Weather Image Classification

Aaron Bergfeld, Daniel Costa, Bella Davies, Theo Hui
MIDS w207, Spring 2025

Motivation

Research Question: How accurately can a ML model recognize weather conditions from images?

Use Case: Autonomous vehicles could adjust their driving behavior based on incremental snapshots of weather conditions.

Goals:

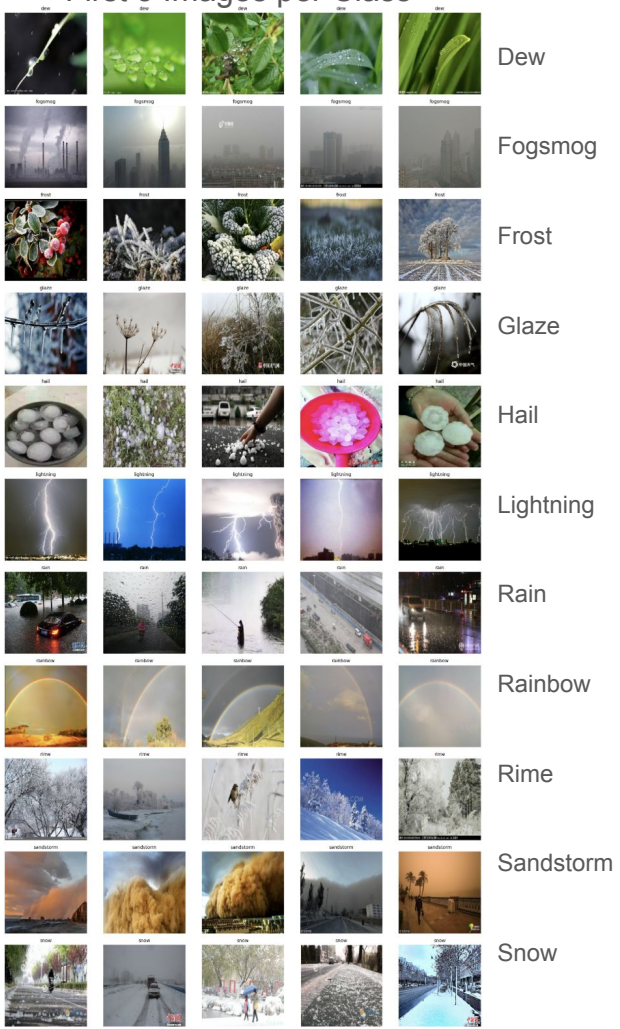
- Image preprocessing for CNNs
- Dimensionality reduction
- Experimentation with new models

Data

Weather Image Dataset:
https://www.kaggle.com/datasets/jehanbhathena/weather-dataset

- 6862 images belonging to 11 classes
  - Dew
  - Fogsmog
  - Frost
  - Glaze
  - Hail
  - Lightning
  - Rain
  - Rainbow
  - Rime
  - Sandstorm
  - Snow
- 80% Train: 5489 images
- 10% Validation: 686 images
- 10% Test: 687 images

First 5 Images per Class



Modeling

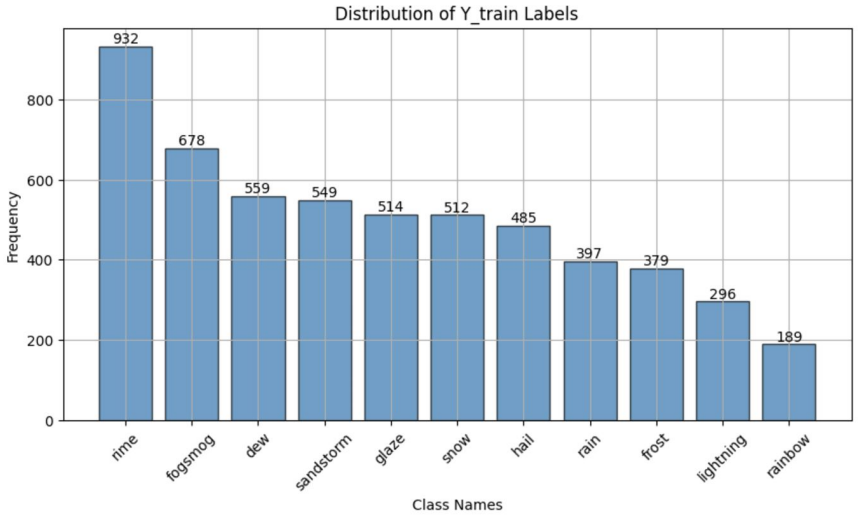
Model	Val Loss	Val Accuracy	*Val F1 Score	Val AUC
1. Baseline Log Reg	**1.50**	**0.51**	**0.46**	**0.88**
2. Log Reg w/ Oversampling	1.48	0.56	0.52	0.88
3. K-Means → CNN	3.85	0.60	0.58	0.93
4. PCA → CNN	1.49	0.59	0.58	0.93
5. Stacked 5+6 → Simple Log Reg	1.00	0.67	0.66	0.82
6. Stacked 5+6 → CNN Tuned	1.01	0.69	0.68	0.94
7. CNN	1.02	0.71	0.73	0.96
*8. CNN Tuned	0.8	0.82	0.83	0.98
*9. Resnet50 + Simple Log Reg	0.45	0.91	0.91	0.99
*10. Resnet50 + CNN Tuned	0.41	0.90	0.90	0.99
*11. SAM + Resnet50 + Log Reg	0.44	0.89	0.90	0.99

