Conformal Predictions for Weather Image Classification

Statistical Learning Project

Group 12 - Cacciatore Davide, Mattei Luca, Romani Luca

Introduction

In recent decades, Weather Recognition has become increasingly important, thanks to the growing interest in Computer Vision and AI applications. Being able to analyze and classify visual conditions at a given time and space has become necessary for many emerging sectors - just think of the emerging world of autonomous driving assistance.

The recognition of weather conditions remains a not thoroughly studied subject, particularly when it comes to the multi-labeling of weather and visual conditions, as they are often addressed individually. The aim of this project is, therefore, to perform Weather Recognition; more precisely, we try to predict to which of the chosen weather classes (*cloudy*, *foggy*, *rainy*, *snowy*, *sunny*) the image belongs.

For this project, we used a *Python* kernel provided by *Google Colab* with an additional GPU for the virtual machine. The main libraries used are *numpy*, *sklearn* and *keras*.

We exploited many methods and algorithms: as a first trial, we tried to classify our images using classifiers as *Gaussian Naive Bayes*, *Random Forest*, *Logistic Regression* and *Support Vector Machines* after appropriate pre-processing of the images. Then, we switched to *CNN (Convolutional Neural Network)* and, subsequently, using the "feature extraction" part of the *CNNs* used, we used the same classifiers as in the first part, noting a significant improvement in the results.

In this section, we added the *Conformal Predictions* as Interpretable Machine Learning task. The weather images classification is a very complex problem, there are many situations were weather conditions are unclear and cannot be identified in a single class. For this reason, we think that Conformal Predictions can really help us describe an image.

As a benchmark model, we exploited *Efficient Net B4* to see how a pre-trained, and much more complex, model would behave to solve our problem. This model is part of a class of pre-trained models introduced by Google AI, specifically it is pre-trained on ImageNet and then applied to the images in our train set.

Finally, we tested our models with their conformal predictions on a group of images taken directly from us over the past few years to see their capability in real situations.

The Data

We decided to collect images of 5 different weather classes: *cloudy*, *foggy*, *rainy*, *snowy* and *sunny*. We took data from different images datasets:

- Multi-Class Weather Dataset for Image Classification from Mendeley Data;
- Rome Weather Image Classification from Kaggle;
- Multi-Class Weather Dataset from Kaggle.

We only took images appropriate for our purpose and relative to our selected classes. At the end of this process, our dataset was really unbalanced, so we also used *Web Scraping* on *Google Images* in order to enlarge and balance our dataset. Thanks to these operations we managed to get a collection of 3694 images. In particular, we got 684 cloudy images, 900 foggy images, 788 rainy images, 670 snowy and 652 sunny images.

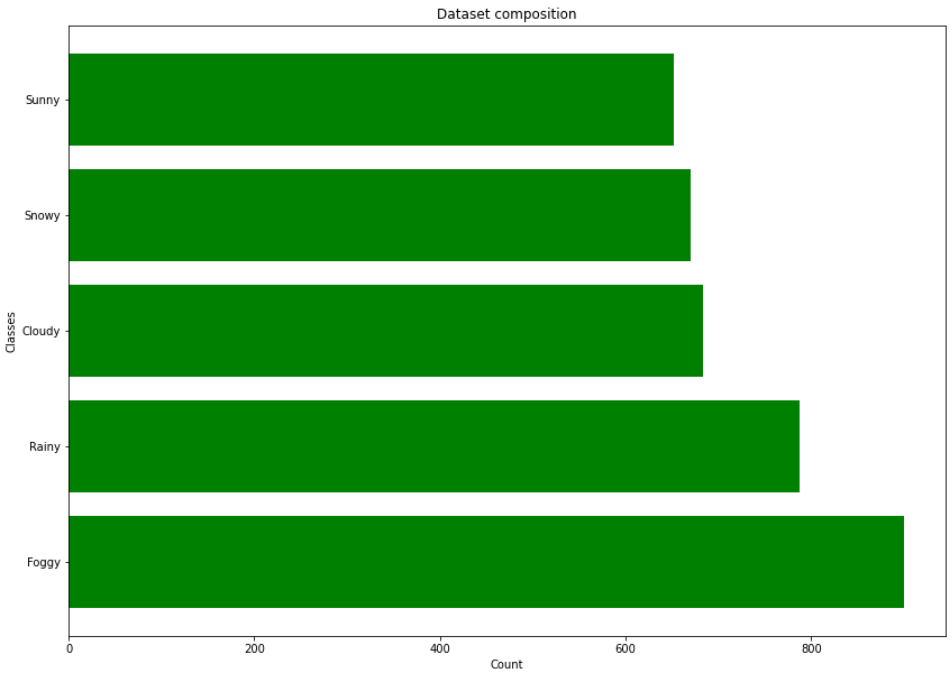

Figure 1: Dataset composition

Furthermore, we decided to use a bunch of pics taken by us in different weather condition as test set to see how our models will classify them.

