UNIVERSIDAD DE BURGOS
Escuela Politécnica Superior
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Sky Classification through Supervised Machine Learning

Presented by Bardia Filizadeh
in the University of Burgos — June 12, 2024
Tutors: Diego Granados López - Nuño Basurto Hornillos

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dr. Diego Granados Lopez, assistant professor of the Department of Digitization, area Computer Science and Artificial Intelligence.

Dr. Nuño Basurto Hornillos, assistant professor of the Department of Digitization, area Computer Science and Artificial Intelligence.

Expone:

That the student Bardia Filizadeh, with DNI L1VC1YV66, has completed the final Degree Project in Computer Engineering entitled Sky Classification through Supervised Machine Learning.

And that said work has been carried out by the student under the direction of the undersigned, by virtue of which its presentation and defense is authorized..

En Burgos, June 12, 2024

V^{o}. B^{o}. del Tutor:	V^{o}. B^{o}. del co-tutor:
Dr. Diego Granados López	Dr. Nuño Basurto Hornillos

Resumen

La clasificación del cielo es un aspecto importante en muchas áreas como la meteorología, específicamente la investigación atmosférica y también para usos arquitectónicos. Sin embargo, tanto la adquisición como el procesamiento de datos de clasificación del cielo según el estándar CIE (Commission Internationale de l'Eclairage), puede resultar una tarea costosa tanto en recursos como en fondos. En este proyecto se ha investigado y desarrollado una forma alternativa de realizar la clasificación del cielo según el estándar CIE basada en el uso de Redes Neuronales Artificiales (RNAs) y su entrenamiento utilizando un conjunto de datos de imágenes del cielo proporcionado por la Universidad de Burgos y diferentes combinaciones de parámetros de red y funciones de entrenamiento. Basado en una investigación previa realizada sobre el uso de preprocesamiento de imágenes para la clasificación del cielo mediante el uso de un RNA [8]. Este proyecto se centra en la comparación de métodos de entrenamiento alternativos que se pueden utilizar para lograr resultados aceptables además del uso del método de gradiente conjugado escalado para el entrenamiento. Los métodos de entrenamiento elegidos son retropropagación resiliente, descenso de gradiente con impulso y retropropagación de tasa de aprendizaje adaptativo, y secante de un paso.

Descriptores

Clasificación del cielo estándar CIE, aprendizaje automático, gradiente conjugado escalado, escáner del cielo, red neuronal artificial, validación cruzada, función de entrenamiento, resplandor del cielo, Retropropagación resiliente, descenso de gradiente con impulso y retropropagación de tasa de aprendizaje adaptativo, secante de un paso.

Abstract

Sky classification is an important aspect of many areas such as meteorology, specifically atmospheric research and also for architectural uses. However both the acquisition and data processing of sky classification under the CIE (Commission Internationale de l'Eclairage) standard, can be a costly task in both resources and funds. In this project, an alternative way to perform sky classification by the CIE standard has been researched and developed based on the use of Artificial Neural Networks (ANNs) and their training using a dataset of sky images provided by the University of Burgos and different combinations of network parameters and training functions. Based on a previous research done regarding the usage of image pre-processing for Sky Classification by using an ANN [8]. This project focuses on the comparison of alternative training methods that can be used to achieve passable results aside from the use of the Scaled Conjugate Gradient method for training. The chosen training methods are Resilient Backpropagation, Gradient Descent with Momentum and Adaptive Learning Rate Backpropagation, and One-step Secant.

Keywords

CIE standard sky classification, machine learning, scaled conjugate gradient, sky scanner, artificial neural network, cross-validation, training function, sky radiance, Resilient Backpropagation, Gradient Descent with Momentum and Adaptive Learning Rate Backpropagation, One-step Secant.

Contents

List of Figures

List of Tables

1. Introduction

The importance of sky classification can be seen in various fields such as architectural design[11][1], environmental studies[12] and the installation of solar energy[10]. The traditional way of achieving the sky classification is under the CIE (Commission internationale de l'Eclairage) standard: CIE S 011/E:2003[33].The CIE standard defines a set of 15 sky types. For each one, an angular distribution of luminance[1] is proposed.

One traditional method of performing this sky classification, is through the use of a sky scanner, which is a device that measures the skies luminance and applies mathematical formulas to the acquired measurements to determine the sky's characteristics[27]. The results of these measurements and calculations are very accurate, however, they are very expensive [6]. Another downside, aside from the high cost, is that the sky scanner has a very low spatial resolution of 145 measurements and a temporal resolution of 5 minutes per scan[8].

In order to alleviate some of these issues, this project proposes the usage of an Artificial Neural Network (ANN), which, in recent years, have emerged as a popular alternative to help with the ever increasing abundance of data. It's use for data mining comes from its fast and autonomous computation of large sets of data, while delivering good results. These ANNs are computational models, inspired by the complexity of the human brain, with the capacity of learning from data and making predictions or classifications based patterns detected within the learned data.

As aforementioned, ANNs can handle large datasets rapidly and efficiently, with the capability of delivering fast predictions/classifications once the network has been trained. Furthermore, ANNs offer a dynamic and

[1]Luminance being the intensity of visible light emission

adaptive way to handle data from various diverse environments and conditions, as they can simply be trained for its required environment. As we also don't require the purchase of a sky scanner, the saved costs of the initial purchase and further future cost of maintenance can be completely saved by using an ANN. It is also important to mention that the acquisition of a high quality sky scanner for sky luminance data gathering is scarce[8].

There are however a few initial downsides to this alternative. In order for the ANN to classify accurately, it requires an extensive dataset of sky images for each sky type defined in the CIE standard. To do this, a digital camera must be set up to capture sky images through a fish-eye lense[8]. The camera will take pictures at a regular interval, while a sky scanner performs scans of the same sky condition and classifies it. The classification process of the scanner is automatic and fast. Once enough images and scans have been acquired, the sky type labelling by the sky scanner needs to be evaluated. This is very time consuming, as the predictions are either compared through visual inspection of the data and/or by comparing the luminance prediction. Luminance prediction can be done using a statistical index, such as the root mean square error. Visual inspection can be very tedious, as it must be done one sample at a time. Data acquisition is the most time consuming part of using an ANN, as the dataset collection and preprocessing is one of the most influential factors in how well the ANN will classify in the end, therefore it is most important to focus on a well maintained and consistent database.

Nevertheless, once a functional ANN has been obtained, its potential use cases can be endless. ANNs could be integrated inside cities for urban monitoring systems, which dynamically adjust the street lighting in accordance to the current sky conditions, thus improving energy efficiency and safety. [2] ANNs could also help with solar energy farms, as it could increase the accuracy of solar power generation forecasts, as it provides precise data on the current sky condition. For environmental studies, the ANN can help forward the studies of sky conditions on local ecosystems and climate. Architects may also use ANNs to make a detailed analysis of the sky conditions for specific locations, in order to assist with the design of buildings, to optimize their usage of natural lighting, which leads to less energy usage by inhabitants.

Previous research done on this topic has focused on image pre-processing to improve ANN performance using the Scaled Conjugate Gradient method for network training. [8] This project however, will be focusing on alternative training methods, that may achieve similar or even better results than the aforementioned training method. We will thus be looking at different training methods and their classification results. The chosen training methods are Resilient Backpropagation (RP), Gradient Descent with Momentum and Adaptive Learning Rate Backpropagation (GDX), and One-step Secant (OSS).

RP is the fastest algorithm on pattern recognition problems and requires lower memory requirements when compared to other algorithms for these types of problems. GDX is usually slower than the other methods, but can still be useful for some problems. Its slower convergence can come in handy for some situations. OSS serves as a bridge alternative between 2 different algorithms and brings its own advantages and disadvantages, which will be shortly touched upon in the "Techniques and Tools" sections. SCG, which was used in the previous work, is a very balanced algorithm that can perform well over a wide variety of problems and is relatively fast on pattern recognition problems [15]. It will be used in this work as a comparative measure that has been tried, tested and proven to be of use in sky classifications using ANNs.

2. Project Objectives

This chapter defines the objectives pursued by the project. The main purpose of this study is: to use Artificial Neural Networks (ANNs) as an alternative for sky classification by the CIE standard, as well as exploring different training functions for the Artificial Neural Network.

General Objectives

1. Investigate the applicability of using ANNs for sky classification under the CIE standard.
2. Use different training functions for the network and compare their performance metrics.
3. Evaluate whether other training functions aside from the Scaled Conjugate Gradient are a viable option for sky classification ANNs.
4. Evaluate the performance of using one of the highest scoring image-processing methods from previous research [8], with the aforementioned training functions.

Technical Objectives

1. Research the basics of sky classification and its use cases.
2. Setup sky image dataset provided by the UBU in such a way that it may be used for ANNs.
3. Load the images into the ANN working environment.

4. Resize images into an even format, so that its computational cost is not too high to not be usable.
5. Extract class labels using the Matlab Deep Learning Toolbox.
6. Choose multiple training functions that focus on the usage of back-propagation.
7. Train the network using each training function and a different combination of training parameters and compare the results.
8. Perform the training again, but this time with image-processing technique applied (the panchromatic channel of each image).
9. With the acquisition of passable results, execute training and testing of the network again, this time using cross-validation.
10. Extract network variation with the best performance.

Personal Objectives

1. Apply my knowledge acquired about Artificial Intelligence and Neural Networks throughout my exchange semester here through the Computer Engineering Degree.
2. Further deepen my knowledge on using the Matlab Deep Learning Toolbox for the development of Artificial Neural Networks.
3. Delve into the field of sky classification and its use cases.
4. Deepen my understanding of ANNs and the training parameters.
5. Learn more about pattern recognition and its application using supervised machine learning.
6. Expand my understanding of multi-class classification problems and their implementation using pattern networks.

Report Structure

The report structure is given below, to give a short overview of the content of each chapter

- Introduction: a short summary of the purpose and motivation for this project

- Objectives: the desired goals and purposes of the project are specified, both on a general and technical level, as well as a personal level.
- Theoretical Concepts: a short inauguration into the basic concepts that will be discussed in this project, mostly in regards to ANNs.
- Techniques and Tools: the main tools and techniques used in the study for development and programming tasks. Explanation of their characteristics and what they have contributed to the work.
- Conclusions: critical assessment of the research results.

