Pattern Recognition for sky recognition

1st Samuel Monsarrat
ETSEIB
UPC
Barcelona, Spain
samuel.monsarrat@estudiantat.upc.edu

2nd Md Zilan Uddn Saif
ETSEIB
UPC
Barcelona, Spain
md.zilan.uddin.saif@estudiantat.upc.edu

3rd Jordi Lladós
ETSEIB
UPC
Barcelona, Spain
jordi.llados@estudiantat.upc.edu

***Abstract*—This paper aimed to predict the hour thanks to a picture of the sky. During the day the sky colors are changing, especially the red channel being enhanced. In this study the data were collected using the camera of a smartphone and have been processed in a computer. Two algorithm have been made, one predicting the hour using PCA (principal component analysis) then GMM (gaussian mixture models) and another using a Convolutional neural network.**

***Index Terms*—PCA, Sky Recognition, CNN,PCA, Gaussian Mixture**

I. Introduction

Since cellphones are so widely available and have such high-quality cameras, taking pictures of the sky has gotten easier. A beautiful dawn, a crisp blue sky, or a stunning sunset are just a few examples of the natural events that people frequently capture on camera phones. The challenge of automatically recognizing and classifying sky regions in these photos, known as "sky recognition," is possible thanks to computer vision and artificial intelligence. There are multiple obstacles in identifying the sky in photographs taken with cellphones. The work may become more difficult due to variations in weather patterns, lighting conditions, and the presence of obstructive objects. Furthermore, because clouds are always shifting in form and location, the sky is dynamic and calls for strong algorithms that can deal with this fluctuation. To address these challenges, AI techniques such as Principal Component Analysis (PCA) and Convolutional Neural Networks (CNNs) have emerged as the best tools for sky recognition.

In a first part, the state of the art is explained which relate a presentation of methods and tools that have been used for sky recognition. The second part describes the proposed approach for sky recognition. The next part will talk about the results of all the experimentation that have been carried out. Finally, in the last part all the conclusions are discussed.

II. State of the art

The sky has always been an element of interest to human beings. Consequently, many scientists have centered their studies around it. We will briefly discuss a few of them that have inspired us to explore this topic. Furthermore, we will examine how similar or different techniques have been implemented in various methods of data collection.

A. Short-term global horizontal irradiance forecasting

The article Cong Feng, "Short-term global horizontal irradiance forecasting based on sky imaging and pattern recognition (IEEE)" develops a short-term forecasting framework for global horizontal irradiance (GHI), crucial for integrating solar energy into power grids. This framework uses pattern recognition and machine learning to predict GHI one hour ahead. It involves data partitioning, model training, and a support vector machine (SVM) classifier for weather pattern identification. The framework, validated with data from the National Renewable Energy Laboratory (NREL), outperforms the persistence benchmark by 16% in normalized mean absolute error and 25% in normalized root mean square error.

The main ideas include a pattern recognition-based forecasting framework, data partitioning for model training, and the use of an SVM classifier for weather patterns. Despite its strengths, this approach is unsuitable for sky pattern recognition projects aiming to predict the hour of sky pictures. The focus on GHI forecasting and irradiance data, rather than temporal patterns in sky imagery, makes it less applicable for this purpose.

B. Identification of stars in astronomical photographs

This article Wojciech Makowiecki, "New Sky Pattern Recognition Algorithm" presents an innovative algorithm for identifying stars in astronomical photographs by utilizing existing star catalogs. This algorithm, implemented in the Skyprint application, uses convex quads formed from star coordinates and geometric hashing techniques to match patterns between photographs and catalogs. The primary focus is on astrometry, determining star coordinates on the celestial sphere, and the algorithm is particularly useful for tasks such as cosmic probe navigation and aligning multiple sky images.

The main purpose behind the algorithm is to create convex quads from stars in both photographs and star catalogs, which are then compared using discretized internal angles and a hash function for efficient matching. The methodology involves creating a grid of seeds (virtual points) to select stars, forming convex quads, calculating internal angles, and using a voting system to identify the correct region in the star catalog.