Experiments

1. Class Distribution Balancing: Oversampling
2. Data Augmentation
3. Hyperparameter Tuning: Keras Hyperband
4. Dimensionality Reduction: K-means Color Quantization
5. Dimensionality Reduction: Principal Component Analysis
6. Ensemble Learning: Combine Model Predictions for Stacking
7. Transfer Learning: ResNet-50 Feature Extraction
8. Image Segmentation: SAM (Segment Anything Model)

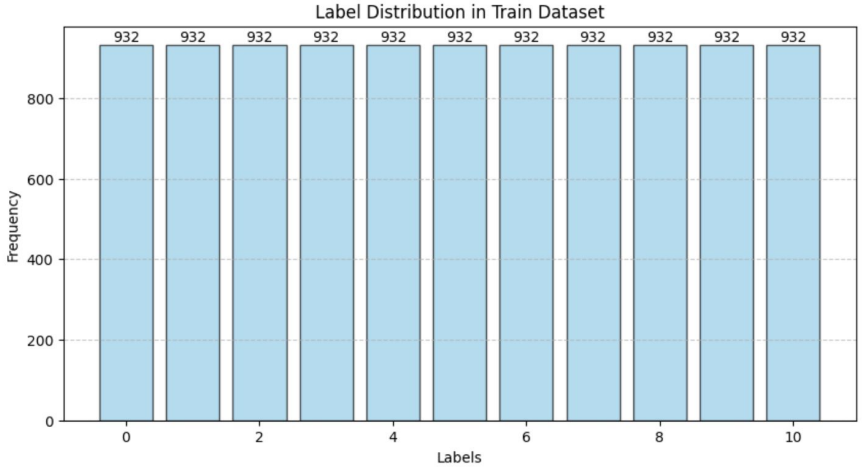
1. Class Distribution Balancing: Oversampling

Before Oversampling:



- X_train.shape = (5490, 224, 224, 3)

After Oversampling:



- X_train.shape = (10252, 224, 224, 3)

Improvement from Baseline Val Accuracy: 5%

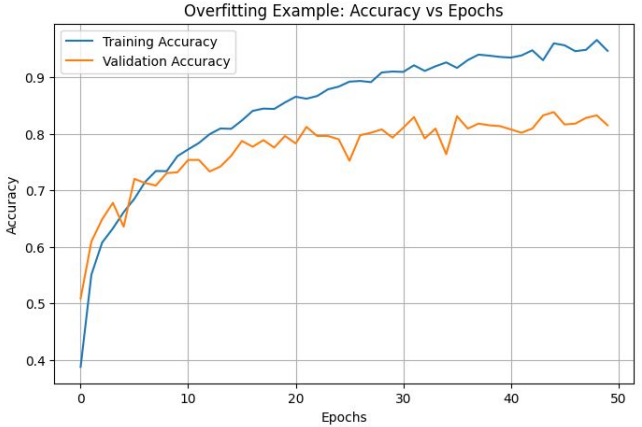
2. Data Augmentation

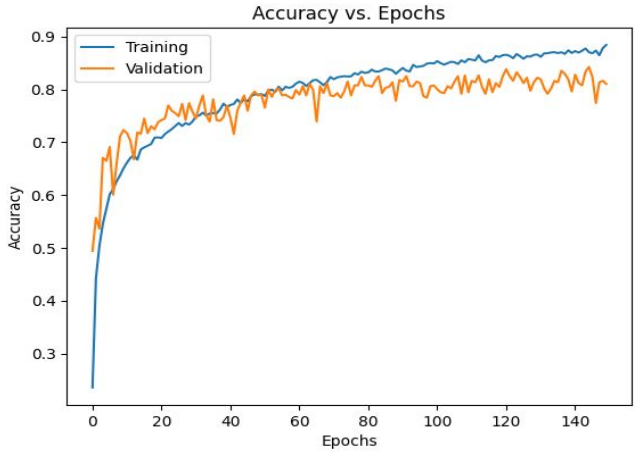
- Random Flip: "horizontal_and_vertical"
- Random Rotation: 0.2
- Random Zoom: 0.2
- Random Contrast: 0.2
- Random Brightness: 0.2

