University of Cambridge

Computer Science Tripos

Part II Project

Time-Lapse Based Weather Classification

Author:
Roman Kolacz

Supervisor:
Advait Sarka

March 31, 2015

A logo/image/something cool

Proforma

Name:	Roman Kolacz
College:	Downing College
Project Title:	TIme-Lapse Based Weather Classification
Examination:	Computer Science Tripos - Part II
Word Count:	
Project Originator:	Alan Blackwell
Project Supervisor:	Advait Sarkar

Initial Project Aims

The initial aim of the project was to associate time-lapse video with external data sources by using a time-lapse video of an outside location to tell the weather. This would then be evaluated against real weather information and demonstrated graphically.

Work Completed

Over a months worth of data has been gathered and a classifier has been built which uses extracted image metrics and actual weather data to train a machine learning classifier which can then infer the weather. The accuracy of several machine learning algorithms has been tested and the best for each data set has been used to maximise overall accuracy. The initial plan was changed from using manually hard-coded rules for weather classification to using machine learning to do so instead.

Special Difficulties

None.

Decleration of Originality

I, Roman Kolacz of Downing College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this disseration and the work described in it are my own work, unaided except as may be specified below, and that the disseration does not contain material that has aready been used to any substantial extent for a comparable purpose.

Signed:

Date:

Acknowledgements

Advait Alan TODO: More detail here

Contents

1 Introduction **8**

1.1 Aims . . . 8

1.2 Challenges . . . 8

1.3 Related Work . . . 8

2 Preparation **9**

2.1 Requirements Analysis . . . 9

2.2 Data Gathering . . . 9

2.3 Time Lapse . . . 10

2.4 Machine Learning . . . 10

2.4.1 Random Forest . . . 10

2.4.2 Multilayer Perceptron . . . 12

2.4.3 J48 . . . 12

2.5 Choice of Tools . . . 13

2.5.1 Programming Language . . . 13

2.5.2 Libraries . . . 13

2.5.3 Backups . . . 13

3 Implementation **14**

3.1 Gathering Image Data . . . 14

3.1.1 Feature Extraction . . . 14

3.2 Data Storing . . . 15

3.3 Machine Learning . . . 15

3.4 Sanity Checking 15

4 Evaluation **17**

4.1 Cross Validation 17

4.2 Description of Data 17

4.2.1 Icon 18

4.2.2 Cloud Cover 18

4.2.3 Precipitation 19

4.2.4 Temperature 20

4.3 Analysis of Classifiers 20

4.3.1 Random Forest 20

4.3.2 J48 23

4.3.3 Multilayer Perceptron 23

4.4 Overall Results 24

5 Conclusions **25**

5.1 Future Work 25

5.2 Changes 25

Bibliography **25**

6 Appendices **27**

List of Figures

2.1 A picture of the Raspberry Pi and Camera setup 9
2.2 An example picture taken from the setup 10
2.3 An example decision tree for classifying the weather 11
2.4 Random Forest Graph of some sort 12

3.1 Processing Pipeline 14

4.1 Icon data bar chart 18
4.2 Some Graph for Cloud Cover 18
4.3 Some Graph for Precipitation 19
4.4 Some graph for Temperature 20
4.5 Icon graph per number of trees 21
4.6 Icon graph per number of trees 21
4.7 Icon graph per number of trees 22
4.8 Icon graph per number of trees 22
4.9 Icon graph per number of trees 23

Chapter 1

Introduction

1.1 Aims

1.2 Challenges

1.3 Related Work

Chapter 2

Preparation

2.1 Requirements Analysis

2.2 Data Gathering

A Raspberry Pi[1] with the Camera Module[1] was used to take pictures at a 5 minute interval out of my window overlooking the Downing College court.

Figure 2.1: A picture of the Raspberry Pi and Camera setup

At each interval, a new line would be added to a csv file which would contain the necessary

image metrics to train the classifier as well as actual weather data and basic information such as the name of the corresponding file, the time, date, etc.

Figure 2.2: An example picture taken from the setup

Raspberry pi, camera, forecast, etc

2.3 Time Lapse

Time lapse is a relatively unexplored way of compressing events which occur over extended periods of time into smaller, more managable videos.

2.4 Machine Learning

A large amount of research was devoted towards learning how different machine learning algorithms can be used and which would excel in different parts of the weather classification system.

