Geeks for Geeks: EcoTech DS Hackathon

Project Documentation

Skycam Images Based Cloud Coverage Prediction using CV & ML

Team: Infyzero

Team Leader: Siddharth Kulkarni

Domain: Climatic Patterns | Energy

Tech Domain: Computer Vision | Machine Learning

Index

1. Problem Statement & Objectives

1.1 Problem Statement

To find the cloud coverage from a Skycam image. Skycam is an automated camera system to periodically record images of the entire sky from dusk until dawn. Skycam Image is generated from a Skycam device.

Domain Knowledge:

- First Image: Low Cloud Coverage
- Second Image: Moderate Cloud Coverage
- Third Image: High Cloud Coverage.

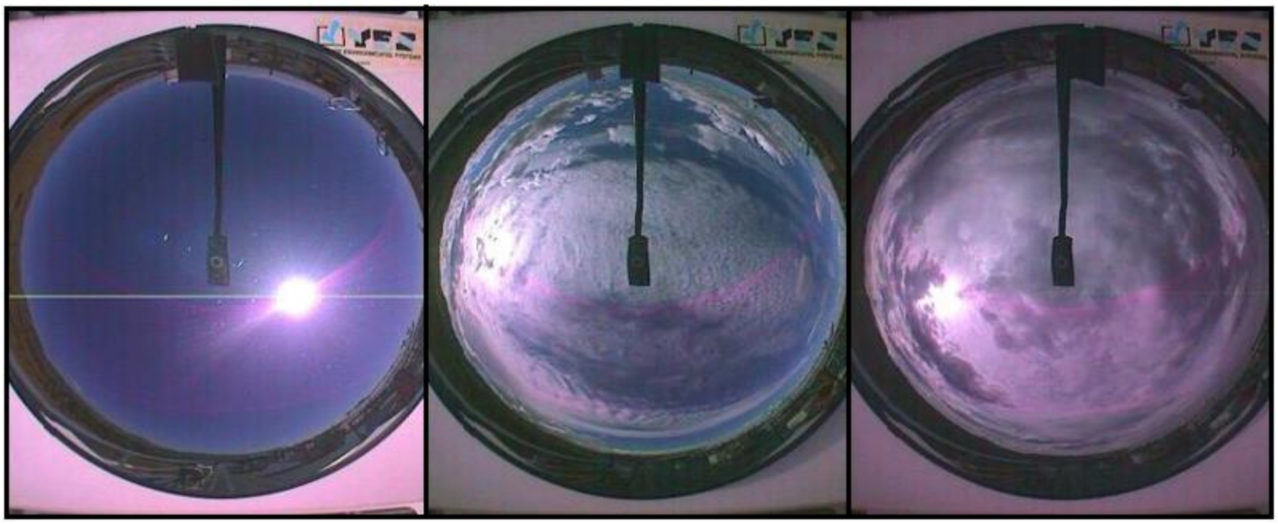


Fig: Sample Skycam Images

1.2 Objectives

- **Cloud Coverage Prediction:** To develop a robust model that accurately calculates cloud coverage from skycam images. This model aims to analyze the cloud formations in the provided images and provide a percentage indicating the extent of cloud coverage.

- **Automation:** Automate the process of cloud coverage assessment using sky images. This will reduce the need for manual monitoring and provide real-time information on the cloud conditions.