3. Hyperparameter Tuning: Keras Hyperband

CNN Tuned Parameters:

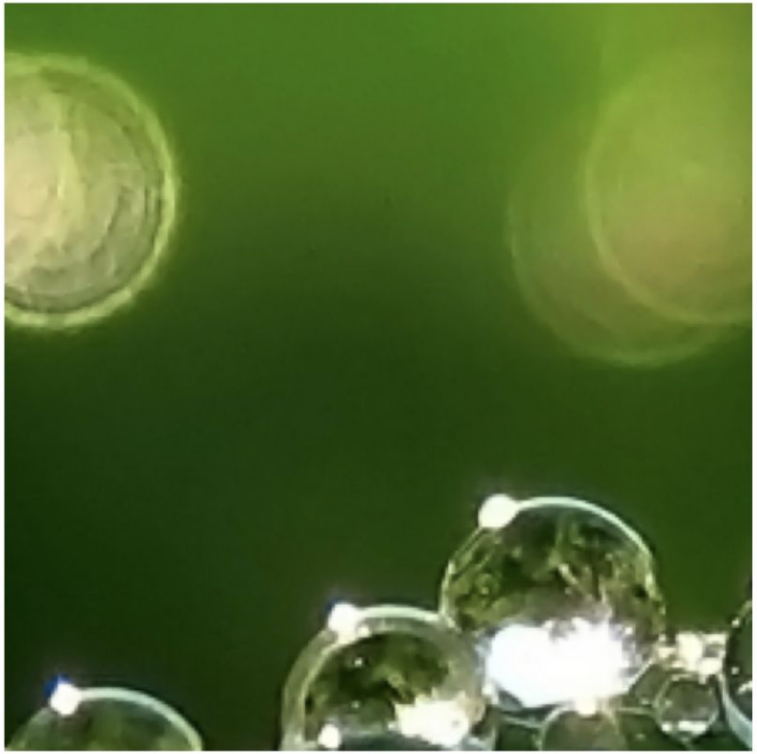
- conv_1 # of filters: 16
- conv_1 kernel size: 7x7
- conv_2 # of filters: 128
- conv_2 kernel size: 3x3
- conv_3 # of filters: 48
- conv_3 kernel size: 3x3
- dropout: 0.1
- # of dense units: 192
- L2 regularization: 0.001
- Learning_rate: 0.001



