Deep Learning - Weather Classification

Vincent Eberwein
Hasso-Plattner-Institute
Potsdam
vincent.eberwein@student.hpi.uni-potsdam.de

Joscha Schroff
Hasso-Plattner-Institute
Potsdam
joscha.schroff@student.hpi.uni-potsdam.de

Abstract

We explore lightweight vision models for weather classification using the Weather Phenomenon Database (WEAPD) in the context of a small-scale student project, focusing on solutions suitable for edge-device deployment. Small-scale CNN architectures and a compact SWIN Transformer are fine-tuned with pretrained weights, data augmentation, and regularization techniques. Our analysis highlights consistent challenges in distinguishing visually similar weather phenomena, emphasizing inherent complexities within the dataset.

1 Introduction

Deep Learning has achieved significant success in image classification tasks, including applications in weather forecasting. In this project, as part of a Deep Learning course, we explore weather classification using image data. Our objective is to train and evaluate models that can classify images into 11 different weather phenomena.

We utilize the Weather Phenomenon Database (WEAPD), which consists of approximately 6,800 images covering various weather conditions. This dataset provides a diverse set of weather-related images, making it a suitable benchmark for evaluating classification models. However, challenges such as class imbalance and visual similarities between certain weather phenomena add complexity to the classification task.

As this is a student project, we have limited access to high-performance computing resources, and all experiments were conducted on a consumer-grade laptop. Given these constraints, we prioritize lightweight models that maintain a balance between efficiency and classification performance. While our primary goal is to explore different architectures under these conditions, the ability to train and deploy models with minimal computational requirements makes them a potential candidate for edge-device applications. In scenarios where cloud-based processing is not feasible—such as remote locations, real-time weather monitoring in the field, or low-power embedded systems—having a model that can function efficiently on local hardware could be highly beneficial.

To accomplish this, we explore different small versions of four convolutional neural networks (CNNs) along with a compact variant of the SWIN Transformer to assess their suitability for weather classification. By leveraging pretrained models, we aim to enhance feature extraction while maintaining efficiency. To increase robustness and prevent overfitting, we incorporate data augmentation and regularization techniques. We compare the different models based on accuracy and F1-score to evaluate their effectiveness in weather classification. Further analysis of misclassifications

Preprint. Under review.

reveals which weather classes are particularly challenging to distinguish, providing insights into common errors and potential areas for improvement.

The goal of this project is to determine the most effective model architecture and training strategy within the constraints of limited computational resources. The findings contribute to understanding how well small-scale deep learning models perform in weather classification tasks under such conditions.

2 Related Work

2.1 Weather Prediction

Weather stations usually contain multiple different sensors for detecting the current weather [BSIAK20] in order to do weather forecasting and warnings. While these sensors may not be expansive, they sum up over multiple stations. As nowadays almost every device has a camera, it might be interesting to use these already deployed sensors. The idea is not new as it was already tried in 2008 [RM08]. This approach and similar approaches [YLZ09, CA11] applied traditional computer vision methods like histogram- and colorspace evaluation.
While these methods either didn't have high accuracies or just as few as 2-3 classes, with the rise of cnns [KSH12] the first approaches for cnn based weather detection [EHE15, AHSZS20, GMST22] allowed for multiple classes or higher accuracies.
More modern approaches include also transformer architectures to solve the task of image based weather classification [Dew24, TLLL23].

2.2 Approaches on WEAPD Dataset

The Weather Phenomenon Database (WEAPD) dataset[Xia21] consists of 6862 images covering 11 weather classes, including some similar looking, but still different weather conditions and thereby adding more complexity compared to previous weather datasets, which typically contained no more than four very distinct classes. Dewi et al.[Dew24] demonstrated notable success on WEAPD, achieving an accuracy of 93% using the Vision Transformer (ViT) architecture.

2.3 Models

Recent surveys, such as those by Dumen et al.[DKYAA24] and Goh et al.[GAS$^+$24], have systematically evaluated various image classification models, highlighting the importance of selecting appropriate architectures and pretrained weights for specific tasks and computational constraints.