2. Software Requirements

Language Used: Python

Coding Platform: Kaggle

To Train CLIP Model: GPU T4

AI/ML Libraries:

- Data Preprocessing: numpy, pandas
- Data Modeling: catboost, transformers, timm, torch, CLIP
- Image Reading & Resizing: open-cv
- Model Saving: pickle

Deployment Libraries: Gradio

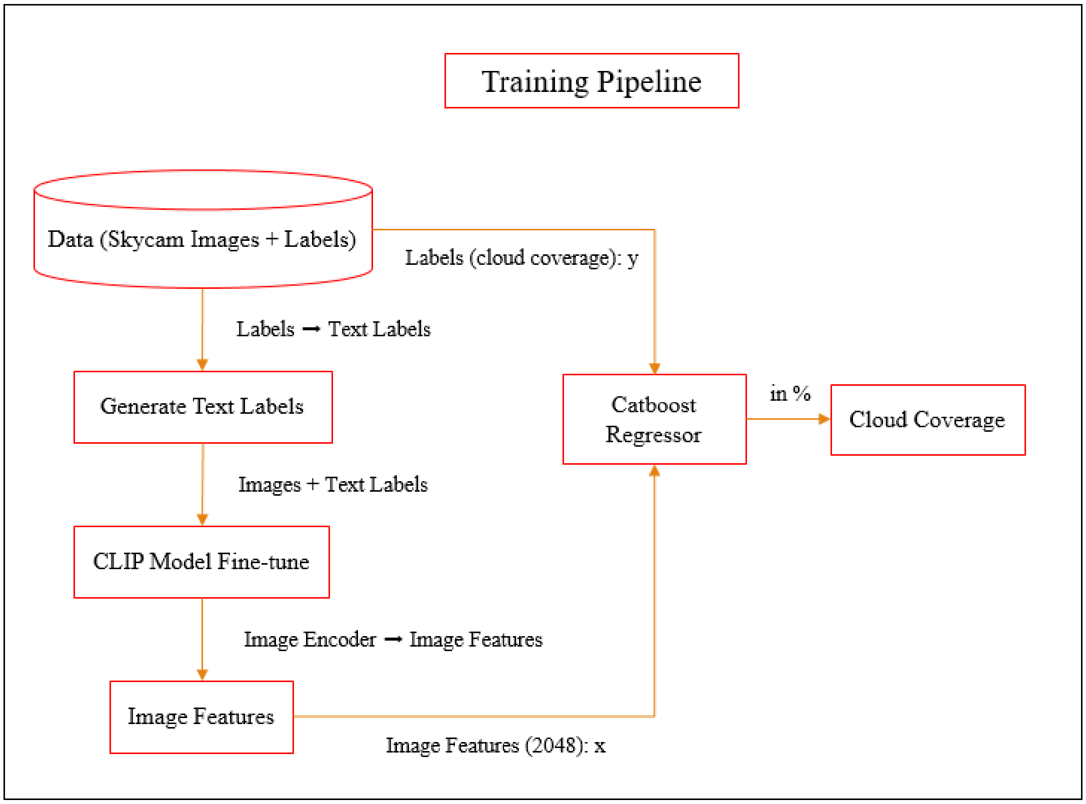
Deployment Platform: Hugging Face

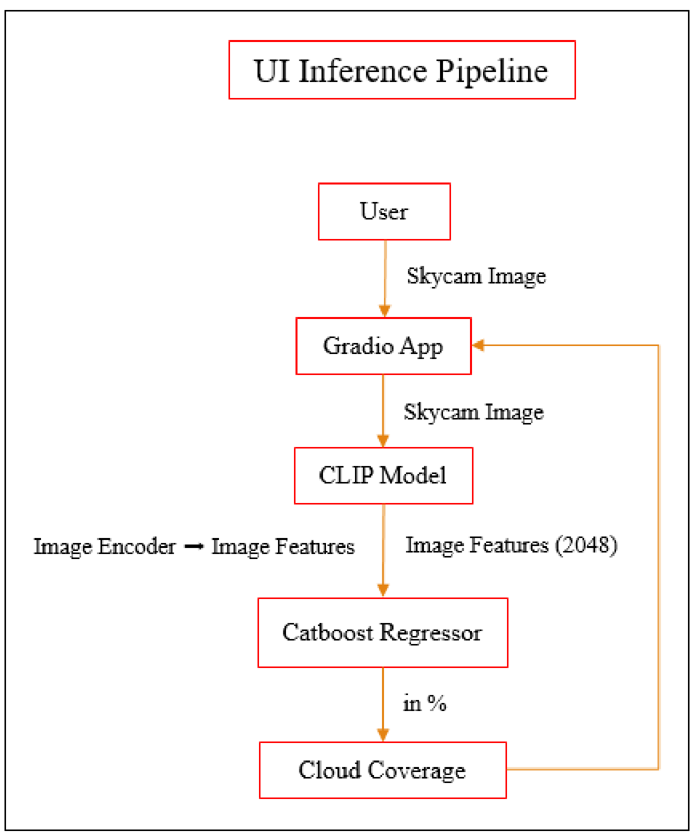
Requirements.txt:

- Gradio
- Timm
- opencv-python
- Catboost
- Transformers
- Torch
- git+https://github.com/openai/CLIP.git

3. Methodology

3.1 System Architecture





3.2 Model Architecture

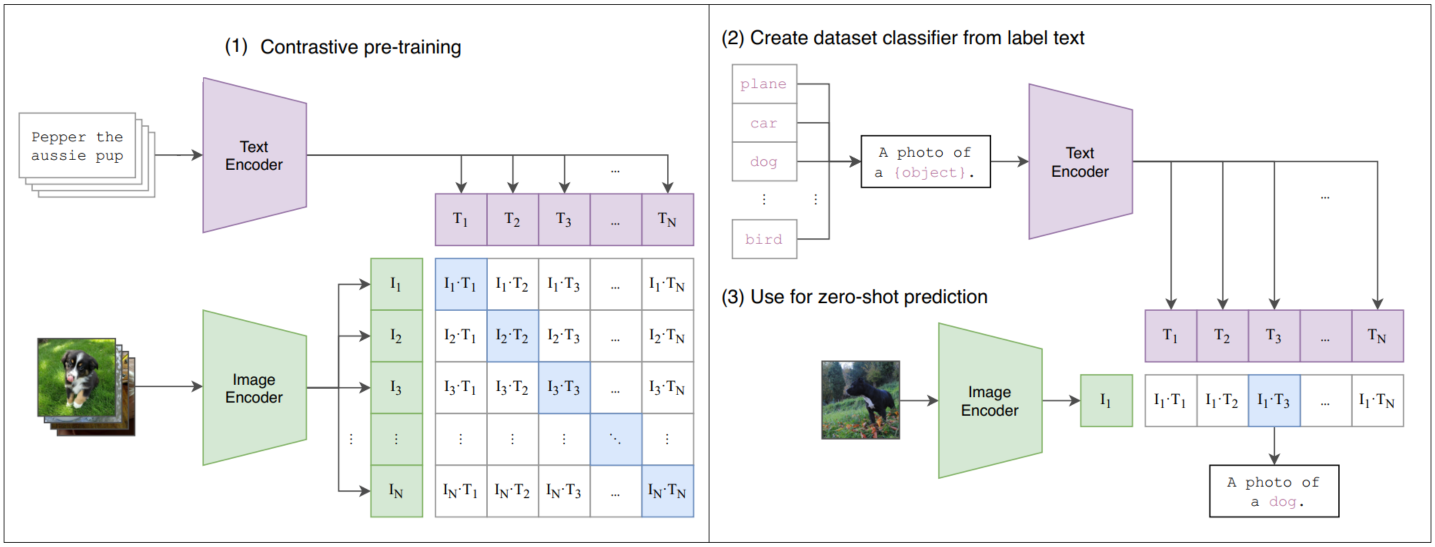


Fig: CLIP Model Architecture

Clip Model Working:

- Contrastive Language Image Pretraining (CLIP) Model is used for image text similarity problems.
- We aim for CLIP model just to extract features out of skycam images.
- To train CLIP Model: Resnet50 Image Encoder is used, Distilled Bert Text Encoder is used.
- Image Encoder: Converts Image to Features in vector format.
- Text Encoder: Converts Text to Features in vector format.
- Projection Head: Transforms image vector & text vector to fixed size of 2048 and apply dot product.
- Calculates Similarity Score between input image, image feature, and text feature.