3. Theoretical Concepts

3.1 Basic concepts

Machine Learning

Machine Learning (ML) describes a subset of Artificial Intelligence (AI) that focuses on the development of algorithms that enable computers to learn and make decisions based on data. Systems can also improve their performance on a specific task over time without being explicitly programmed for every scenario, thus making it dynamic. [3] When talking about ML we can differentiate between 3 main types of learning; **Supervised** Learning, **Unsupervised** Learning and **Reinforcement Learning**. Each type has its own methodologies and applications.

Unsupervised Learning

During unsupervised learning, a model is trained on data without labels or specified target output. Its purpose is to identify patterns or structures within the data without outside help. Common techniques used are clustering (e.g. k-means, hierarchical clustering) and dimensional reduction (e.g., PCA, t-SNE).

Supervised Learning

Supervised learning on the other hand, involves training a model on a labeled dataset, which means that each training sample is paired with an output label/value. The goal is to learn a mapping from inputs to outputs. Common techniques include regression and classification. After each training prediction, the prediction provided by the model is compared to the actual label/value. If the label/value is correct, training continues. If it isn't correct, then the model is adjusted according to a specified rule within the model and training is continued.

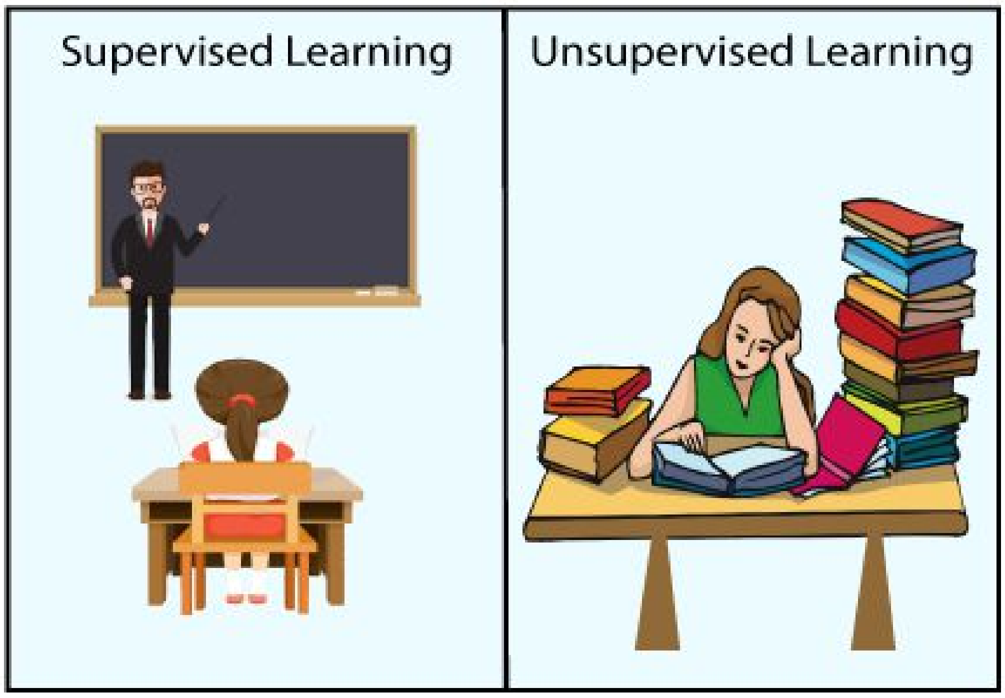


Figure 3.1: Example Comparison of what Supervised and Unsupervised Learning can look like [9]

Reinforcement Learning

Reinforcement learning serves as a middle ground between supervised- and unsupervised learning. It works based on the trial and error principle [2]. It differentiates itself from the previous 2 learning methods by being told during training whether an output was incorrect, but without being told the "solution" in this case. Therefore the model will have to adjust itself with only the directive that the previous output was incorrect. This leads to a severe increase in iterations as the model will have to try repeatedly to figure out the correct adjustment direction to go into. It can be described as a model that learns through experience from its environment [21].

[2]A fundamental problem-solving method that consists of performing repeated attempts to solving a problem, until success is achieved.

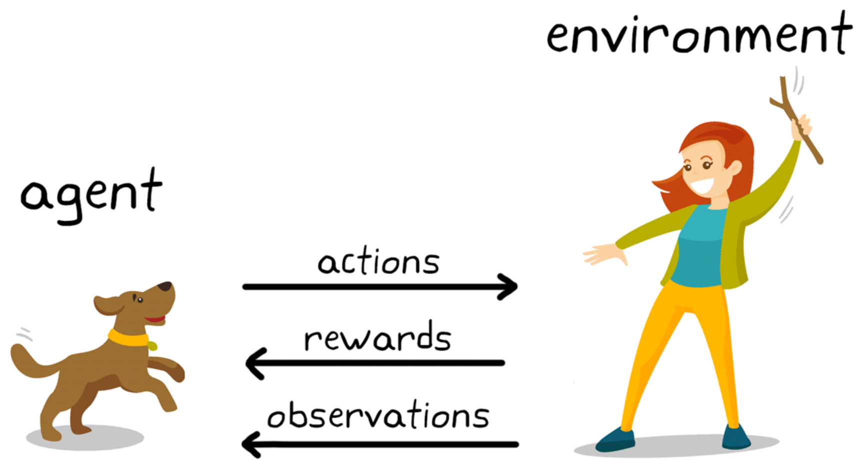


Figure 3.2: Reinforcement Learning Real Life example [21]

Artificial Neural Networks (ANNs)

In this work we will be using the AI tool known as Artificial Neural Networks (ANNs). ANNs are adaptive systems that adapt by using interconnected nodes or neurons in a layered structure (hence neural network), which resemble that of a human brain [13]. The adaptation is achieved by learning through data, so that the network may be trained to classify data, recognize patterns and even forecast future events based on minor data correlation. Networks consist of multiple layers which are interconnected, each layer containing a set amount of neurons. There is an input layer, one or multiple hidden layers and one output layer. Each neuron is a computation unit that processes its inputs and passes an output to the subsequent layers. Once an input has passed all layers, the output layer computes the final output.

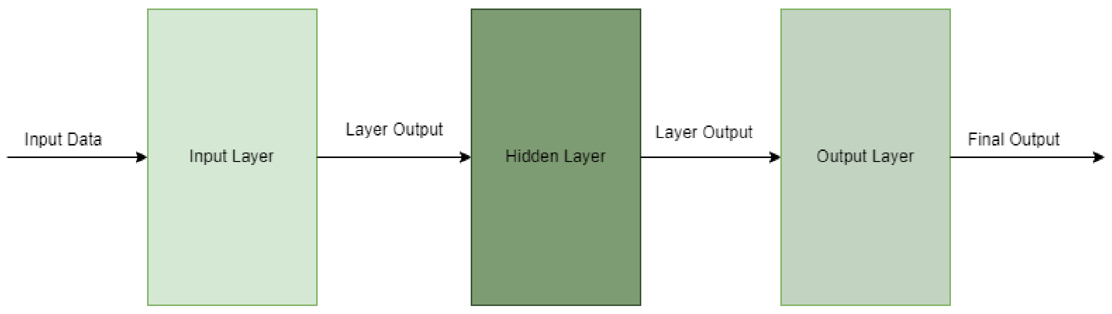


Figure 3.3: Artificial Neural Network Basic Structure

Layers

Input Layer: Recipient of the initial data, each neuron in this layer represents a feature in the input data.

Hidden Layers: Layer(s) between input and output layer. Perform transformations on the inputs using a series of weights between the edges of neurons and **activation functions** on the sum of the input per neuron. This computation process is performed by every neuron in all layers.

Output Layer: Returns the final output of the network, the given solution of the network for the given task (e.g. classification or regression).[7]

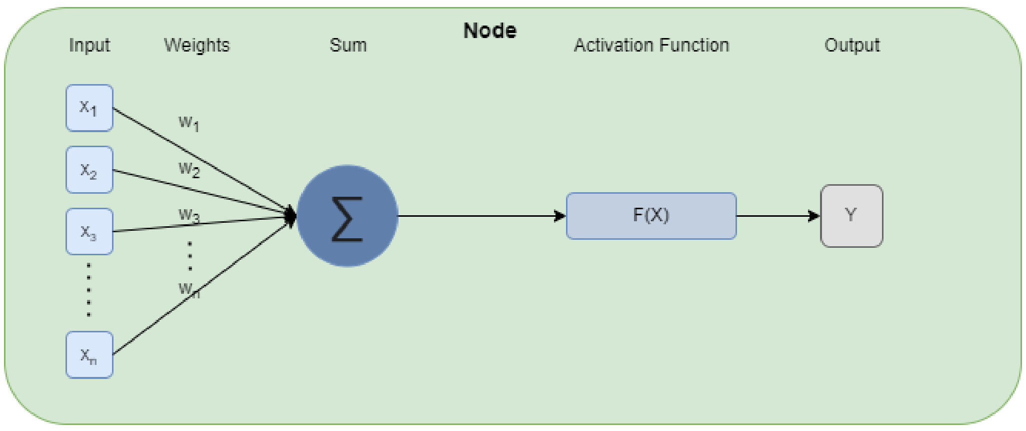


Figure 3.4: Node Structure

Weights

The ANNs ability to perform correctly all depends on the weights being correct on each edge between neurons, so that the input can be transformed into the desired output. The weights of the network are adjusted during training. In addition to weights, there is a possibility to add an additional parameter to the edges called **bias**. This bias can be used to shift the input into a more desired direction, and can help with performance for specific models. [7]

Activation Functions

As mentioned before, each neuron applies the sum of their inputs to the activation function. This function returns a new output which is supplied to the next connected neuron/layer. Using these functions allows the network to learn more complex patterns, making the model less linear. The most common activation functions are **Sigmoid** functions, which return an output value between 0 and 1, often used for classification problems, **Rectified Linear Unit (ReLU)** functions, that forward the input directly if its a positive number, else it outputs 0, and **Tanh** functions, which output values are between -1 and 1.

