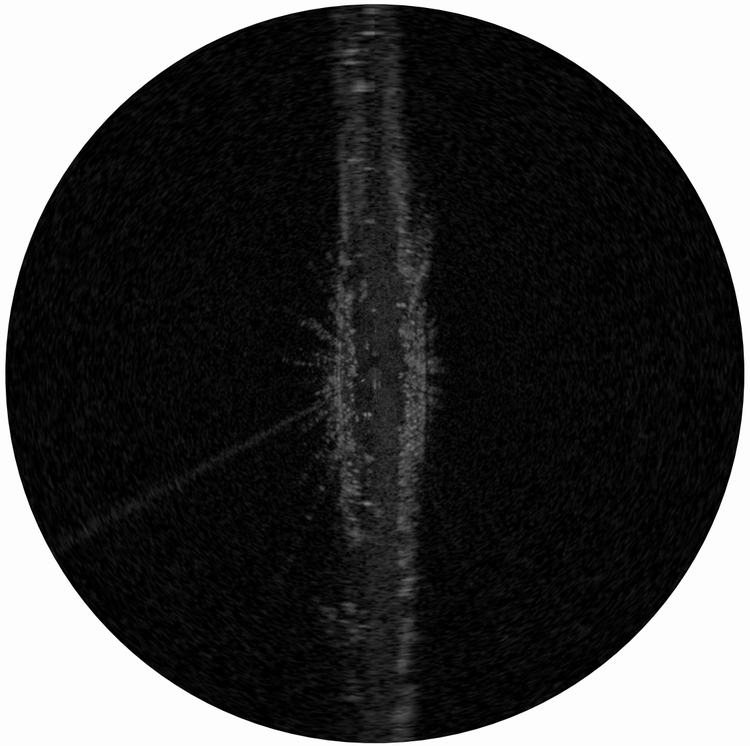
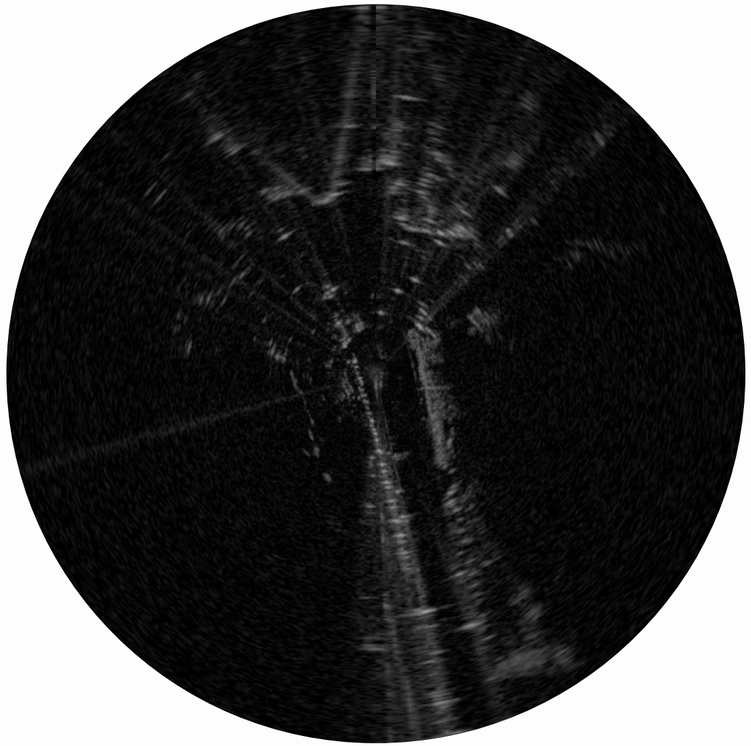
Waether Classifier besed on Automotive Radar Data (POC)



Rain



Normal

Raghad sayed

Brief Description

- This presentation will showcase the complete pipeline of our weather classification project using the RADIATE dataset and YOLOv8n-cls.

- I will walk you through the process, starting from dataset acquisition, the exploration of potential models, and the transfer learning of YOLOv8n-cls to classify weather conditions (normal, rain, fog, and snow).

- Finally, I will present the results achieved through this approach, offering insights into the model's performance.