3.3 Model Training & Evaluation

- CLIP : Total Epochs : 12
- CLIP : Image Encoder : Resnet50
- CLIP : Text Encoder : DistilledBert
- Catboost Model : Iterations : 700
- Catboost Model : Learning Rate : 0.1
- Catboost Model : Max Depth : 8
- Catboost Model : Eval Metrics : RMSE
- Catboost Model TEST Metrics : RMSE : 10.19 | R2 : 0.88
- Catboost Model Output:
    - Low Cloud Cover: 0% - 33%
    - Moderate Cloud Cover: 33% - 66%
    - High Cloud Cover: 66% - 100%

```
# Pretrained Model Usage
model = CLIPModel().to(CFG.device)
params = [
{"params": model.image_encoder.parameters(), "lr": CFG.image_encoder_lr},
{"params": model.text_encoder.parameters(), "lr": CFG.text_encoder_lr},
{"params": itertools.chain(
    model.image_projection.parameters(), model.text_projection.parameters()
), "lr": CFG.head_lr, "weight_decay": CFG.weight_decay}
]
optimizer = torch.optim.AdamW(params, weight_decay=0.)
lr_scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
optimizer, mode="min", patience=CFG.patience, factor=CFG.factor
)
step = "epoch"
```

```
Downloading model.safetensors:   0%|          | 0.00/102M [00:00<?, ?B/s]

Downloading model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]
```

```
# Model Training
best_loss = float('inf')
for epoch in range(CFG.epochs):
    print(f"Epoch: {epoch + 1}")
    model.train()
    train_loss = train_epoch(model, train_loader, optimizer, lr_scheduler, step)
    model.eval()
    with torch.no_grad():
        valid_loss = valid_epoch(model, valid_loader)

    if valid_loss.avg < best_loss:
        best_loss = valid_loss.avg
        torch.save(model.state_dict(), "best.pt")
        print("Saved Best Model!")

    lr_scheduler.step(valid_loss.avg)
```

Fig: CLIP Model Fine-tuning Code: Total Epochs: 12

Catboost Model

```
CB_model = CatBoostRegressor(iterations = 700, learning_rate = 0.1, max_depth = 8, eval_metric = 'RMSE', random_seed = 48)

CB_model.fit(train_features.cpu().numpy(), train_labels.cpu().numpy(),
             eval_set = (valid_features.cpu().numpy(), valid_labels.cpu().numpy()),
             use_best_model = True, plot = True, verbose = 50)
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))

0:      learn: 28.1361841       test: 28.2423136        best: 28.2423136 (0)    total: 2.13s    remaining: 24m 49s
50:     learn: 11.5614561       test: 11.9335237        best: 11.9335237 (50)   total: 1m 3s    remaining: 13m 21s
100:    learn: 10.7263689       test: 11.4059249        best: 11.4059249 (100)  total: 2m 1s    remaining: 12m 1s
150:    learn: 10.0566562       test: 11.0617557        best: 11.0617557 (150)  total: 3m       remaining: 10m 55s
200:    learn: 9.5172739        test: 10.8473396        best: 10.8473396 (200)  total: 3m 58s   remaining: 9m 51s
250:    learn: 9.0923719        test: 10.6886373        best: 10.6886373 (250)  total: 4m 55s   remaining: 8m 47s
300:    learn: 8.7042622        test: 10.5734544        best: 10.5734544 (300)  total: 5m 51s   remaining: 7m 45s
350:    learn: 8.3755575        test: 10.4773273        best: 10.4773273 (350)  total: 6m 47s   remaining: 6m 45s
400:    learn: 8.0759744        test: 10.3938604        best: 10.3938604 (400)  total: 7m 44s   remaining: 5m 46s
450:    learn: 7.7814581        test: 10.3233375        best: 10.3233375 (450)  total: 8m 42s   remaining: 4m 48s
500:    learn: 7.5160766        test: 10.2628795        best: 10.2628795 (500)  total: 9m 39s   remaining: 3m 50s
550:    learn: 7.2897423        test: 10.2027638        best: 10.2027638 (550)  total: 10m 35s  remaining: 2m 51s
600:    learn: 7.0611325        test: 10.1574324        best: 10.1574324 (600)  total: 11m 33s  remaining: 1m 54s
650:    learn: 6.8320990        test: 10.1136860        best: 10.1136860 (650)  total: 12m 30s  remaining: 56.5s
699:    learn: 6.6529638        test: 10.0780409        best: 10.0780409 (699)  total: 13m 25s  remaining: 0us

bestTest = 10.07804086
bestIteration = 699
```

