Solar Eclipse Image Classification

Luca Moresca, Nicholas Suozzi, Valerio Santini
Sapienza University of Rome
lucamoresca12@gmail.com, nicholassuozzi@gmail.com, valerio.santini.97@gmail.com

Abstract

In this project, we tackle the problem of classifying the phases of a solar eclipse. The purpose is to develop a machine learning model capable of accurately identifying the different phases of a solar eclipse, such as total, partial and annular eclipse, by analysing images. **In the project we use the 'Eclipse Megamovie' dataset, a collection of solar eclipse images taken by citizens and scientists during 'The Great American Eclipse' of 2017**. **The dataset presents several challenges, including variability in lighting conditions, image quality and the presence of noise elements**. To prepare the data for model training, we implemented several image pre-processing techniques, including greyscale conversion, noise reduction, scaling and normalisation. **We use a convolutional neural network (CNN)** model for image classification, this choice is motivated by its effectiveness in extracting features from images. The results obtained demonstrate the effectiveness of the proposed model in accurately classifying the phases of the solar eclipse. We have also implemented an **image retrival** model using **histograms**, which is able, by means of distance measures, to classify images by finding the closest one in the dataset.

1 Introduction

The project is part of the 'Eclipse Megamovie' [2] science initiative, which was launched during the 2017 'Great American Eclipse'. This initiative involved thousands of volunteers contributing to a large database of high-resolution images of the total solar eclipse, captured along its path across the United States. This archive represents a fundamental resource for the in-depth study of the solar corona, the Sun's outer atmosphere visible only during total eclipses.

The project described aims to contribute to the 'Eclipse Megamovie' database by developing a system for the automatic classification of eclipse phases. This approach aims to facilitate the analysis of the collected data by automating the identification of scientifically relevant phases and promoting the study of the solar corona.

In the context of the Kaggle competition hosted on the official platform [3] , the project *'Categorising Solar Eclipse Phases'* involves the creation of a machine learning model capable of recognising the different phases of the eclipse (total, partial, annular) and correctly identifying irrelevant images. The dataset provided, consisting of thousands of images collected during the 'Great American Eclipse' of 2017, provides the basis for this challenge.

The competition encourages participants to develop models capable of meeting the following challenges:

- Image diversity: Photos come from different sources and devices, with varying lighting conditions and backgrounds.
- Inclusion of irrelevant images: the model must be able to distinguish between images that show an eclipse and those that do not.

Our target in this project is to implement a robust and effective system that improves data processing and understanding in the context of the Eclipse Megamovie initiative.

2 Dataset overview and Preliminary Analysis

2.1 Dataset "Eclipse Megamovie"

The dataset consists of hundreds of images captured during the 2017 'Great American Eclipse'; this dataset provides a rich source of data for training and evaluating machine learning models.

2.1.1 Main Features

- **Dimensions**: The dataset is divided into two sets: a training set with 495 images and a test set with 140 images. This division allows the model to be trained on datasets and its performance evaluated on a separate, independent set.
- **Type of Images**: The dataset includes a variety of images. Some images were taken by semi-professional photographers, while others are from astronomy enthusiasts with amateur equipment.
- **Conditions of Acquisition**: Images were acquired in a wide range of conditions, including different geographical locations, varying weather conditions and different camera settings.

Figure 1: Visualization of the dataset

2.1.2 Dataset Challenges

The heterogeneous nature of the dataset introduces a number of challenges for the development of an accurate classification model.

- **Variety of Images**: The presence of semi-professional and amateur images, taken with different devices and under different conditions, makes the dataset highly variable. The model must be able to generalise well to this variety of images to achieve robust performance.
- **Presence of Irrelevant Images**: The dataset includes images that do not contain solar eclipses, posing an additional challenge for the model. The model must be able to correctly identify these images as 'impostors' to avoid false positives.
- **Variability in Size and Orientation**: The images in the dataset have different resolutions and the eclipse is not always centred in the image. These variations require image pre-processing techniques to standardise the input data to the model.

2.1.3 Data Format

The dataset is organised in a file structure that includes:

- **File .json**: This file contains the mapping between the numerical labels used in the .csv file and the corresponding phases of the solar eclipse.