Project Pipeline Steps

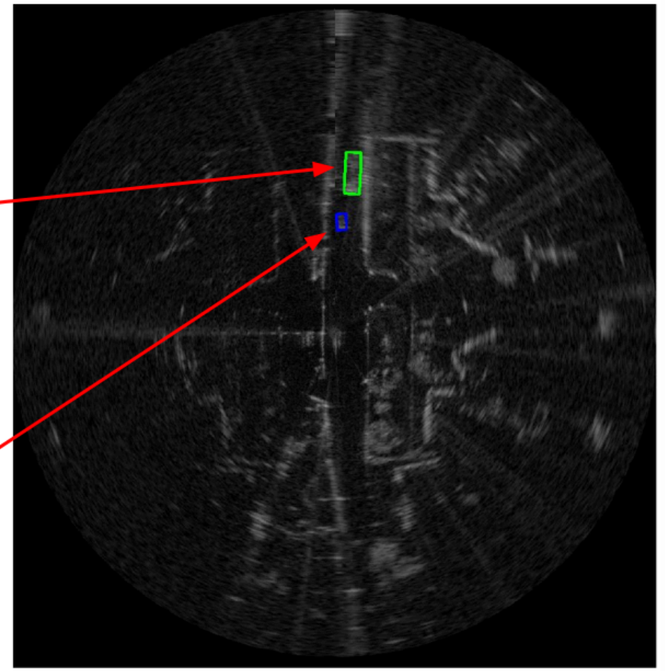
1. Search for dataset K-radar, Radiate
2. Request Radiate dataset
3. Downolad the Dependicies
4. Perpare the dataset(Train, VAL,Test)
5. Search fot best classification models
6. Adjust parametrs of the yolov8 model
7. Transfer Learning
8. Evaluate the model and make some predictions