Network Training

In order for the ANN to be able to perform a given task, it has to be trained on the desired task first, by giving it example data that represents the problem at hand. As we will be performing supervised learning on the ANN, we provide the network with input data of the given problem, along with the desired output for that input data. Before we start training we must first specify a few additional key parameters for the training process.

Learning Rate: Controls by how much the network weights are adjusted, a smaller learning rate can make the training process slow, while on the contrary a large learning rate can cause the training process to converge too quickly.

Epochs: The number of epochs specifies how many times the training process passes the entire input dataset through the network, depending on the complexity of the problem, a higher number of epochs can allow the network more opportunity to become familiar with the problem.

Number of Hidden Layers: This determines the depth of the network. Adding more hidden layers to the network allows for more complex functions to be adjusted to, but requires more computational resources. Additionally, the risk of **overfitting** (the networks ability to generalize, degrading due to it leaning itself too much on the training data) increases.

Number of Neurons per Hidden Layer: In the same sense as the number of hidden layers, the number of neurons per hidden layer can increase the networks ability to capture more features, but also increases computational complexity and the risk of overfitting.

Loss Function: Measures the difference between the predicted output of the network and the actual desired target value. Chosen depending on the type of problem (e.g. Cross-Entropy Loss for classification problems, Mean Squared Error for regression problems).

Weights Initialization: The Method used to set the initial values of the network's weights.

Stopping Criteria: It's possible to specify a set stopping criteria to end the training process early, such as: after a fixed number of iterations, training time, error threshold etc. This is done to prevent overfitting and also save unnecessary computation time.

Training Function: This function defines how the weights and bias values are updated after each input.

Once training starts, the network's weights are adjusted constantly using the loss function output and the training function to lean further into the correct direction of the problem, until a stopping criteria is met. Then it is time to test the network using a separate set of data, the test set. The test set data has to be different from the training data as to measure the network's ability to generalize, that being its ability to provide appropriate output regardless of the input data having previously been presented to it or not.

The CIE standard sky classification

Sky classification by the CIE (Commission Internationale de l'Eclairage) standard is defined into 15 standard sky types, each one representing a specific distribution of sky luminance. The sky types range from overcast,partial and clear sky conditions. It is based on theoretical models and empirical data that describe how the light from the sky dome is distributed across different parts of the sky under various weather conditions. The light distribution of each sky can be described mathematically using specific equations. Each type, takes into consideration different atmospheric conditions to clearly differentiate between the types that are within the similar range (overcast, partial and clear).

The usage of this standard are very important for architectural design[11] [1], atmospheric research and solar energy calculations [12], as knowing the the average sky luminance of an area can help in designing buildings with optimal natural lighting use, which reduce the need for artificial light and therefore improving energy efficiency [23].

The different sky types can be shortly described as follows[5]:

- **Type 1**: Overcast sky with steep gradation[3].
- **Type 2**: Overcast with steep gradation and slight brightening towards the sun.
- **Type 3**: Overcast with moderate gradation.
- **Type 4**: Overcast with moderate gradation, with slight brightening towards the sun.
- **Type 5**: Overcast, foggy or cloudy.
- **Type 6**: Partly cloudy, with uniform gradation and slight brightening towards the sun.
- **Type 7**: Partly cloud, with uniform gradation and bright circumsolar[4] effect.
- **Type 8**: Partly cloudy, rather uniform, with a clear solar corona [5].
- **Type 9**: Partly cloudy, with obscured sun.
- **Type 10**: Partly cloudy, with brighter circumsolar region.

[3]Gradual transition of sky colors and brightness due to different atmospheric conditions.

[4]Revolving or surrounding the sun

[5]Outermost part of the sun, being able to see the "sun's circular form" clearly, so to say.

- **Type 11**: White–blue sky with distinct solar corona.
- **Type 12**: Clear sky, low illuminance turbidity[6].
- **Type 13**: Clear sky, polluted atmosphere.
- **Type 14**: Cloudless turbid sky[7] with broad solar corona.
- **Type 15**: White–blue turbid sky with broad solar corona.

[6]The effect of atmospheric conditions (water vapor, pollutants etc.) on the distribution and intensity of light.

[7]Turbid sky refers to a sky that appears hazy, muddy or opaque due to the presence of turbidity(see previous footnote 6).

4. Techniques and Tools

This chapter will describe the techniques and tools used in the project, as well as the structure of the data used to carry out the project.

4.1 Data Description

The data used to train and test the ANN proposed in this study were experimental sky images recorded at a meteorological weather station located on the roof of the Higher Polytechnic School building at the University of Burgos [8]. The photographs can be classified into 15 different types as per CIE standard [23] depending on the sky condition and luminance distribution. Each image has a resolution of 1158x1172 pixels, recorded with the RGB color model (8 bits per pixel, ranging values from 0 to 255). In Figures 4.2, you can see examples of each sky type. The overcast type covers types 1-5, partially overcast covers 6-10 and clear sky covers types 11-15. The dataset consists of 1500 images in total (selected from more than 80000 sky images). 100 images have been provided for each sky type. The images selected were characterized by their superior consistency with the pattern defined by the CIE standard for the specific sky category, thus making for great samples to be used to train the neural network.

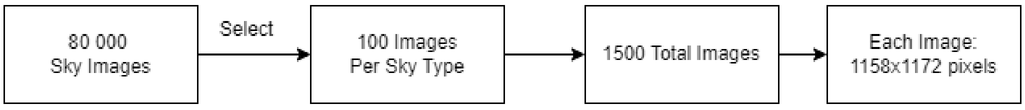


Figure 4.1: Simple Visualization of the dataset structure

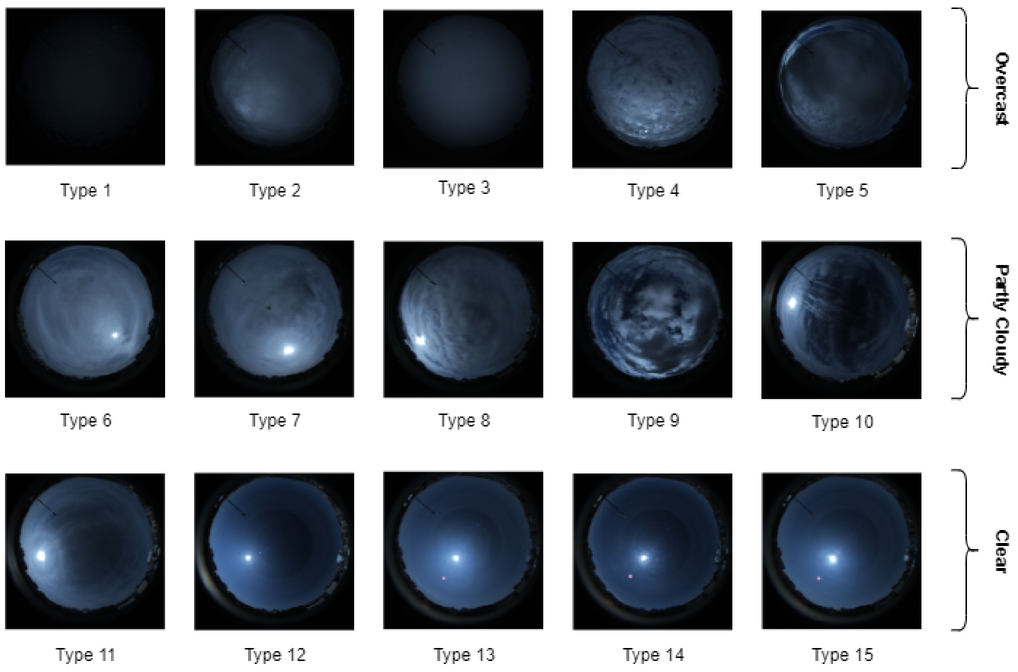


Figure 4.2: Images of all sky types recorded in Burgos, Spain

4.2 Techniques and Tools

Languages

MATLAB

MATLAB is a high-level programming language designed for engineers and scientists that expresses matrix and array mathematics directly. It can be used for basically anything, from running simple interactive commands to developing large-scale applications [14]. Its dynamic range of usage, stems from its wide variety of toolboxes, readily available for the user to download and install. The most prominent toolbox we will be using for this project, will be the Deep Learning Toolbox, which allows us to create and train Artificial Neural Networks. Where MATLAB comes handy the most, are its thousands of built-in functions for common mathematical, scientific and engineering calculations. The availability to quickly plot and visualize your data, using the variety of built-in plot functions, makes for quick and simple visualizations of many types of data, including numeric, string, date-time, categorical and even structures.

One of MATLABs most common uses, is the creation of scripts to automate work. This allows for entire programs to be run automatically or to be run individually in separate sections. Adding high-level programming constructs, allows for more dynamic scripts, and with its easy to access plot and save functions, script results can be easily saved for later analysis.

In this work, MATLAB has been selected as the programming language, not only because of its aforementioned dynamic range of usage, but also due to it being one of the most accessible programming languages for Neural Network application, as it both offers easy to understand examples of different types of Neural Networks [13], as well as the Deep Learning Toolbox containing the desired ANN models that are required for this project.

Deep Learning Toolbox

MATLABs Deep Learning Toolbox, is an extension that allows the user to easily develop or test simple or complex Neural Networks. MATLAB offers plenty of example networks for each type of Machine Learning problem (classification, regression, clustering etc.) and makes AI easily accessible to beginner developers.

Python

One alternative to using MATLAB, is the high-level, general-purpose programming language Python, which design philosophy emphasizes code readability and dynamics [31]. It is a very popular programming language, due to its ease of use and its wide range of applications. As mentioned, Pythons design, emphasizes dynamics, meaning it is a dynamically typed programming language, which means that variables do not need to specify their data type when declared. When using Python for Artificial Neural Networks, one of the available libraries for such a task, are the Scikit-learn (sklearn) libraries.

Tools

LaTeX

This report and its attached annexes were made using LaTeX, which is a software system for typesetting documents. It is widely used in academia for publication of scientific documents and technical note-taking in many fields. LaTeX focuses its approach on simplifying content format by separating format and content in such a way that the user can focus their attention on describing the content, while the system automatically handles the formatting. To help with that, the software provides the user with ready-made commands for formatting and layout requirements for chapter headings, footnotes, bibliographies, cross-references etc. [29].