However, this article is not suitable for predicting the hour of sky pictures in our project. The primary focus of the article is on identifying star positions and patterns, which does not provide temporal information. Additionally, the features and

models used (such as star coordinates, angles, and convex quads) are not relevant to the appearance and lighting conditions of the sky at different times of the day.

C. Feature Extraction from Whole-Sky Ground-Based Images for Cloud-Type Recognition

The referenced article Josep Calbó, "Feature Extraction from Whole-Sky Ground-Based Images for Cloud-Type Recognition" proposes an innovative approach to classifying cloud types through the analysis of digital sky images. It employs a combination of image texture statistics, Fourier transform features, and pixel-level cloud detection to enhance classification accuracy. The study utilizes data captured by Total Sky Imagers (TSI) and Whole Sky Imagers (WSC) located in Toowoomba, Australia, and Girona, Spain, respectively.

The methodology involves examining statistical measurements of image texture to discern between different cloud types, utilizing Fourier transform features to capture frequency domain representations associated with various cloud patterns, and employing pixel-level classification to segregate cloudy pixels from clear-sky pixels. Evaluation against manual visual inspections demonstrates a commendable agreement index of 76% for five sky conditions: clear, low cumuliform clouds, stratiform clouds (overcast), cirriform clouds, and mottled clouds (altocumulus, cirrocumulus).

However, while effective for cloud-type classification, the approach lacks applicability to projects aimed at predicting the hour of sky pictures. This misalignment arises due to fundamental differences in objectives, features, and model optimization.

D. PCA in image processing

Images are inherently high-dimensional data, with each pixel representing a dimension. PCA effectively reduces this dimensionality by transforming the original data into a new set of variables called principal components. In the article Omar et al., "Image Compression using Principal Component Analysis", all the steps required to compress an image using PCA are detailled as well as the advanges, drawbacks and limitations of this method. PCA has been widely use for face recognition techniques, an example is shown in Alkandari and Aljaber, "Principle Component Analysis algorithm (PCA) for image recognition". Those steps can be also applied to sky recognition in order to understand fully the image and classify it.

E. CNN in Image recognition and processing

The CNN model for CIFAR-10 dataset achieved an accuracy of 80.17% on the test set, demonstrating the efficacy of advanced deep learning techniques such as data augmentation and dropout in enhancing model performance on complex image classification tasks Chauhan et al., "Convolutional Neural Network (CNN) for Image Detection and Recognition". The implementation of data augmentation techniques such as rotation, width shift, height shift, shear, and zoom enhances the robustness of the Convolutional Neural Network by artificially expanding the training dataset and introducing variability, which aids in preventing overfitting. Utilizing a pre-trained model such as MobileNetV2 leverages transfer learning to improve classification accuracy by applying the learned features from a large dataset to a specific task, thereby enhancing performance on limited training dataWu, "Introduction to convolutional neural networks"..

III. Experimentation

The following steps have been carried out. Our hypothesis in those experiments are that every cellphone would take the same picture, and that the location of the picture is not relevant.

A. Data acquisition

All sky pictures were captured using our cell phones. We hypothesized that pictures taken from different cameras (cell phones) would have the same quality. As a result, we assumed that using different cameras did not introduce variability into our model. We designated specific hours for these captures: 8 AM, 12 PM, 4 PM, and 8 PM. This four-hour interval between shots was crucial to document the varying hues of the blue sky throughout the entire day. Furthermore, all photographs were taken on consecutive days to mitigate changes in lighting conditions due to seasonal shifts.

In addition to adhering to the designated hours, several rules were enforced. Firstly, any obstructions such as buildings, planes, or birds were to be omitted from the frame, allowing only the sky to be captured. Moreover, we made efforts to minimize the presence of clouds in the images. The objective was to capture the variability of the blue zones over time for accurate time prediction. Including extraneous elements not only failed to contribute useful information but also constrained the amount of relevant data captured.

B. Data pre-processing

Once we obtained the pictures, some adjustments were necessary to remove unwanted information that could reduce accuracy. Our aim was to eliminate bias from our algorithm by removing irrelevant features that do not have a meaningful relationship with the target variable and by avoiding noise that complicates the learning process.

