Giovanni Michele Miranda, 507567
Maria Caterina Mottini, 502322

Data Collection

We are asked to perform image classification and image correction using MATLAB. The first step to be
carried out is the creation of the dataset. Our goal is to classify the exposure level of a total of 101 pictures, specifically to classify them into three classes, namely correctly exposed, overexposed, or underexposed.

The classes are distinct and exclusive, meaning that no image can be at the same time overexposed and
correctly exposed, or overexposed and underexposed, or correctly exposed and underexposed.

We therefore created three folders, appropriately named, containing a balanced distribution of pictures:
31 underexposed images, 33 overexposed images, and 37 correctly exposed images. The images were taken using our smartphones (both are IPhones).

- Correctly exposed images: the pictures collected were strategically chosen to depict different landscapes (changing seasons), and different light settings (at night, during the day, with artificial light), and at different ranges of closure to the object (from close-ins to panoramic).
- Overexposed images and Underexposed images: similarly to the correctly exposed images, the other pictures have been taken in different natural-lighting scenarios, manually obtaining the desired exposure.

Importantly, we deactivated the live photo option in the Camera settings. When taking a live photo,
indeed, the IPhone saves a .HEIC file, which is not needed. Moreover, we deactivated HDR (High Dynamic Range). The higher dynamic range the camera has, the closer the picture will compare to how
The human eye sees reality. Notwithstanding the fact that this is an aesthetically pleasing feature in
real-life, for our intents and purposes, it is not needed. In fact, it would correct the images that we are
purposefully taking in extreme ranges of exposure.

Image Preparation

During the preprocessing phase, all images are converted to grayscale and resized to dimension [480,640] for consistency and computational efficiency: processing a single channel

(grayscale) instead of three channels speeds up calculations and reduces memory usage, making algorithms more efficient, moreover the size [480, 640] is a balance between retaining sufficient detail and reducing the data size.
The pictures are then saved back to the original location.

Image Classification

Feature Extraction

To extract features from our pictures, we use a function *fe_low_level*, which takes an image as input and returns a vector of extracted features: mean, variance, skewness, kurtosis, and percentiles. Having low-level features will be useful to perform classification, specifically:

- **Mean intensity** provides a measure of the overall brightness;
- **Variance** indicates the spread or dispersion of pixel intensities;
- **Maximum value** indicates the highest proportion of pixels at any intensity level;
- **Minimum value** indicates the lowest nonzero proportion;
- **Skewness** shows the asymmetry of the pixel intensity distribution; positive skewness indicates a distribution with a long tail on the right (more dark pixels), while negative skewness indicates a distribution with a long tail on the left (more bright pixels);
- **Entropy** is a statistical measure of randomness in pixel intensity values. It provides valuable information about the complexity within an image.
- **Percentiles** summarize the distribution of pixel values.

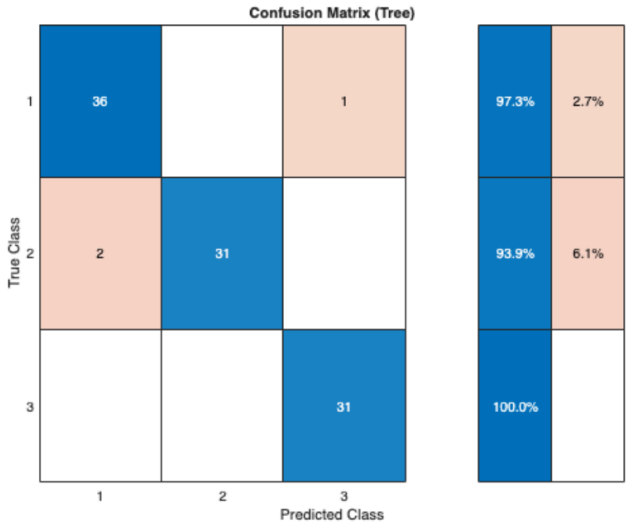
After the extraction of the features, we proceeded to the classification process, by firstly organizing the images in three classes, differentiated by the features described above. We define three classes and use
a for-loop to loop through each picture class, each of which corresponds to a different type of exposure.
Another for loop, inside the former one, iterates through each file path, reads the images (while converting
into double precision), extracts their features, and appends the current class label to the labels array. This ensures that each feature vector has a corresponding label indicating its class.

Model Training and Evaluation

The classification involves using K-fold cross-validation to train and evaluate three types of classifiers: Support Vector Machine (SVM), and Decision Tree. Again, we use a for loop to go through each fold (we decided to use 5 fold, as it is a common choice to train models). In the same loop, we get the indexes of training and test sets using a random cross-validation partitioning, and we train the three models. We used two MATLAB built-in functions: *fitcecoc* and *fitctree*.

- **fitcecoc** for SVM: trains multiclass models by combining binary SVM classifiers.
- **fitctree** for Decision Tree: constructs a tree-structured model for predictions.