GitHub

GitHub is a developer platform, that allows developers to create, store, manage and share their code. Through its use of Git software, it allows enables the distributed version control of Git in addition to access control, bug tracking, task management, software feature requests and wikis for every project [28].

In this project, a repository has been created[8] that contains all the source code for the project. As this is not a team project, the repository has only been used to keep track of issues, and for version control after major changes within the code. The repository has been carried out privately without public access.

The features of this tool have been very useful for work, being able to track issues (objectives) while at the same time keeping an overview of which version commit has closed which issue. However as the project itself does not necessitate a very complex nor grand code structure, its usage during development in regards to issues and milestones has been very limited. Additionally, the documentation of the project remains very scarce as well, as most of the code can be easily read through thanks to MATLABs accessibility. The repository does however remain a surefire way to keep access to the content of the project at any time and from any place as long as the GitHub user has access to it.

Scikit-learn (sklearn)

The aforementioned alternative to MATLAB, is using Python with the Scikit-learn library, specifically its Neural Network package, however it is important to note, that sklearn does not offer GPU [9] support, therefore not making it optimal for larger scale applications [25].

Sklearn is a free and open-source machine learning library for Python programming. It offers a various selection of classification, regression and

[8]`https://github.com/acy003/TFG_Sky_Classification`

[9]GPU is a graphics processing unit that allows for faster execution of calculations in regards to graphics and visualizations on computers. In machine learning, they may be used to speed up the training process.

clustering algorithms. It is a very popular library, that is based on other popular libraries such as NumPy, SciPy and matplotlib. Its Application Programming Interface (API) is easy to use, which allows beginners in the Deep Learning field, to quickly develop prototyp machine learning models and experiment using the vast selection of algorithms and parameters [32] .

Microsoft Excel

Microsoft Excel is a spreadsheet editor developed by Microsoft. It allows users to calculate, analyse and organize data efficiently and easily. [30] It has been used in this project to store the ANN results, more specifically the evaluation metrics, in data tables. All values have been saved in the csv (comma separated values) format. Excel was chosen as it is a calculation tool that offers both ease of access and a variety of options to analyse and visualize data, as well as MATLABs options, to save results easily inside of csv files.

Techniques

Preprocessing

As mentioned in the introduction, the preprocessing of the database is one of the most influential factors for the ANNs final performance, and its importance was taken into account for this project. Before the dataset can be used for training, there are multiple steps it has to go through before it is ready.

Resizing

First and foremost, the images inside the dataset have to be reduced in size, as the captured images have a very high resolution of 1158x1172 pixels, which would make them very costly in terms of computational complexity for the ANN, as well as making it drastically more difficult for the network to classify based on recognized patterns, as having more pixels would mean that there are more values that the network has to "memorize" and adjust to. Thus, we begin the preprocessing by shrinking the image size down to a more manageable size (in this case, a size of 117x128 pixels was chosen), without inhibiting that network's classification performance. The network now has less values to deal with and doesn't require as much computational power to work through each image.

The resizing performed by the previous research [8] had shown that a reduction to 110x110 pixels did not negatively influence performance and therefore was safe to use. For this project, the decision was made to shrink the images to 117x128 pixels instead, in order to keep the height and width ratio of the images in tact.

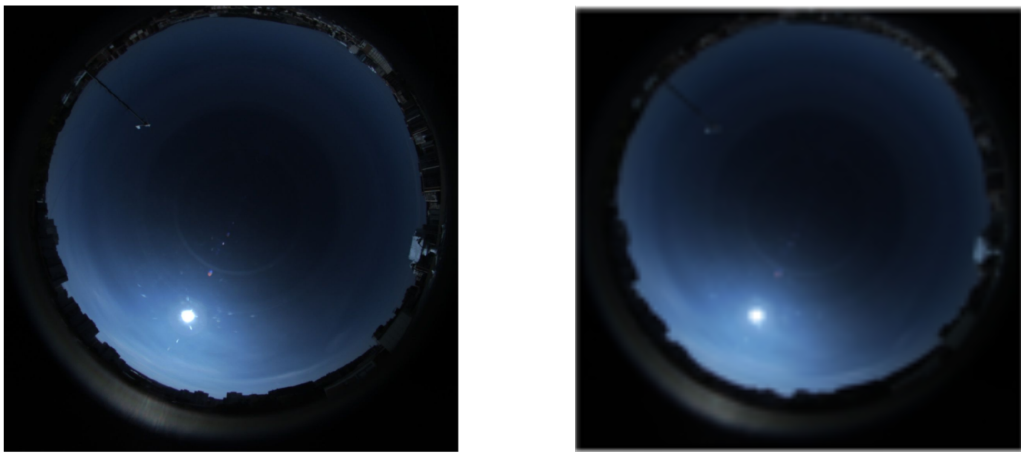


Figure 4.3: Sky Image before and after resizing.

Color channel

As mentioned in the project objectives, we have results from previous research on this topic [8] available, that tells us of the improving qualities of image-preprocessing in regards to the color channel used. Thus it was decided that using one of the best scoring image processing methods used in the previous work will be tested in this work.

The method chosen, is to convert the RGB images into panchromatic images, which are images that "combine" the information from the R, G and B bands[10] into a singular gray-scale value [19]. The resulting image is a grayscale image with a high spatial resolution that uses a single-band, the so called Y channel [24]. Figure 4.4 shows 3 panchromatic images of the clear, partly cloudy and overcast sky types respectively. Therefore, for this work, training was performed using RGB images and panchromatic images respectively, comparing their performance and evaluating them respectively.

[10]The red, green and blue color values of a pixel respectively.

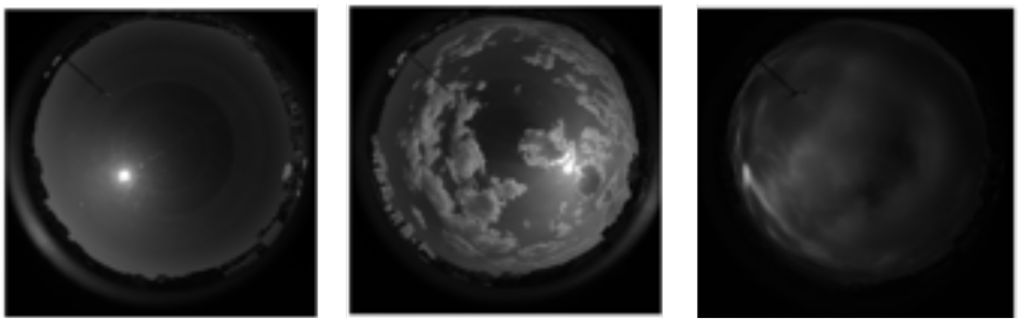
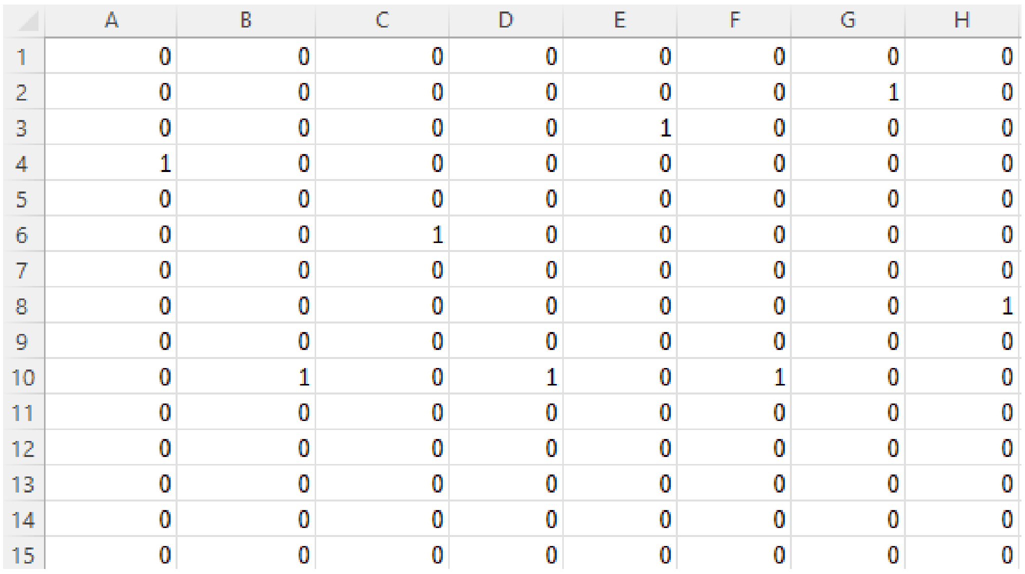


Figure 4.4: Sky Images after transformation to panchromatic images

Class Extraction

When creating a supervised neural network it is important to have the "solution", the so called target value, for each sample that will be used for training and testing, which in our case would be the sky category for each image inside of the dataset. The acquisition of the target values for the dataset is as mentioned in the introduction, a long and arduous process of manually labelling each image to its corresponding class.

In order to use the target values that have been set during the dataset acquisition, MATLAB offers a multitude of ways. The most common one is to simply create a csv spreadsheet, containing a matrix of binary values that represent each sample's class values, e.g. a sample of sky type 4, would have 0 values in every row except for the 4th row, depicting their class label. This spreadsheet can be used during training as a guide for the network in if it needs to adjust its corresponding output for each sample.



	A	B	C	D	E	F	G	H
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
3	0	0	0	0	1	0	0	0
4	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0
10	0	1	0	1	0	1	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0

Figure 4.5: Example structure of a class spreadsheet, each column represents a sample image, their row in which the value is 1, points to their corresponding class

The more simple and convenient way, is offered by MATLABs "imageDatastore" function, in which a sample's class label, is simply the name of its parent folder. This makes it possible for users to simply create a folder for each desired class, and move the corresponding images inside the corresponding class folder and MATLAB will apply the label for each sample automatically.