Data Loading

To load our images we used the *Open CV* library and we follwed these steps:

- We set the image size to 256×256 pixels;
- We transformed all the images in *RGB* format;
- We set the label equal to the folder name where the image was originally stored.

We can see, as example, one image for each class:

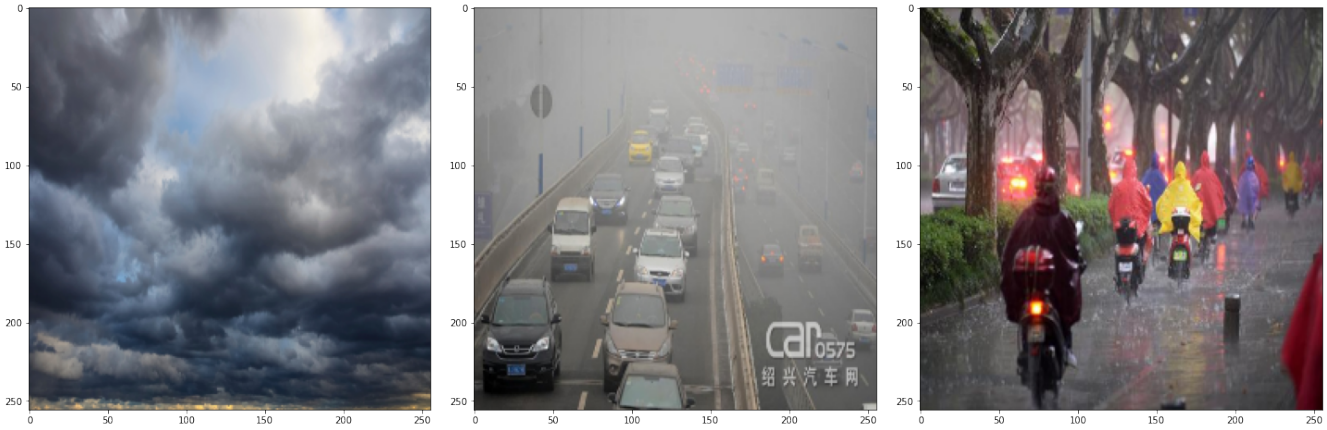

Figure 2: Cloudy, Foggy, Rainy

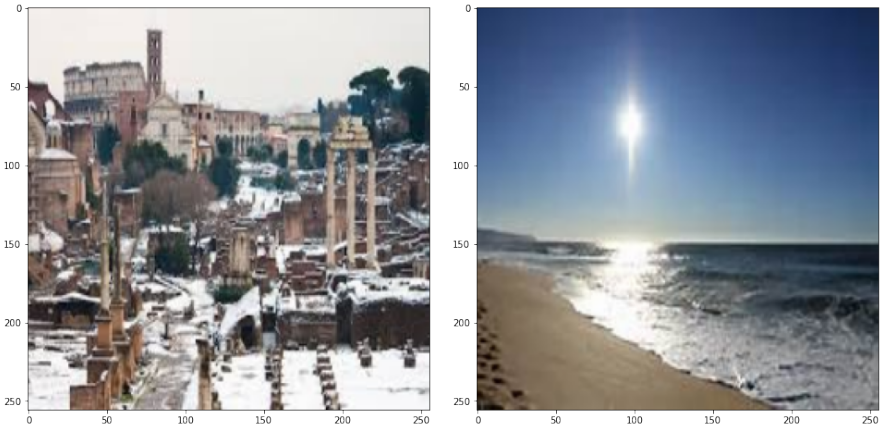

Figure 3: Snowy, Sunny

Image preprocessing and classification models

In order to perform our analysis, first we need to preprocess our images. All the methods we know cannot deal with 3D matrices, so we had to transform our images in a gray scale format and used the *flatten* function from *Numpy* which return a copy of the array collapsed into one dimension. We can see an example of this kind of transformation in Figure 4.