- **File .csv**: This file contains a list of image file names and their numerical labels indicating the phase of the solar eclipse in the image. Each line in the file corresponds to a single image in the training set.
- **Folder with training and test images**: The dataset images are stored in a separate folder, organised in subfolders for the training set and the test set.

2.2 Preliminary Analysis

A preliminary analysis of the dataset is essential to understand the characteristics of the data and to identify the specific challenges that the classification model will face.

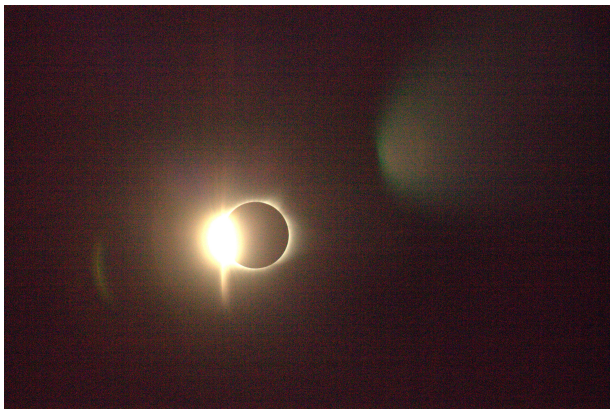

Figure 2: Example of an image from the dataset

The image in figure 2 was taken with an amateur device and has a significant level of noise. The eclipse is not perfectly centred in the image and the brightness varies in different areas of the image. These characteristics do not apply to all images in the dataset, this variability is a challenge that complicates the classification of the images, as it would be simpler if the analysis were carried out under the same conditions.

These preliminary observations suggest the need to apply image pre-processing techniques to improve data quality and to standardise images prior to model training.

3 Image Preprocessing

As discussed above, the dataset is characterised by a high variability in images with different resolutions, perspectives and acquisition conditions. To address this challenge and prepare the data for training the classification model, an image preprocessing process was implemented.

3.1 Preprocessing pipeline

The image preprocessing process was designed to standardise the images in the dataset and prepare them for training the classification model. Here is a detailed description of the steps involved:

1. **Grayscale Conversion**: Each image is converted from an RGB colour image to a greyscale image. This operation reduces data complexity as it removes colour information that is not essential for eclipse phase classification. To convert the image to greyscale, we took advantage of the **OpenCV** [4] library's function, *cvtColor*.
2. **Noise Reduction**: A Gaussian filter is applied to the greyscale image to reduce random noise. The Gaussian filter averages the values of the surrounding pixels, smoothing out intensity fluctuations and making the image more uniform.

3. **Normalisation**: Image pixel values are normalised to a range between 0 and 1. Normalisation ensures that all images have a similar intensity scale, improving the stability and efficiency of the model training process.

4. **Eclipse Detection**: An eclipse detection algorithm is applied to identify the region of the image that contains the sun. This algorithm uses the Otsu threshold to separate bright pixels (corresponding to the sun) from dark pixels (corresponding to the background). The largest region of bright pixels is identified as the eclipse.

5. **Crop**: The image is cropped around the detected eclipse region, removing irrelevant areas of the image, such as the background. Cropping focuses on the region of interest for classification.

6. **Resizing**: The cropped image is resized to a fixed predefined size. Resizing ensures that all input images have the same size, which is a requirement for training many machine learning models.

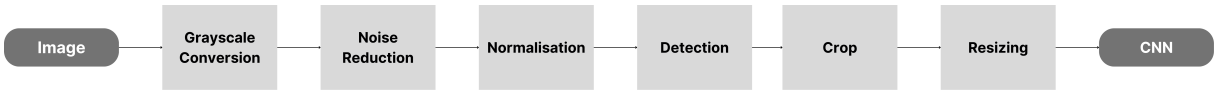

Figure 3: Preprocessing pipeline

This preprocessing process produces standardised images that are ready for feature extraction and training of the classification model.