After the images are managed by the two models, we compare the performances. We obtain the accuracy percentage of each of the models, obtaining 89.11% in the former case and 90.10% in the latter.
Moreover, we can have some insights using confusion matrices:



Some notable evidence in the matrices is that SVM performs well for correctly exposed images, as expected. The model struggles more with overexposed images, often confusing them with correctly exposed images.

Decision Tree classifier has better results, with the highest number of correctly classified pictures. The **overall accuracy of Decision Tree**, indeed, is about **90%,** so we decided to save this model and related data.

Image Correction

To proceed in our goal, we need to perform a correction of wrongly exposed images. To achieve this, we use the extracted features to determine thresholds for gamma correction.

An empty structure (*features_all*) is initialized to store the features for each class of images. We will
use this structure to store the features from the three classes, using the same function defined in the
classification process, and compute the mean values for each of those features. The results are then

printed:

```
-Class: correctlyexposed
Mean Intensity: 0.48
Variance: 0.04
Entropy: 7.43
Skewness: 1.12
```

```
-Class: overexposed
Mean Intensity: 0.69
Variance: 0.07
Entropy: 7.04
Skewness: 6.99
```

```
-Class: underexposed
Mean Intensity: 0.21
Variance: 0.03
Entropy: 6.53
Skewness: 2.89
```

Having these values, we proceed with the gamma correction process for adjusting the exposure of images. To enhance the exposure accuracy of our image classification model, we implemented an **automated gamma correction** process. This technique adjusts the luminance of each image, aiming to reclassify underexposed and overexposed images into the "correctly exposed" category.

Firstly, all images are processed to **grayscale** and **resized** to maintain consistency. For each image not initially predicted as correctly exposed, we iteratively apply gamma adjustments to modify its brightness.
Underexposed images have their brightness increased by reducing the gamma value, while overexposed images have their brightness decreased by increasing the gamma value. This iterative process continues until the image is classified as correctly exposed or a maximum number (35) of iterations is reached.

During each iteration, we extract features from the gamma-corrected image and use our trained **decision tree model (best in performance)** to predict its class. If the corrected image is successfully classified as correctly exposed, it is saved along with the applied gamma value. If the image cannot be corrected within the maximum iterations, the original image is saved with a "_NOTcorrect" suffix. This approach allows for dynamic adjustment based on initial predictions and ensures a comprehensive correction process, allowing for a visual feedback of the applied corrections.

The values of gamma applied to the single images are stored in the *gamma_applied* matrix. We can therefore use an histogram and a box plot check the frequencies with which gamma values have been applied:

Metric Evaluation: Histogram analysis

The obtained results allowed us to make some considerations about the gamma correction process and give valuable interpretations with respect to the corrected images and the original dataset in general.
The analysis of skewness values in our gamma-corrected images reveals a very **nuanced** distribution. The **80% CI**, chosen to focus more on central tendency, spanning from 0.26 to 9.40, delineates where the central 80% of skewness values reside.
The **mean** skewness value of **3.37** indicates a high level of right-skewness across the dataset, suggesting that gamma correction tends to **bias pixel intensities towards brighter values**.

This observation is compounded by the presence of several images with notably **high skewness values**, contributing to an elevated mean and highlighting the varied impact of gamma correction across the dataset.

On the contrary, the **median** skewness value of **1.85** serves as the midpoint of the skewness distribution, indicating that half of the images have lower skewness values than this median. This disparity between the mean and median underscores a right-skewed distribution with a tail extending towards higher skewness values, likely due to outliers and images with significant brightness adjustments. Importantly, visual analysis through histograms illustrates a clustering of skewness values around lower levels, with a visible extension towards higher values. This variability confirms that gamma correction effectively adjusts image brightness but also reveals the diverse responses of different images to this correction.

It's crucial to underline that the initial images before correction exhibited considerable skewness themselves. This suggests that a lot of images already showed a considerable bias towards either darker or brighter pixel values, resulting in a worse correction accuracy overall.

Conclusion

We conclude by classifying the exposure levels of corrected images using the saved Decision Tree model.
Using a for loop, we read the corrected images, extract their features, predict their classes, and count the
occurrences of each class. Finally, we calculate the percentage of correctly exposed images.

The final result is that we have roughly **86% of correctly exposed images**, which is a good score. Using cross-validation provides a robust estimate of model performance, and we used an effective feedback loop between correction and classification to fine-tune the brightness of images until they are correctly adjusted.
However, some observations can be raised: even though the images are converted to grayscale, retaining some color information through histogram analysis of color channels could be beneficial for distinguishing exposure levels. The method we used, moreover, relies heavily on mean intensity and variance given their fundamental relevance in features estimation, which may not capture all aspects of an image’s exposure. Additionally, complex images with difficult patterns and varying local brightness levels might require more sophisticated analysis not satisfied by our model.