Weather Monitoring with Deep Learning
Milestone report

Chau Chun Kai
ckchauag@connect.ust.hk

Chau Chi Hang
chchauag@connect.ust.hk

CHEN Hong quan
hchendq@connect.ust.hk

Abstract

Traditional weather monitoring methods are costly, time-consuming, and lack timeliness, which makes them unsuitable for real-time applications. To tackle this problem, we developed a deep learning framework to classify thirteen common weather conditions from 8000 images. Our study explored three primary approaches: a custom convolutional neural network, transfer learning with pre-trained ResNet models, and a hierarchical model structure. The custom CNN, with three convolutional layers, achieved a validation accuracy of 71%, acting as the baseline for comparison with other models. We fine-tuned ResNet models and using class weights to tackle the dataset imbalance, achieving the highest validation accuracy of 93%. The hierarchical approach grouped similar classes, for instance, 'fogsmog' and 'sandstorm' into 'foggy', and built three ResNet-based models: a grouped model with 94% validation accuracy, a 'foggy' model with 94% accuracy, and a 'cold' model with 85% accuracy. However, its overall accuracy of 89% is lower than the expectation, possibly due to the extra complexity and error propagation in multi-stage inference. Our real-time feature enabled immediate weather classification, which allows it to be used dynamically. The further improve the classifier, a more balanced dataset and further fine-tuning would be necessary.

1. Introduction

As the advancement of society, accurate and timely weather monitoring has been crucial for various fields, such as autonomous navigation, agricultural automation, outdoor event planning and climate research. Traditional weather monitoring methods, such as meteorological sensors, manual observations, or periodic data collection, are often costly, time consuming, and lack the ability to provide real-time updates. These limitations hinder their effectiveness in dynamic applications which require immediate weather conditions. To address these limitations, we aimed to develop a weather classifier to identify various weather conditions in real time without relying on traditional meterological equipment.

Our team focused on using computer vision and deep learning techniques to classify thirteen common weather types, such as 'snow', 'cloudy', 'lightning', and 'sandstorm' from imagery. To achieve our aim, we tried different approaches: a custom convolutional neural network (CNN) model, transfer learning [1] with pre-trained ResNet models [2], and a hierarchical model structure with grouped classes. Additionally, we implemented class weights to mitigate dataset imbalance and developed a real-time classification pipeline to enable practical deployment.

Among all the approaches, the custom CNN model, serving as a baseline, achieved a validation accuracy of 70.94%, limited by its simple architecture. Transfer learning with ResNet18 improved accuracy to around 86%, while ResNet152, enhanced with class weights, achieved the highest validation accuracy of 93.08%. The hierarchical model, with a final accuracy of 94.19% and 85.52% for the 'foggy' and 'cold' models respectively, despite achieving 94% accuracy on grouped classes, attained an overall accuracy of 88.99% due to its complex multi-stage inference. This project contributes to automated weather monitoring by showing the potential of deep learning for real-time, image-based classification, offering a scalable alternative to traditional methods.

2. Related Works

As a powerful potential application in climate research, agriculture, outdoor activity planning, and so on, weather classifiers using computer vision and deep learning have always been an interesting project for researchers. Some approaches have been published.

An approach is offered by Wang et. al in 2020 [3], which customized a lightweight CNN architecture like what the team did in the first approach.

It consists of four convolution layers (blue and brown blocks). Batch normalization and hard swish are placed between them. Then it passes the output to a Squeeze and Excitation (SE) model.

Instead of using standard convolution layers, Wang's model uses depthwise separable convolutions which are

Figure 1. Lightweight model structure

separated into two parts. It first does a pointwise convolution with a 1x1 filter to scan across each input pixel, then applies a separate 3x3 filter to extract features for each channel in the two Depth-wise 3x3 blocks, and finally a 1x1 pointwise convolution to combine all the information captured from the previous Depth-wise 3x3 block.

Furthermore, noticing that his model uses a hard swish activation function, it can therefore be more efficient compared to the ReLU activation function in our custom CNN approach.