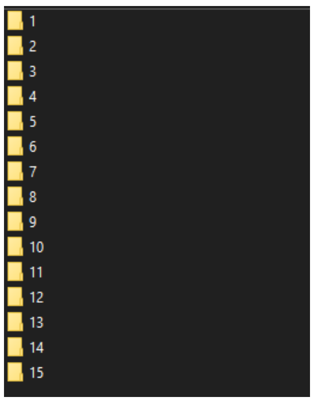


Figure 4.6: Example structure of class folders, each containing the images of their corresponding class

Network Training

Cross-Validation

Cross-validation is a model validation technique used to evaluate a neural networks performance in making predictions on unknown data, that it hasn't been trained on. It is an important technique for ANN training as it is crucial that the model does not overfit its algorithm on the training set, meaning that it should not adjust its weights to simply perfectly fit the training examples, but that it should be capable of classifying samples on any chosen sample, thus making sure its ability to generalize is functioning well [16].

The cross-validation process can be described as followed:

1. Data Division: Firstly, the training dataset is divided into k folds of equal size. A common number of folds is usually 5 or 10. For this project, 5 folds were used, as the dataset is not big enough to warrant smaller fold sizes.
2. Training and validation cycle: During training, a training set and a validation set are used. These are chosen from the folds, such that training will use the majority of the folds and validation uses the remaining fold(s).
3. Training cycle: In each training cycle, the network is trained using the training data chosen and network weights are adjusted accordingly.
4. Validation: After each training cycle, the network predicts the target values of the validation data set and the results are evaluated to measure the networks performance. If the networks performance on the validation set worsens (thus failing a so called validation check) multiple times in a row, then training is terminated.
5. After training and validation completion, the network is used on a separate test set, and its performance is evaluated.
6. Repetition: Steps 2-5 are repeated k times, with the chosen fold combination for training and validation being different in each iteration.
7. Results: After the k repetitions, the average results obtained of each iteration are averaged out, which shows the the model's overall performance.

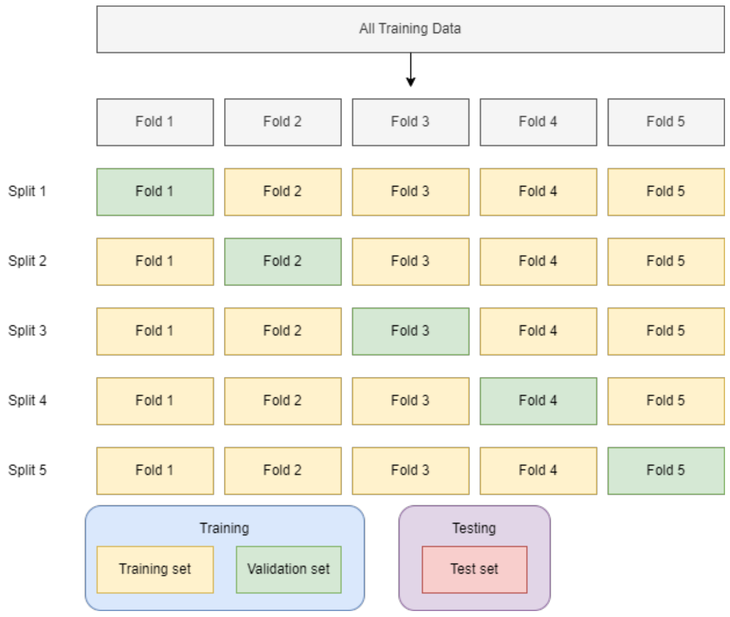


Figure 4.7: Cross-Validation Structure

Training Functions

One of the main objectives of this study is to research the availability of other viable training functions that can be used for sky classification. As a starting point 3 alternative training functions were chosen for the network to train with in addition to the Scaled Conjugate Gradient training function from the previous research [8].

Scaled Conjugate Gradient

Scaled Conjugate Gradient (SCG) is an algorithm that combines the features of the conjugate gradient method and the Levenberg-Marquadt algorithm to train neural networks [20]. Through the usage of the initial gradient of the error function, it establishes a search direction for weight adjustment. During training, it uses scaling to ensure the stability of the weight adjustment [22].

Resilient Backpropagation

Resilient backpropagation is an algorithm used mainly to tackle harmful effects caused by the usage of sigmoid transfer functions inside the hidden layers of a neural network. It introduces a separate update value that determines the size of the weight change according to a factor delta. The update value increases by a factor "delta_inc" whenever the derivative of the performance function has the same sign for two successive iterations. In the same sense, the update value decreases whenever the derivative of the performance function is different from the previous iteration [18].

$$\text{dX} = \text{deltaX.*sign(gX)}$$

Gradient Descent with Momentum and Adaptive Learning Rate Backpropagation

Gradient Descent with Momentum and Adaptive Learning Rate Backpropagation (Gdx) is an algorithm that combines adaptive learning rate (lr) [11] with momentum training (mc)[4].

$$\text{dX} = \text{mc*dXprev} + \text{lr*mc*dperf/dX}$$

dXprev represents the previous change to the weight or bias.

One-step Secant

The one-step secant (OSS) method, is an attempt to bridge the gap between conjugate gradient algorithms and secant algorithms [26]. The OSS method requires less storage and computations per epoch than BFGS (Broyden–Fletcher–Goldfarb–Shanno) algorithms, but needs more storage and computation per epoch than the conjugate gradient algorithms. Its purpose as a bridge between conjugate gradient and secant algorithms is also made in hopes of creating a lower computational requiring alternative to the BFGS algorithm [17].

[11]Adaptive learning rate, is a method for the model to adjust its learning rate during training. Usually this results in a high learning rate at the beginning, which decreases towards the end of training.

5. Relevant Aspects of Project Development

The project development will be summarized in this chapter, elaborating on the most important points and problems that occurred during the process.

5.1 Project Planning

During the project planning phase, the first step was to delve into the sky classification topic and read into the methodology of how and why it is important. Additionally, it was important to understand the CIE standard for sky classification and how its classification works.

Once the surface level research had been done, further research in previous work related to the topic was made. The research itself revealed that most classification projects based on the sky were more focused on cloud classification than sky classification itself.

Finally it had to be established what exactly the ANN was going to classify and which classes were to be defined. The final decision was made to focus the project on sky classification by the CIE standard.

5.2 Data Preparation

As the sky images were provided by the UBU already as it had been used for previous research [8]. To prepare for the implementation, image measurements were documented and the database was made available on the local device.

Once prepared, it was time to plan how the dataset would be split for the neural network to use. As the dataset "only" contains 1500 images, a 70-15-15 split was chosen; 70% for the training set, 15% for the validation set and 15% percent for the final testing. It was also decided that this split should be easily replicable for repeated training and experimenting, therefore, the seed used to choose the random split was the same for every training execution cycle.

5.3 Implementation

Class Labelling

As previously mentioned in the Techniques and Tools chapter, MATLABs user friendly accessibility, allowed for simple label extraction for each image. This however, would turn out to be slightly more complicated for this database, as each image was inside its class folder as intended, however it would also be inside a nested subfolder inside of the class folder for each single image, and since MATLABs automatic labelling only uses the direct parent folder of the image, some slight work around was required

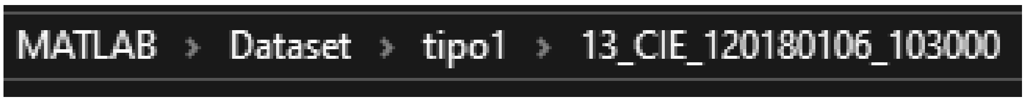

MATLAB > Dataset > tipo1 > 13_CIE_120180106_103000

Figure 5.1: Example path of how each image is stored inside the dataset

To deal with this issue, a function was defined that takes the file path of each file, and splits the file path at the root dataset folder and continues to split the path until only the class folder has been acquired. This cell structure of labels is then returned and can be assigned to the networks "label" parameter.

The last requirement is to create a 15x1500 matrix that stores binary values in each row, indicating that samples class. To do this, a short function was created that creates a 15x1500 matrix of zeroes and loops through the newly established labels to assign their corresponding class value for their respective column.

Resizing

To be able to use the current images, without requiring excessive computational power, it was important to resize the images to a more manageable size. To do this, another function was defined that loops through all files inside the dataset, and resizes them to a smaller format. In order to not overwrite the old dataset, a new directory with the resized images is created and the images are saved inside the new directory with a new name.

An issue however, occurred during initial executions. It turns out that a few images inside the dataset were either a few pixels smaller or bigger than the previously established standard. This caused problems during training, as the array dimensions of the images were not the same. This was also not prevented by the first implementation of the resizing function, as that resizing was using a 80% reduction of each image to create the new format, therefore each images that were inconsistent, remained inconsistent after resizing as well.

To fix this issue, a simple change from percentage resizing, to resizing using set dimensions was made. All images inside of the dataset would now be resized to a 117x128 pixel image.

Color Channel

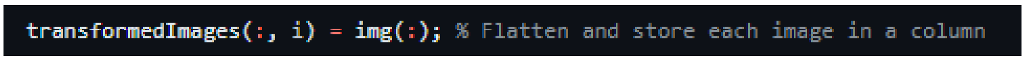
Since it was established, that all experiments would also be performed using the Y color channel as well as the regular RGB channel, there would be a need to transform the RGB images into Panchromatic ones.

To facilitate this change, it was decided that adding a parameter to the resizing function, that determined whether or not the images should be transformed to the panchromatic channel before resizing, was the most efficient. Images would now be transformed into Panchromatic images (using the Y channel) before resizing, if the corresponding parameter of the function was set.

Dimensionality

The network function used for this project, was the patternnet function provided by MATLAB, which is a ANN that specializes in pattern recognition, making it perfect to be used for this project. The usage of this method did come with its own requirements however. The network itself, could only work with 1 dimensional sample values and since images are 2 dimensional, a transformation had to be done before training could begin.

For this particular issue MATLABs accessibility regarding matrix and array mathematics came in handy, as a simple one line of code expression, handled the dimension transformation perfectly.