The first step was reducing the image size by cropping its boundaries. This helped homogenize the blue color in our pictures. Typically, the blue tone is brighter near the sun. By reducing the boundaries of the picture, we eliminated most of the bright blues that could confuse our algorithm.

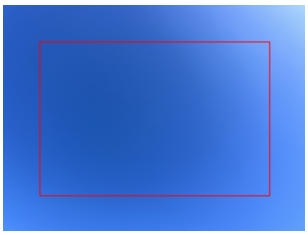

Fig. 1: Example of Cropped Image

Secondly, we generated patches from our images. This allowed us to expand our dataset without increasing the number of original images. Additionally, reducing the size of the elements improved the performance of our code. Importantly, all patches were generated randomly to avoid bias, overfitting, and underfitting, ensuring they were representative and generalized.

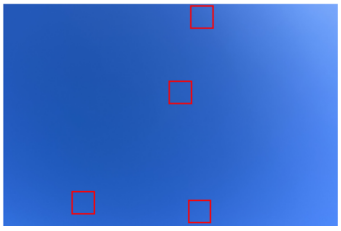

Fig. 2: Example of Random Patch Selection

Finally, it was crucial to filter the content of the patches. This involved removing white pixels from clouds in our model. There are two scenarios here. The first scenario occurs when the patch contains some white pixels from a cloud. In this case, the mean of the blue pixels in that patch is calculated, and all the white pixels (R>150, G>150, B>150) are replaced by this mean value. The second scenario occurs when the entire patch is a cloud. In this case, the patch is removed as it does not provide any useful information.

C. Principal Component Analysis (PCA)

In order to understand the images, a PCA algorithm, an orthogonal linear transformation is applied. The original dimension of our problem are patches of the images which have a dimension of 256*256*3.

1) Data Preparation: To perform Principal Component Analysis (PCA) on a color image, it is necessary to convert the image into a grayscale representation. However, to improve accuracy, rather than simply converting the image to grayscale, we opted to select one of the three color channels (Red, Green, or Blue). To determine the most suitable color channel, we analyzed the values of each color channel across several randomly selected pixels from images representing four different classes.

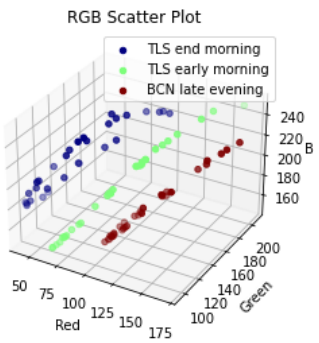

Fig. 3: 30 Random pixels value per channel in 3 different classes

In Figure 3, the red channel exhibited the most significant variation among the different classes, leading to the decision to use the red channel for subsequent analysis. Prior to applying PCA, the image patches are flattened from their original 256x256 dimensions to a 1x65536 vector. Additionally, the data were standardized using the following formula to ensure consistency and improve the performance of PCA:

$$Z = \frac{X - \mu}{\sigma} \quad (1)$$

X : the observations

μ : the mean

σ : the standard deviation

Using PCA allows for the reduction of data from a shape of (Number of patches, 65536) to (Number of patches, Dimension chosen). This reduction is achieved by computing the principal eigenvectors of the standardized data and projecting the data onto these vectors. This method captures the maximum variance in the data. Dimensionality reduction enables us to represent the entire image with only a few features, which can then be used for classification. Consequently, PCA significantly reduces the complexity of the problem. This reduction makes the model more efficient and less susceptible to overfitting.

2) Classification method: Subsequently, the data need to be classified into clusters. For this purpose, Gaussian Mixture Model (GMM) clustering has been utilized. To determine the optimal number of clusters, the Bayesian Information Criterion (BIC) is employed, with the optimal number of clusters ranging between 4 and 10. Once the clusters have been formed using the training set, a method for classifying the test set must be implemented. To assign a label to each cluster, the number of samples from each class within the cluster is counted. The cluster is then labeled based on the majority class it contains.

3) Validation method: Two validation methods have been implemented. The first method involves splitting the data into a training set and a test set, with the training set comprising 86% of the data and the test set comprising 14% (1 image per class). The second method uses k-fold cross-validation with 7 folds. The algorithm currently processes 7 images for each time of day; however, this does not define the number of patches. The number of patches can be specified as a parameter, thus altering the size of the training and validation datasets. Similarly, the number of dimensions after PCA can be set according to a parameter. In cases where the dimensions are reduced to 2 or 3, the results are plotted on a graph for visualization.