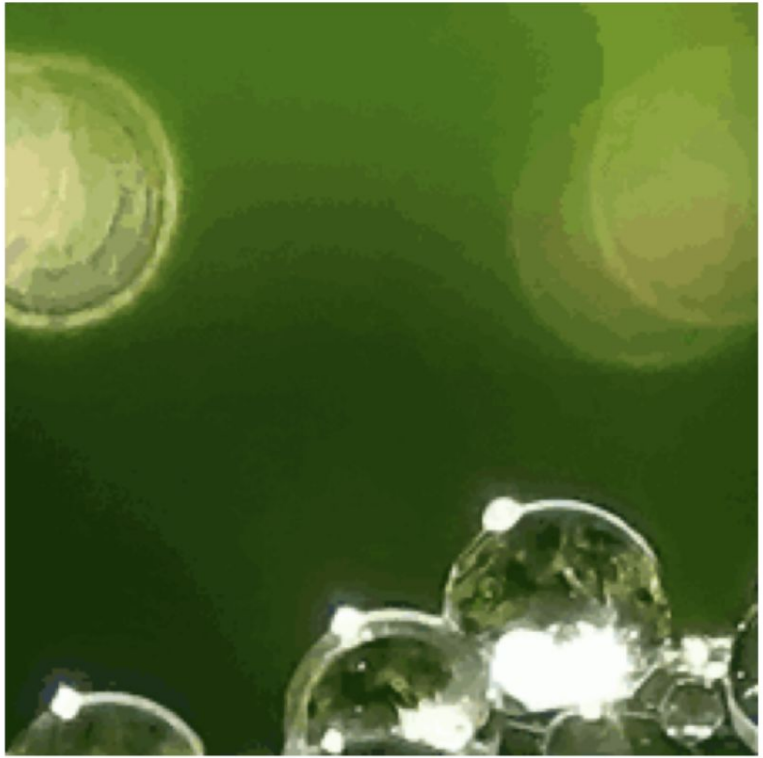


4. Dimensionality Reduction: K-Means Color Quantization

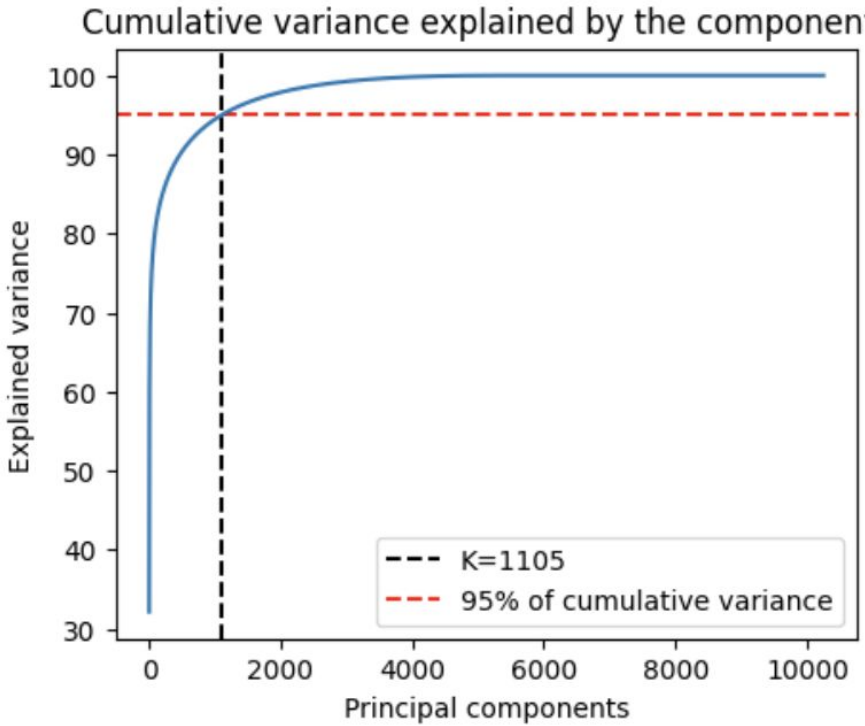
Original image (96,615 colors)



Quantized image (64 colors, K-Means)