The SE module is a layer that applies global average pooling to squeeze each feature map in each channel into a single number. After all that, the output is passed through a fully connected layer to perform classification.

Model name	Acc1	Acc2	Memory usage (MB)
InceptionV3	82.97	90.95	733.33
Resnet152	**92.98**	**96.55**	829.00
Densenet201	92.61	95.26	510.66
Vgg16	89.45	95.26	735.52
Vgg19	89.66	93.07	775.68
→ Proposed model	91.43	**96.55**	**32.97**
Performance gap	-1.55	**0.00**	**25.14 times**

Figure 2. Comparison with transfer learning models

His proposed model achieves an accuracy of around 93%, which outperforms our first approach: Custom CNN. Comparing it with our second approach: Transfer learning with ResNet18, it achieved 85 99% of accuracy which is slightly lower than what Wang's model did. However, the ResNet18 transfer learning approach is more generalized as it evaluated on 11 classes where Wang's model evaluated on only 6 classes which include "dew", "frozen", "haze", "rain", "dust" and "snow". Also, as ResNet18 is pre-trained on millions of diverse images, it is more powerful to adapt to varied real-world weather images and possibly be able to achieve real-time monitoring.

Another approach was proposed by Kang et al. in 2018 [4]. They constructed a weather classification model based on GoogLeNet with four categories. The GoogLeNet architecture that they used was based on the Hebbian principle and multi-scale processing, further deepening resource internal computation utilization and creating a 22-layered deep network. For weather image recognition, they turned the output size to 4 in GoogLeNet and collected a dataset of 20,000 images for training. To improve the model, they applied stochastic gradient descent (SGD) for softmax loss minimization. The accuracy of their model reached 92%. In comparison to our model, they utilized a larger dataset and achieved slightly higher accuracy. However, the model proposed by Kang was limited to identifying only three types of weather conditions: haze, rain, and snow, whereas our model is capable of recognizing 11 different types, offering a broader range of applications.

Overall, whether it is the CNN method proposed by Wang or the Transfer learning proposed by Kang, their trained models have good performance in accuracy. However, the types of weather they can train in are too limited to be widely used. Meanwhile, our model is more generalized with a wider range of target weather classes, which is more suitable for real-time weather monitoring.

3. Data

In this project, it is impractical to collect weather images manually as it is very time-consuming. More importantly, certain weather like sandstorms or snow is not common or does not even exist in Hong Kong due to its subtropical climate. To overcome this limitation, the team utilized publicly available datasets, which consist of weather conditions captured all over the world, including those rarely seen or even absent in Hong Kong, from Kaggle and other open-access repositories. By combining all the collected datasets, we obtained a total of 8000 labeled images, namely fogsmog, sandstorm, snow, rime, frost, glaze, cloudy, dew, hail, lightning, rain, shine, and sunrise. The distribution of the images are shown in Figure 3, which indicates a significant imbalance issue among the weather classes.

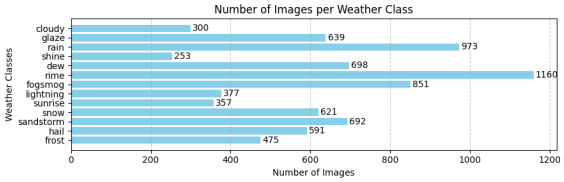

Figure 3. Image distribution in our dataset

To offer a better quality of training and ability to generalize for the model, several adjustments and transformations need to be preprocessed before training. Firstly, the images are resized to 224×224 pixels to match the input size requirements of the models. Secondly, perform random adjustments:

- flip the image horizontally with a probability of 50%
- adjust the brightness, contrast, saturation, and hue within a specified range
- apply affine transformations (10% translations in this case)

- apply a perspective transformation with 20% distortion, and a 50% probability

All these are aimed to simulate various changes in the real-life environment. Thirdly, the data is randomly cropped and resized back to 224×224. Fourthly, the randomly transformed data is translated to a PyTorch tensor and normalized using the specified mean and standard deviation so that the dataset can be trained in ResNet transfer learning [5]. Finally, the dataset is splitted into training subset and validation subset with a 80 to 20 ratio. When the data is ready, it will be passed to train the model.