To evaluate the model's performance, accuracy and a confusion matrix are used. For k-fold cross-validation, the total confusion matrix and the average accuracy across all folds are displayed. It should be noted that there is inherent randomness

in the results due to the random selection of patches, which can vary in location within the image.

D. Results

In all the tests conducted, the first principal component of the PCA accounted for more than 98% of the variance. This indicates that the problem is largely linear, making it unnecessary to perform PCA with more than 2 dimensions. Despite this, it proved to be very challenging to accurately determine the time of day using PCA. One reason for this difficulty is that even though the images were taken during the same period and at the same times, variations in the orientation of the pictures could have influenced the results. An example of PCA output can be seen in Figure 4.

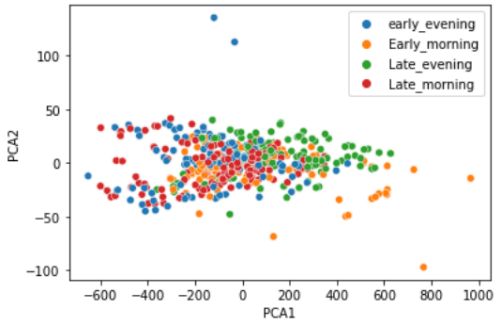

Fig. 4: Principal component analysis in 2D with 20 random patches per image

The late evening and the early morning goes towards high value of PCA1, while the "day picture" which represent the late morning and early evening goes towards low values. This is probably simply because the sky is darker in general in those pictures. No clear pattern between the 4 classes can be seen. The result of the GMM can be in Figure 5 :

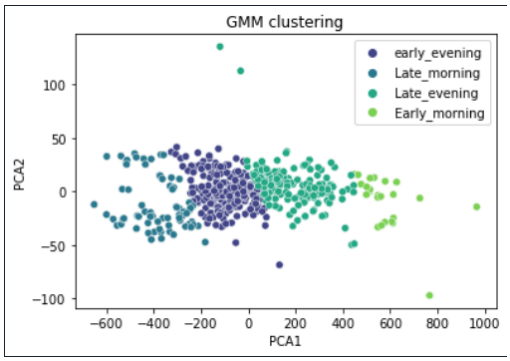

Fig. 5: GMM clustering with 20 bacthes

The GMM generally identified a pattern where data collected earlier in the day were grouped on one side, while data collected near sunset and sunrise were grouped on the other side. Based on these results, a binary classification model was tested to distinguish between daytime data and data collected around sunrise and sunset, to see if this approach would yield better performance.

In this experiment, the training accuracy was 0.4 and the test accuracy was 0.41. These accuracy rates indicate that the model made a significant number of errors.

The confusion matrices was :

$$Training : \begin{bmatrix} 13 & 7 & 53 & 47 \\ 0 & 32 & 57 & 31 \\ 0 & 25 & 69 & 26 \\ 11 & 6 & 26 & 77 \end{bmatrix} Testing : \begin{bmatrix} 5 & 2 & 3 & 10 \\ 0 & 4 & 9 & 7 \\ 0 & 5 & 15 & 0 \\ 0 & 0 & 11 & 9 \end{bmatrix} \quad (2)$$

When those patterns have been seen, it has been tried to classify the data using only two classes which would represent a picture during the day and a picture close to the night. A k-fold cross validation method has also been implemented in the code but the result where not shown here because the accuracy in general was lower.

