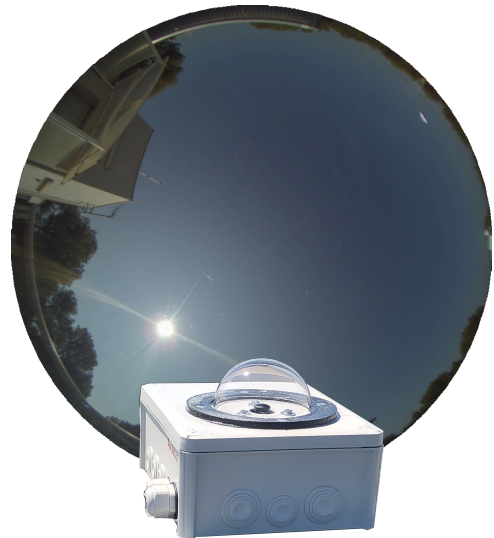


Applied Deep Learning, 2024 WS


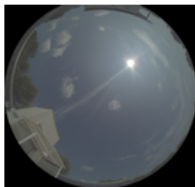
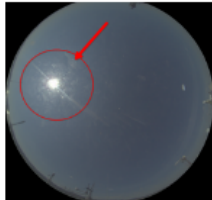
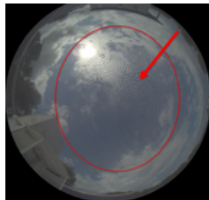
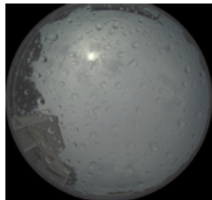
Assignment 3, Report by Valentin Bauer

Introduction – Hemispherical Sky Images

Hemispherical sky images, like the example on the right, can be used to estimate or forecast solar irradiation for solar energy applications [1, 3], such as photovoltaics or concentrating solar power plants. My master's thesis aims to provide the most accurate possible estimation of diffuse sky radiation per pixel from hemispherical sky images using a custom-built, low-cost, all-sky camera based on a Raspberry Pi 4. I collected a dataset of approximately 10,000 HDR images over four months, captured with two camera prototypes located on building rooftops in Vienna, Austria, and Sophia Antipolis, France, and will continue to collect more images in the future.



While working with the camera prototype, I encountered recurring challenges where image artifacts - primarily soiling, dew formation, and rain droplets - led to systematic errors in radiation measurements. Adding to these issues is my need to identify clear-sky conditions for radiometric calibration, making image quality assessment a fundamental prerequisite for accurate data collection. Examples of these conditions are shown below.

| Clear Sky | Clouds | Soiling | Water droplets inside | Water droplets outside |
|---|---|---|--|---|
|  |  |  |  |  |

I could not regularly maintain the cameras due to the high effort required to access them, which led to soiling, dew, and rain on the optics, persisting longer than optimal. I wanted to explore the possibility of an automated system detecting artifacts through multi-label image classification, potentially also enabling condition-based maintenance triggers. I decided to experiment with a classification task that aims to detect up to four simultaneous conditions: clouds (or clear sky), soiling, water droplets inside (from dew), and water droplets outside (from rain). Multiple labels can be present simultaneously - for example, an image might be classified as having both clouds and soiling, reflecting the real-world scenario where multiple artifacts can affect image quality at the same time.

Methodology – Multi-Label Classification

While initial research pointed me towards specialized architectures like TripleNet [2] for resource-constrained devices, practical constraints led me to adopt MobileNetV3 [4] as the foundation model, which can do inference on images on the Raspberry Pi 4 in less than a second. This decision was primarily driven by the availability of pre-trained weights and the model's native support for higher resolution inputs - a crucial factor given the detailed nature of dew and soiling artifacts.

I labeled **5,201 images** out of the 10,000 I had available, with 9,303 labels using the CVAT software¹. After experimenting with various data splitting strategies, I settled on grouping images by day of capture to prevent data leakage while maintaining label distribution balance across sets (more on that in the section [Lessons learned](#)), resulting in 4,103 training, 584 validation, and 514 test images. The distribution of labels in the subsets can be seen in the table below.

| Subset | # Clouds | # Rain | # Dew | # Clear sky | # Soiling |
|------------|----------|--------|-------|-------------|-----------|
| Train | 3261 | 381 | 1161 | 842 | 1695 |
| Validation | 498 | 86 | 86 | 86 | 247 |
| Test | 413 | 70 | 190 | 101 | 186 |

For model evaluation, I chose the **Jaccard Score** (also known as Intersection over Union), which measures how well predicted labels match true labels by dividing the number of correctly predicted labels by the total number of unique labels (both predicted and true). In my multi-label classification setting, where sky images can simultaneously exhibit multiple conditions (clouds, rain, dew, soiling, clear sky), a high Jaccard Score indicates that the model accurately predicts the exact combination of present conditions while avoiding false positives. Additionally, I analyzed **macro-averaged precision and recall** to assess balanced performance across all classes, calculated by averaging the per-class precision and recall scores.