Lightweight convolutional neural networks (CNNs), including MobileNet[HZC$^+$17] and EfficientNet[TL20], are especially suited to applications with resource limitations or deployment on mobile and edge devices. The smaller variants of these models, such as MobileNetV2 and EfficientNet-B0, provide a balance of computational efficiency and performance.

ResNet[HZRS15] introduced residual connections in order to prevent the vanishing gradient problem. Versions such as ResNet-18 and ResNet-50 have proven highly effective in various image classification tasks.

Transformer-based architectures, such as Swin Transformer[LLC$^+$21] and Vision Transformer (ViT)[DBK$^+$21], have emerged as robust alternatives to CNNs in recent years, utilizing the initially for LLMs developed attention mechanism. Compact versions, such as Swin-Tiny and ViT-Small, allow the use in scenarios with constrained computational resources while retaining high performance.

3 Dataset

The *Weather Phenomenon Database (WEAPD)* serves as the foundation for our experiments, containing approximately 6,800 images categorized into 11 different weather phenomena. The dataset exhibits natural variations in image quality, perspective, and environmental conditions, making it a valuable benchmark for weather classification.

3.1 Image Characteristics

The images in the dataset vary significantly in resolution, with a mean size of 520×373 pixels and a median size of 400×280 pixels. While most images fall within a reasonable range, some exhibit notably higher resolutions, introducing variability in input dimensions. To ensure consistency and compatibility with pretrained models, all images are resized during preprocessing. Figure 1 presents a representative sample from each of the 11 classes.

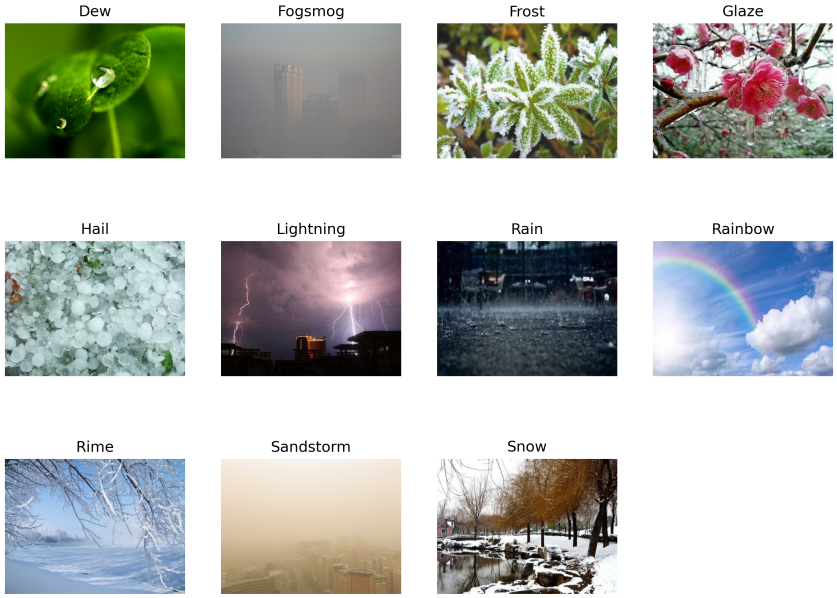

Figure 1: Sample images representing each of the 11 weather classes.

3.2 Dataset Splits

The dataset is split into 70% training, 15% validation, and 15% test data. The test set remains untouched during training and is used exclusively for final evaluation. This split ensures a sufficient number of samples for model optimization while maintaining an independent set for performance assessment.

3.3 Preprocessing and Data Augmentation

To maintain compatibility with pretrained models, all images are resized according to model-specific input requirements and standardized using ImageNet mean and standard deviation normalization. Additionally, various data augmentation techniques are applied during training to enhance generalization and robustness:

- **Random Resized Cropping**: Crops images to 224×224 pixels, scaling within 50% to 100% of the original size.
- **Random Horizontal Flip**: Simulates variations by mirroring images.
- **Random Rotation**: Rotates images by up to ± 10 degrees to introduce angular variability.
- **Color Jittering**: Adjusts brightness (0.5 to 1.5), contrast (0.5 to 1.0), saturation (0.5 to 1.5), and hue (± 0.1) to account for lighting variations.

These augmentations help mitigate overfitting and improve the model's ability to recognize weather patterns under diverse conditions.

