, Train Accuracy: 69.80%, Val Loss: 0.9674, Val Accuracy: 66.90%
Epoch [6/25], Train Loss: 0.7498, Train Accuracy: 72.13%, Val Loss: 0.8256, Val Accuracy: 70.00%
Epoch [7/25], Train Loss: 0.6810, Train Accuracy: 75.07%, Val Loss: 1.1259, Val Accuracy: 60.60%
Epoch [8/25], Train Loss: 0.6551, Train Accuracy: 75.27%, Val Loss: 0.8770, Val Accuracy: 69.30%
Epoch [9/25], Train Loss: 0.6344, Train Accuracy: 76.67%, Val Loss: 0.9026, Val Accuracy: 72.80%
Epoch [10/25], Train Loss: 0.5586, Train Accuracy: 78.93%, Val Loss: 0.9348, Val Accuracy: 68.10%
Epoch [11/25], Train Loss: 0.5216, Train Accuracy: 80.57%, Val Loss: 0.9223, Val Accuracy: 69.50%
Epoch [12/25], Train Loss: 0.4674, Train Accuracy: 82.93%, Val Loss: 0.9701, Val Accuracy: 68.30%
Epoch [13/25], Train Loss: 0.4622, Train Accuracy: 82.87%, Val Loss: 0.9829, Val Accuracy: 68.50%
Epoch [14/25], Train Loss: 0.4393, Train Accuracy: 83.70%, Val Loss: 0.9909, Val Accuracy: 69.50%
Epoch [15/25], Train Loss: 0.3919, Train Accuracy: 85.70%, Val Loss: 0.9900, Val Accuracy: 70.70%
Epoch [16/25], Train Loss: 0.4071, Train Accuracy: 85.33%, Val Loss: 0.9811, Val Accuracy: 68.90%
Epoch [17/25], Train Loss: 0.3520, Train Accuracy: 87.23%, Val Loss: 0.9332, Val Accuracy: 70.80%
Epoch [18/25], Train Loss: 0.3655, Train Accuracy: 86.90%, Val Loss: 1.0178, Val Accuracy: 69.60%
Epoch [19/25], Train Loss: 0.3121, Train Accuracy: 89.10%, Val Loss: 1.1700, Val Accuracy: 66.60%
Epoch [20/25], Train Loss: 0.3186, Train Accuracy: 87.97%, Val Loss: 0.9215, Val Accuracy: 71.90%
Epoch [21/25], Train Loss: 0.2681, Train Accuracy: 90.50%, Val Loss: 1.0090, Val Accuracy: 71.20%
Epoch [22/25], Train Loss: 0.2678, Train Accuracy: 90.17%, Val Loss: 1.0444, Val Accuracy: 70.10%
Epoch [23/25], Train Loss: 0.2752, Train Accuracy: 89.87%, Val Loss: 1.0638, Val Accuracy: 72.20%
Epoch [24/25], Train Loss: 0.2572, Train Accuracy: 90.47%, Val Loss: 1.0260, Val Accuracy: 71.30%
Epoch [25/25], Train Loss: 0.2408, Train Accuracy: 91.80%, Val Loss: 1.0359, Val Accuracy: 72.20%
```



Training and Validation loss and Accuracies plots vs Epochs

Validation Accuracy: 72.20%

	precision	recall	f1-score	support
0	0.40	0.53	0.45	36
1	0.48	0.65	0.55	60
2	0.47	0.58	0.52	102
3	0.36	0.27	0.31	15
4	0.41	0.53	0.46	97
5	0.92	0.81	0.86	670
6	0.38	0.50	0.43	20
accuracy			0.72	1000
macro avg	0.49	0.55	0.51	1000
weighted avg	0.76	0.72	0.74	1000



The Confusion Matrix shows the performance of a classification model by comparing actual versus predicted labels.