4. Methods

To address the challenge of classifying diverse weather conditions from images, we explored different approaches and compared their results to determine the most effective strategy. These approaches included developing a custom convolutional neural network (CNN) model, using pre-trained ResNet models with transfer learning, and designing a hierarchical model structure.

4.1. Custom CNN

We built a custom CNN model to serve as a baseline for comparison with more advanced architectures, as shown in Figure 4. The model comprises three convolutional layers with increasing filter sizes (32, 64, and 128), each followed by a ReLU activation function and a max-pooling layer to reduce spatial dimensions. The output from the final convolutional block is flattened and fed into a dense layer with 512 units, followed by a dropout layer to mitigate overfitting. The final layer consists of an 11-unit dense output with softmax activation for multi-class classification.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_4 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_5 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten_1 (Flatten)	(None, 86528)	0
dense_2 (Dense)	(None, 512)	44,302,848
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 11)	5,643

Figure 4. Architecture of the custom CNN model

4.2. Transfer Learning with ResNet

To enhance classification performance, we employed transfer learning using pre-trained ResNet models. Initially, we fine-tuned a ResNet18 model, modifying its final fully connected layer to match the number of weather classes in our dataset. While ResNet18 outperformed the custom CNN, its lightweight architecture struggled to differentiate visually similar weather classes. Consequently, we adopted ResNet152, a deeper model, and experimented with unfreezing different numbers of its final layers to optimize feature extraction for our specific task. This approach leveraged ResNet152's robust pre-trained weights to capture intricate visual patterns.

4.3. Hierarchical Model Structure

The hierarchical model approach addressed the dataset's class imbalance and visual similarities among certain weather types, which often led to misclassifications, as observed in the confusion matrices. To tackle this issue, we grouped visually similar weather classes into broader categories, as shown in Figure 5. We defined nine grouped classes: 'foggy' ('fogsmog' and 'sandstorm'), 'cold' ('snow,' 'rime,' 'frost,' and 'glaze'), 'cloudy,' 'dew,' 'hail,' 'lightning,' 'rain,' 'shine,' and 'sunrise.'

```
grouped_dict = {
    'foggy': ['fogsmog', 'sandstorm'],
    'cold': ['snow', 'rime', 'frost', 'glaze'],
    'cloudy': ['cloudy'],
    'dew': ['dew'],
    'hail': ['hail'],
    'lightning': ['lightning'],
    'rain': ['rain'],
    'shine': ['shine'],
    'sunrise': ['sunrise']
}
```

Figure 5. Grouped class dictionary mapping original weather classes to broader categories

We trained three specialized models using pre-trained ResNet architectures: a grouped model based on ResNet152 to classify images into the nine grouped classes, a foggy model using ResNet50 to distinguish between 'fogsmog' and 'sandstorm', and a cold model using ResNet50 to differentiate among 'frost', 'glaze', 'rime'. and 'snow'. We selected ResNet50 for the foggy and cold models to avoid overfitting, which could arise from using the deeper ResNet152 for these binary and four-class tasks. During inference, an image is first processed by the grouped model. If classified as 'foggy' or 'cold', it is subsequently passed to the corresponding foggy or cold model for fine-grained classification. This hierarchical structure aimed to enhance accuracy by employing specialized models for visually similar classes.