3.4 Challenges and Observations

Several dataset-specific challenges impact classification performance:

- **Class Imbalance**: Certain weather phenomena, such as *rainbow* and *lightning*, are underrepresented. This reflects their natural rarity in real-world scenarios. Figure 2 illustrates the dataset's class distribution.
- **Image Artifacts**: Some images contain *TV overlays or watermarks*, introducing noise that may hinder model performance.
- **Overlapping Weather Phenomena**: Certain weather conditions frequently co-occur (e.g., *rime, glaze, frost and snow*), making classification ambiguous. In such cases, one phenomenon may visually dominate while another remains present, increasing the likelihood of misclassification. These issues are further analyzed in the evaluation section.

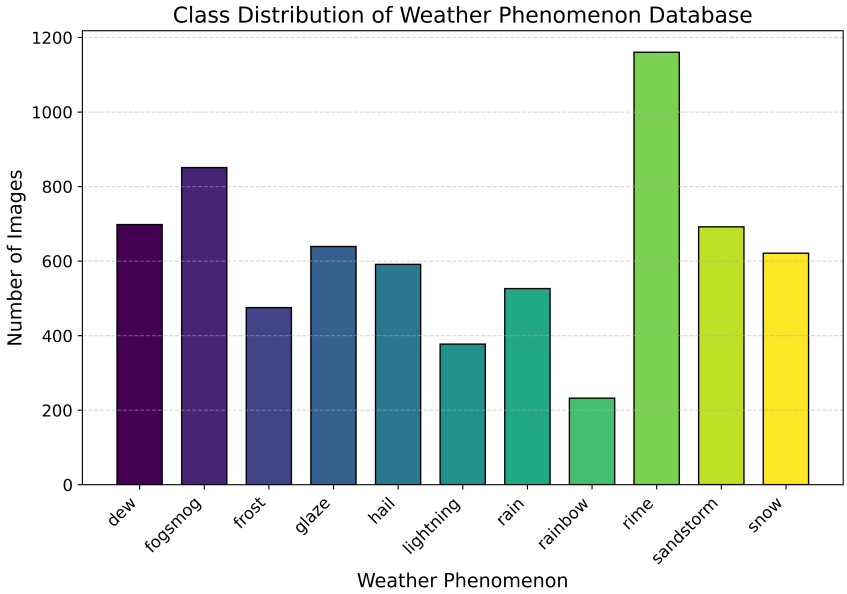

Figure 2: Class distribution of the dataset.

4 Methods

4.1 Architecture

We employ four different Convolutional Neural Network (CNN) architectures alongside a small Transformer-based model to compare their performance on our task. Specifically, we utilize ResNet50 [HZRS15], MobileNetV2 [HZC+17], EfficientNetB0, and EfficientNetB1 [TL20]. To contrast these with a lightweight Transformer-based approach, we also integrate a Swin Transformer Tiny (Swin-T) [LLC+21] model.

To evaluate the computational complexity of these models, we summarize their parameter counts in Table 1. The Swin-T model was included to assess whether a small Transformer-based approach can outperform the CNN models, despite requiring slightly more computational time.

4.2 Training

To enhance the robustness of the model and improve generalization, various data augmentation techniques were applied as mentioned in the data section.

The model was trained using the cross-entropy loss, which is commonly used for classification tasks. For the training, we used a batch size of 32 according to our hardware limitations and used early stopping to prevent overfitting.

Model	Parameters (Millions)
ResNet50	25.6
MobileNetV2	3.5
EfficientNetB0	5.3
EfficientNetB1	7.8
Swin-T	28.3

Table 1: Comparison of model architectures and their parameter counts.

After each epoch, the model was evaluated and saved if it had the highest accuracy compared to previous epochs, in order to take advantage of small peaks in the training.

All experiments were run on an Nvidia GeForce GTX 1650 GPU with 4GB memory.

4.3 Experiments

4.3.1 Sweep

In order to find the optimal hyperparameters for each model architecture, we swept over the most important hyperparameters using grid search. For each architecture, we trained 24 models to select the best for each.