3.2 Parallelisation of the pipeline

After implementing the preprocessing pipeline, we realised that in order to process the large number of images in the dataset, the time required to complete the process was extremely long. Therefore, we decided to use python's *concurrent.futures* library. Several processes are used to perform the pre-processing, the image paths are distributed over the available threads. Each thread performs the pre-processing function on one of the image paths, which means that multiple images can be pre-processed and saved at the same time, reducing processing time compared to a sequential approach. In this way, we were able to significantly reduce the execution time of the rendering pipeline:

- Time with parallelisation: 97.14 seconds (1.6 minutes)
- Time without parallelisation: 379.42 seconds (6.31 minutes)

4 Histogram classification

We decide to apply a simple resolution model for our task. We decided to use classification using histograms. This type of classification is based on the computation of the histogram of each image, in our case it will be a histogram of the type dxdy that takes into account the partial derivatives along the vertical and horizontal axes of each image. Thinking of the sun's corona and the edge of the moon in front of the sun, it is easy to see that this type of technique allows us to detect the edge of the sun and understand what percentage of it is in front of it. Furthermore, it is difficult to detect anything else in the image besides the sun, because other objects are rarely present in eclipse images.

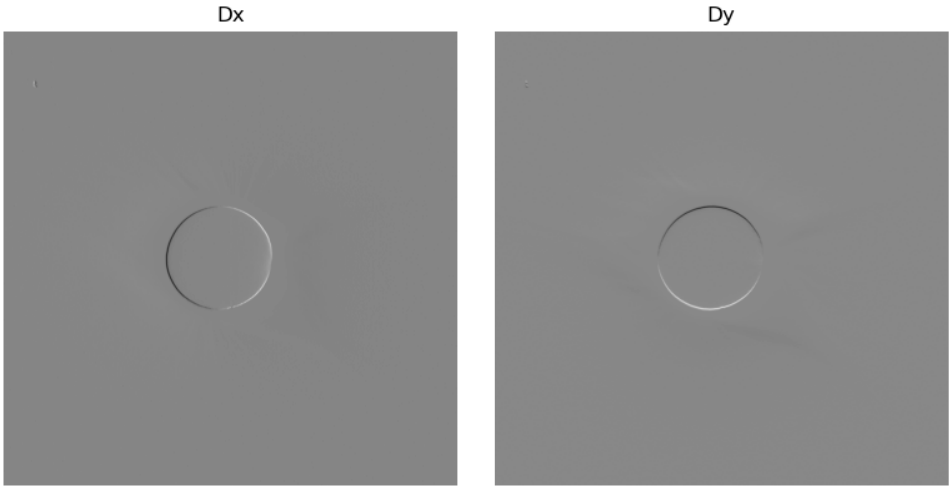

Figure 4: Derivatives along x and y axes

As can be seen in figure 4, the edge of the moon in front of the sun is detected. This image is obtained by applying a Gaussian filter and convolving the filter and image twice along both axes.

Afterwards, the combined histogram of the Gaussian partial derivatives of the image in the x and y direction is calculated.