4.4. Class Weights

To address the dataset's class imbalance, we explored the use of class weights to balance the influence of under-represented classes during training. We developed a function, `get_class_weights()`, to compute class weights

based on the inverse frequency of each class, as shown in Figure 6. These weights were applied to the cross-entropy loss function during model training to penalize misclassifications of minority classes more heavily, and hence improving the model's ability to learn from imbalanced data.

```python
def get_class_weights(class_counts):
    class_counts = np.array(class_counts)
    beta = 0.999
    effective_num = 1.0 - np.power(beta, class_counts)
    weights = (1.0 - beta) / effective_num
    weights = weights / weights.sum() * len(class_counts)
    return torch.tensor(weights, dtype=torch.float32, device=device)
```

Figure 6. Function `get_class_weights()` for computing class weights to address dataset imbalance

5. Experiments

We evaluate the performance of our weather classification approaches: a custom CNN model, transfer learning with ResNet models, and a hierarchical model structure. Below, we will show the training configurations, performance metrics, and observations for each approach, supported by loss, accuracy, and confusion matrix plots.

5.1. Custom CNN Model

We trained the custom CNN model and achieved a training accuracy of 71.09% and a validation accuracy of 70.94%. As shown in Figure 7, the training and validation loss and accuracy plots show steady improvement over epochs, indicating a well-balanced model with no evident signs of underfitting or overfitting.

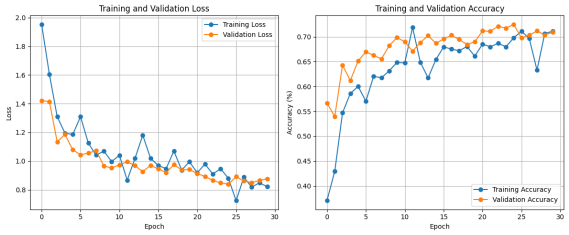

Figure 7. Loss and accuracy plots for the custom CNN model

5.2. Transfer Learning with ResNet

5.2.1 ResNet18

We fine-tuned a pre-trained ResNet18 model, adjusting only its final fully connected layer to match the number of weather classes. We trained the model for 15 epochs using the Adam optimizer with an initial learning rate of 0.01 and a decay factor of 0.8 applied every three epochs. As shown in Figure 8, the model achieved a training accuracy of 92% and a validation accuracy of 86%, significantly outperforming the custom CNN, particularly in the initial epochs.

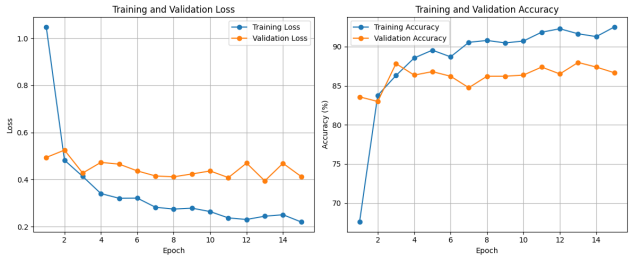

Figure 8. Loss and accuracy plots for the ResNet18 model

After collecting more data, we added a dropout layer with a probability of 0.3, keeping other hyperparameters unchanged. However, as shown in Figure 9, the loss and accuracy plots indicate unstable training, with a validation accuracy similar to the previous configuration.

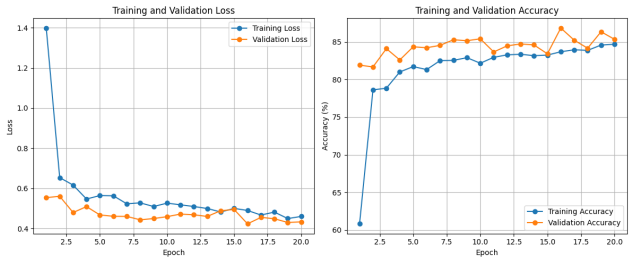

Figure 9. Loss and accuracy plots for the ResNet18 model with dropout

We also applied class weights for the imbalanced dataset, but the results are similar to the unweighted model. The confusion matrix, shown in Figure 10, reveals challenges in distinguishing certain visually similar classes like 'rime' and 'glaze'.