- **Learning Rate**: For the initial learning rate, we chose $1e^{-2}, 1e^{-3}, 1e^{-4}$, as higher and lower values didn't seem to work at all in early experiments.
- **Learning Rate Decay Gamma**: The learning rate is updated after each epoch with $lr_t = lr_{t-1} \cdot \gamma$. We chose $0.1, 0.5, 0.9$ for γ to adjust the learning rate by one order of magnitude per epoch, halve it, or reduce it only minimally. In addition, we added 1.0 for no learning rate decay.
- **Use of Classweights**: As the dataset has huge class imbalances as shown in the dataset section, we added class weights while training. Class weights ensure that each class has the same influence on the training loss, by multiplying the loss for each image by the class weight of it's class. In the sweep, we trained each hyperparameter combination with and without the use of class weights.

4.3.2 Model Ensemble

We chose the best weights swept weights for each architecture and made predictions on the test dataset. We then tried two methods of combining the results of each model:

- **Average Voting**: For average voting, we averaged for each prediction the softmax values between the different models and then chose the prediction.
- **Max Voting**: For max voting, we first chose the predictions for each model and then chose the dominant predicted class for each model.

5 Evaluation

5.1 Metrics

For the evaluation of our trained models, we utilize the test split to ensure a fair and consistent comparison. The primary evaluation metrics used are **Accuracy** and **F1-Score**. Accuracy provides a straightforward measure of overall correctness, while the F1-Score accounts for class imbalances by considering both precision and recall.

5.2 Model Performance

We conducted an extensive hyperparameter sweep to identify the best-performing model for each architecture. The comparison of all models in terms of Accuracy and F1-Score is shown in Figure 3.

The results indicate that all architectures achieve high and consistent performance, with only minor differences. EfficientNetB0 and B1, along with Swin-T, demonstrated the best overall performance. Given that EfficientNet models are both smaller and computationally more efficient than Swin-T, they would be the preferred choice for deployment.

Interestingly, despite the class imbalance in our dataset, the best-performing models were trained *without* using class weights in the loss function. This suggests that the architectures effectively learned discriminative representations without requiring explicit weighting adjustments.

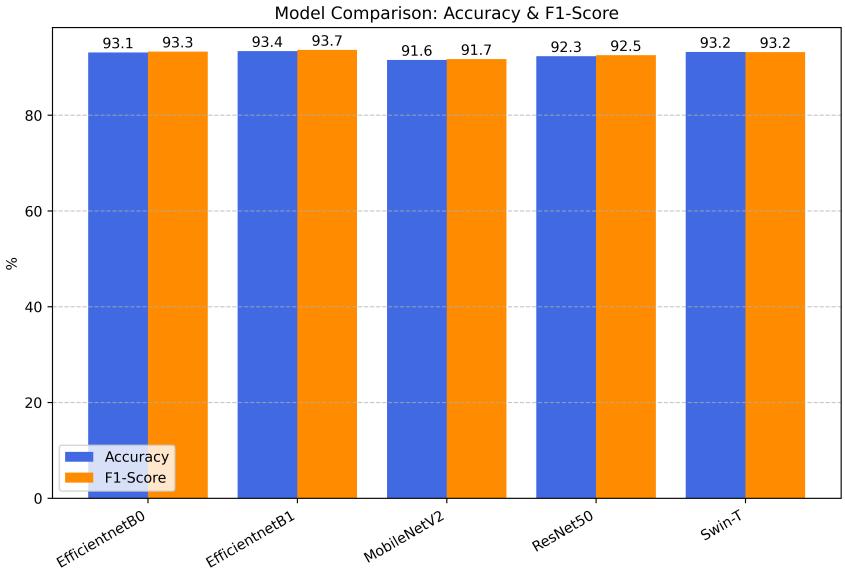

Figure 3: Comparison of model architectures based on Accuracy and F1-Score.

Overall, these results demonstrate that all architectures can classify weather phenomena with high accuracy and robustness, making them suitable for deployment in real-world scenarios.

5.3 Ensemble Learning

To further explore performance improvements, we experimented with ensemble learning by combining all five models using both max voting and mean voting techniques. Despite theoretical expectations of performance gains through ensemble methods, our results indicate no significant improvement, as both ensemble strategies yielded an accuracy of 93%, identical to the best-performing single models. Additionally, ensemble learning introduced a higher computational cost due to the need to run multiple models simultaneously, making it less practical for deployment. Given that the individual architectures already capture sufficient discriminative information, the additional computational burden of ensembling does not justify its use in this case.