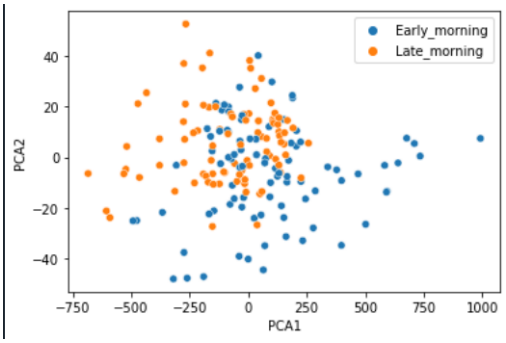

Fig. 6: Principal component analysis in 2D with 15 random patches per image and 2 classes

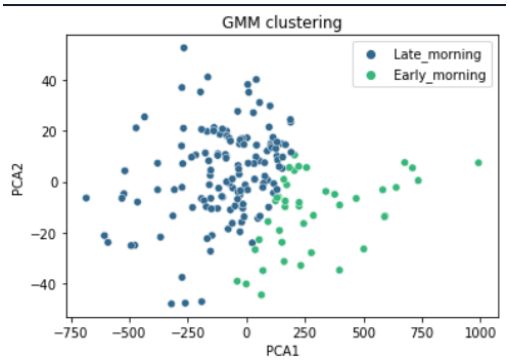

Fig. 7: GMM clustering with 15 bacthes and 2 classes

As it can be seen in Figure 6 and Figure 7 the model performs much better, on this experiment the accuracy of the training was 0.68 and the test accuracy was 0.8 which show a pattern with some mistakes.

In this case the confusion matrices was :

$$Training : \begin{bmatrix} 37 & 53 \\ 5 & 85 \end{bmatrix} Testing : \begin{bmatrix} 9 & 6 \\ 0 & 15 \end{bmatrix} \quad (3)$$

The model performs significantly better with two classes, suggesting that increasing the number of classes leads to lower

accuracy. Additionally, it has been observed that distinguishing pictures of the sky taken during the day is particularly challenging. Despite these difficulties, the model successfully captures a general pattern differentiating between daytime and near-nighttime images.

IV. Convolutional Neural Network (CNN)

Classifying the time of day from images of the sky is a challenging problem that can be effectively addressed using Convolutional Neural Networks (CNNs). This section outlines the steps involved in preparing, training, and evaluating a CNN for this task, including data preprocessing, augmentation, model building, and evaluation.

A. Data Labelling and Splitting

Before processing the images, they are normalised to a range of 0 to 1.Normalisation standardises pixel values, enabling efficient and successful model training. Handling errors throughout this process enhances robustness. Afterwards, labelling is needed. In this context, labels indicate the time of day each photograph was taken (e.g., morning, noon, evening, night). The categorical labels are translated into numerical indices and then encoded as one-hot vectors. This translation is essential as neural networks work with numerical data. One-hot encoding converts labels into binary vectors with a label index set to 1 and all other indices set to 0. After labelling, the dataset is divided into training and testing sets. By splitting the data, the model may be trained on one part and evaluated on another, ensuring generalisation to new data.

B. Data Augmentation and Model Building

To enhance model resilience and generalisation, data augmentation approaches are used. To provide fresh training examples, data augmentation utilises random changes of existing images, such as rotations, shifts, shearing, zooming, and flipping. This method is similar to practicing a skill in several settings to improve proficiency. Augmenting the training set with variety prevents overfitting and enhances model learning of robust characteristics. Creating the CNN model requires an architecture that learns from picture input. Different layers of the CNN provide different purposes. Convolutional layers use convolution processes to detect local characteristics like edges and textures in images. To perform these actions, a kernel is dragged over the input image and the dot product is calculated. Activation functions like ReLU (Rectified Linear Unit) introduce non-linearity for sophisticated pattern learning in models. Pooling layers, like MaxPooling, reduce input dimensionality and computational complexity while preserving key properties. Data is flattened into a one-dimensional vector and fed into dense (completely connected) layers for classification after convolutional and pooling layers.

C. Convolution Operation

The convolution operation involves applying a filter to the input image to extract features which is edges and textures. The formula for a 2D convolution is:

$$(I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n) \tag{4}$$

where:

- I is the input image
- K is the kernel (filter)
- i, j are the coordinates of the output pixel

D. Activation Function (ReLU)

The Rectified Linear Unit (ReLU) activation function introduced non-linearity into the model, allowing it to learn complex patterns. The formula is:

$$f(x) = \max(0, x) \tag{5}$$

E. Max Pooling Operation

Max pooling is reduced the dimensionality of the input while retaining important features. The formula for max pooling is:

$$P(i,j) = \max_{(m,n) \in R(i,j)} I(m,n) \tag{6}$$

where $R(i,j)$ is the pooling region.