2.4.1 Random Forest

A decision tree in machine learning is a tree used to reach a conclusion based on given parameters. At each branch, a path is chosen based on a condition, and the tree is traversed

in this way until a leaf is reached with an outcome.

Decision trees are incredibly simple to generate and use, though have a very high variance with results (even small changes in inputs can cause drastically different branch paths to drastically different results) as well as a tendency to overfit data (so outliers are significantly accounted for and general patterns are not formed), and are thus not usable as a reliable machine learning algortihm without modifications.

Figure 2.3: An example decision tree for classifying the weather

A random forest is a machine learning algorithm which builds upon decision trees and corrects for their variance and tendency to overfit. They work by using a random set of decision trees which individually classify the data- then either the mode or the mean of the results is used as the final outcome.

Figure 2.4: Random Forest Graph of some sort

Random forests run relatively accurately and also incredibly quickly and so were used from an early stage in the project to see if classification was working, and to give a rough estimate of the overall accuracy of classification (primarily to see if it was better than randomly guessing the weather).

2.4.2 Multilayer Perceptron

A single perceptron simply returns a 0 or 1 based on a given imput and a given activation function based on the input. It aims to divide the data into two sets and so the number returned determines which side of the division the given value should lie.

A single layer perceptron recursively trains itself and adjusts weightings of the output of each of the perceptrons within the layer on each recursion to improve the overall outcome of the classification (of the training set).

A multilayer perceptron uses multiple layers of perceptrons, including various hidden layers between the input and output layers.

2.4.3 J48

J48 is an open source implementation of the C4.5 algorithm which also uses a decision tree to build a classifier. At each node in the tree, the algorithm chooses the attribute which most effectly splits the set of training samples into consistent subsets. This, done recursively eventually forms the complete tree.

2.5 Choice of Tools

2.5.1 Programming Language

I used Java for the entirety of the project. This was chosen to ease development accross different operating systems, to be able to use the WEKA library (which my supervisor was very familiar with), and from sheer familiarity and simplicity of the language. Performance wasn't an issue as the classifying was done in a single run whenever results were required.

2.5.2 Libraries

WEKA

The WEKA[1] library was used for the machine learning portion of the project. It provides the functionality to train it's many included classifiers on CSV files with whatever options need to be set.

Some problems arose with training the classifiers to use a mixed data set (continious and discrete) which only existed in the newer versions of WEKA, so older versions were used until a workaround was found TODO: what was the workaround.

Forecast.io

The Forecast.io[1] API was used for gathering the real weather data. It provides an incredibly accurate forecast based on longitude and latitide for all the data which was required, and more.Forecast.io allows 1000 free queries daily, which is well above the amount required for the 5-minutely querying of my project.

The foreast.io API takes a longitude and latitude and returns JSON file with detailed weather information subdivided into minutely, hourly and daily; each of which contain data for several TODO: how many? units of time in either direction. I used a wrapper[1] library to deserialize the JSON format into classes with the relevant data.

2.5.3 Backups

Dropbox[1] and github[1] were both used to ensure the project was safe should my hard drive become inaccessible for whatever reason. This was done by creating the git repository within a dropbox folder. Versioning and branching wasn't used due to the linear nature of the project and the fact that the work was done by a single person on a single computer at a time.

Chapter 3

Implementation

Figure 3.1: Processing Pipeline

3.1 Gathering Image Data

3.1.1 Feature Extraction

Once images had been obtained with the hardware detailed above, image features had to be extracted in order to train the classifier with. Initially OpenCV[1] was going to be used, though soon into development, there was a realisation that there was little need for it.

The features extracted include:

- Average Hue
- Brightness Histogram

- Each of RGB histograms

These were obtained by sampling each pixel in the images using the java BufferedImage[1] class' ability to return the RGB vales of the pixels of an image in sequence as an array of bytes. The brightnss value of a set of RGB values was determined by the luminance function[1].

$$Y = 0.2126R + 0.7152G + 0.0722B$$

where Y is the luminance and R, G and B are Red, Green and Blue respectively.

The histograms are a size 256 array containg the number of pixels with the given RGBY value.

The average hue was obtained by simply averaging the RGB values of each of the pixels in each image.