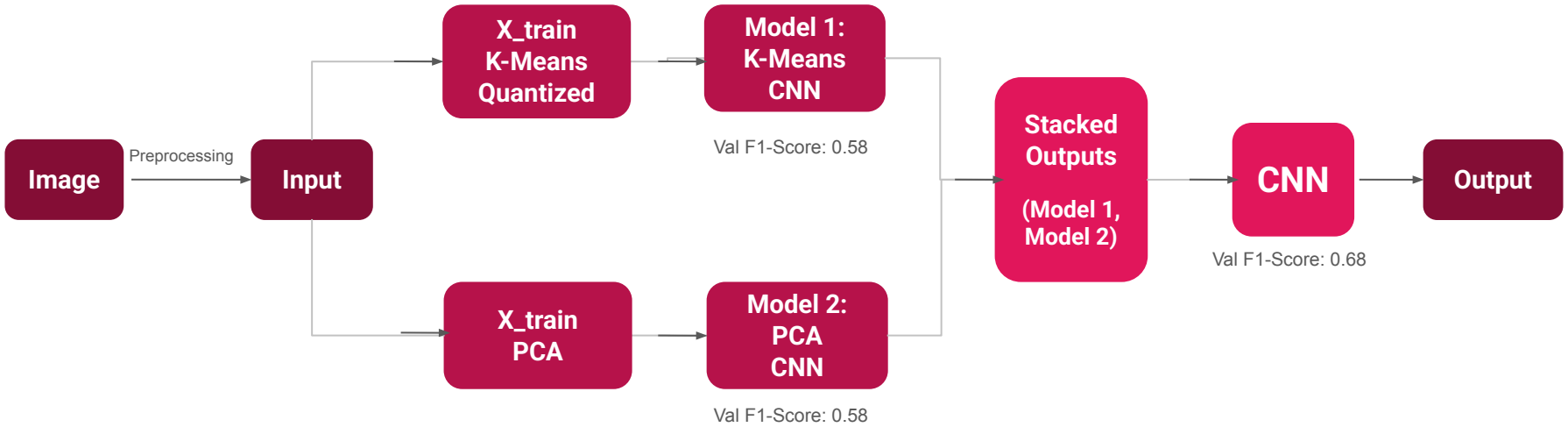


5. Dimensionality Reduction: Principal Component Analysis



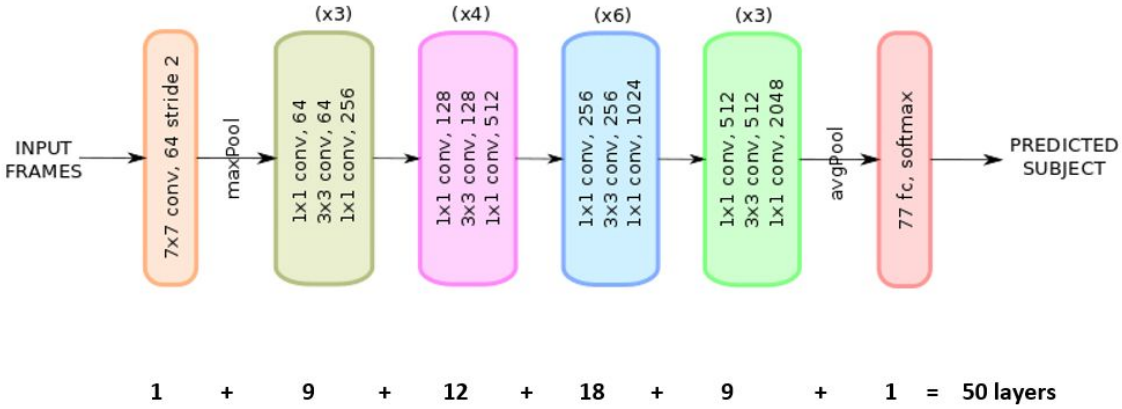
Choosing k = 1200:	X_train.shape
Before PCA:	(10252, 224, 224, 3)
Flatten data for PCA:	(10252, 150528)
After PCA:	(10252,**1200**)
Reshape data for CNN:	(10252, **20, 20, 3**)

6. Ensemble Learning: Stacking Model Outputs



7. Transfer Learning: ResNet-50 Feature Extraction

- Much larger than anything we could field
- 25.6 million parameters, 50 convolutional layers, 1.3 million training set,1000 result categories
- Convert from (224, 224) to (2048,) vector



SAM (Segment Anything Model)