My final model is a fine-tuned **MobileNetV3Large** model processing 224x224 pixel images to predict the likelihood (values between 0 and 1) of **five distinct labels** being present in a given hemispherical sky image: **clouds, rain, dew, clear sky, and soiling**. I assign a label if the model's output for the class is greater than 0.5. I trained the best-performing model (more on that in the section [Lessons learned](#)) without data augmentation at a learning rate of 0.001 using the Adam optimizer with weight decay for 60 epochs. I selected the best model from the training run based on its Jaccard Score on the validation set, which occurred in the 20th epoch.

Results

The final model achieved strong performance metrics on the test set, with a **mean Jaccard Score of 86.5%, macro-averaged precision of 87.2%, and recall of 86.6%**. The model excels at detecting common conditions like clouds with high precision (98.4%) and recall

¹ <https://www.cvat.ai>

(92.0%) but shows lower performance on rare conditions like rain (precision: 72.2%, recall: 74.3%). The complete class-wise performance metrics are shown in the table below.

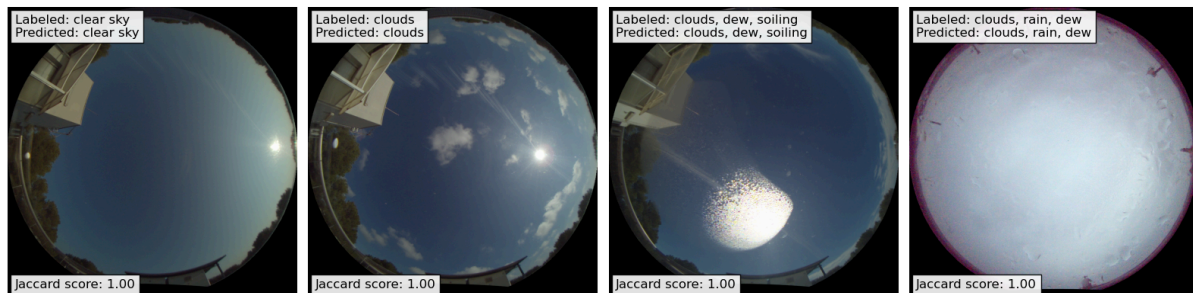
| Class | Precision (one-vs-rest) | Recall (one-vs-rest) | # train set labels | # test set labels |
|-----------|----------------------------|-------------------------|-----------------------|----------------------|
| Clouds | 98.4% | 92.0% | 44% (3261) | 43% (413) |
| Rain | 72.2% | 74.3% | 5% (381) | 7% (70) |
| Dew | 100.0% | 94.2% | 16% (1161) | 20% (190) |
| Clear sky | 74.2% | 94.1% | 11% (842) | 11% (101) |
| Soiling | 91.2% | 78.5% | 23% (1695) | 19% (186) |

Discussion

Classification performance

While the above metrics indicate reliable classification performance, examining concrete prediction examples is necessary to examine a model's capabilities thoroughly. The figure below shows cases where the model achieves perfect label prediction (Jaccard score of 1.0) across various conditions - clear skies, clouds, dew, rain, and soiling. These examples represent straightforward classification scenarios where visual features are distinct and well-defined, demonstrating the model's ability to learn clear visual patterns.

Classifications with Jaccard score equal to 1.0 (test set)

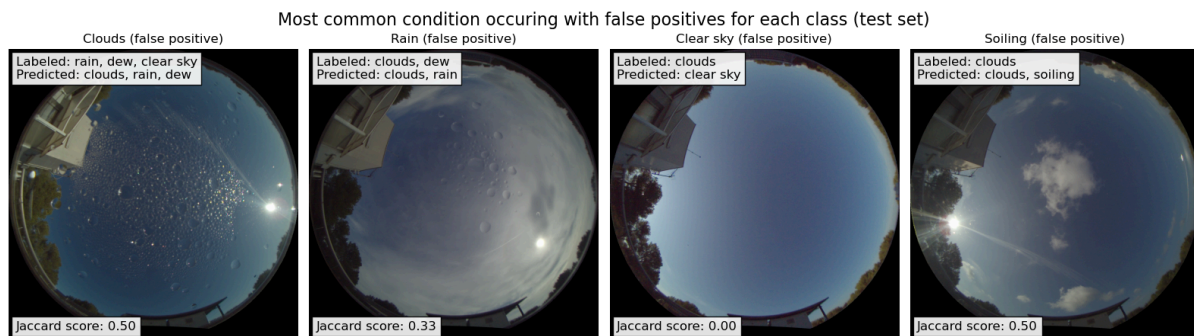


I analyzed misclassification patterns by examining false positives and false negatives for each class (except dew, which had no false positives in the test set). The confusion between clouds and clear sky classes emerged as a notable pattern, often associated with the presence of dew on the optics. However, several edge cases revealed potential ground truth labeling inconsistencies:

1. The clear sky false positive example shows minimal cloud coverage at the image periphery, suggesting the ground truth label might be incorrect rather than the model prediction.
2. Similarly, the soiling false positive case exhibits visible artifacts around the sun that could warrant a soiling label.