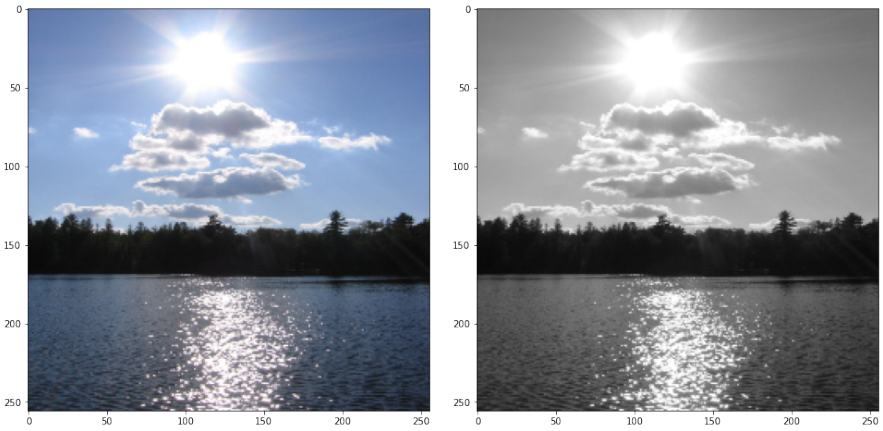

Figure 4: Original image and gray-scaled

In addition, we created a copy of the original dataset where we applied a *Gaussian blur* filter on the images. This is a widely used technique in image pre-processing in particular when reducing the size of an image. This way we smooth the image and we remove details and noise. We can see in Figure 5 an example of this transformation both for the original image and for the gray-scaled one.

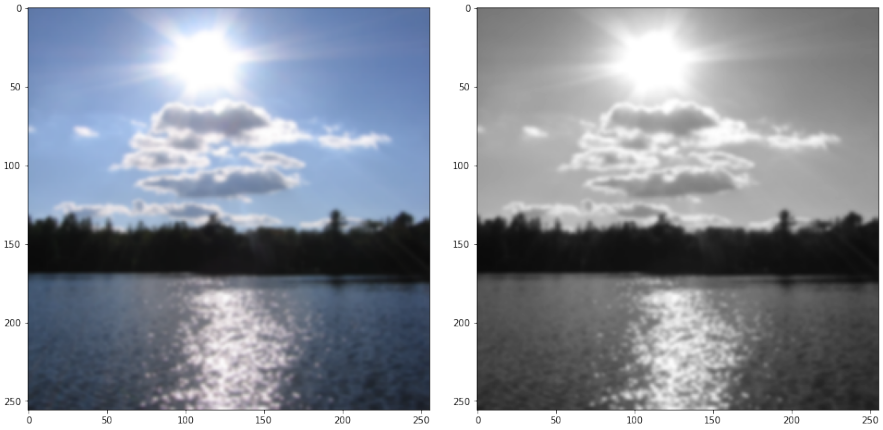

Figure 5: Blurred image and gray-scaled

Then, we splitted our dataset into train and validation test, setting the validation size equal to the 33% of the dataset. We decided to use classical classification methods on our data such as *Gaussian Naive Bayes*, *Random Forest*, *Logistic Regression* and *Support Vector Machines*.

Gaussian Naive Bayes Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. In the Gaussian Naive Bayes algorithm for classification the likelihood of the features is assumed to be Gaussian. Due to this strong assumptions we did not expect good results with this model, that is rarely used for image classification. We did not specify any prior probabilities, the algorithm will calculate them directly from data.

Random Forest The Random Forest classifier fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control overfitting. In this analysis, we kept the number of decision trees equal to 50.

Logistic Regression Logistic Regression, despite its name, is a linear model for classification rather than regression. We used as solver the *'newton-cg'* which supports L_2 regularization, it is used for the multinomial models and it was found to converge faster for high-dimensional data. We set 1000 maximum iterations for the algorithm to ensure the convergence.

Support Vector Machines Support Vector Machines (SVMs) are a set of supervised learning methods capable of performing binary and multi-class classification on a dataset. We decided to use a *linear* kernel which resulted to be the best in terms of accuracy compared with the other kernels.