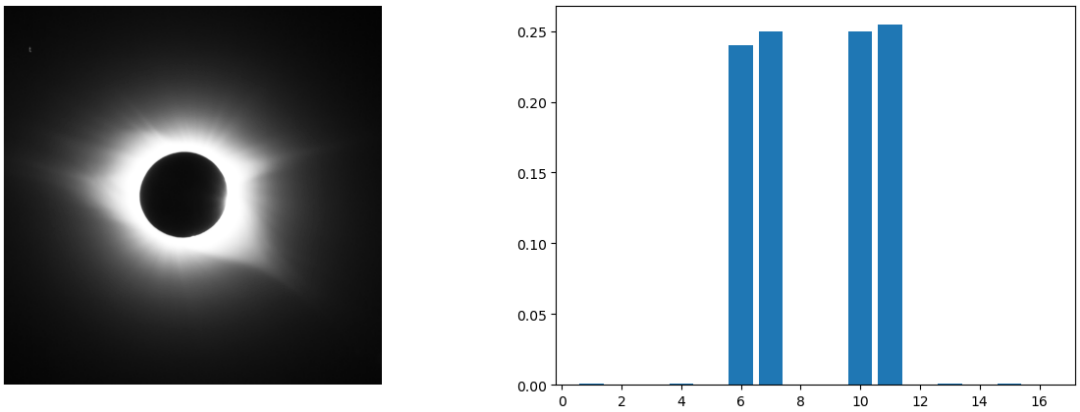

Figure 5: Histogram DxDy

The histogram in figure 5 provides us with important information regarding the nature of the image. It can be seen that there is a predominant concentration of values in a narrow range, which tells us that the derivatives represent well-defined variations in a certain direction. The predominance on a few bins also shows us areas where there is a sharp transition between light and dark (between the sun and the edge of the moon). The non-uniform distribution also indicates the strong break that occurs between the edge of the moon and the sun.

4.1 Similarity Measurement

After calculating the **dxdy** histograms for each image, we measured the similarity between the histograms. This helps identify images with similar gradient distributions, suggesting potential membership in the same phase of the solar eclipse.

We used various metrics including:

- **Intersection Distance (*Intersect*)**: Sum of minimum values between corresponding bins, robust to outliers.

$$\bigcap(Q,V) = \frac{1}{2}\left(\frac{\sum_i min(q_i, v_i)}{\sum_i q_i} + \frac{\sum_i min(q_i, v_i)}{\sum_i v_i}\right) \tag{1}$$

- **Euclidean distance ("L_2")**: Sum of quadratic differences between corresponding bins, sensitive to outliers.

$$d(Q,V) = \sqrt{\sum_i (q_i - v_i)^2} \tag{2}$$

- **Chi-Square Distance χ^2 (*Chi2*)**: Sum of weighted quadratic differences, sensitive to differences in the shapes of the distributions.

$$\chi^2(Q,V) = \sum_i \frac{(q_i - v_i)^2}{q_i + v_i} \tag{3}$$

After several attempts, trying out various metrics, we chose to use the **Euclidean distance** metric with a **bin number of 16**

4.2 Best matches

To find the best match, a distance matrix was created after calculating the histograms, with the distances between all histogram pairs of images in the dataset. This matrix is then used to find the images with the highest level of similarity. Images with low distance between their histograms are candidates to belong to the same eclipse phase. The dataset provides a CSV file to obtain the actual labels of the images in the training set. These labels are used to evaluate the accuracy of the classification method based on histogram similarity. The label assignment process consists of two steps:

- **True label assignment:**. For each image, its corresponding label is read from the CSV file.
- **Assignment of predicted labels:** For each image, we search for the most similar image within the training set, using the distance matrix D computed earlier. The label of the most similar image is assigned as the predicted label to the image under consideration.

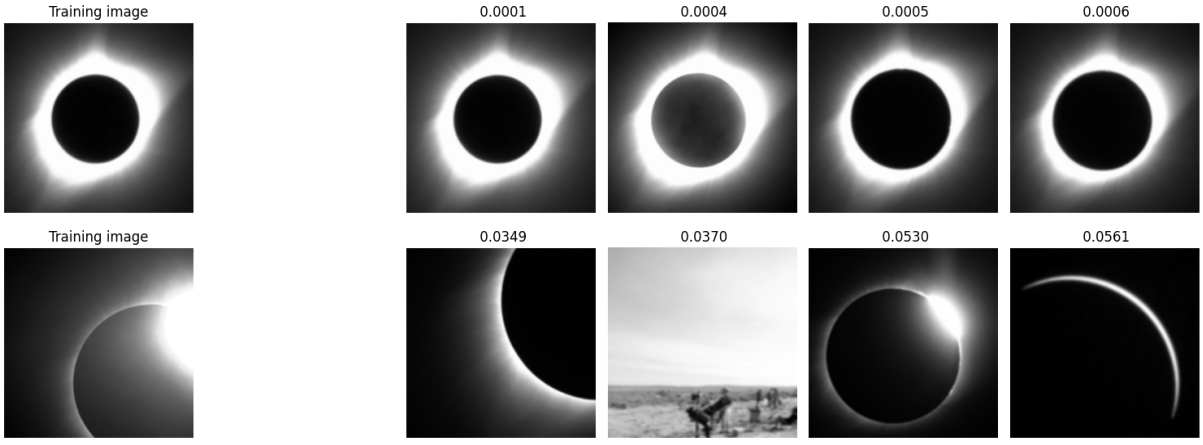

Figure 6: Top-5 match with the quary image

As can be seen in the fiugra 6 the model finds fairly accurate matches, with minimal distances between the various images. It should be noted that the model is forbidden to classify images that are equal to itself, otherwise a best match with itself would always exist, bringing the accuracy value to 100%, thus losing a reference on how it actually works.