Fig: Hyperparameter Tuned Catboost Regressor Model

CatBoost Regressor Metrics			
-	Train Data	Validation Data	Test Data
MAE	4.43	6.3	6.36
RMSE	6.65	10.07	10.19
R2	0.95	0.89	0.88
Total Records	70168	30072	33414

Fig: Catboost Model Metrics

4. Deployment & Screenshots

- Developed a Gradio app which takes a skycam image as input and outputs predicted cloud coverage in percentage.
- Developed app consists Fine-tuned CLIP model and Trained Catboost Model.
- Deployed the app to Hugging Face Spaces.

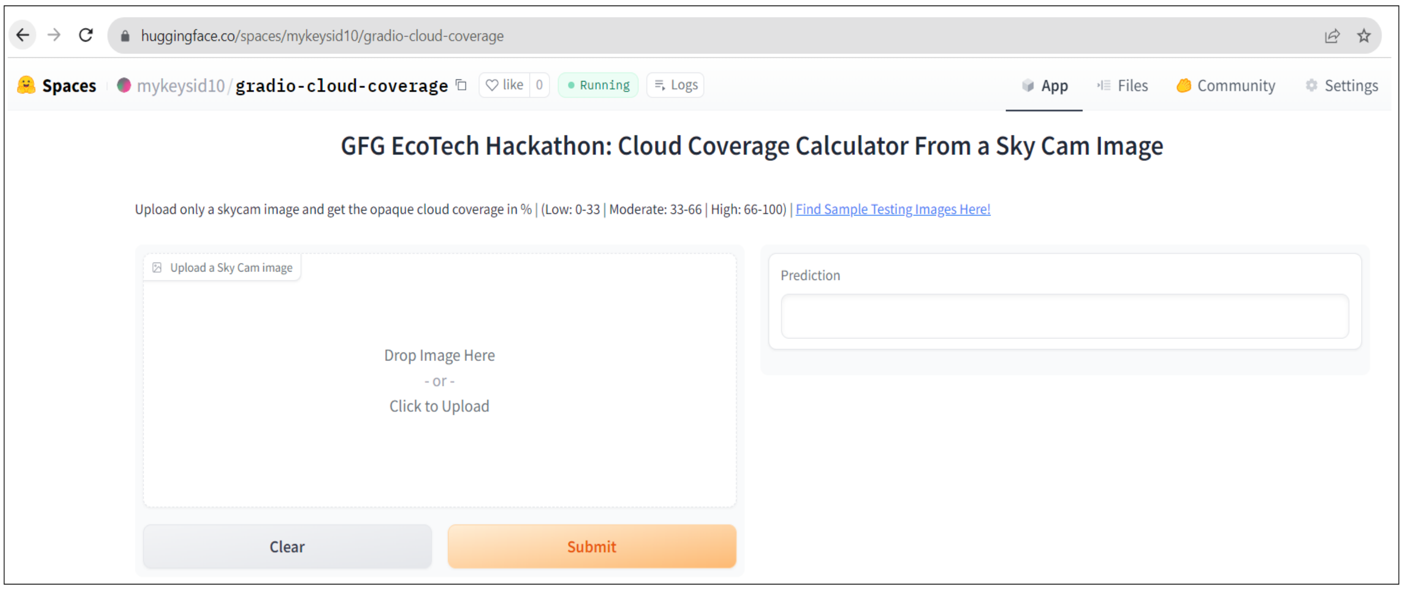


Fig: Hugging Face Deployed UI