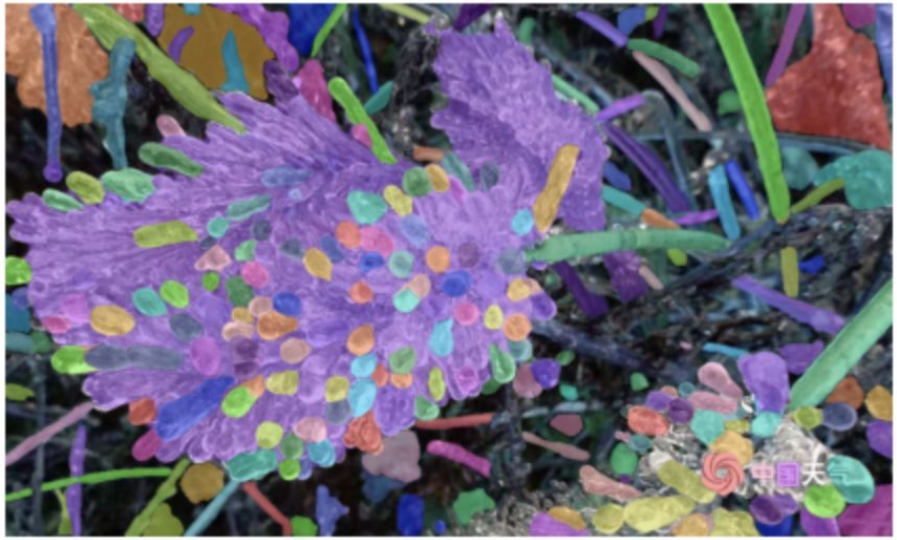
Original Image

Segmented Image

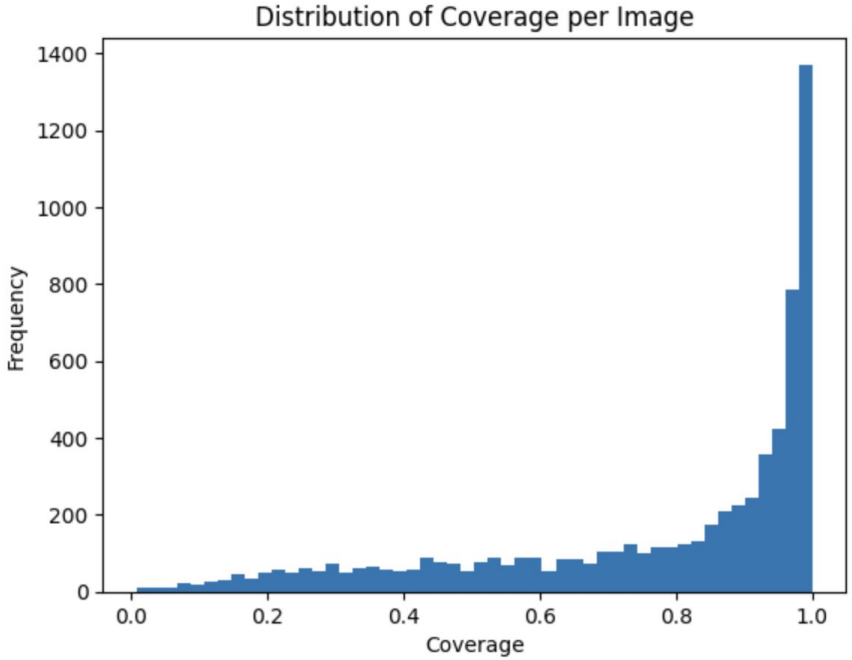
Computational Constraints

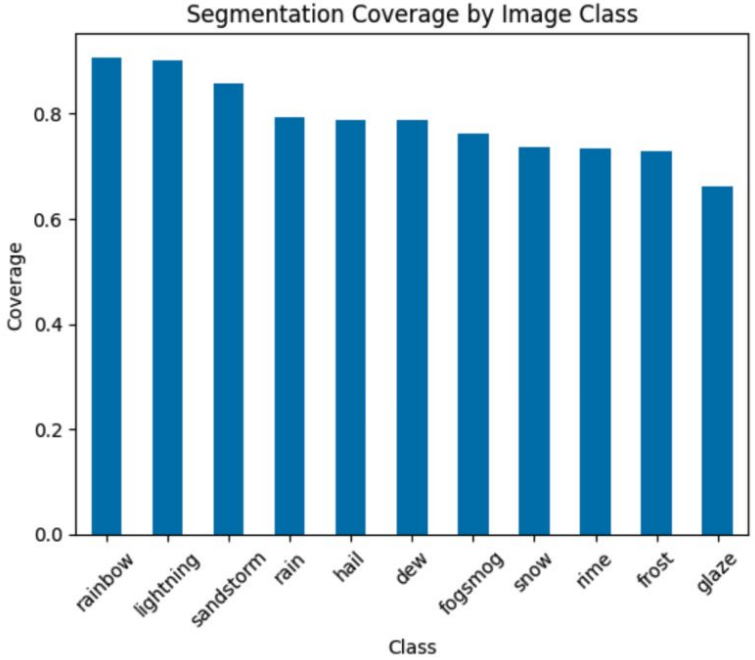
Resized Image: 28% Segmented

Original Image: 65% Segmented



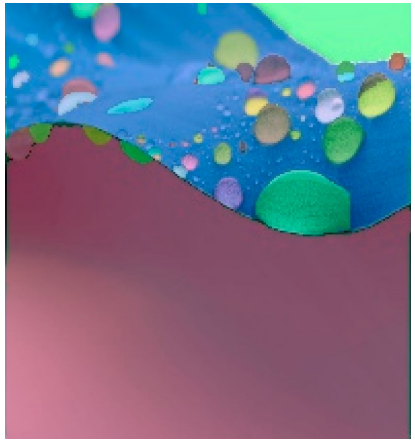
Segmentation Coverage





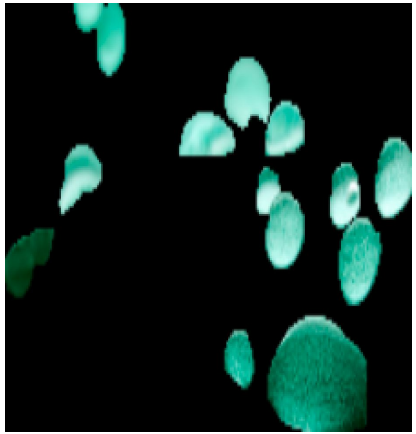
Preprocessing Pipeline