5.4 Misclassification and Confusion Matrix Analysis

To gain deeper insights into the challenges of classifying weather phenomena, we analyzed the confusion matrices for all five models, as shown in Figure 4. Our findings indicate that all models exhibit similar error patterns, suggesting that the individual architectures already capture sufficient discriminative information. However, some weather conditions remain difficult to classify, either due to their inherent visual similarities or because certain images are particularly challenging.

We observed notable misclassifications between:

- **Sandstorm and Fogsmog**: The visual similarity in density and texture likely contributes to the misclassification.
- **Rime, Glaze, Frost, and Snow**: These phenomena often co-occur, making it challenging to isolate one specific category.

- **Snow and Rain**: The models frequently confused these categories, suggesting challenges in distinguishing between different precipitation types.

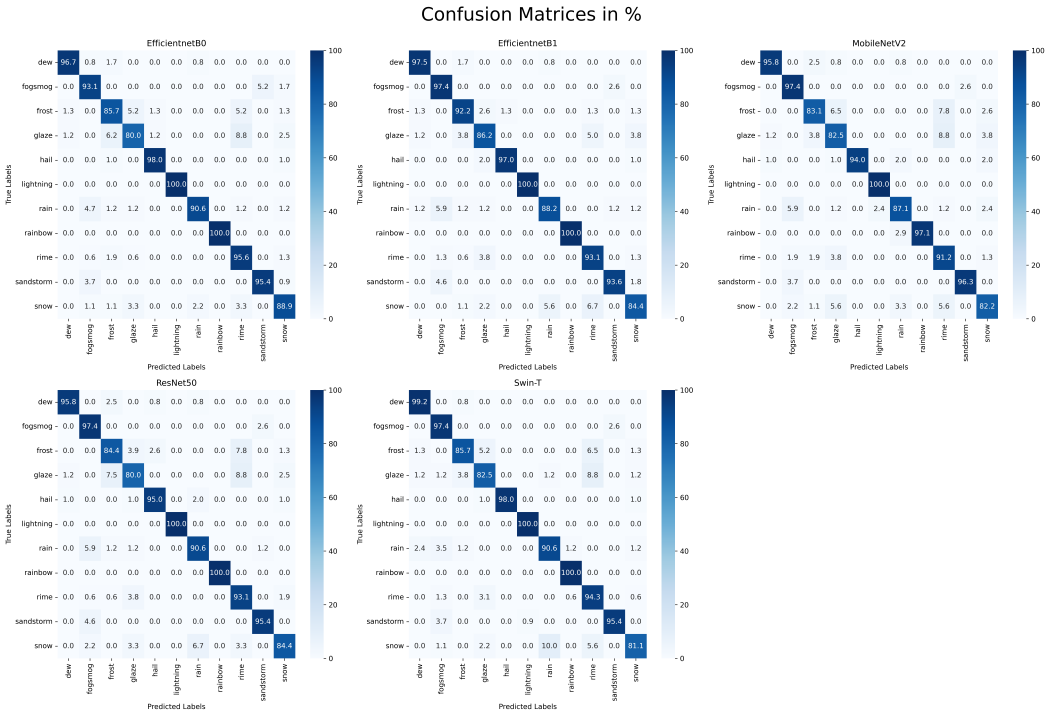

Figure 4: Confusion Matrices of all models

Furthermore, we identified 23 images that were misclassified by all five models, accounting for approximately one-third of all misclassifications. This suggests that some weather conditions are inherently difficult to distinguish, even for deep learning models trained on a diverse dataset. Figure 5 illustrates nine such examples, highlighting the complexity of these cases.

While some misclassifications may stem from model limitations, certain images might also be ambiguous by nature. For example, sandstorms and fog/smog can appear visually similar under specific conditions, making even human labeling difficult in some cases. Additionally, some images could plausibly belong to multiple categories—such as the image labeled as "rain" in Figure 5, which also prominently features a rainbow. This raises the question of whether a multi-label classification approach might be beneficial for certain scenarios, allowing the model to assign multiple relevant categories rather than being forced into a single-label classification.