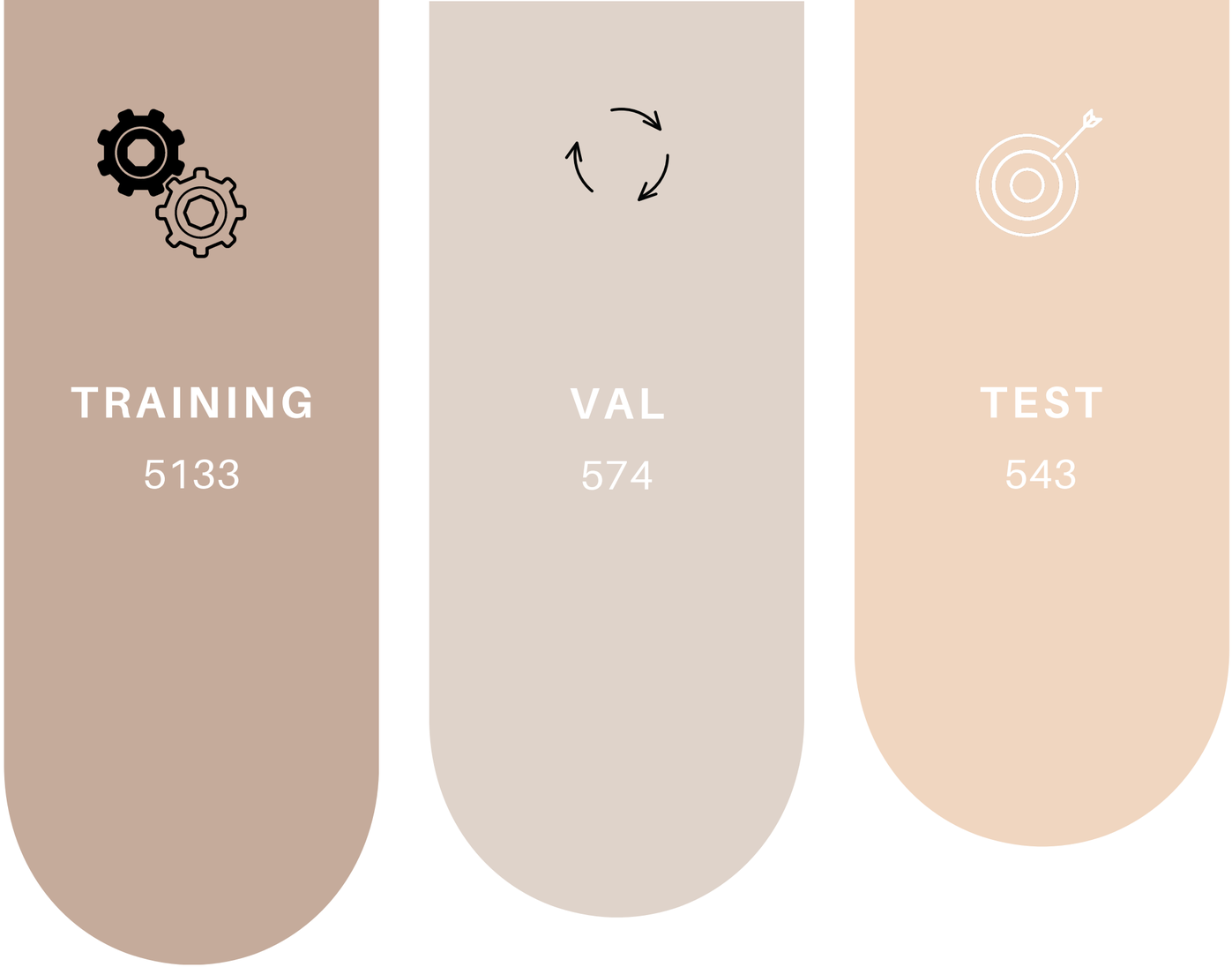
RADIATE Dataset

- RADIATE includes 3 hours of annotated radar images with more than 200K labelled road actors in total
- Adopts the Navtech CTS350-X [24] radar.It is a scanning radar which provides 360° high-resolution range-azimuth images.
- It has 100 meters maximum operating range with 0.175m range resolution, 1.8° azimuth resolution and 1.8° elevation resolution.
- Navtech_Cartesian date provide .png with resolution 1152 x 1152, The each pixel represents a 0.17361 m x 0.17361 m.
- It does not provide Doppler information.
- It covers 8 different categories of actors in a variety of weather conditions (e.g., sun, night, rain, fog and snow) and driving scenarios (e.g.,parked, urban, motorway and suburban)
- RADIATE dataset Link

Radar_cartesian Image and Annotation


```
[
     {"id": 1,
      "class_name": "bus",
      "bboxes": [{ ... },
          {
            "position": [594.8, 269.2, 26.6, 73.1],
            "rotation": 177.7},
          { ... }
      ]},
     {"id": 2,
      "class_name": "car",
      "bboxes": [{ ... },
          {
            "position": [578.9, 364.0, 17.1, 28.7],
            "rotation": 181.1},
          { ... }
      ]},
{ ... }]
```



Training

normal:1296
fog:1929
rain:1287
snow:621

DATASET SPLITTING

Yolov8-cls

YOLOv8 is a versatile deep learning model primarily known for object detection but also highly effective for classification tasks.
In classification mode, YOLOv8 processes input images and assigns them to predefined categories by learning features through convolutional layers.
Its architecture allows for fast and accurate predictions, making it ideal for real-time applications.
YOLOv8's ability to be fine-tuned on custom datasets makes it an excellent choice for image classification problems,
enabling high performance across a variety of tasks, such as identifying weather conditions or other visual patterns.

Model ARCH

- **Backbone Network:** YOLOv8 uses a backbone network (typically CSPDarknet) to extract features from input images.
- **Neck:** YOLOv8 employs a feature pyramid network (FPN) or path aggregation network (PANet) to combine features from different layers.
- **Head:** For classification tasks, the head outputs class probabilities based on the extracted features. In YOLOv8, the head is optimized for real-time performance and accurate predictions.
- In YOLOv8, the backbone consists of convolutional layers, C2f layers, and an SPPF layer
- YOLOv8 Link

Model	Input Size (pixels)	Top-1 Accuracy (%)	Top-5 Accuracy (%)	Speed (CPU ONNX) (ms)	Speed (A100 TensorRT) (ms)	Parameters (M)	FLOPs (B) @ 640px
YOLOv8n-cls	224	69.0	88.3	12.9	0.31	2.7	4.3

HP-Parameters

- **Model weights**
  tmodel=yolov8n-cls.pt
- **Num Of epochs**
  epochs=10
- **The batch size**
  batch=16
- **Img size**
  imgsz=224
- **pretrained**
  pretrained=True
- **The learning rate**
  lr0=0.01
- **Augmentation**
  auto_augment=randaugment

RandAugment: Automatically applies a random set of augmentation transformations, such as rotations, flips, and color distortions.