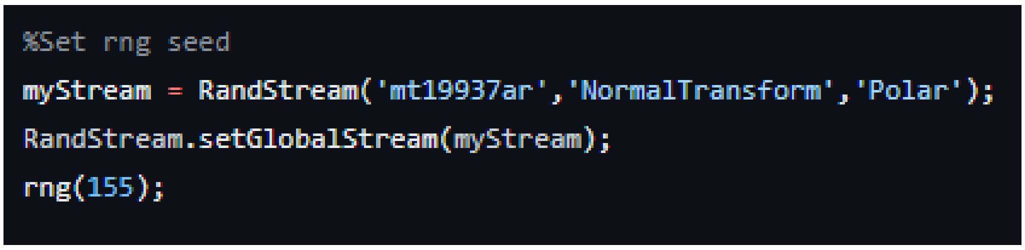


```
transformedImages(:, i) = img(:); % Flatten and store each image in a column
```

Figure 5.2: Code snippet showing the part of the transformation function that assigns all values from an image data matrix, to a single row of a new matrix

Split Replicability

In order to have the data split replicability, we have to specify a set seed, which the script will use to perform all its functions that involve randomisation. To achieve this, a simple call of the rng (Random number generator) method in MATLAB would suffice. However, this doesn't set a global seed for all machines from which this script can be run. To help with this, MATLAB offers the option to set a Global Stream variable, which defines a specific global stream for randomisation, after the global stream variable has been set, the rng seed can be specified as usual.



```
%Set rng seed
myStream = RandStream('mt19937ar','NormalTransform','Polar');
RandStream.setGlobalStream(myStream);
rng(155);
```

Figure 5.3: Code snippet showing the Global Stream and RNG seed assignments

Training

Training ANNs can take up lots of time when working with bigger datasets or when trying to experiment with different parameter combinations. One of the best ways to alleviate having to waste time manually changing given parameters and/or document results is to automate as much of the process as possible. For this project the usage of different training functions with 2 different color channels was the main focus. Thus it was important to setup all the functions so that they can either be called multiple times per parameter combination. Or that they go through the given parameter combination themselves by passing the desired parameters to the function. The option opted for in this project, is to pass the sets of parameters with which training shall be performed and the function will then perform loops over the entire training process using each parameter combination possible. Most notably, for each training function, different hidden layer neuron quantities were performed and tested. As this process can result in multiple nested loops, it is important to set aside longer time frames so that training can be performed without interruption. It is also essential to save the results after each finished training and evaluation cycle for the given parameter combination. Doing this is a simple matter of saving the results inside a csv file.

5.4 Performance Evaluation

When performing Multi-class classification, the resulting predictions can be difficult to evaluate, as once would have to go over each class and identify the correctly predicted samples for them. Luckily, MATLAB offers simple solutions to these types of problems. To acquire the desired evaluations, it is easiest to simply get the confusion matrix using the predicted and actual classes for the test set. To do so MATLAB has the confusionmat method, taking the predicted classes and actual classes as parameters to return the confusion matrix. Now it is simply a matter of accumulating all values that align on the diagonal on the matrix as true positive values and the ones that do not align, are accumulated for false positives and false negatives respectively.

Metrics

To understand the evaluation metrics acquired from the work, their calculations and meanings will be explained.

Confusion Matrix

The most basic metric to evaluate the classification of a neural network is to make a confusion matrix. A confusion matrix represents the number of correct and incorrect predictions of the network while also showing how many exactly of each class has been predicted correctly or incorrectly. A simple confusion matrix for a 2 class classification problem could look like this:

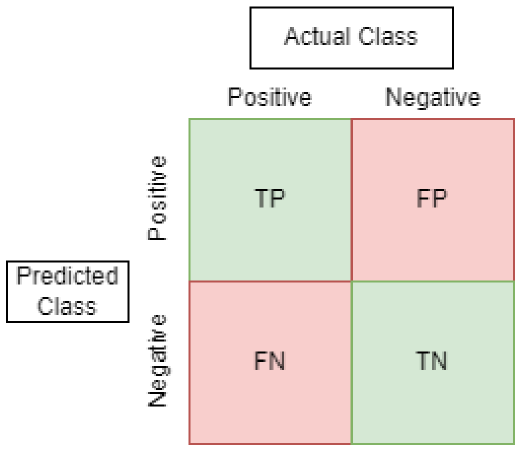


Figure 5.4: Simple 2 Class Confusion Matrix

The rows of this matrix show the class predicted by the network, while the columns show their actual class. Therefore, the diagonal shows where the actual class and the predicted class align. The cell "TP" represents the True Positives, meaning the case where the actual class was "positive" and the predicted class was also "positive". In the same sense the cell "TN" represents the predictions where actual and predicted class align with a "negative" class prediction. "FP" and "FN" reflect the predictions where the network has either predicted the class falsely as positive (false positive) or falsely as negative (false negative).

As the number of classes increase for a classification problem, so do the rows and columns of its corresponding confusion matrix, making its readability more complex and slightly less readable.

Using the information acquired from the confusion matrix, there are a number of possible metrics we can calculate to evaluate the networks performance:

- **Accuracy**: The accuracy is the fraction of correctly predicted samples divided by the total number of samples of the prediction set. It is a measure to show how accurate the model can predict data and is a widely used metric for ANNs. It can however be misleading when using unbalanced datasets, as e.g. when having a dataset that consists of 90% class 1 samples, a network could achieve a 90% accuracy simply by predicting class 1 on every sample, which may be accurate, but makes the network as good as useless in actual practice. To counteract such occurences, there are additional metrics that can be used to prevent such misleading results.

Accuracy = (TN + TP) / Total number of predictions

- **Precision**: The precision is the fraction of correctly predicted samples for a specific class divided by the total number of samples that predicted that specific class. It is used to determine how precise the model is for predictions of a specific class.

  Precision = TP / (TP+FP)

- **Recall** : The recall is the fraction of correctly predicted samples for a specific class divided by the total number of samples of that class. It shows how well the model can identify samples from that class among samples of all classes.

  Recall = TP / (TP+FN)

- **F1-score**: This metric is used as a combination of the precision and recall values to evaluate the overall performance of the model. It is considered the most important metric for networks in which there is an imbalance between classes. This project used the f1-score on the best network results to further gain an overall view any dissonances between precision and recall, if there were any.

  F-1 Score = (2*precision*recall)/(precision+recall)

Precision, recall and f1-score are important metrics to prevent misleading results of unbalanced datasets. However, as the dataset used in this project is perfectly balanced, the usage of accuracy is a good enough measure, when trying to compare network results that have performed well.

6. Related Works

Having mentioned the previous work done on the topic of sky classification multiple times before, it is only natural to briefly summarize and reiterate its importance to this work.

Pixel-Based Image Processing for CIE Standard Sky Classification through ANN

This work [8], focuses mainly on the application of image-processing techniques to improve the performance of ANNs (in this case foucsed on the accuracy) for sky classification under the CIE standard. It reviews 22 image-processing methods and applies them to the sky image dataset. It takes into account both the 15 CIE standard sky categories as well as a more generalized 3 category application.

It also highlights that the usage of image-processing methods, depends on the target sky categories required, as the regular RGB channel, performed best when classifying for the simplified 3 category sky conditions of clear, partial and overcast.

7. Conclusions

In this final chapter, the most significant results of the work will be presented, as well as possible future improvements on which the work can be expanded upon.

Results

The development of this work went through multiple general stages.

1. Firstly, the dataset was acquired and preprocessed, using the sky images provided by the meteorological weather station located on the roof of the Higher Polytechnic School building at the University of Burgos.

2. Secondly, the neural network was setup starting with the chosen dataset split using a set rng seed and then specifying the desired parameters and training function.

3. Thirdly, the network was trained using all parameter combinations, as well as repeating this step with the panchromatic version of the images.

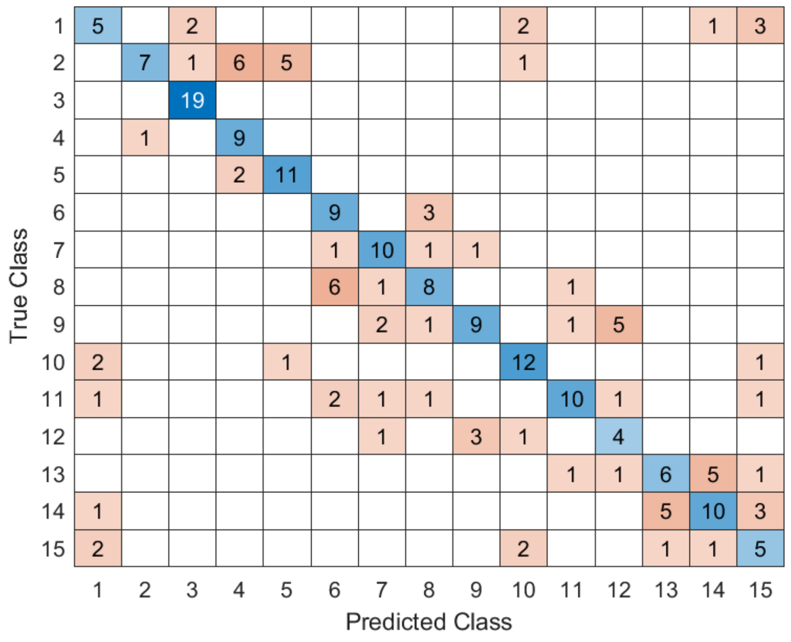
4. Lastly, the performance of these networks was evaluated using the confusion matrix acquired from using the network to predict the classes of the test set. The evaluations for each parameter combination was then saved inside a csv file.

Training time

The training time is the duration the network needs to go through its entire training process and can be used to evaluate a network in terms of it being of interest. Its usefulness stems from use cases in which time is a critical factor for the training of the network. In this project, time was a factor as it allowed for even further comparisons between both the training functions as well as between the application of image processing before training. Network configurations that require longer training time scale even worse in this project, as the use of cross validation increases the total time spent training by almost k fold (k being the number of folds used). This was also made very apparent during the experiment and its training.

Evaluation

As previously mentioned, the more classes a classification problem has, the more complex and difficult to read its confusion matrix becomes. The confusion matrix structure for this project looks like this:



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	5		2							2				1	3
2		7	1	6	5					1					
3			19												
4		1		9											
5				2	11										
6						9		3							
7						1	10	1	1						
8						6	1	8			1				
9							2	1	9		1	5			
10	2				1					12					1
11	1					2	1	1			10	1			1
12							1		3	1		4			
13											1	1	6	5	1
14	1												5	10	3
15	2									2			1	1	5

Figure 7.1: Confusion matrix created by evaluating one iteration of the ANN

The numbers on the axes signify the corresponding sky category and the values inside each cell are the amount of samples that have been predicted for the specific value. Using the information acquired from the confusion matrices, we could calculate the evaluation metrics mentioned in chapter 5 and compare the results of the ANNs.