3. Lastly, the rain false positive demonstrates a challenging case where large dew drops were misclassified as rain, highlighting the model's difficulty in distinguishing between similar artifacts occurring for rain and dew.

These observations suggest that some apparent classification errors may stem from labeling ambiguities rather than model limitations, particularly in cases with subtle or overlapping visual features. Example images exhibiting the above-mentioned edge cases leading to false positives are shown below for each class, except for the dew class.



Lessons learned

Caching the entire dataset in memory reduces training time by orders of magnitude.

Initially, I struggled with long training times (2.5h+ for 30 epochs), which made experimenting with different hyperparameters tedious. After noticing low GPU utilization (less than 10%) during training, I suspected the training speed was I/O bound. I realized that it is not only possible to cache my entire dataset (2.6GB on disk) in memory, but it also reduces my training time massively. Without caching, every single epoch of the training run took around 7 minutes. With caching, every epoch after the first took only a few seconds (on an NVIDIA RTX 3080 FE).

The baseline model, trained by fine-tuning a MobileNetV3Large on sky images, performed the best.

I trained a baseline without additional data augmentation or architecture adaptation, and it performed the best out of all the enhancements I tested. After that, I experimented with data augmentation (color jittering, random rotation, and random flip) with arbitrarily chosen parameters, leading to worse model performance on the validation set. I hypothesize that these other models performed worse because I did not optimize the data augmentation parameters but arbitrarily chose some that I deemed sensible. I also experimented with changing the architecture by doubling the input image size from 224 to 448 pixels, but the resulting model also performed worse than the baseline.

Dataset splitting of time series data has many pitfalls. Every 15 minutes during the daytime, my cameras take an image of the sky. The weather conditions (clouds, rain, clear sky) and camera conditions (dew, soiling) change slowly from one image to the next. Halfway through the project, I realized that randomly shuffling the images before splitting them into the three subsets for training, validation, and testing would leak information from test and validation to the training set because a similar condition would be present in most successive images. However, splitting in time is problematic, too, because the label and image distribution changes over time (e.g., less rain in summer, more clouds in winter),

which reduces the efficacy of metric comparisons between the validation and test sets. Ultimately, I decided to group the images by day and split the days between the three subsets, which should mitigate the information leakage and give comparable distributions.

Labeling images is an iterative process and depends heavily on the human labeler. Starting with labeling, I implicitly expected consistency when labeling the images. Instead, I change my criteria for labeling an image over time. For the clear sky, clouds, and soiling classes, I decided to reexamine already labeled images a few times and change some labels. The motivation for relabeling came from the difficulty of deciding how images should be labeled, which showed tiny clouds on the image edges, thin clouds covering the entire sky, or tiny dust specs. Lastly, I should have combined the clouds and clear sky labels into one label, reducing the number of predictable labels to four because they are mutually exclusive and, therefore, can be represented more concisely with one label.

Working Time

| Category | Tasks | Hours |
|---------------------------------|--|-------------|
| Dataset Preparation | Image collection, preprocessing, and manual labeling | 21 (17%) |
| Model Development | Training pipeline setup, architecture adaptation, experimentation, debugging, and analysis | 67 (53%) |
| Application Development | Raspberry Pi testing, web interface development, and deployment | 21 (17%) |
| Documentation and Dissemination | Intermediate and final Report, Video | 17 (13%) |
| Total | | 126 |

I underestimated the time I would need for model development, especially its analysis, primarily because I did not include time for the analysis in my initial plan. Also, I got carried away trying to understand which images were causing the model to mispredict. However, I do not regret diving deeper into the model analysis, compared to simply showing some error metrics, because the misclassified images revealed problems with the ground truth labels rather than the trained model itself.

References

- [1] B. Nouri *et al.*, "Probabilistic solar nowcasting based on all-sky imagers," *Solar Energy*, vol. 253, pp. 285–307, Mar. 2023, doi: [10.1016/j.solener.2023.01.060](https://doi.org/10.1016/j.solener.2023.01.060).
- [2] R.-Y. Ju, T.-Y. Lin, J.-H. Jian, and J.-S. Chiang, "Efficient convolutional neural networks on Raspberry Pi for image classification," *J Real-Time Image Proc*, vol. 20, no. 2, p. 21, Feb. 2023, doi: [10.1007/s11554-023-01271-1](https://doi.org/10.1007/s11554-023-01271-1).
- [3] N. Blum *et al.*, "A Benchmark of Simple Diffuse and Direct Irradiance Measurement Systems," presented at the EU PVSEC, Lissabon, Portugal, Sep. 2023. Accessed: Mar. 20, 2024. [Online]. Available: <https://elib.dlr.de/199101/>
- [4] A. Howard *et al.*, "Searching for MobileNetV3," Nov. 20, 2019, *arXiv*: arXiv:1905.02244. doi: [10.48550/arXiv.1905.02244](https://doi.org/10.48550/arXiv.1905.02244).