Model config

- **Data Size & Paths**
  train: /content/drive/MyDrive/radiate/Dataset_2/train... found 5133 images in 4 classes ✅

  val: /content/drive/MyDrive/radiate/Dataset_2/val... found 574 images in 4 classes ✅
- **Dependicies**
  Ultralytics YOLOv8.2.87 🚀 Python-3.10.12 torch-2.4.0+cu121 CPU
  !pip install ultralytics

Input Image Strucuture and Format

Input image is a radar image in cartesian coordinates, gray scale image with one channel and dimensions of 1152x1152

1. Polar Image Overview:

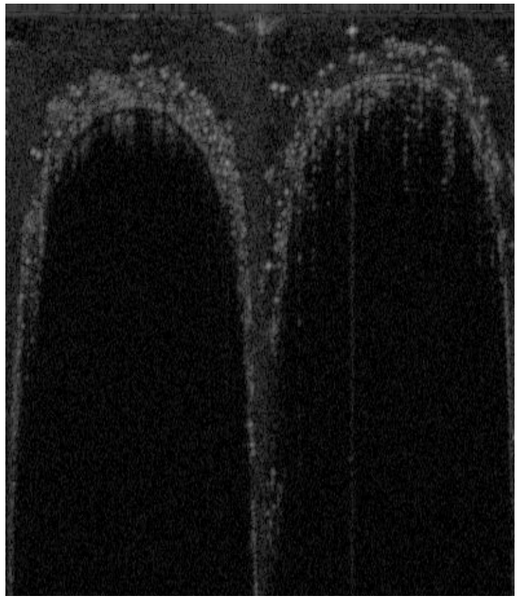
- **Resolution:** 400 (Azimuth) x 576 (Range)
  - **Azimuth Axis:**
    - Index 0: Represents angle 0°.
    - Index 399: Represents angle 360°.
  - **Range Axis:**
    - Index 0: Represents 0 meters.
    - Index 575: Represents 100 meters.
- **Raw Data:**
  - Data captured using Fast Fourier Transform (FFT).
  - Power values are converted to decibels (dB) and quantized between 0 to 255. Pixel value represents the received power (depends on object material and shape).

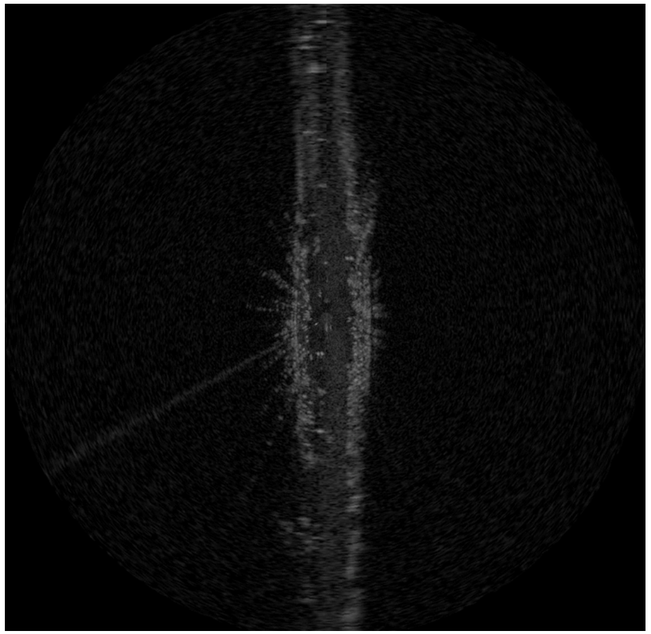
2. Conversion to Cartesian Image:

- **Input to the Model:** The Cartesian image is used as the input for the model.
- **Conversion Process:**
  - Polar image is converted to Cartesian coordinates by projecting each point onto an (x, y) plane.
  - Bilinear interpolation is applied to fill any gaps, ensuring smooth pixel transitions. Cartesian Image Resolution: 1152 x 1152

Key Points:

- No elevation data is considered in the image (2D format only).
- Power values are scaled between 0-255 (normalized).





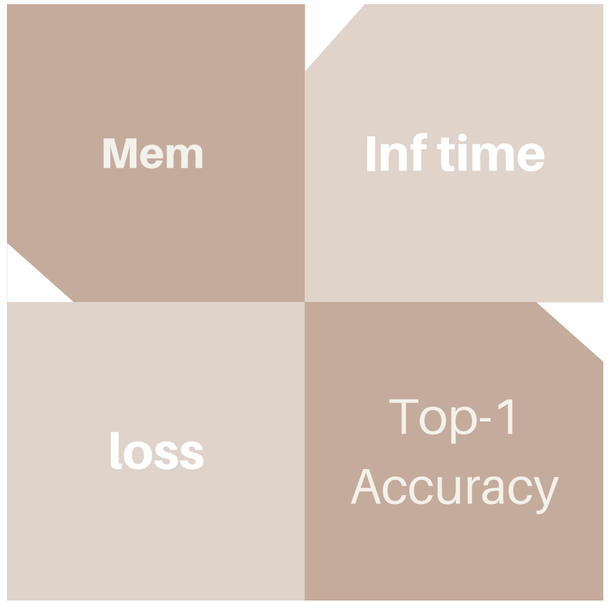
Polar Image (400 x 576) → Converted to Cartesian Image (1152 x 1152)