3.2 Data Storing

Comma-Separated Value (csv) files were used to store the data in a convenient way for both anyone reading the data and for the WEKA library. The data stored included the name of the image, the time and date it was taken, as well as the actual weather information occuring in the image and the features extracted to train a classifier. Blank columns were left for the predicted weather information which would be filled in by the classifier once it was trained and ran.

3.3 Machine Learning

Once the features and real weather data had been stored in a csv file, the machine learning aspect of the project could begin. The data was split into training and testing with an 80:20 split such that 80% of the data was used to train a classifier and the remaining 20% was then used to evaluate the trained classifier. The data was split randomly, and placed into two seperate csv files.

The irrelevant columns (such as the name/date/etc) in the csv files then had to be filtered so that they were not used in the training of the classifier to not allow training using data which wasn't extracted from the images themselves.

3.4 Sanity Checking

Sanity checking ensures that everything is going well, and if it isn't, that a user responsible is made aware sooner rather than later to fix whatever problems may have arisen.

A sanity check which was employed in this project involved making sure that the Raspberry Pi was taking images and uploading them to dropbox continiously, as a large set of data was an integral part of the success of this project.

TODO: This section is a bit pointless... Remove?

Chapter 4

Evaluation

4.1 Cross Validation

Cross validation is a technique used to evaluate machine learning algorithms by testing them on 100% of the data set. In this case, the data was split into training:testing with an 80:20 ratio. This was done five times such that each of the testing splits were unique and formed the complete data set, so each point of data was used to train four classifiers and was tested by one (which it didn't train).

It is important not to train and test machine learning classifiers using the same data, as classifiers which don't learn (and rather, remember) would perform extraordinarily well.

4.2 Description of Data

The input data had to be analysed and described to give meaning to the evaluation.

4.2.1 Icon

Figure 4.1: Icon data bar chart

There were initial concerns with how varied the icon would be given the high likelihood of cloudy weather in the UK, especially during the period of data gathering. Fortunately, forecast.io distinguishes between cloudy and partly cloudy, with the latter being further distinguised into day and night classes. Due to this, the data is relatively spread out and a highly accurate classifier would not be able to be built by simply guessing cloudy all the time, as per the initial concerns.

4.2.2 Cloud Cover

Figure 4.2: Some Graph for Cloud Cover

As the graph above shows, the cloud cover is distribted qute uniformly between 0 (no cloud cover) and 1 (complete cloud cover), with the mean and median being almost equal. The standard deviation is 31% cloud cover, so the classifier should aim to classify within that percentage.

4.2.3 Precipitation

Figure 4.3: Some Graph for Precipitation

Much unlike the cloud cover, the precipitation is massively skewed toward the lower 10th percentile, which indicates that there was generally very little rain in the data gathering stage of the project. Deviation of 23%.

4.2.4 Temperature

Figure 4.4: Some graph for Temperature

The temperature is somewhat better distributed than the precipitation, though not as uniformly as the cloud cover. The data was gathered over the winter and early spring months, and so the temperature is generally very low with later dates having the higher temperatures. This would likely be a more uniform distribution if the data was to be taken over a longer period of time encompassing at least all four seasons.

Deviation of 2.9 degrees.

4.3 Analysis of Classifiers

The discrete data was analysed by simply comparing the percentage of correct classifications for each class. The continious data was analysed by calculating the root-mean-square error, given by

$$RMSD = \sqrt{\frac{\sum_{t=1}^{n}(x_t - y_t)^2}{n}}$$

where x is the actual value and y is the predicted value in each line of data.

4.3.1 Random Forest

Random forests were analysed with a varying number of trees ranging from 1 to 100, and then 200. Results wer otained with each of the number of trees to analyse how performance changes for different classes with the increase in trees.

Icon

Figure 4.5: Icon graph per number of trees

The accuracy of the classifier for icon increased with the number of trees up to about 55-56% at 50 trees. There was negligable increase in accuracy after this point, suggesting that the classifier has reached its potential for the feature set.

56% as an accuracy is rather good- well above randomly guessing and well above the mode value percentage.

Sunrise and Sunset

Figure 4.6: Icon graph per number of trees

Cloud Cover

Figure 4.7: Icon graph per number of trees

Precipitation

Figure 4.8: Icon graph per number of trees

Temperature

Figure 4.9: Icon graph per number of trees

4.3.2 J48

16% accuracy on icon and nothing else becuase it only works on discrete data.