4.3 Evaluation

The accuracy of the classification is based on the similarity of histograms by comparing the predicted labels with the actual labels. For each image, we check whether the predicted label matches the real label. If the labels match, the correct prediction count is incremented. Finally, accuracy is calculated by dividing the number of correct predictions by the total number of images. The result, a percentage, indicates how accurate the classification method was in identifying the correct eclipse phase for each image. The accuracy obtained is 76.36% using our best setting.

	Number of Bins	Metric (Distance Metric)	Accuracy (%)
1	2	Intersect	47.87
2	4	Intersect	65.85
3	8	Intersect	72.32
4	16	Intersect	76.36
5	32	Intersect	76.36
6	2	L_2	49.09
7	4	L_2	63.43
8	8	L_2	68.48
9	16	L_2	70.71
10	32	L_2	70.10
11	2	Chi^2	49.09
12	4	Chi^2	64.04
13	8	Chi^2	69.49
14	16	Chi^2	71.72
15	32	Chi^2	71.52

Table 1: Metric and Accuracy by Number of Bins

The table 1 shows the various attempts made with the various metrics to figure out which metric and which number of bins would be optimal for our purpose, we chose the highest accuracy value.

5 Convolutional Neural Network

The classification model chosen is a **CNN (Convolutional Neural Network)**. CNNs have increasingly demonstrated irreplaceable superiority in image classification. [1]

CNNs are particularly suitable for **classification problems** due to their ability to **automatically extract spatial features** from images. Unlike other machine learning algorithms, CNNs do not require manual feature extraction. The convolutional **layers** in CNNs are able to learn **patterns and shapes** from images, making them extremely efficient for machine vision tasks.

5.1 Theoretical Description of the Model

CNNs are composed of different types of layers, including:

- **Convolutional Layers**: These layers apply convolutional filters to the input image to extract features such as edges, textures and shapes.
- **Pooling Layers**: Pooling layers reduce the dimensionality of data extracted from convolutional layers, while retaining the most important features.
- **Layer Fully Connected**: These layers receive the data extracted from the previous layers and use them to classify the image into one of the predefined categories.

The training of a CNN consists of optimising the parameters of convolutional filters and fully connected layers to minimise a loss function. The loss function measures the difference between model predictions and actual image labels.

Our Convolutional Neural Network is realised using Pytorch library. We took the CNN from the Pytorch manual [5] as a basic model, customising it according to our needs. We added a convolution layer a in order to achieve greater attention to detail. In addition, we added a fully connected layer in order to have a better mapping of the extracted features. Each value of the output of the fully connected layer represents a raw probability that the image belongs to a certain class. A function of *softmax* is applied to these raw protabilities to convert them into normalised probabilities that determine belonging to a particular class.

5.2 Results

After the model implementation, we performed the CNN training, initially without going through the pre-proccesing pipeline. This resulted in very long compilation times ($\approx 77min$ using the GPU). The level of accuracy was low (around 70%), we realised that the images were being classified inaccurately. This initially made us think of a *overfitting* problem, which we tried to avoid by reducing the number of *epochs*. Subsequently, by implementing the pre-processing pipeline, we were able to obtain significant results. The end result of this operation is the *confidence*, a value that we then used to determine whether an image belongs to a class.