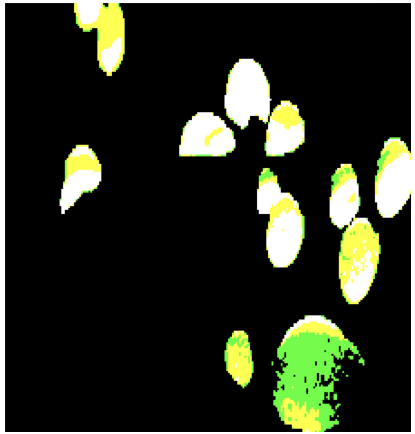
1
Resized
Image



2
Segmented
Image

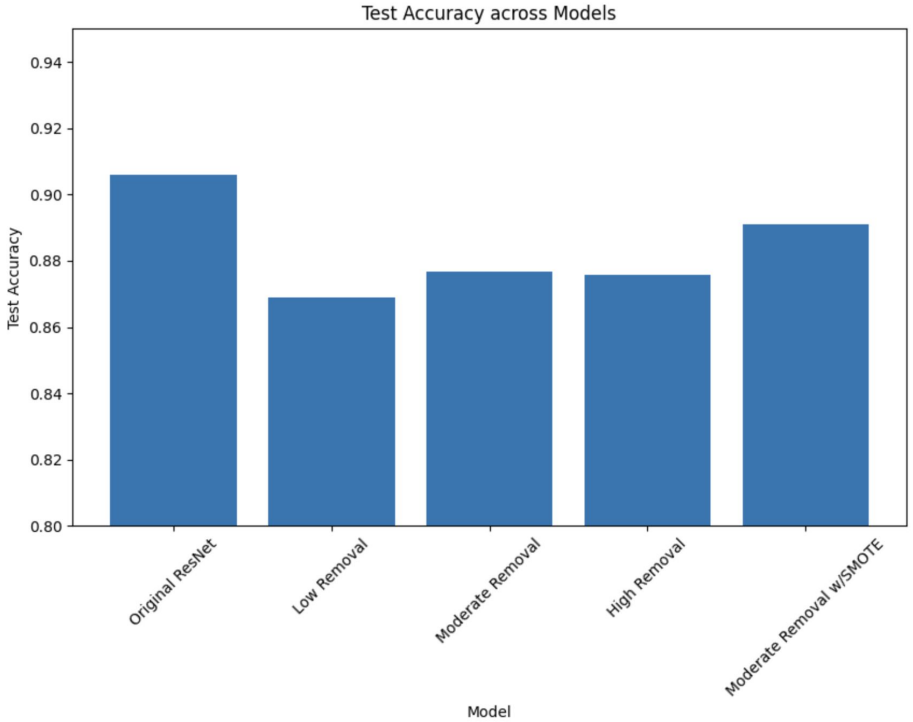
3
De-Noised
Image

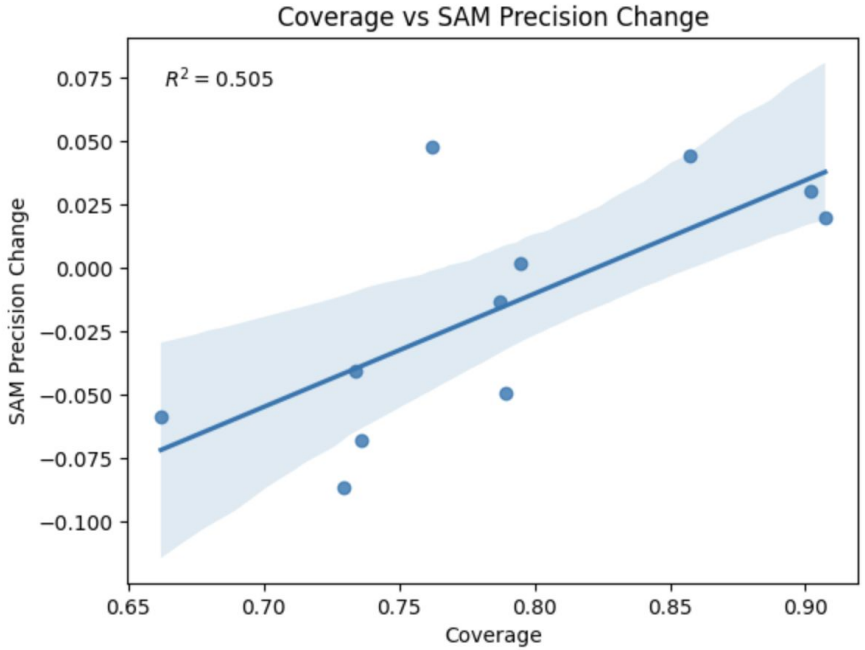




4
ResNet
Normalized
Input

Evaluation


Test Accuracy across Models
Test Accuracy
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
Original ResNet
Low Removal
Moderate Removal
High Removal
Moderate Removal w/SMOTE
Model