F. Softmax Function

The softmax function is used in the output layer for multi-class classification, converting logits into probabilities. The formula is:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{7}$$

where:

- K is the number of classes
- z_i is the logit for class i

G. Loss Function (Categorical Cross-Entropy)

The categorical cross-entropy loss function measured the performance of a classification model whose output is a probability value between 0 and 1. The formula is:

$$L = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(p_{i,c}) \tag{8}$$

where:

- N is the number of samples
- C is the number of classes
- $y_{i,c}$ is the true label
- $p_{i,c}$ is the predicted probability

H. Adam Optimizer

The Adam optimizer is used for updating the weights of the network. It combines the advantages of two other extensions of stochastic gradient descent, namely Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). The update rules are:

1) Compute the gradients:

$$g_t = \nabla_\theta J(\theta_{t-1}) \tag{9}$$

2) Update biased first moment estimate:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{10}$$

3) Update biased second raw moment estimate:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{11}$$

4) Compute bias-corrected first moment estimate:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{12}$$

5) Compute bias-corrected second raw moment estimate:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{13}$$

6) Update parameters:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{14}$$

where:

- θ_t are the parameters at time step t
- α is the learning rate
- β_1, β_2 are the exponential decay rates for the moment estimates
- ϵ is a small constant to prevent division by zero

I. Results from CNN and Comparison with Predefined Model

The pre-trained MobileNetV2 model achieves higher and more stable accuracy compared to the custom CNN. The MobileNetV2 model effectively leverages pre-trained features for better generalization and performance, even with limited training data. In contrast, the custom CNN shows a gradual learning curve with fluctuations, indicating potential overfitting and instability.

TABLE I: Key Differences Between Custom CNN and MobileNetV2

Aspect	**Custom CNN**	**MobileNetV2**
Architecture	Custom Conv2D and MaxPooling layers	Pre-trained MobileNetV2
Data Augmentation	Moderate	Aggressive
Training Accuracy	Fluctuating	Rapid convergence
Validation Accuracy	Fluctuating	Stable
Training Loss	Fluctuating decrease	Rapid, stable decrease
Validation Loss	Variable	Steady decrease

MobileNetV2 is generally the better choice for image classification tasks, particularly in scenarios with limited data. It demonstrates superior generalization and advanced feature extraction capabilities provided by transfer learning. Despite a slightly lower final accuracy (50% vs. 55% for the custom CNN), MobileNetV2 shows more stable training and validation curves, indicating better consistency and reliability. It effectively handles small datasets through pre-trained weights, making it more data-efficient and less prone to overfitting compared to the custom CNN. Additionally, MobileNetV2's aggressive data augmentation helps improve performance in limited data scenarios, making it more effective overall.

TABLE II: Key Analysis of Custom CNN and MobileNetV2

Criteria	**Custom CNN**	**MobileNetV2**
Generalization	Less robust	More robust
Overfitting	Higher	Lower
Data Efficiency	Lower	Higher
Feature Extraction	Basic	Advanced
Limited Data Performance	Less effective	More effective
FInal Accuracy (%)	55	50
Training Accuracy (%)	48.71	58.10