4.3.3 Multilayer Perceptron

Icon

The multilayer perceptron had a 36.1% accuracy rate with classifying the icon. This is marginally better than assigning the mode result to every instance, and also better than the average case of randomly assigning icons. However, this is by no means a reliable classifier for classifying the icon as 36% is a generally low success rate for classification (it's wrong more often than it's right). Other options will have to be explored.

Sunrise and Sunset

The sunrise and sunset had a root-mean-square error of 8:12 and 5:11 minutes respectively. TODO: data descriptor for this

Cloud Cover

The multilayer perception had a 35% root-mean-square error for cloud cover, which is outside of the 31% standard deviation, though only just. Being within two standard deviations, it's still a somewhat accurate way of deducing cloud cover, though not ideal.

Precipitation

The precipitation predictions had a root-mean-square error of 33%, which is well outside of the stanard deviation of 23%.

Temperature

The multilayer perceptron had a 1.6 degree root-mean-square error when telling the temperature, which is well within a standard deviation of 2.9 degrees. This is an excellent result and is unlikely to be beaten by the performance of any other classifiers.

Summary

Overall the Multilayer Perceptron is a rather inaccurate classification algorithm in all cases with the exception of the rather surprisingly accurate temperature reading.

4.4 Overall Results

Summary of overall results/accuracy, etc?

See which pair of in/out was the most and least successful with classification. Maybe pictures of which were the most innacurate??

Chapter 5

Conclusions

5.1 Future Work

Extensions, etc? - More classifiers - Better feature extraction- perhaps a histogram for each segment in the image (divide into 9 or something)

5.2 Changes

Could be merged with above?

Bibliography

[1] Missing citation.

Chapter 6

Appendices

Roman Kolacz
Downing College
rk476

Part II Project Proposal

Time-Lapse Based Weather Classification

19th October 2014

Project Originator: *Alan Blackwell*

Resources Required: See attached Project Resource Form

Project Supervisor: *Advait Sarkar*

Signature:

Director of Studies: *Robert Harle*

Signature:

Overseers: *Markus Kuhn* and *Neal Lathia*

Signatures:

Introduction and Description of the Work

Time-lapse videos are a relatively unexplored aspect of data visualisation, and have a rather unique property amongst video of being able to hold information about incredibly large periods of time in a relatively small amount of time and space. This is especially true in the case of events which happen slowly and gradually, such as the weather conditions in a particular place over a long period of time.

This project will aim to develop an alternative way of associating time-lapse videos with external data sources by attempting to determine the weather from time-lapse footage and evaluating it against true weather information. This will be demonstrated graphically.

The time-lapse video will be made as part of the project, though any existing video will also suffice for the project.

Literature Review

Classification of Weather Situations on Single Colour Images

An approach which forms the basis of the weather-labelling per frame is described in the paper "Classification of Weather Situations on Single Colour Images"[4]. This paper describes classifying weather using image metrics to be used to assist driving-assistance systems which are heavily dependent on clear conditions. It achieves very accurate results using only two labels (clear and rain), and reasonable accuracy when adding a third, "light rain" label. This project will build on this by adding further labels to the weather classifying, as well as incorporating a cloud-cover measure.

The paper also mentions inaccuracies relating to weather situations containing heavy fog or rain, as visibility is greatly reduced. These will have to be taken into consideration and/or mentioned in the evaluation.

Weather Identifying System Based on Vehicle Video Image

"Weather Identifying System Based on Vehicle Video Image"[5], describes using image comparison to get the weather conditions on a road. A 'background' image is obtained (and updated should a more suitable image be detected), and then weather conditions are inferred by analysing the images obtained by subtracting each of 1000 frames in a video from the background image. Although this technique isn't going to be used to label weather in the approach which will be presented in this project, a similar technique may very well be employed to determine cloud cover percentage.

Classification of Road Conditions

The paper, "Classification of Road Conditions"[6], also applies weather classification to driving conditions. It uses an inverse-least-squares method of inferring the weather from not only images, but a multitude of weather data including temperature, humidity and wind speed. This project will not be using these additional variables, though the results will likely be evaluated with their aid.