Class	Precision	Recall	F1-Score	Support
0	0.98	0.99	0.99	187
1	0.96	0.98	0.97	52
2	1.00	0.97	0.99	38
3	0.98	1.00	0.99	57
4	0.98	0.98	0.98	63
5	1.00	0.95	0.98	44
6	0.95	1.00	0.97	18
7	1.00	0.92	0.96	36
Accuracy: 0.98 (495 samples)				
Macro Avg: Precision 0.98, Recall 0.98, F1-Score 0.98, Support 495				
Weighted Avg: Precision 0.98, Recall 0.98, F1-Score 0.98, Support 495				

Figure 7: Classification report

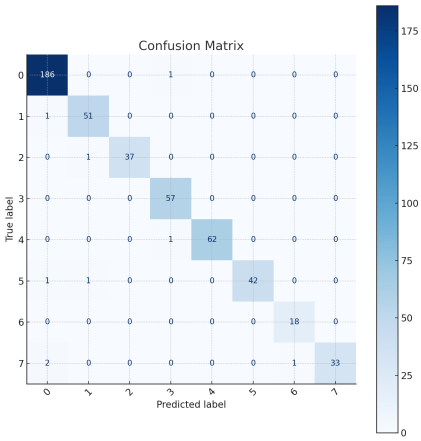

Figure 8: Confusion matrix

As is highlighted in the table 7, after applying the pre-processing the performance is still high, and from a preliminary analysis the risk of overfitting seems to have decreased considerably, thanks to the application, for example, of a normalisation on the image vectors. In addition, we have considerably reduced the batch size (by varying it from 64 to 4) in such a way as to obtain an introduction of noise, which, however, improves the generalisation of the model and decreases the risk of overfitting, reducing the computational weight of the model. Furthermore,

by analysing the *confusion matrix* (shown in figure 8), it shows that the classes are well balanced in the training and the predictions, as can be seen from the accuracy value, are correct.

Class	Precision	Recall	F1-Score	Support
0	0.73	0.96	0.83	187
1	0.93	0.81	0.87	52
2	1.00	0.50	0.67	38
3	0.87	0.58	0.69	57
4	0.86	1.00	0.93	63
5	0.96	0.59	0.73	44
6	0.38	0.89	0.53	18
7	1.00	0.19	0.33	36

Accuracy: 0.78 (495 samples)
Macro Avg: Precision 0.84, Recall 0.69, F1-Score 0.70, Support 495
Weighted Avg: Precision 0.83, Recall 0.78, F1-Score 0.76, Support 495

Figure 9: Clasification report (No pre-process)

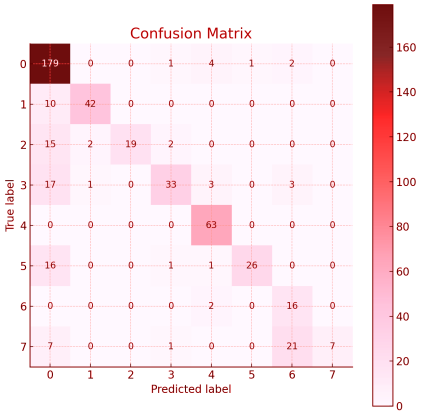

Figure 10: Confusion matrix (No-preprocess)

In the figures 9 and 10 we can see how, without applying the pre-processing pipeline, the performance of the model collapses to an accuracy level of 0.78%. In addition to having implications on the efficiency of the model, time is obviously also affected, with runtime going from *2m 46.2s* to *7m 12.2s* (considering the use of the GPU in both cases)