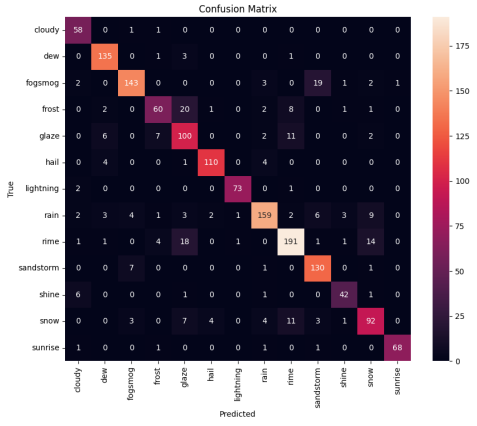

Figure 10. Confusion matrix for the ResNet18 model with class weights

5.2.2 ResNet152

For the ResNet152 models, we unfroze the last two blocks to capture more details from the images. We trained the model for 20 epochs with the Adam optimizer, an initial learning rate of 5e-4, and a decay factor of 0.9 every five epochs. This configuration achieved a validation accuracy of 92.45%, as shown in Figure 11, marking a notable improvement over ResNet18.

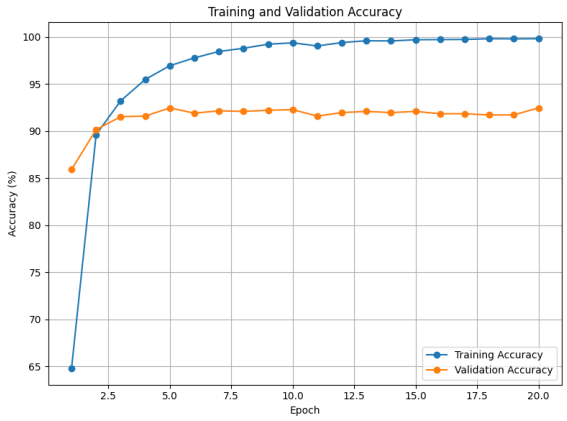

Figure 11. Accuracy plot for the ResNet152 model

We also applied class weights, and train the model for 30 epochs with an initial learning rate of 1e-3 and a decay factor of 0.8 every three epochs. This model achieved the highest validation accuracy of 93.08%. The confusion matrix, presented in Figure 12, shows only a few misclassifications, primarily between 'rime' and 'glaze,' indicating robust performance.

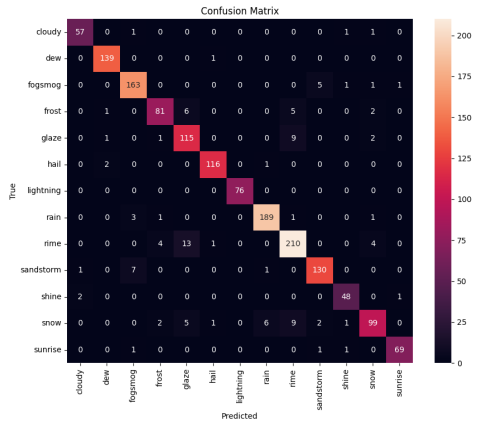

Figure 12. Confusion matrix for the ResNet152 model with class weights

5.3. Hierarchical Model Structure

5.3.1 Grouped Model

We trained a ResNet152 model to classify images into the nine grouped classes, using 30 epochs, an initial learning rate of 5e-4, and a decay factor of 0.8 every three epochs. The model achieved a validation accuracy of 94%, as shown in Figure 13, showing strong performance on the grouped classification task.

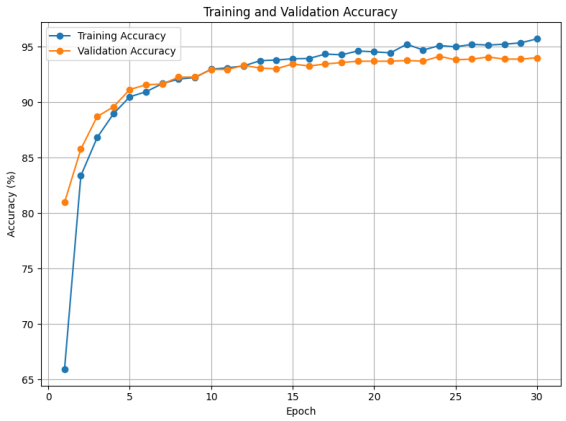

Figure 13. Accuracy plot for the grouped model

5.3.2 Foggy Model

For the foggy model, our initial attempt used a ResNet50 model trained for 30 epochs with an initial learning rate of 5e-4 and a decay factor of 0.8 every three epochs, achieving a validation accuracy of 91.34%, as shown in Figure 14.