You might wonder how a grayscale image with a single channel and dimensions of 1152x1152 can be processed by a model that expects a 3-channel RGB input and a size of 224x224.

The answer lies in the preprocessing steps that happen before the image is fed into the model. For grayscale images, the preprocessing converts the single channel into 3 channels by duplicating the grayscale data across the Red, Green, and Blue channels. Then, the image is resized to the model's expected input size, which for YOLOv8n-cls is typically 224x224.

Evaluation Measurments

0.268GB = 405 MB



0.389 MS INFERENCE

0.0004

1

On Validation Data

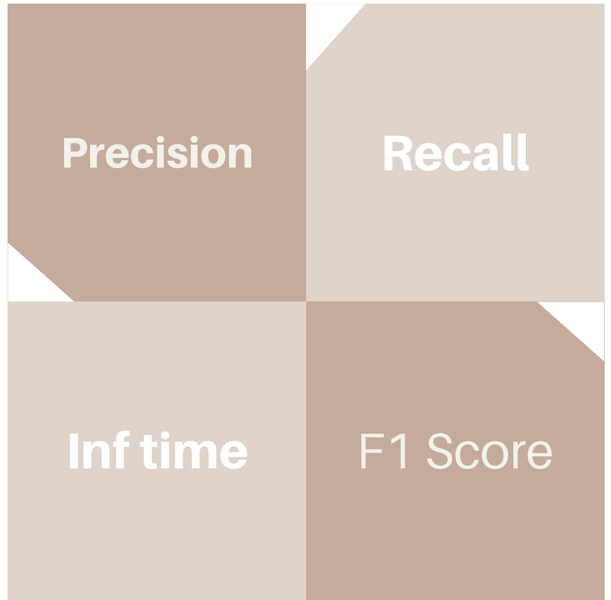
Top-1 Accuracy
The percentage of times the model's top prediction (the class with the highest confidence score) matches the true label.

Precision
Precision measures how many of the predicted positive instances are actually

Recall
Recall measures how many of the actual positive instances were correctly predicted by the model.