7.1 Project Results

After extensive training using different combinations of training functions and hidden neurons per hidden layer, the best networks for each training function came out like this:

Gradient Descent with Momentum and Adaptive Learning Rate Backpropagation

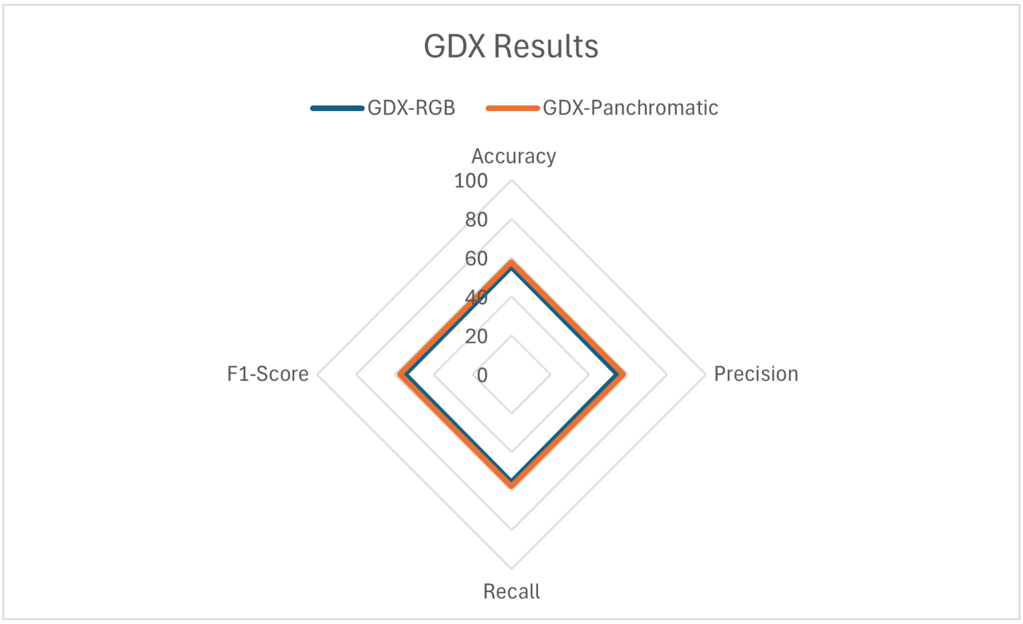


Figure 7.2: GDX evaluation results of both color channels in a radar plot

As we can see on the radar plot, GDX performed nicely in both iterations of its usage, that being using RGB images as well as Panchromatic ones. Both networks scored an above 50% accuracy, with the panchromatic one scoring higher with 57% and the RGB one with 55%.

As previously mentioned, this graph shows very well how precision, recall and the f1-score are not critical for the evaluation of a network using a balanced dataset, as their respective values are pretty much identical with the accuracy.

Though the panchromatic network used more hidden neurons per layer, with a number of 80 neurons per hidden layer, making its network structure more complex, its training time remained substantially shorter than its RGB counterpart, which needed 64 seconds on average to perform one training cycle. The panchromatic network finished on average after 24 seconds.

This was a common occurrence during the entire experiment, as using panchromatic images, leads to only having to work through one third of the total values in comparison to RGB images which have 3 color channels and therefore 3 times the pixel values to deal with.

One-step Secant

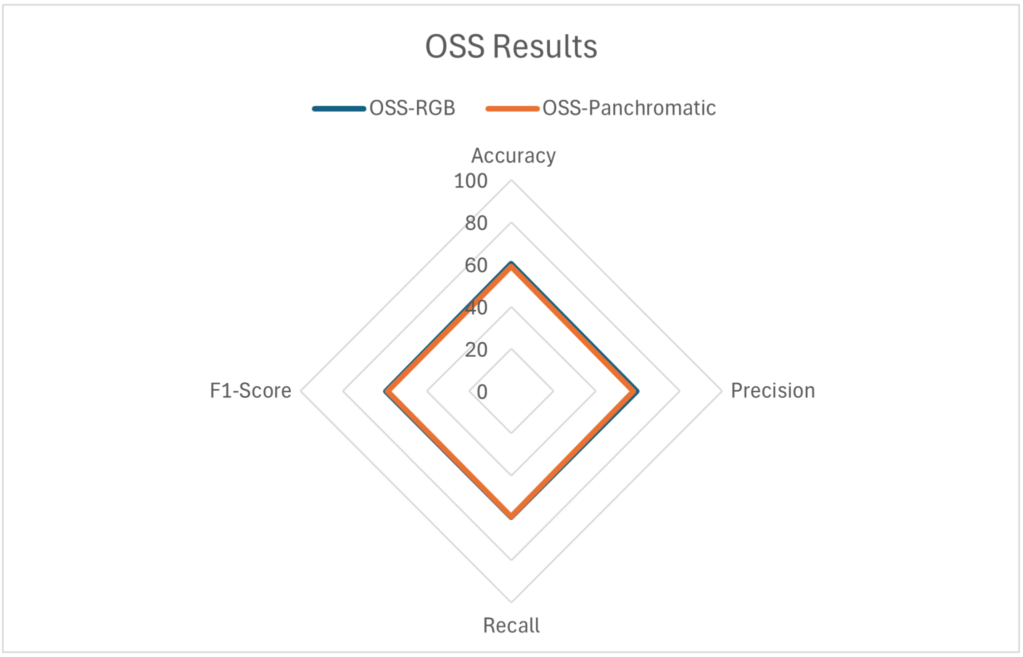


Figure 7.3: OSS evaluation results of both color channels in a radar plot

OSS performed even better than GDX in terms of accuracy and hidden neuron requirement. The radar plot shows an accuracy of 60% for OSS using RGB images and a hidden neuron usage of 40. The OSS network with panchromatic images lands right behind with a 59% accuracy and a hidden neuron usage of 50.

However, there is a big downside in the OSS' RGB network, that being the enormous training time, which was the highest out of every training function. The RGB network of OSS needed on average 202 seconds to finish training, which is a lot worse compared to all the other networks. Its panchromatic counterpart however only needed 44 seconds on average, making it a viable alternative that doesn't require such strenuous training time.

Both networks remain consistent with the previous assumption that precision, recall and f1-score remain consistent with the accuracy, due to the balanced dataset.

Resilient Backpropagation

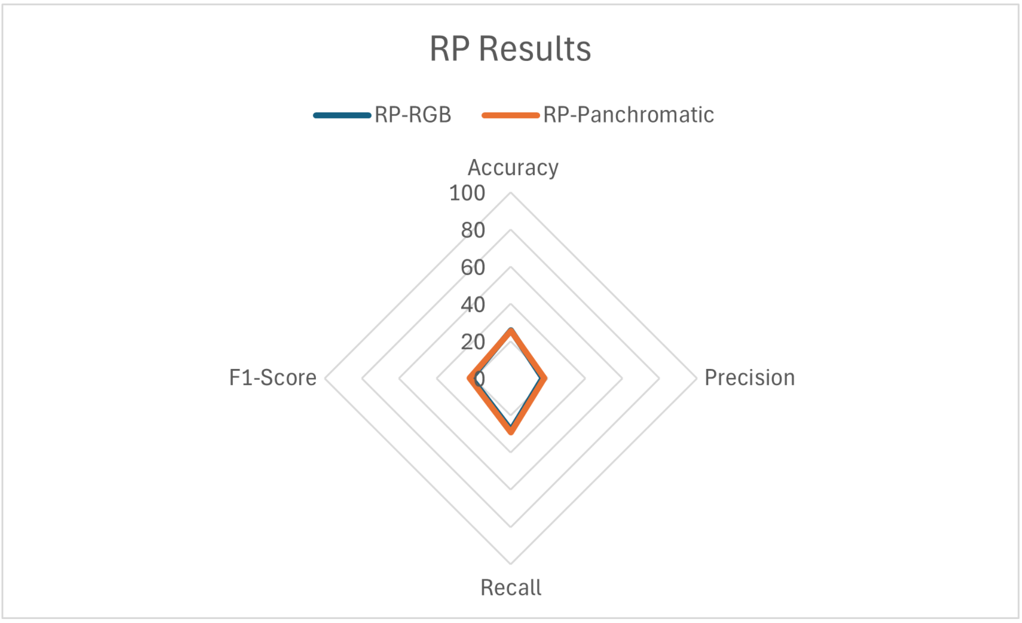


Figure 7.4: RP evaluation results of both color channels in a radar plot

The worst performing training function in this work was the rp function. As seen on the radar plot, its accuracy is severely lacking, with a low 25% for both versions. It is also the only network where its precision, recall and f1-score values differ from the accuracy value. In this case the precision was always lower than the accuracy while the recall was higher, the f1-score therefore, scored in the middle of both values.

The training functions poor performance may be due to it not being suitable for this type of classification problem, or it simply requires a more substantial amount of parameter fine tuning to perform well.

Its training time was the lowest out of all networks for the panchromatic channel this however is only due to the fact that it reached the stopping criteria of failing too many validation checks [12] in a row very fast.

Its RGB counterpart however delivered a poor training time on top of the poor accuracy results.