Coverage vs SAM Precision Change
$R^2 = 0.505$
SAM Precision Change
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
0.65
0.70
0.75
0.80
0.85
0.90
Coverage

Modeling

Model	Val Loss	Val Accuracy	***Val F1 Score**	Val AUC
1. Baseline Log Reg	1.50	0.51	**0.46**	0.88
2. Log Reg w/ Oversampling	1.48	0.56	**0.52**	0.88
3. K-Means → CNN	3.85	0.60	**0.58**	0.93
4. PCA → CNN	1.49	0.59	**0.58**	0.93
5. Stacked 5+6 → Simple Log Reg	1.00	0.67	**0.66**	0.82
6. Stacked 5+6 → CNN Tuned	1.01	0.69	**0.68**	0.94
7. CNN	1.02	0.71	**0.73**	0.96
***8. CNN Tuned**	**0.8**	**0.82**	**0.83**	**0.98**
***9. Resnet50 + Simple Log Reg**	**0.45**	**0.91**	**0.91**	**0.99**
***10. Resnet50 + CNN Tuned**	**0.41**	**0.90**	**0.90**	**0.99**
***11. SAM + Resnet50 + Log Reg**	**0.44**	**0.89**	**0.90**	**0.99**

Final Model Evaluation Results

Test loss: .44

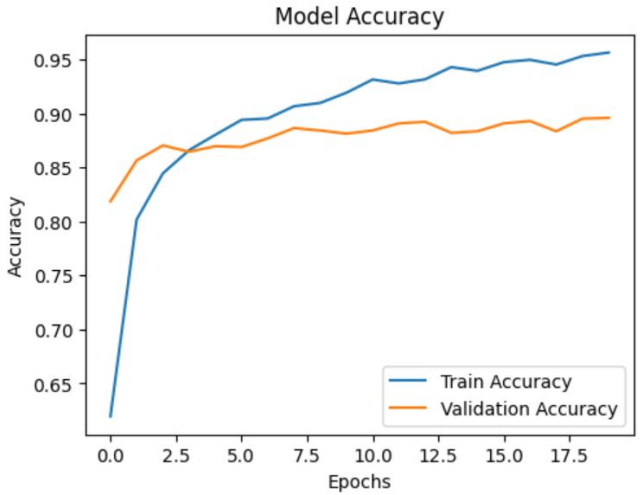
Test accuracy: .90

Test f1: .90

Test auc: .99

Classification Report:

	precision	recall	f1-score	support
dew	1.00	0.95	0.97	129
fogsmog	0.90	0.96	0.93	182
frost	0.89	0.73	0.80	100
glaze	0.77	0.88	0.82	144
hail	0.95	0.97	0.96	129
lightning	0.98	0.97	0.98	63
rain	0.82	0.93	0.87	95
rainbow	0.96	0.98	0.97	47
rime	0.87	0.87	0.87	217
sandstorm	0.95	0.90	0.92	136
snow	0.88	0.79	0.83	130
accuracy			0.90	1372
macro avg	0.91	0.90	0.90	1372
weighted avg	0.90	0.90	0.90	1372



Conclusions

Improvement from baseline: 44%

Best model: Resnet50 + Simple Log Reg

Most interpretable best model: SAM + Resnet50 + Log Reg

***9. Resnet50 + Simple Log Reg**	**0.45**	**0.91**	**0.91**	**0.99**
***10. Resnet50 + CNN Tuned**	**0.41**	**0.90**	**0.90**	**0.99**
***11. SAM + Resnet50 + Log Reg**	**0.44**	**0.89**	**0.90**	**0.99**


Thank You!
Any Questions?

Git/Contributions

Git Link: https://github.com/djcosta2/207_Final_Project

Aaron
- Class Distribution Balancing: Oversampling
- Data Augmentation
- CNN Hyperparameter Tuning: Keras Hyperband

Bella
- Baseline Logistic Regression, & Logistic Regression with Oversampling
- Dimensionality Reduction: K-Means Color Quantization, Principal Component Analysis with CNN
- Ensemble Learning: Stack Model Predictions from K-Means and PCA for Simple Log Reg, then CNN

Daniel
- Image Segmentation Analysis
- SAM (Segment Anything Model) with ResNet
- Class balancing with SMOTE

Theo
- Data Loading
- CNN
- Transfer Learning: ResNet-50 Feature Extraction with CNN