Time-Lapse Photography Applied to Educational Videos

Although unrelated to weather classifying, "Time-Lapse Photography Applied to Educational Videos"[7] explores applying time-lapse videos to educational videos to aid with analysing long term trends and discarding short term 'noise'. This is comparable to what this project is aiming to achieve through the use of time-lapse video.

Two-Class Weather Classification

"Two-Class Weather Classification"[8] also uses image processing techniques to classify weather into two distinct labels: cloud and sun. It initially explores analysing pixel brightness distribution, which appears to fail in a general case. The paper then explores various techniques for looking for signs of a sunny day. One such technique involves detecting the sky in the image (something which this project may well have to incorporate) and checking for a blue hue. The paper also explores using edge detection to find shadows, which is found to be very vulnerable to false positives in a cloudy case; as well detecting bright reflections in reflective objects- two techniques which will not be employed in this project, though may be useful for discussion purposes.

Resources Required

- Obtaining the data will require a Raspberry Pi[1] with an attached camera module.
- Depending on the frequency of frame capture and resolution, some additional disk space may be required to store the data.
- OpenCV[2] will need to be installed.
- If the machine-learning part of the project is to be undertaken, Weka[3] will need to be installed.

Substance and Structure of the Project

The OpenCV library will be used to obtain the following image metrics from individual frames:

- Brightness
- Average Hue/Saturation
- Brightness curve

This data will then be analysed and compared to the images so that a set of hard-coded rules can be derived which will determine which weather label can be attached to each image.

Frames in which the sun is shining are likely to have have brightness curves with very high peaks and low troughs, in contrast to relatively flat curves of cloudy images. This would come as a result of shadow and reflections skewing the curve in sunlit images, whereas cloudy images will be generally more evenly (and dimly) lit as a result of the clouds dispersing and absorbing sunlight.

Snow will likely cause the entirety of the image to generally appear brighter than expected and with less saturation, as a result of the reflective properties of snow.

Rain will likely be the most challenging weather label to place, as the conditions will decrease the brightness of the frame in a similar way to dawn and dusk would, and otherwise be relatively similar to cloudy conditions. One potential situation to look out for would be reflections in puddles, though these aren't guaranteed to happen, and may have insignificant effects on the image even if they do. A probable situation is that the saturation of the frame decreases and the hue shifts towards a blue/grey shade, though this will need to be explored during the progress of the project, and the accuracy of which will need to be heavily analysed.

The cloud cover will be inferred by isolating the sky from the rest of the image, and then analysing the brightness and hue of the relevant part of each frame.

Evaluation

The weather classifier will need to be evaluated alongside real weather data. This will be obtained from forecast.io[9], which contains an API[10] for obtaining precise (latitude and longitude) weather information. This is free to use for up to 1000 polls a day, which will easily suffice for this project.

The data will be split into training and testing sections with a 4:1 split respectively, as (especially in the extension case involving machine learning), data which is used to train

the classifier will have a positively skewed accuracy. The testing section will be used to evaluate the classifier.

Each aspect of the classifier will then be evaluated separately.

Weather Labels

The weather labels, being discrete sets of data, will be tested for accuracy by simply calculating the percentage of correctly assigned labels. A truth matrix will be created to highlight the concentration of errors on a per-case basis.

Cloud Cover

The cloud cover will be calculated as a percentage and compared to the real value which is available from the Forecast API. The accuracy of this will be manually checked, given that a camera pointing outside will cover a relatively large section of the sky. The accuracy of this aspect of the classifier will be calculated using root mean square error.

Sunset and Sunrise

The sunset and sunrise times will be tested for accuracy by also using root mean square error, using the difference between actual and inferred time, in minutes.

Extensions

Depending on the progress of the project at specific points, the following additional features may be implemented, in order of priority:

- Multiple images per frame may be taken at varying exposure and gain settings as a large difference in luminance within a single frame (which will likely happen on a sunny day), may make other parts of the image unnaturally dark should the camera use automatic settings.
- Weather prediction and likely trends can be derived from existing data. Information predicted in such a way can be compared to inferred data and accurate data when the related time period passes.
- Machine learning can be used instead of hard-coded rules on determining weather, as both the image metrics and accurate weather information will be available as a learning data set. Weka will be used for this.

Success Criterion

For the project to be deemed a success the following items must be successfully completed.