6 Discussion

Our evaluation results show that the smaller EfficientNet architectures slightly outperformed the other CNN models and were on par with Swin-T in terms of classification accuracy and F1-score. Given their smaller size and computational efficiency, they present the most practical choice for deployment. Despite class imbalance, the best models were trained without class weighting, indicating that modern architectures can learn effective representations without explicit adjustments. Ensemble learning, including average and max voting, did not significantly improve performance over the best single models while introducing additional computational overhead. Notably, all models exhibited similar misclassification patterns, suggesting that the dataset itself presents inherent classification difficulties rather than individual model shortcomings.

Several strategies could enhance model performance. Additional data augmentation techniques, such as mixup or cutmix, could improve generalization by generating more diverse training samples. Exploring alternative architectures, including larger models, may yield better feature extraction

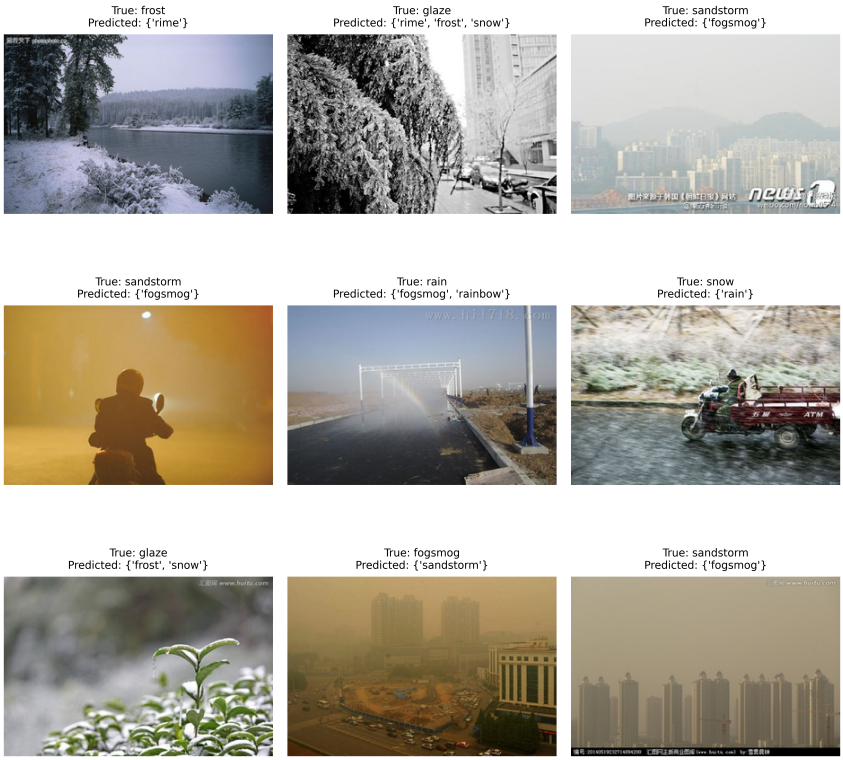

Figure 5: 9 misclassifications made by all 5 models

capabilities. Adjusting the learning rate decay strategy—such as using exponential decay or cosine annealing—could optimize convergence. Additionally, implementing a warmup period for learning rate scheduling might help stabilize early training, preventing suboptimal local minima. These enhancements could refine classification performance and improve model robustness.

A key challenge remains the classification of overlapping weather phenomena, such as *rime, glaze, frost, and snow*, which share similar visual characteristics. The confusion matrix analysis highlights frequent misclassifications, indicating that all models struggle with the same set of images. Filtering out particularly ambiguous images or refining the dataset labeling process could help mitigate these issues. Another approach could involve testing the models on other similar datasets to evaluate their generalization capabilities. If the same misclassifications persist across datasets, more advanced feature extraction techniques or alternative architectures may be required.

Future work could explore additional techniques to improve classification accuracy. One promising direction is contrastive learning, which could help distinguish visually similar weather conditions by learning more robust representations. Another avenue is implementing multi-label classification, allowing models to assign multiple categories to ambiguous images rather than forcing single-label predictions. Additionally, leveraging temporal information from weather video data could provide better contextual clues, improving classification performance for dynamic weather conditions.