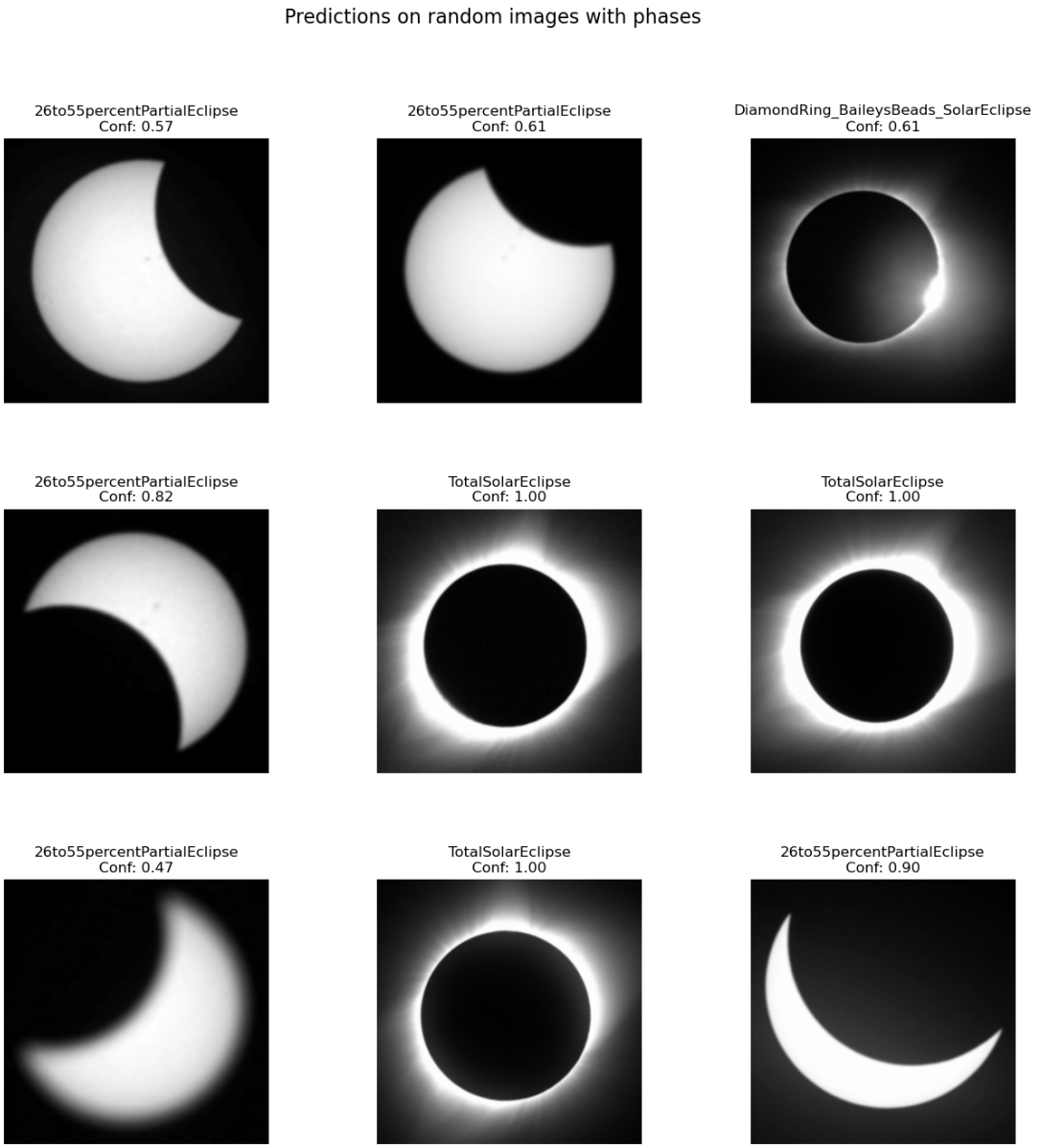

Figure 11: Output of the classification

The image in the figure 11 is the result of the classification, with the labels and confidence value shown above each image.

6 Conclusion

In conclusion, the problem of classifying solar eclipses was tackled with two different techniques, with different characteristics and different results. Classification using histograms appears to be a good compromise between accuracy and performance, allowing the images closest to each other to be found and thus enabling classification based on the proximity of the images in the dataset. The result of classification using histograms is satisfactory. Classification using convolutional neural networks is extremely precise, providing a clear division between eclipse classes with a high accuracy value. However, this type of model requires high computational costs and thus longer runtimes.

References

[1] Leiyu Chen et al. "Review of image classification algorithms based on convolutional neural networks". In: *Remote Sensing* 13.22 (2021), p. 4712.

[2] *eclipse-megamovie*. URL: https://opencv.org.

[3] *Kaggle*. URL: https://www.kaggle.com/competitions/eclipse-megamovie.

[4] *OpenCV*. URL: https://opencv.org.

[5] *Pytorch*. URL: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html.