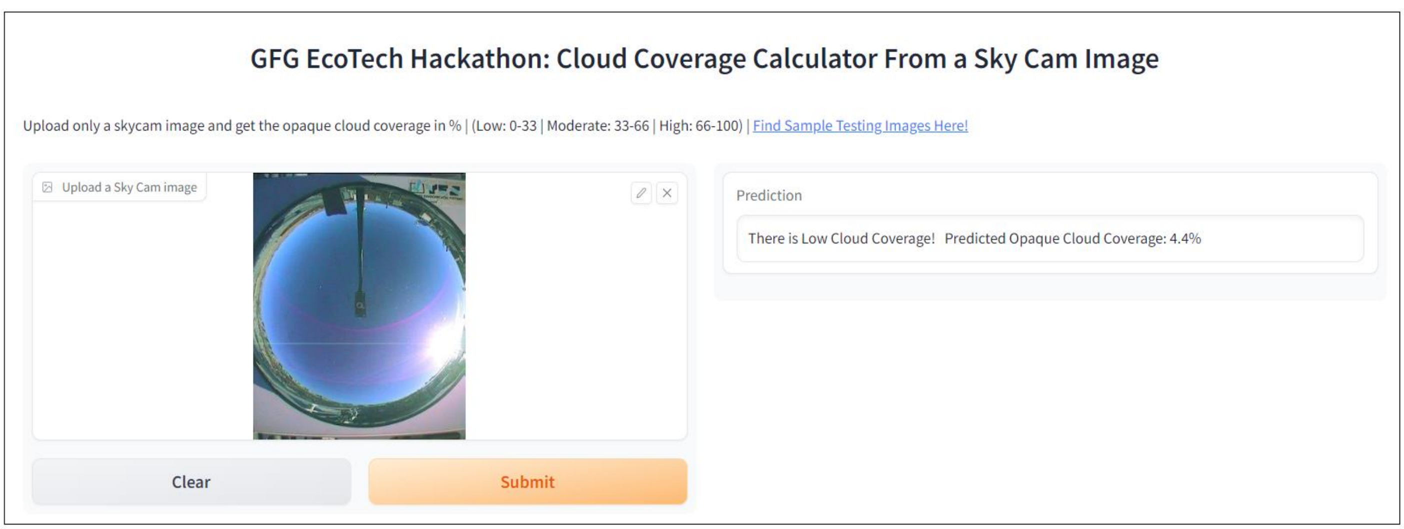


Fig: Low Cloud Coverage Sample

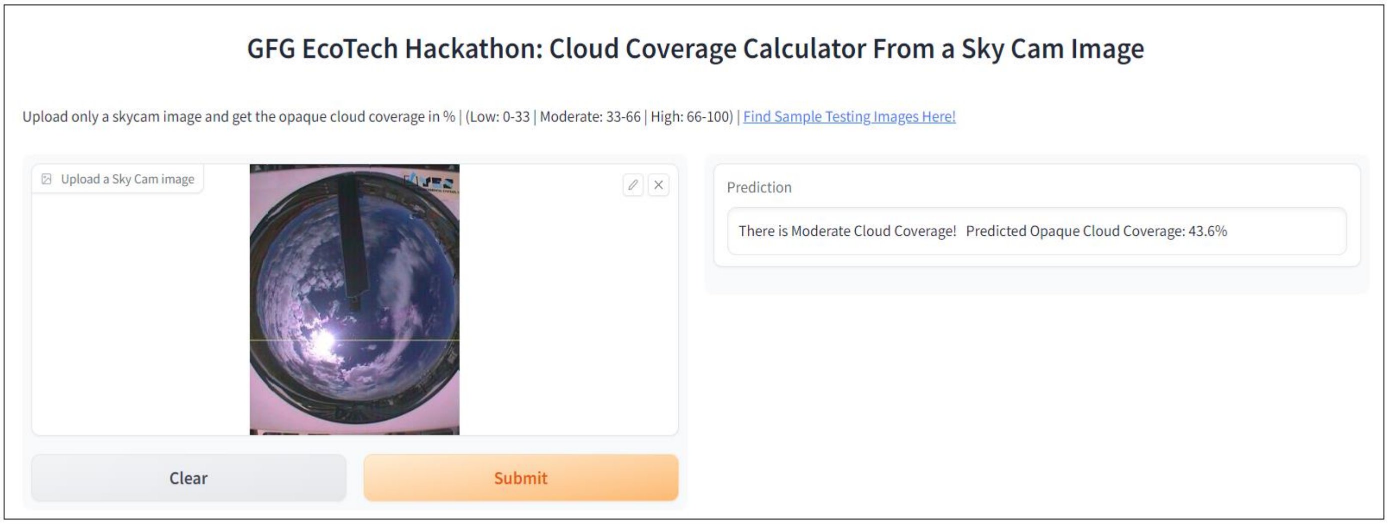


Fig: Moderate Cloud Coverage Sample

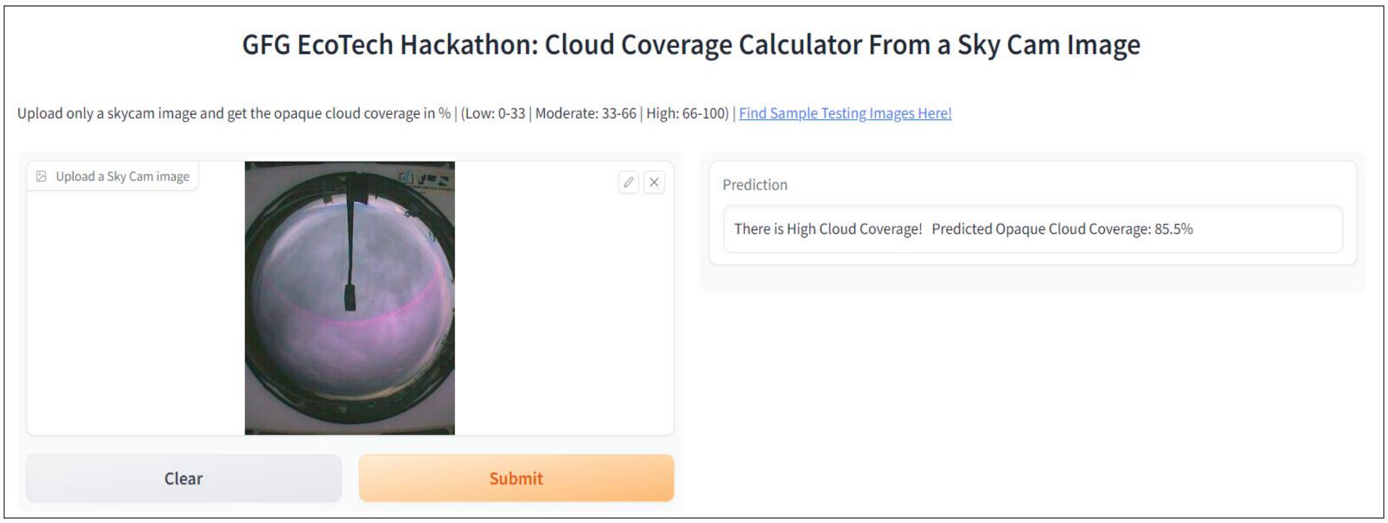


Fig: High Cloud Coverage Sample

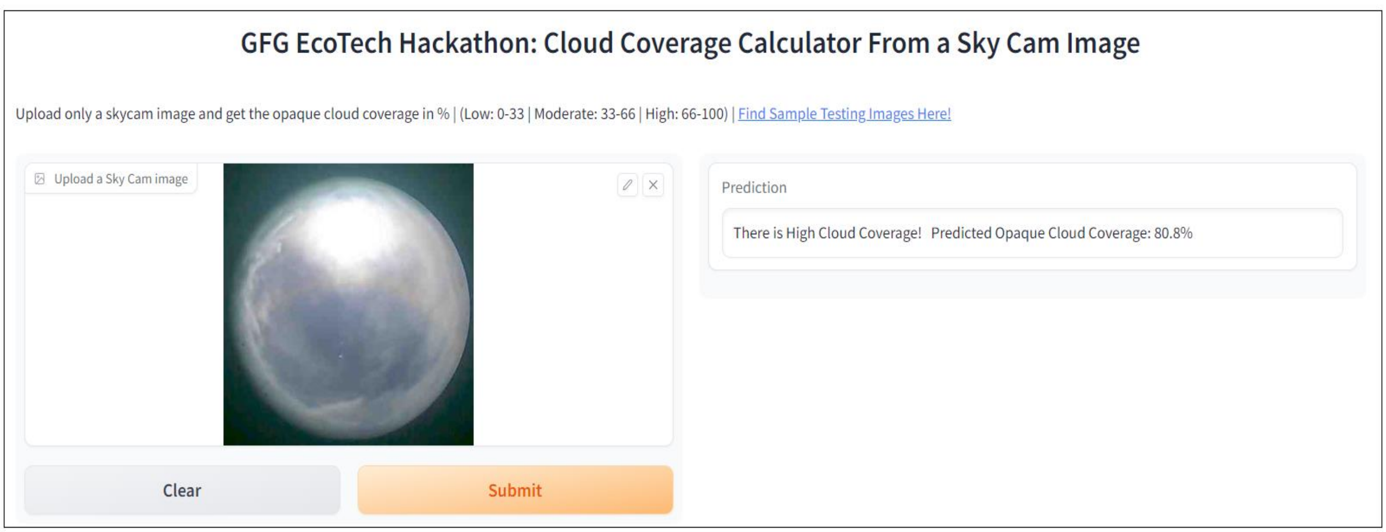


Fig: Other Variants High Cloud Coverage Sample

5. Conclusion

- Successfully build an application which intakes a skycam image and predicts the opaque cloud coverage in percentage with a **RMSE** of **10.19** on Test Data & **R^2** of **0.88**.
- This model aims to analyze the cloud formations in the provided images and provide a percentage indicating the extent of cloud coverage and also outputs Cloud Coverage Level.

6. Future Scope

- Accurate **weather monitoring** is crucial for various applications including agriculture and disaster management. Cloud coverage is a key parameter in weather forecasting and automating its assessment can improve weather predictions.
- Providing **real-time information** on cloud coverage can benefit industries that rely on weather conditions, such as renewable energy generation, outdoor event planning, and transportation.
- The integration of the cloud coverage model with skycam can serve as an **early warning system** for impending storms or heavy rains and climatic drifts. This can help in taking preventive measures and ensuring public safety.

7. Project Links

Github Repo URL: https://github.com/mykeysid10/EcoTech-Data-Science-GfG-Hackathon-Cloud-Coverage-Calculator

Webapp URL: https://huggingface.co/spaces/mykeysid10/gradio-cloud-coverage

Data & Models URL: https://drive.google.com/drive/folders/14Fk5nWNNQT5Dk0J7KVO4VNCgUxxJTG6Y?usp=drive_link

Demo Video Link: https://www.youtube.com/watch?v=b8qGr6CowWs