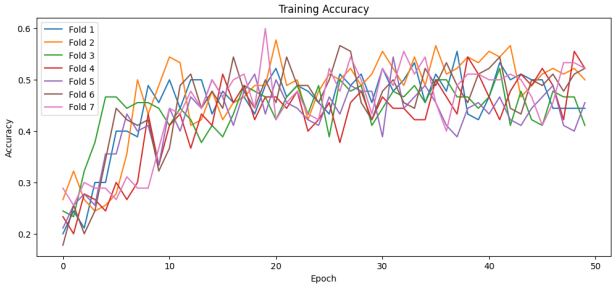

Fig. 8: Validation Loss for custom CNN

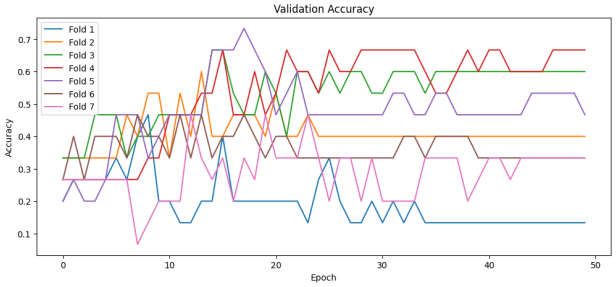

Fig. 9: Training Loss for custom CNN

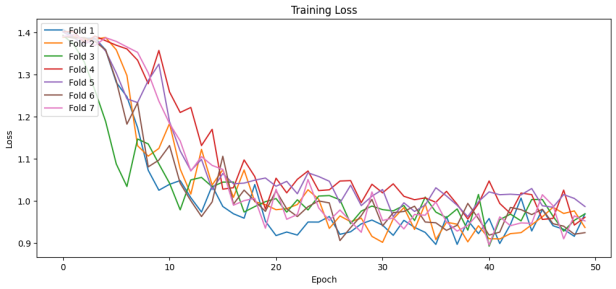

Fig. 10: Validation Accuracy for custom CNN

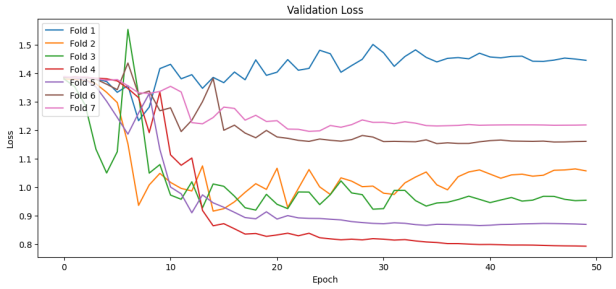

Fig. 11: Training Accuracy for custom CNN

V. Conclusion

Overall, we have met our objectives. By following two different approaches, we have developed algorithms capable of predicting the sky hour with just a picture. Even though the accuracy may not be extremely high, we believe it has good performance, particularly with the PCA preparation, GMM classification. While performing with this approach, it has been seen that the general pattern of the images in the PCA projection was with 2 classes. In terms of deep learning model, while the custom CNN achieved a slightly higher final accuracy, MobileNetV2's robustness, stability, and efficiency make it a more suitable choice for small datasets. Further fine-tuning and data augmentation could potentially enhance the performance of both models.

However, the accuracies indicate that there is room for improvement. One possible enhancement would be to take more pictures at the same established times. This would provide more information about the evolution of the red channel over the course of the day. Another possibility would be to take pictures on more widely separated days. This way, we could also account for seasonal changes. Finally, we would like to explore sky pattern recognition further by investigating different approaches beyond the RGB channels.

References

Alkandari, A., & Aljaber, S. J. (2015). Principle component analysis algorithm (pca) for image recognition. *2015 Second International Conference on Computing Technology and Information Management (ICCTIM)*, 76–80. https://doi.org/10.1109/ICCTIM.2015.7224596

Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018). Convolutional neural network (cnn) for image detection and recognition. *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 278–282. https://doi.org/10.1109/ICSCCC.2018.8703316

Cong Feng, M. C. (2017). Short-term global horizontal irradiance forecasting based on sky imaging and pattern recognition (ieee). *Short-term global horizontal irradiance forecasting based on sky imaging and pattern recognition (IEEE)*. https://doi.org/10.1109/PESGM.2017.8274480

Josep Calbó, J. S. (2008). Feature extraction from whole-sky ground-based images for cloud-type recognition. *Journal of Atmospheric and Oceanic Technology*, 3–14. https://doi.org/https://doi.org/10.1175/2007JTECHA959.1

Omar, H. M., Morsli, M., & Yaichi, S. (2020). Image compression using principal component analysis. *2020 2nd International Conference on Mathematics and Information Technology (ICMIT)*, 226–231. https://doi.org/10.1109/ICMIT47780.2020.9047014

Wojciech Makowiecki, W. A. (2008). New sky pattern recognition algorithm. *Computational Science – ICCS 2008*, 749–758. https://doi.org/https://doi.org/10.1007/978-3-540-69384-0_80

Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University*, *5*(23), 495.