```
Accuracy for class 0: 0.53
Accuracy for class 1: 0.65
Accuracy for class 2: 0.58
Accuracy for class 3: 0.27
Accuracy for class 4: 0.53
Accuracy for class 5: 0.81
Accuracy for class 6: 0.50
```

Accuracies for each Classes

MOBILENETV2 WITH HARD DATA MINING:

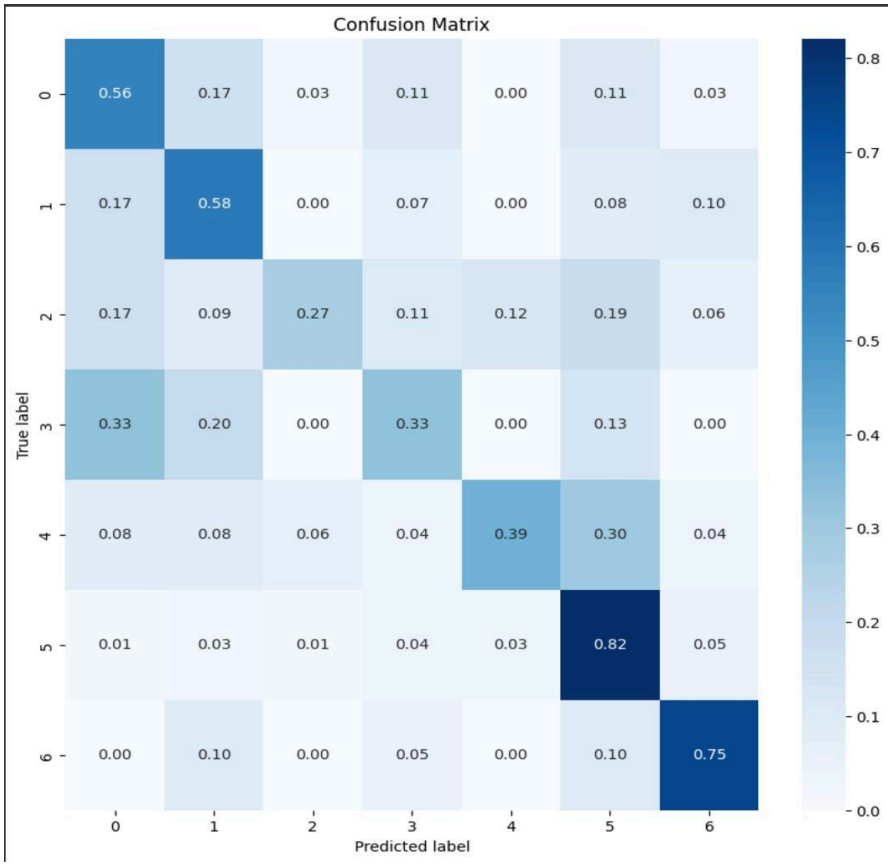
In the second phase of our approach, we enhanced the baseline MobileNetV2 model using a **hard data mining** strategy. After initially training the model on the entire dataset, we evaluated its performance on the training set to identify **hard samples**—i.e., those images the model either misclassified or predicted with low confidence. We then created a **new training subset** consisting exclusively of these difficult examples. The MobileNetV2 model was further fine-tuned on this subset to help it focus on edge cases and improve its generalization. By explicitly forcing the model to re-learn from its mistakes, this hard data mining phase helped improve classification performance, particularly for underrepresented and visually ambiguous classes.

RESULTS FOR MOBILENETVE WITH HARD DATA MINING:

Our model MobileNetV2 with Hard Data Mining achieved an accuracy of 69.10% on the validation data.

Validation Accuracy: 69.10%

	precision	recall	f1-score	support
0	0.29	0.56	0.38	36
1	0.43	0.58	0.49	60
2	0.62	0.27	0.38	102
3	0.09	0.33	0.14	15
4	0.52	0.39	0.45	97
5	0.90	0.82	0.86	670
6	0.23	0.75	0.35	20
accuracy			0.69	1000
macro avg	0.44	0.53	0.44	1000
weighted avg	0.76	0.69	0.71	1000



The Confusion Matrix shows the performance of a classification model by comparing actual versus predicted labels.

```
Accuracy for class 0: 0.53
Accuracy for class 1: 0.65
Accuracy for class 2: 0.58
Accuracy for class 3: 0.27
Accuracy for class 4: 0.53
Accuracy for class 5: 0.81
Accuracy for class 6: 0.50
```

Accuracies for each Classes

CONCLUSION

Upon examining the distribution of images across the different classes in our dataset, we observed a significant class imbalance, as illustrated in the accompanying bar chart. Notably, Class 5 constituted the majority of the dataset, while several other classes had comparatively few samples. This uneven distribution posed challenges during model training, often causing the model to overfit on the dominant class while underrepresented minority classes, leading to misclassifications.

To address this issue and improve the model's ability to generalize across all classes, we implemented a hard data mining strategy. By retraining the model specifically on samples that were misclassified or predicted with low confidence, we encouraged the network to focus on difficult cases and underrepresented patterns. This approach helped mitigate the effects of imbalance and contributed to a more robust and balanced classification performance.

REFERENCES

1. https://pytorch.org/vision/main/models/mobilenetv2.html
2. https://pytorch.org/tutorials/
3. https://pytorch.org/vision/main/models/generated/torchvision.models.mobilenet_v3_large.html
4. Lecture Slides: Lecture materials provided by Professor Bharat Richhariya on Deep Learning.
5. Goodfellow, I., Bengio, Y., and Courville, A. "Deep Learning." In preparation for MIT Press, 2016.
6. https://www.researchgate.net/publication/349174886_Cancer_Detection_Using_Convolutional_Neural_Network
7. https://www.sciencedirect.com/science/article/abs/pii/S0893608023006287