References

[AHSZS20] Qasem Abu Al-Haija, Mahmoud A. Smadi, and Saleh Zein-Sabatto. Multi-Class Weather Classification Using ResNet-18 CNN for Autonomous IoT and CPS Applications. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1586–1591, Las Vegas, NV, USA, December 2020. IEEE.

[BSIAK20] Abu Saleh Bin Shahadat, Safial Islam Ayon, and Most. Rokeya Khatun. Efficient IoT based Weather Station. In *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, pages 227–230, December 2020.

[CA11] Aurélien Cord and Didier Aubert. Towards rain detection through use of in-vehicle multipurpose cameras. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 833–838, June 2011. ISSN: 1931-0587.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. arXiv:2010.11929 [cs].

[Dew24] Christine Dewi. Enhancing Weather Scene Identification Using Vision Transformer. 2024.

[DKYAA24] Sezer Dümen, Esra Kavalcı Yılmaz, Kemal Adem, and Erdinç Avaroglu. Performance of vision transformer and swin transformer models for lemon quality classification in fruit juice factories. *European Food Research and Technology*, 250(9):2291–2302, September 2024.

[EHE15] Mohamed Elhoseiny, Sheng Huang, and Ahmed Elgammal. Weather classification with deep convolutional neural networks. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 3349–3353, September 2015.

[GAS+24] Jocelyn Hui Lin Goh, Elroy Ang, Sahana Srinivasan, Xiaofeng Lei, Johnathan Loh, Ten Cheer Quek, Cancan Xue, Xinxing Xu, Yong Liu, Ching-Yu Cheng, Jagath C. Rajapakse, and Yih-Chung Tham. Comparative Analysis of Vision Transformers and Conventional Convolutional Neural Networks in Detecting Referable Diabetic Retinopathy. *Ophthalmology Science*, 4(6):100552, November 2024.

[GMST22] Moshira Ghaleb, Hala Moushier, Howaida Shedeed, and Mohamed Tolba. Weather Classification using Fusion Of Convolutional Neural Networks and Traditional Classification Methods. *International Journal of Intelligent Computing and Information Sciences*, 0(0):1–13, May 2022.

[HZC+17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, April 2017. arXiv:1704.04861 [cs].

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. arXiv:1512.03385 [cs].

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[LLC+21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, August 2021. arXiv:2103.14030 [cs].

[RM08] Martin Roser and Frank Moosmann. Classification of weather situations on single color images. In *2008 IEEE Intelligent Vehicles Symposium*, pages 798–803, June 2008. ISSN: 1931-0587.

[TL20] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, September 2020. arXiv:1905.11946 [cs].

[TLLL23] Jun Zhi Tan, Jit Yan Lim, Kian Ming Lim, and Chin Poo Lee. Weather Image Recognition Using Vision Transformer. *2023 IEEE 11th Conference on Systems, Process & Control (ICSPC)*, pages 50–55, December 2023. Conference Name: 2023 IEEE 11th Conference on Systems, Process & Control (ICSPC) ISBN: 9798350340860 Place: Malacca, Malaysia Publisher: IEEE.

[Xia21] Xiao Xiao. Weather phenomenon database (WEAPD), 2021.

[YLZ09] Xunshi Yan, Yupin Luo, and Xiaoming Zheng. Weather Recognition Based on Images Captured by Vision System in Vehicle. volume 5553, pages 390–398, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. Book Title: Advances in Neural Networks – ISNN 2009 Series Title: Lecture Notes in Computer Science.

Appendix

6.1 Sweep Parameters

General Training Parameters	
num_workers	4
accelerator	cuda
seed	42
num_epochs	20
batch_size	32
val_before_train	False
Learning Rate Schedule Parameters	
learning_rate	[0.01, 0.001, 0.0001]
lr_step_size	1
lr_gamma	[0.1, 0.5, 0.9, 1.0]
Model Parameters	
model_name	['efficientnetb0', 'efficientnetb1', 'mobilenet', 'resnet50', 'swin']
num_classes	11
save_model	False
save_model_path	./models
Data	
data_dir	./data/weather-dataset
use_class_weights	[True, False]
Early Stopping	
early_stopping	True
early_stopping_patience	3
early_stopping_metric	val_loss
early_stopping_mode	min
early_stopping_delta	0.001
Wandb	
wandb_project	sweep-models
wandb_entity	dl25-weather

Table 2: Hyperparameters used while sweeping the best model configuration. Multiple values in square brackets mean that this parameter was part of the sweep. Single values were fixed during the sweep.