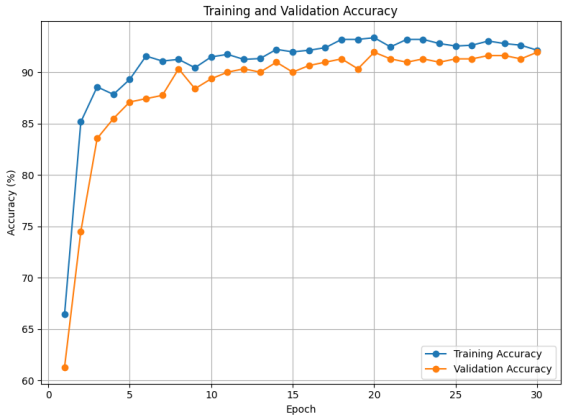

Figure 14. Accuracy plot for the initial foggy model

After fine-tuning, we adopted an initial learning rate of 1e-4 with a decay factor of 0.9 every five epochs and used binary cross-entropy loss, achieving an improved validation accuracy of 94.19%. The accuracy and confusion matrix

plots, shown in Figures 15 and 16, confirm enhanced performance.

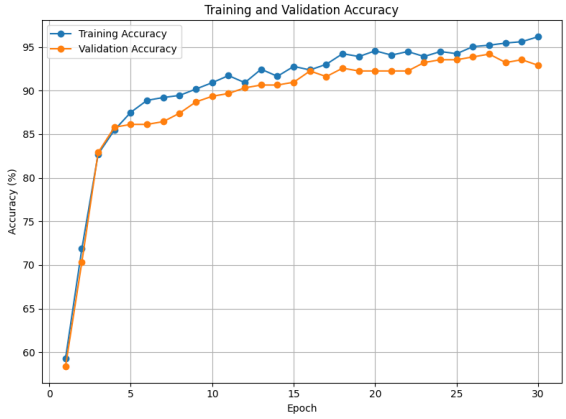

Figure 15. Accuracy plot for the fine-tuned foggy model

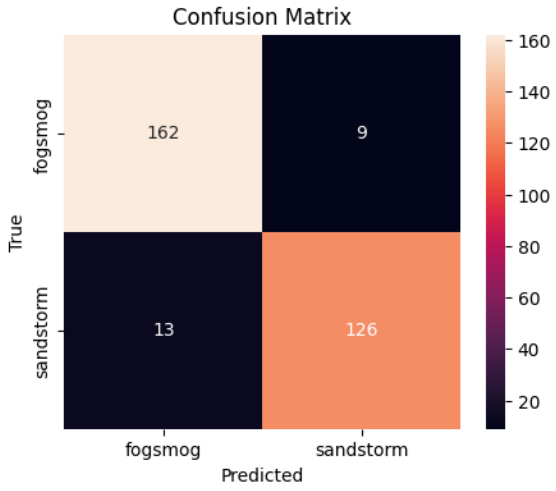

Figure 16. Confusion matrix for the fine-tuned foggy model

5.3.3 Cold Model

The initial cold model, based on ResNet50, was trained for 30 epochs with an initial learning rate of 5e-4 and a decay factor of 0.8 every three epochs, yielding a validation accuracy of 81.55%, as shown in Figure 17.

After fine-tuning, we used an initial learning rate of 2e-4 with the same decay schedule and implemented early stopping, which stopped the training at the 26th epoch out of 50. This configuration achieved a validation accuracy of 85.52%. The accuracy and confusion matrix plots, shown in Figures 18 and 19, indicate improved differentiation among cold classes.