We can see in a summary table how they performed:

Method	Accuracy not-blur	F_1 not-blur	Accuracy blur	F_1 blur
Gaussian Naive Bayes	0.565	0.556	0.552	0.541
Random Forest	0.771	0.765	0.761	0.753
Logistic Regression	0.597	0.554	0.580	0.541
Support Vector Machines	0.643	0.609	0.634	0.604

The Random Forest model was the best in terms of *accuracy* and F_1 score. We can notice how we can obtain better results with the non-blurred dataset.

CNN (Convolutional Neural Network)

We decided to apply a Convolutional Neural Network (CNN) which are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images.

First thing first, we encoded class labels from text to integers and perform a *one-hot encoding*. This kind of encoding is strictly needed when we deal with CNNs.
The network we used is structured as follow:

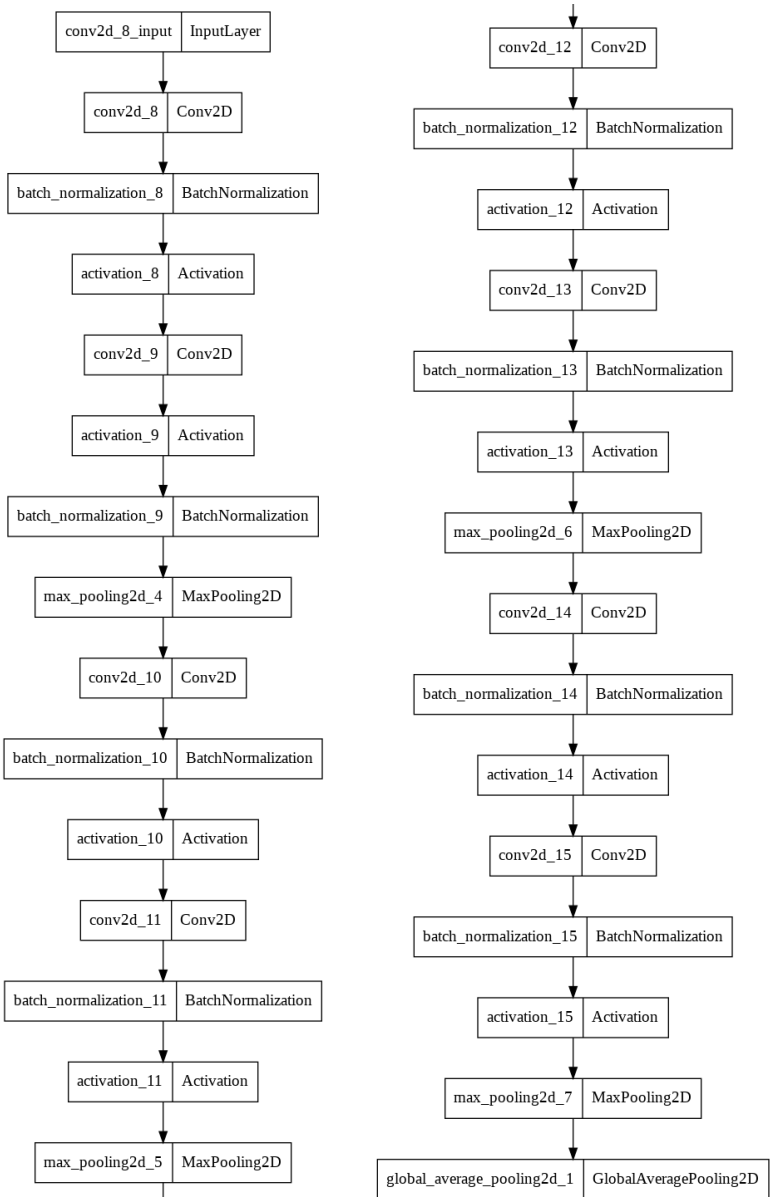

Figure 6: CNN feature extraction

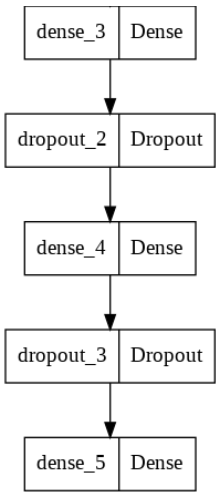

Figure 7: CNN dense

Our *CNN* is basically composed by 5 blocks: the first 4 blocks are used for the features extractions from the images while the last one is exploited for the classification purpose.
The 4 pre-processing blocks are all pretty equal: there is a "pipeline" of three layers (2D Convolution, Batch normalization and activation function) repeated twice and then a pooling.