6.2 Best Model Configs

The following configs were the result of the sweep and the best config for each architecture. The results presented in the paper were achieved with weights trained using these train parameters.

6.2.1 EfficientNetB0

General Training Parameters	
num_workers	4
accelerator	cuda
seed	42
num_epochs	20
batch_size	32
val_before_train	True
Learning Rate Schedule Parameters	
learning_rate	0.0001
lr_step_size	1
lr_gamma	0.9
Model Parameters	
model_name	efficientnetb0
num_classes	11
save_model	True
save_model_path	./models
Data	
data_dir	./data/weather-dataset
use_class_weights	False
Early Stopping	
early_stopping	True
early_stopping_patience	5
early_stopping_metric	val_acc
early_stopping_mode	max
early_stopping_delta	0.001
Wandb	
wandb_project	final-runs
wandb_entity	dl25-weather

Table 3: Best config for EfficientNetB0

6.2.2 EfficientNetB1

General Training Parameters	
num_workers	4
accelerator	cuda
seed	42
num_epochs	20
batch_size	32
val_before_train	True
Learning Rate Schedule Parameters	
learning_rate	0.001
lr_step_size	1
lr_gamma	0.5
Model Parameters	
model_name	efficientnetb1
num_classes	11
save_model	True
save_model_path	./models
Data	
data_dir	./data/weather-dataset
use_class_weights	False
Early Stopping	
early_stopping	True
early_stopping_patience	5
early_stopping_metric	val_acc
early_stopping_mode	max
early_stopping_delta	0.001
Wandb	
wandb_project	final-runs
wandb_entity	dl25-weather

Table 4: Best config for EfficientNetB1

6.2.3 MobileNetV2

General Training Parameters	
num_workers	4
accelerator	cuda
seed	42
num_epochs	20
batch_size	32
val_before_train	True
Learning Rate Schedule Parameters	
learning_rate	0.001
lr_step_size	1
lr_gamma	0.5
Model Parameters	
model_name	mobilenet
num_classes	11
save_model	True
save_model_path	./models
Data	
data_dir	./data/weather-dataset
use_class_weights	False
Early Stopping	
early_stopping	True
early_stopping_patience	5
early_stopping_metric	val_acc
early_stopping_mode	max
early_stopping_delta	0.001
Wandb	
wandb_project	final-runs
wandb_entity	dl25-weather

Table 5: Best config for MobileNetV2

6.2.4 ResNet50

General Training Parameters	
num_workers	4
accelerator	cuda
seed	42
num_epochs	20
batch_size	32
val_before_train	True
Learning Rate Schedule Parameters	
learning_rate	0.0001
lr_step_size	1
lr_gamma	0.5
Model Parameters	
model_name	resnet50
num_classes	11
save_model	True
save_model_path	./models
Data	
data_dir	./data/weather-dataset
use_class_weights	False
Early Stopping	
early_stopping	True
early_stopping_patience	5
early_stopping_metric	val_acc
early_stopping_mode	max
early_stopping_delta	0.001
Wandb	
wandb_project	final-runs
wandb_entity	dl25-weather

Table 6: Best config for ResNet50

6.2.5 Swin-T

General Training Parameters	
num_workers	4
accelerator	cuda
seed	42
num_epochs	20
batch_size	32
val_before_train	True
Learning Rate Schedule Parameters	
learning_rate	0.0001
lr_step_size	1
lr_gamma	0.9
Model Parameters	
model_name	swin
num_classes	11
save_model	True
save_model_path	./models
Data	
data_dir	./data/weather-dataset
use_class_weights	False
Early Stopping	
early_stopping	True
early_stopping_patience	5
early_stopping_metric	val_acc
early_stopping_mode	max
early_stopping_delta	0.001
Wandb	
wandb_project	final-runs
wandb_entity	dl25-weather

Table 7: Best config for Swin-T