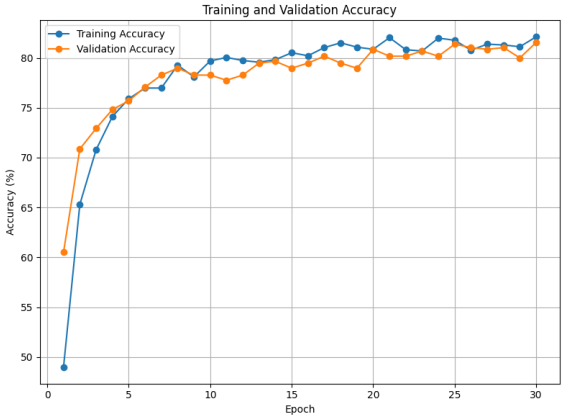

Figure 17. Accuracy plot for the initial cold model

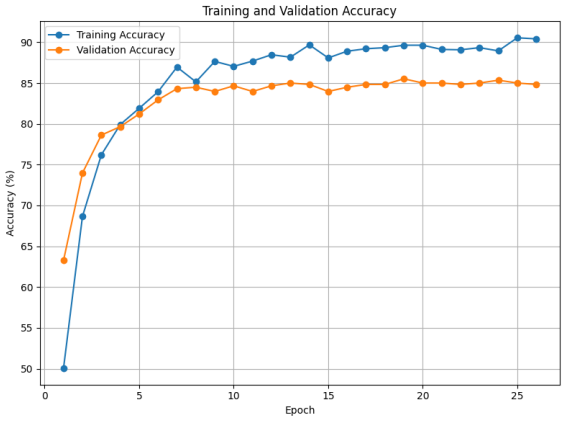

Figure 18. Accuracy plot for the fine-tuned cold model

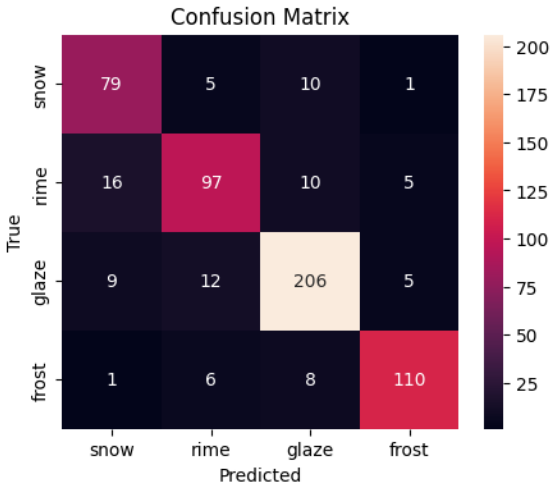

Figure 19. Confusion matrix for the fine-tuned cold model

5.3.4 Overall Result

Using the initial foggy and cold models, we implemented the hierarchical structure for inference on all 8,000 images, achieving an overall accuracy of 88.26%, as shown in the confusion matrix in Figure 20. This performance was lower than the expected.

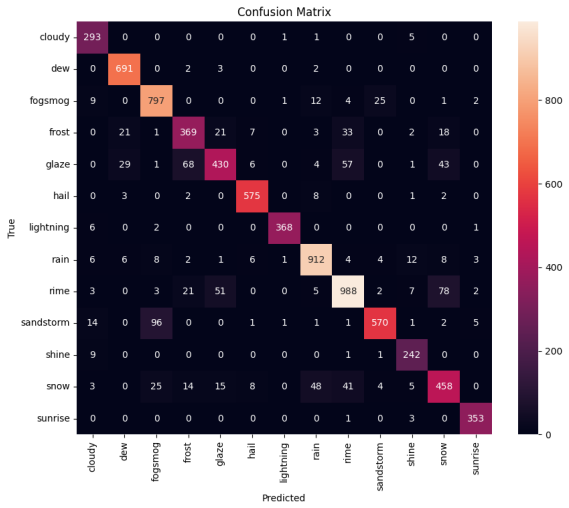

Figure 20. Confusion matrix for the hierarchical model with initial foggy and cold models

After fine-tuning the foggy and cold models, the hierarchical structure's overall accuracy improved slightly to 88.99%, as indicated in the provided notebook. However, this remained lower than the single ResNet152 model with class weights, which achieved 93.08% validation accuracy.