Scaled Conjugate Gradient

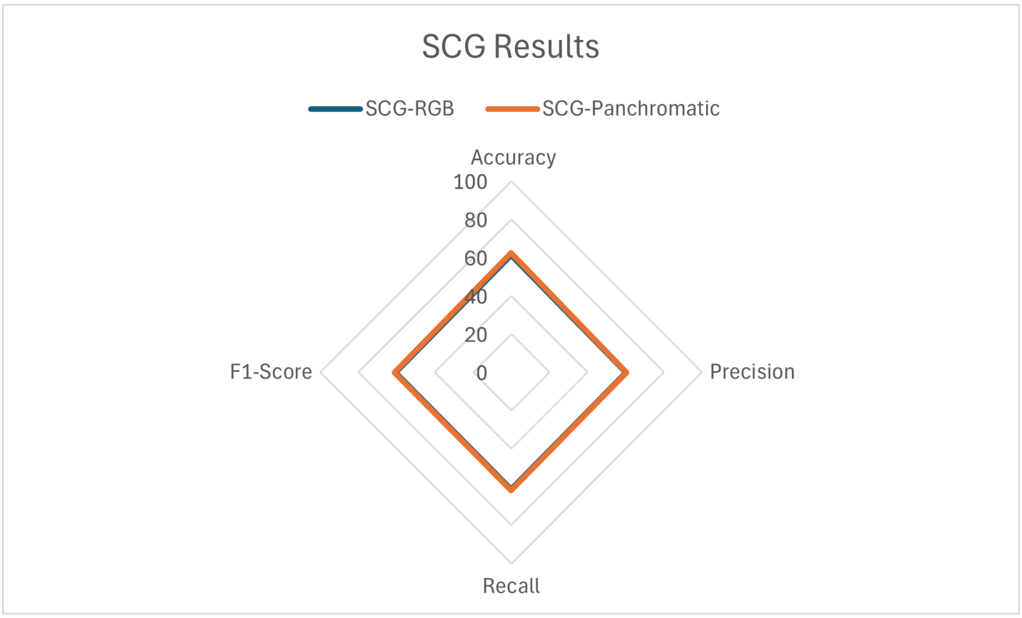


Figure 7.5: SCG evaluation results of both color channels in a radar plot

SCG was the most consistent training function during its entire training phase. It scored slightly above the rest with a higher accuracy by a small margin, securing itself the highest accuracy with 61.3% and 62.4% for its RGB and panchromatic version respectively.

Its recall, precision and f1-score values remain consistent with the accuracy, as is expected. Its performance was the best using a higher number of neurons, when comparing it to the other training functions' performances

[12]Validation check failure being the aforementioned validation set seeing a worse performance in a training iteration than the previous iteration.

using such a higher number. Its required training time does leave some room for improvement, as it finished with a decently high time of 85 seconds on its RGB version.

Overall Results

After reviewing the results of each training function and its color channel alternative, it can be clearly said that there are alternatives to using the Scaled Conjugate Gradient when it comes to sky classification under the CIE standard. While SCG has the highest accuracy of all the functions, OSS and GDX remain close behind with only a marginally smaller accuracy. It is also important to note that these accuracies are not final, there is still the possibility that these values can be further improved with more in depth parameter fine tuning, which may take an extensive amount of time.

It can also be said that using GDX may come in handy more than SCG in some use cases where a lower training time is required. Adding to that, it can be clearly seen in the table 7.1 that in almost every case, the performance of the panchromatic version of the network both performed better in terms of accuracy as well as training time. This revelation is also very useful, as models that may have more time critical appliances may prefer using panchromatic images. It is important to note, that if panchromatic images are used for training, then panchromatic images need to also be used when using the trained network later on to predict, thus there remains the possibility that the time loss from having to transform every sample to the Y channel may outweigh its time saving from training.

Overall it can be said that sky classification under the CIE standard using Artificial Neural Networks is a potential alternative to using the traditional way of using a sky scanner. The initial acquisition of data to use for the network may be strenuous, but it can pay off in the long term once the network has been fully trained. It is also certain that the SCG training function is not the only viable training function for networks specializing in this task.

T Function	Color channel	Accuracy	Hidden Neurons	Training Time s
GDX	RGB	55.46	50	64.15
GDX	Y	57.69	80	23.92
OSS	RGB	60.27	40	202.66
OSS	Y	58.93	50	44.305
RP	RGB	25.96	80	84.615
RP	Y	25.42	50	11.36
SCG	RGB	61.33	70	85.3
SCG	Y	62.4	90	36.086

Table 7.1: Best experiment result for each training function and color channel

7.2 Future Lines of Work

In this last section of the report, possible improvements that can be applied to the project in the future and further research aspects are presented.

Parameter Fine Tuning

While the results delivered by the experiments are viable and comparable to other works, there still remains the option of further fine tuning of parameters specifically for each training function. This step of fine tuning is a very time consuming process and may require deeper insight into each training functions calculations. It is also possible that other training function alternatives remain that may work just as well or even better than current experiment results. Training functions discovered in the future can also be applicable to this classification problem and should be experimented with.

In the same sense, other image processing techniques may harmonize better with specific training functions than the one used in this work. The same applying to any future image processing techniques that may be discovered.

Interface

As it stands now, the interface of this project remains rather barren. Through the usage of a MATLAB script can network training be performed. The required setup for it however is very specific, with a very specific dataset structure. The same can be said about the MATLAB script that is responsible for independent predictions, using a saved network.

Both scripts could be implemented into a proper user interface that visually helps setup the data and guides the user through the training/prediction process.

Deployment of the ANN for Sky Classification

Once the ANN has been fine tuned to such a degree that its prediction accuracy can be considered consistent (preferably at least a 70-80% accuracy), deployment of the network could be considered. One form of using the network could be to make local projects that require the classification of the sky for either e.g. architectural design or solar energy installation. To do so, the local researchers would need to set up a digital camera pointing towards the sky through a fish-eye lens to capture sky images. After enough images have been taken, the ANN can be used to classify all the sky types of the newly acquired dataset. Using the new prediction data (even if not a 100% accurate), can help with the planning of the aforementioned use cases, by providing a general overview of the typical sky types that occur frequently at the given location.

Bibliography

[1] K.A. Alshaibani. Classification standard skies: The use of horizontal sky illuminance. *Renewable and Sustainable Energy Reviews*, 73:387–392, 2017.

[2] Mohamed Elasyed Abd Elaziz Ammar Hamed Elsheikh. *Artificial Neural Networks for Renewable Energy Systems and Real-World Applications*. Academic Press, 2022.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] Geeks for Geeks. What is momentum in neural network. https://www.geeksforgeeks.org/what-is-momentum-in-neural-network/, 2024.

[5] Ignacio García, Marian de Blas, and José Luis Torres. The sky characterization according to the cie standard general sky: Comparative analysis of three classification methods. *Solar Energy*, 196:468–483, 2020.

[6] Ignacio García, Carlos Sáenz, Begoña Hernández, Rafael García, and José Luis Torres. Luminance calibration of a full sky hdr imaging system using sky scanner measurements. *Solar Energy*, 239:147–169, 2022.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[8] D. Granados-López, A. García-Rodríguez, S. García-Rodrígue, A. Suárez-García, M. Díez-Mediavilla, and C. Alonso-Tristán. Pixel-

based image processing for cie standard sky classification through ann. *Complexity*, 2021, 2021.

[9] Javatpoint. Difference between supervised and unsupervised learning. https://www.javatpoint.com/difference-between-supervised-and-unsupervised-learning, 2024.

[10] Hana Kasht. Sky conditions classification and estimation of solar radiation for clear sky days. Master's thesis, Middle East Technical University, 2018.

[11] Danny H.W. Li, T.C. Chau, and Kevin K.W. Wan. A review of the cie general sky classification approaches. *Renewable and Sustainable Energy Reviews*, 31:563–574, 2014.

[12] Danny H.W Li and Joseph C Lam. An analysis of climatic parameters and sky condition classification. *Building and Environment*, 36(4):435–445, 2001.

[13] Mathworks. What is a neural network? https://www.mathworks.com/discovery/neural-network.html, 2017.

[14] Mathworks. Programming with matlab. https://www.mathworks.com/products/matlab/programming-with-matlab.html#:~:text=MATLAB%20is%20a%20high-level,to%20developing%20large-scale%20applications., 2024.

[15] Matlab. Choose a multilayer neural network training function. https://www.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html, 2024.

[16] Matlab. Cross validation. https://www.mathworks.com/discovery/cross-validation.html, 2024.

[17] Matlab. One-step secant backpropagation. https://www.mathworks.com/help/deeplearning/ref/trainoss.html, 2024.

[18] Matlab. Resilient backpropagation. https://www.mathworks.com/help/deeplearning/ref/trainrp.html, 2024.

[19] Matlab. rgb2gray. https://www.mathworks.com/help/matlab/ref/rgb2gray.html, 2024.

[20] Matlab. Scaled conjugate gradient backpropagation. https://www.mathworks.com/help/deeplearning/ref/trainscg.html, 2024.

[21] Matlab. What is reinforcement learning? https://www.mathworks.com/discovery/reinforcement-learning.html, 2024.

[22] Martin Møller. A scaled conjugate gradient algorithm for fast supervised learning. neural networks vol. 6, pp. 525–533. *Neural Networks*, 6:525–533, 1993.

[23] Ignacio García Ruiz. *Daylight modeling in complex environments considering the angular distribution of sky luminance*. upna, 2021.

[24] ScienceDirect. Panchromatic image. https://www.sciencedirect.com/topics/computer-science/panchromatic-image#:~:text=A%20panchromatic%20image%20is%20a,%2C%20G%2C%20and%20B%20bands., 2020.

[25] Scikit-Learn. Neural network models (supervised). https://scikit-learn.org/stable/modules/neural_networks_supervised.html, 2024.

[26] StudySmarter. Understanding the secant method in computer programming. https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/secant-method/#:~:text=The%20Secant%20Method%20works%20by,new%20estimate%20for%20the%20root., 2024.

[27] Peter Tregenza. Analysing sky luminance scans to obtain frequency distributions of cie standard general skies. *Lighting Research & Technology - LIGHTING RES TECHNOL*, 36:271–281, 12 2004.

[28] Wikipedia. Github. https://en.wikipedia.org/wiki/GitHub, 2024.

[29] Wikipedia. Latex. https://en.wikipedia.org/wiki/LaTeX, 2024.

[30] Wikipedia. Microsoft excel. https://en.wikipedia.org/wiki/Microsoft_Excel, 2024.

[31] Wikipedia. Python (programming language). https://en.wikipedia.org/wiki/Python_(programming_language), 2024.

[32] Wikipedia. scikit-learn. https://en.wikipedia.org/wiki/Scikit-learn, 2024.

[33] S. Aydinli Y. Uetani, A. Joukoff, J.D. Kendrick, R. Kittler, Y. Koga, K. Matsuura, T. Nagata, H. Nakamura, M. Oki, R. Perez, P.R. Tregenza, and P. Valko. Spatial distribution of daylight - cie standard general sky. 2004, 2004.