The *Convolutional layer* is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter.

The *Batch normalization* (also known as batch norm) is a method used to make training of artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling.

The *activation function* is used to learn and approximate any kind of continuous and complex relationship between the variables of the network. We used the rectified linear activation function (or 'relu') which is easier to train, learn faster and perform better.

The aim of the *Pooling layer* is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. More specifically for our case, Max Pooling takes the largest element from feature map.

The *Dense layers* consist of the weights and biases along with the neurons and are used to connect between different layers. Moreover, in this CNN is also used the *dropout*: to prevent over-fitting a few neurons are dropped from the neural network during training process resulting in reduced size of the model.

Using this model we got nice results; after the classification on validation set we reached both an *accuracy* and F_1 score of 84%.

As we can observe from the below confusion matrix the model mess up between *snowy* and *rainy* pics, while sometimes it erroneously predict *sunny* for *cloudy* pics.

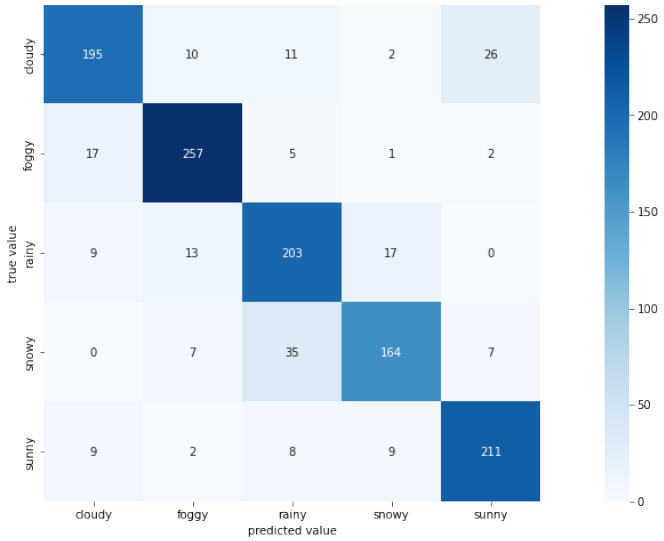

Figure 8: CNN confusion matrix

Models on CNN extracted features

We moved on applying the algorithms used in the first part on the data retrieved from the *feature extractor* of the previous *CNN*, with the addition of a flatten layer.
The resulting dataset had 65536 features. This way we managed to improve the results from the first part for all the used models, as we can see in the table below.

Method	Accuracy	F_1
Gaussian Naive Bayes	0.642	0.627
Random Forest	0.835	0.832
Logistic Regression	0.821	0.814
Support Vector Machines	0.834	0.830

We can notice how the best models are the *Random Forest* and the *Support Vector Machine*, but also the *Logistic Regression* registered very good results. The *Gaussian Naive Bayes* model improved its performance with respect to the previous preprocessing but the final results are not satisfactory.

Furthermore, as from the *feature extractor* of the previous *CNN* returned a massive dataset, we gave a try on using *PCA* before applying the classifier, more precisely we tried with *Random Forest*. Unfortunately, there were no improvements, so we gave up on this method and kept all our obtained features.

This high number of features created us some problems, in particular with *Logistic Regression* and *Gaussian Naive Bayes*. The Logistic Regression algorithm was not able to converge even with 1000 iterations, so the obtained results come from a "partial" run of the algorithm. For the Gaussian NB we encountered problems in the Conformal Predictions step as we will see.

We can give a look at the confusion matrices of the Random Forest and of the SVM:

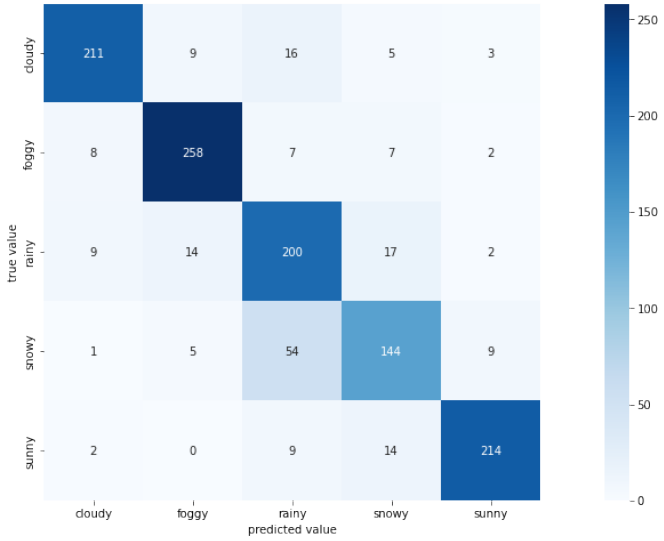

Figure 9: Random Forest confusion matrix

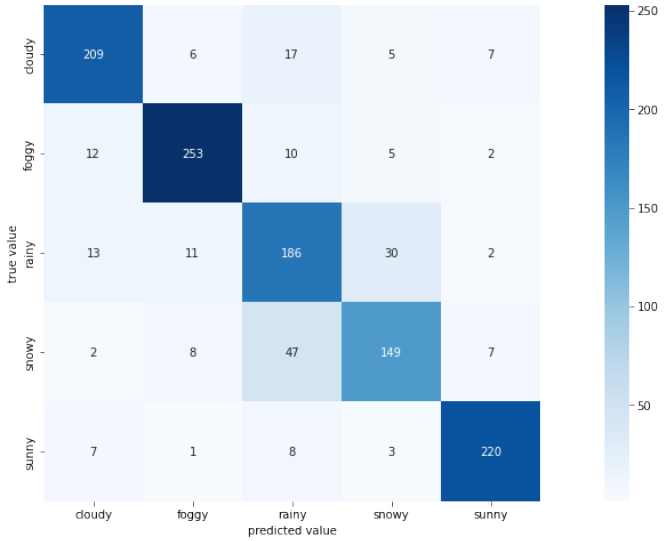

Figure 10: SVM confusion matrix

We can draw the same conclusions from these two confusion matrices. The models still mess up between *rainy* and *snowy* classes, but this time (especially for Random Forest), it has been made confusion also between *rainy* and *cloudy* images. Anyway, we can expect this kind of result. As said before, some weather conditions are not really distinguishable, and in particular, rainy and cloudy images can be easily confused.

Efficient Net B4

As a model benchmark, we exploited the *Efficient Net B4* to see how a pre-trained, and much more complex, model would perform to solve our problem. This model is part of a class of pre-trained models introduced by Google AI, in particular it is pre-trained on ImageNet and then applied to the images in our train set.
Using this pre-trained *CNN* the *accuracy* approached the 96.5%, while the F_1 score was equal to 96.4%. We can see below the relative confusion matrix.

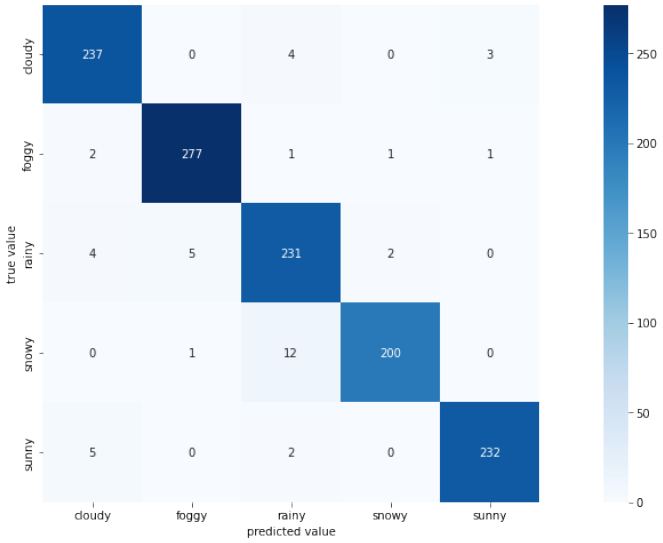

Figure 11: ENB4 confusion matrix

These are really outstanding results, there are really few errors. We can only notice that some snowy images are mis-predicted as rainy, but as said before, this kind of errors can be acceptable. An important thing is that the number of errors between sunny images and bad weather conditions images are really low.

In any case, this was a big Neural Network model pre-trained especially for this kind of tasks. We can use it as benchmark to compare with our implemented models.

Conformal Predictions

With conformal predictions, we perform a set of predictions instead of point predictions while maintaining some statistical guarantees. This can increase confidence in the use of Machine Learning techniques in critical areas. The classification of weather image is one of these, as this is a very complex problem, there are many situations were weather conditions are unclear and cannot be identified in a single class. For this reason, we believe that Conformal Predictions can help us describe an image.