After researching and testing, we believed that the unexpected low performance of the hierarchical approach was due to its greater complexity. The multi-stage inference process, where there were successive predictions from the grouped, foggy, and cold models, introduced potential error propagation, particularly for visually similar classes. This additional complexity, coupled with the need for precise coordination between the three models, decreased overall accuracy below that of the solitary ResNet152 model, which achieved a 93.08% validation accuracy with a less complicated architecture.

5.4. Real-Time Classification

To enable practical deployment, we implemented a real-time classification pipeline integrated with the trained models. As shown in Figure 21, we designed a system using OpenCV to process continuous image streams, applying the same preprocessing steps (resizing to 224x224 pixels, tensor conversion, and normalization) as in training. The result is satisfactory and reliable for real-world applications.

Figure 21. Real-time Application with a time-lapse video

6. Conclusion

In conclusion, we developed a weather classifier with deep learning techniques to identify diverse weather conditions from images in real time, addressing the limitations of traditional meteorological methods. Our project employed four approaches: a custom convolutional neural network (CNN), transfer learning with pre-trained ResNet18 and ResNet152 models, a hierarchical model structure, and class weights to mitigate dataset imbalance. The custom CNN, serving as a baseline, achieved a validation accuracy of 70.94%, limited by its simple architecture. Transfer learning with ResNet18 improved performance to 86%, while ResNet152, enhanced with class weights, yielded the highest validation accuracy of 93.08%. The hierarchical model, comprising grouped, foggy, and cold models, achieved 94% accuracy on grouped classes but only 88.99% overall due to its complex multi-stage inference and error propagation. Additionally, we implemented a real-time classification pipeline using ResNet152, enabling immediate weather predictions for dynamic applications.

Through this project, we gained valuable insights into the impact of model architectures on image classification tasks. We learned that while basic CNNs provide a foundational approach, advanced pre-trained models like ResNet152 significantly enhance performance, particularly for visually similar classes. The application of class weights proved effective in addressing dataset imbalance, though challenges persisted with underrepresented classes. We also recognized the trade-offs of hierarchical models, where increased complexity can compromise accuracy. These experiences deepened our understanding of transfer learning, data preprocessing, and the critical role of balanced datasets in achieving robust model performance.

For future work, we propose expanding the dataset to balance class distributions, reducing biases in predictions for underrepresented weather types. Further testing of the

real-time classification pipeline under diverse weather conditions could enhance its reliability for applications like autonomous navigation, public transportation routing, and travel planning. Additionally, integrating multi-modal data, such as sensor inputs alongside images, could improve classification accuracy and broaden the system's applicability in real-world weather monitoring.

References

[1] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," IEEE TKDE, 2009. 1

[2] K. He et al., "Deep Residual Learning for Image Recognition," in CVPR, 2015. 1

[3] C. Wang, P. Liu, K. Jia, X. Jia, and Y. Li, "Identification of Weather Phenomena Based on Lightweight Convolutional Neural Networks," *Comput. Mater. Contin.*, vol. 64, no. 3, pp. 2043–2055, 2020. [Online]. Available: https://doi.org/10.32604/cmc.2020.010505 1

[4] C.-H. Chen, C.-T. Shen, and L.-W. Chang, "Deep Learning-Based Weather Image Recognition," in *Proc. 2018 IEEE 3rd Int. Conf. Image, Vision and Computing (ICIVC)*, Chongqing, China, Jun. 2018, pp. 631–635. [Online]. Available: https://ieeexplore.ieee.org/document/8644946 2

[5] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. [Online]. Available: https://pytorch.org/ 3