1. The sunrise and sunset times can be inferred from a time lapse video.
2. The weather in each frame can be inferred and labelled one of the following:
    - Clear
    - Cloud
    - Rain
    - Snow
3. The cloud cover can be inferred from each frame within the video.
4. A measure of confidence can be assigned to each of the above criteria to reflect the probability of incorrectly inferred data.
5. The above data can be graphically demonstrated and shown to work to some degree of accuracy.
6. The accuracy of the data must be evaluated.
7. The dissertation must be planned and written.

Timetable and Milestones

24th October - 6th November

Research how to use the Raspberry Pi and camera module to take time lapse videos, and set up a fixed position for it to gather data from for the rest of the project.

Research the OpenCV library and how it can be used to gather the data required, as well as how the data gathered can be used to infer the conditions described in the success criteria.

Milestones

- Produce a short, not necessarily usable time lapse video using the Raspberry Pi.
- Begin gathering data for remainder of project.
- Found/made a relatively hassle free mount for the Raspberry Pi and camera.
- Produce some small demo's and examples using OpenCV.

7th November - 18th December

Use the research gathered to design and implement algorithms which can establish weather conditions from frames based on hard coded rules.

Milestones

- A working library for inferring weather conditions from images.
- Tests written for said library.

Key Dates

- NPR End: 14th December

19th December - 1st January

Review algorithms and adjust timetable if further tests and research is needed for a working, usable library.

Implement the library to the thus-gathered time lapse video, and design a graphical demonstration program to view the data through. Assign a confidence value to each frame to reflect the chance of a correct result.

Milestones

- A graphical visualisation of the data extracted from the time lapse video.
- A confidence measure for the weather at each given frame.

2nd January - 29th January

Work on evaluating the data by gathering accurate weather information for each obtained frame and automatically labelling them with said data. Compare some specific instances to vaguely gauge the accuracy of the program.

Milestones

- Accurate data gathered for evaluation.
- A general idea of how accurate the data is.

Key Dates

- NPR Start: 4th January

30th January - 19th February

Work on more formally evaluating the accuracy of the data, along with the reliability of the confidence measure assigned to each piece of data. This can also be graphically shown.

Milestones

- Complete data on accuracy of algorithm.
- Graphical visualisation of said accuracy.

Key Dates

- Progress Report Deadline: 30th January
- Progress Report Presentation: 5th, 6th, 9th or 10th February
- Dissertation Lecture: 11th February

20th February - 5th March

Review work and adjust timetable if needed. Enhance test program for a more friendly UI which is more suitable for demonstrative purposes.

Milestones

- Program should be fully complete, usable and presentable.

6th March - 2nd April

Write dissertation. Notes should have been gathered throughout the previous stages, so these should be read first for familiarisation of the previous weeks of work.

Milestones

- First draft of dissertation should be complete.

Key Dates

- NPR End: 15th March

3rd April Onwards

Take a look at what needs attention, if anything, and work on that. Look through dissertation and ensure it's up to submission standards.

Milestones

- Final dissertation should be complete.

Key Dates

- NPR Start: 12th April
- Project Deadline: 15th May
- Supervisor Report Deadline: 20th May
- Exams: 2nd-4th June
- Viva Announcement: 12th June
- Vivas: 15th June
- NPR End: 21st June

References

[1] Raspberry Pi Computer
`http://www.raspberrypi.org/`

[2] Open CV Library
`http://opencv.org/`

[3] Weka 3 Library
`http://www.cs.waikato.ac.nz/ml/weka/`

[4] Classification of Weather Situations on Single Colour Images
Martin Roser and Frank Moosmann, 2008
`http://www.mrt.kit.edu/z/publ/download/Roser_al2008iv.pdf`

[5] Weather Identifying System Based On Vehicle Video Image
Hong Jun Song, 2011
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5970722

[6] Classification of Road Conditions
Patrik Jonsson, 2011
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6059917

[7] Time-Lapse Photography Applied To Educational Videos
Wei Liu and Hongyun Li, 2012
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6202303&tag=1

[8] Two-Class Weather Classification
Cweu Lu, Di Lin, Jiayi Jia and Chi-Keung Tang, 2014
http://www.cse.cuhk.edu.hk/leojia/papers/weatherclassify_cvpr14.pdf

[9] Forecast.io
https://forecast.io/

[10] Forecast.io Developers API
https://developer.forecast.io/docs/v2