On Test Data

0.99



0.89

AROUND 3MS

0.92

10 epochs completed in 1.616 hours

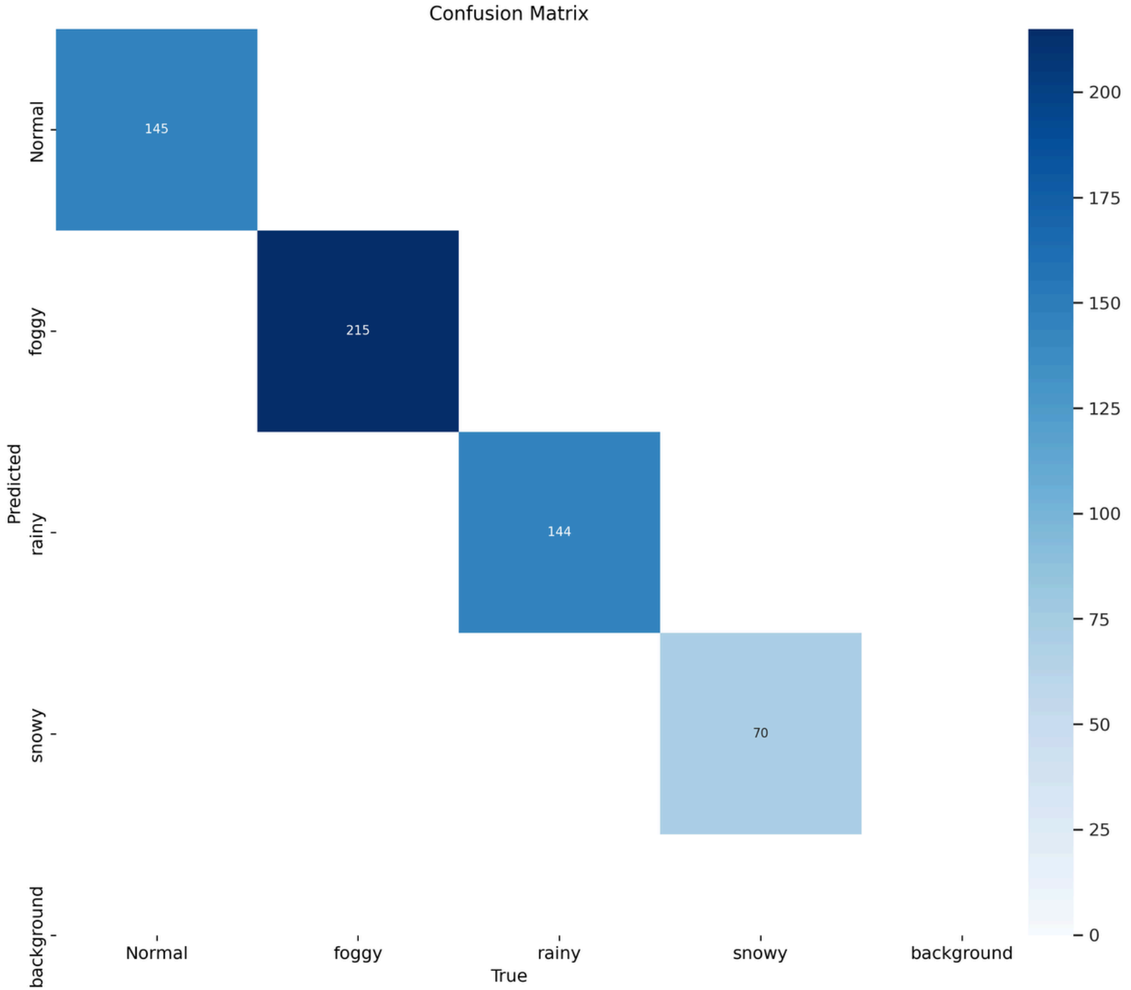
The reason some classes have a lighter color than others in the confusion matrix is that the number of images for those classes in the validation dataset is smaller.

Normal:145

foggy:215

rainy:144

snowy:70



```
train: /content/drive/MyDrive/radiate/Dataset_v0.1/train... found 5133 images in 4 classes ✅
val: /content/drive/MyDrive/radiate/Dataset_v0.1/val... found 574 images in 4 classes ✅
test: /content/drive/MyDrive/radiate/Dataset_v0.1/test... found 543 images in 4 classes ✅
val: Scanning /content/drive/MyDrive/radiate/Dataset_v0.1/val... 574 images, 0 corrupt: 100%|██████████| 5
               classes   top1_acc   top5_acc: 100%|██████████| 36/36 [00:21<00:00,  1.68it/s]
                   all          1          1
Speed: 0.3ms preprocess, 0.6ms inference, 0.0ms loss, 0.0ms postprocess per image
Results saved to runs/classify/val2
Top-1 Accuracy: 1.0
Top-5 Accuracy: 1.0
Fitness Score (equivalent to top-5 accuracy): 1.0
Speed metrics (time taken for each step in ms): {'preprocess': 0.2649823132292319, 'inference': 0.59868566
Full results dictionary: {'metrics/accuracy_top1': 1.0, 'metrics/accuracy_top5': 1.0, 'fitness': 1.0}
```

Eval on Val data

```
image 1/1 /content/drive/MyDrive/radiate/Dataset_v0.1/test/foggy/000434.png: 224x224 foggy 1.00, Normal
Speed: 9.6ms preprocess, 2.9ms inference, 0.0ms postprocess per image at shape (1, 3, 224, 224)
Processed 000434.png: Predicted class - foggy with confidence - 1.00
Precision: 0.99
Recall: 0.89
F1 Score: 0.92
```

Prediction on Test data

Some Important Links

- RADIATE Dataset Repo
- YOLOV8 Repo
- My Google Colab Notebook for preparing dataset, fine tuning the model, evaluation and prediction
- YOLOv8 Architecture
- Radar Papers Summary, Presentation Assets

My Contact Info:

- **E**mail: raghadsayed770@gmail.com
- LinkedIn URL
- Phone: 01014665408

Finally, I am truly grateful to the DEBI scholarship for their generous support and to SEITech for providing me with an invaluable internship experience, both of which have significantly contributed to my growth and learning in the field.