Our aim is that Pr(Correct Prediction in the Set) $\geq 1 - \alpha$. The prediction sets should be as small as possible and should be adaptive, which means that the harder the example, the larger the set should be.

We calculated the estimated probabilities of belonging to each class for all images in the validation dataset and sorted them from most likely to least likely. Then, we took the sum (in order of likelihood) of all estimated probabilities up until the true class. This represents the amount of estimated probability required to obtain the true class. It can be expressed as follows:

$$\sum_{j=1}^{k} \hat{f}(X)_{\pi_j}$$

where $\hat{f}(X)$ represents the probabilities and π_j is the j-th permutation of $\{1, ..., k\}$ ordering them. At this point, we can estimate the quantile using the validation data:

$$\hat{q} := \frac{\lceil (n+1)(1-\alpha) \rceil}{n}$$

Then, we perform estimations by taking the k most likely classes, so that the estimated probability required to obtain the true class is greater than or equal to the quantile of these scores.

$$k = \inf\{k : \sum_{j=1}^{k} \hat{f}(X)_{\pi_j} \geq \hat{q}\}$$

In our analysis, we set $\alpha = 0.05$ and thus expect a coverage of 95%. For EfficientNetB4, the coverage with single predictions was already high, so we set $\alpha = 0.005$ to obtain a coverage of 99.5%, which will probably lead to a higher size for conformal sets. The following table shows the different quantiles of the scores obtained in our models:

Method	$\hat{q}$
CNN	0.875
Random Forest	0.66
Support Vector Machines	0.828
Logistic Regression	0.999
EfficientNetB4	0.984

We encountered a problem with the Gaussian Naive Bayes algorithm. Due to the large number of feature, the algorithm was giving not reliable probabilities for each class for each image. We obtained probabilities equal to 1 in correspondence of the single predicted class and 0 for all the others. So, we were not able to compute correctly the conformal predictions and we did not include that model in this last analysis.

Finally, we can see the result on 5 images chosen from our test set consisting of images taken by us in different (and maybe controversial) weather conditions.

Figure 12: Test 1

In this image, we can see some clouds in the sky, but behind us is the sun, as we can see from the shadows on the sand.

Method	Prediction	Conformal Set
CNN	Rainy	Rainy
Random Forest	Rainy	Rainy, Snowy, Cloudy
Support Vector Machines	Rainy	Rainy
Logistic Regression	Rainy	Rainy
EfficientNetB4	Cloudy	Cloudy, Snowy, Rainy, Sunny, Foggy
Gaussian Naive Bayes	Sunny	/

All our models, with the exception of Gaussian NB, predicted a rain class for this image. Only RF has included the cloudy class in the conformal set. This can be explained by the very dark colour of the clouds.

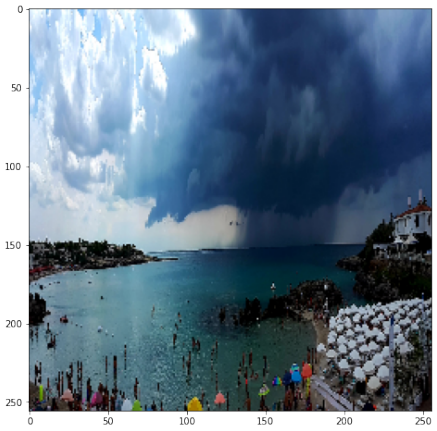

Figure 13: Test 2

This is a very interesting image. Half the sky was sunny with some clouds and the second half was rainy.

Method	Prediction	Conformal Set
CNN	Sunny	Sunny, Cloudy, Snowy, Rainy
Random Forest	Sunny	Sunny, Rainy, Foggy
Support Vector Machines	Snowy	Snowy, Cloudy, Rainy
Logistic Regression	Snowy	Snowy, Rainy, Cloudy, Sunny
EfficientNetB4	Cloudy	Cloudy, Snowy, Sunny, Foggy, Rainy
Gaussian Naive Bayes	Rainy	/

It can be seen that in most of the conformal sets there are sunny, cloudy and rainy at the same time. It is interesting to see that some models predicted 'snowy'. Perhaps this is due to the white beach umbrellas in the bottom right corner which, mixed with the dark sky, lead to an incorrect prediction. This was a very confusing image.

Figure 14: Test 3

Here, the condition is clearly snowy.

Method	Prediction	Conformal Set
CNN	Snowy	Snowy
Random Forest	Snowy	Sunny, Rainy
Support Vector Machines	Snowy	Snowy, Rainy
Logistic Regression	Snowy	Snowy, Rainy
EfficientNetB4	Snowy	Snowy
Gaussian Naive Bayes	Snowy	/

It was quite easy for all our models.

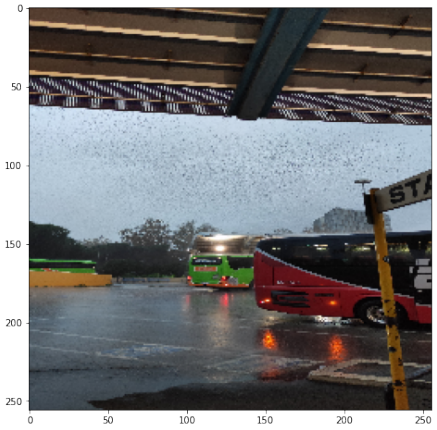

Figure 15: Test 4

This picture taken at the Tiburtina bus station in Rome shows a day of heavy rain.

Method	Prediction	Conformal Set
CNN	Rainy	Rainy
Random Forest	Rainy	Rainy, Sunny, Cloudy
Support Vector Machines	Rainy	Rainy
Logistic Regression	Rainy	Rainy
EfficientNetB4	Rainy	Rainy, Foggy, Snowy
Gaussian Naive Bayes	Rainy	/

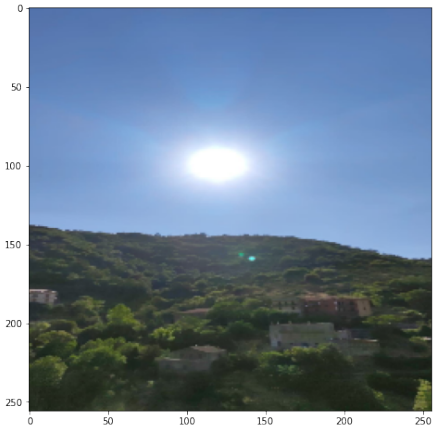

Figure 16: Test 5

This image was taken on a sunny day.

Method	Prediction	Conformal Set
CNN	Sunny	Sunny, Cloudy
Random Forest	Sunny	Sunny, Cloudy
Support Vector Machines	Sunny	Sunny
Logistic Regression	Sunny	Sunny, Cloudy
EfficientNetB4	Sunny	Sunny, Cloudy
Gaussian Naive Bayes	Sunny	/

Even for these last two images, all our models agree on one weather condition.

Conclusions

In conclusion, we can say that the conformal predictions helped a lot in this type of task to identify a suitable set of possible weather conditions. Looking at the models we implemented, the best seems to be the Random Forest applied to the features extracted with the CNN layers.

Improvements in this project can be various, e.g. a deeper tuning of the models can be attempted, which was not possible in terms of time and computational effort due to the limited capacity of the Google Colab notebook. The main goal may be to get closer and closer to the performance of EfficientNetB4.

This type of models can be applied in various fields where a certain weather condition is required to be detected and, in our opinion, with the implementation of the conformal sets of predictions they can do a good job.

References

Aaditya Ghag - *Weather Image Classification*.

Gurucharan M K - *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network*.

Martin Isaksson - *Use Case: Automated Weather Analysis Using Image Recognition* (2021).

Mingxing Tan and Quoc V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*.

Mohamed R. Ibrahim, James Haworth, Tao Cheng - Department of Civil, Environmental and Geomatic Engineering, University College London (UCL) *WeatherNet: Recognising weather and visual conditions from street-level images using deep residual learning*.

M. T. Ribeiro, S. Singh, C. Guestrin - *"Why Should I Trust You?" - Explaining the Predictions of Any Classifier*.

Ryan J. Tibshirani, Rina Foygel Barber, Emmanuel J. Candès, Aaditya Ramdas - *Conformal Prediction Under Covariate Shift*.

Sreenivas Bhattiprolu - *Python for microscopists - Classification with CNN and Random Forest*.

W. J. Murdocha, C. Singhb, K. Kumbiera, R. Abbasi-Aslb and Bin Yua - *Definitions, methods, and applications in interpretable machine learning*.