Cloud Coverage Prediction System

Phase 2: Cloud Coverage Detection

Ayush Kumar Mishra, Akshat Kumar , Aditya Prakash

October 27, 2024

Abstract

[Project Overview]:

This document presents Phase 2 of our comprehensive weather prediction system, focusing on cloud coverage detection using deep learning and computer vision techniques. The system implements a hybrid approach combining CLIP-based feature extraction with CatBoost regression, achieving robust performance in cloud coverage estimation. This phase lays the groundwork for Phase 3, which will implement temperature prediction based on cloud coverage patterns.

Contents

1 Introduction

1.1 Project Context

This project represents Phase 2 of a three-phase implementation:

Phase 1: SOP Submission

Phase 2: **Cloud Coverage Detection** (Current Phase)

Phase 3: Temperature Prediction from Cloud Coverage prediction , we are getting from phase-2 code (Upcoming)

1.2 Problem Statement

Cloud coverage prediction serves as a crucial intermediate step in our temperature prediction system. This phase focuses on developing accurate cloud coverage estimation using ground-based sky cameras and deep learning techniques.

2 Technical Implementation

2.1 Model Architecture

2.1.1 CLIP-Based Feature Extraction

```
class MemoryEfficientImageEncoder(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.model = timm.create_model(
            cfg.model_name,
            pretrained=cfg.pretrained,
            num_classes=0,
            global_pool='avg'
        )
```

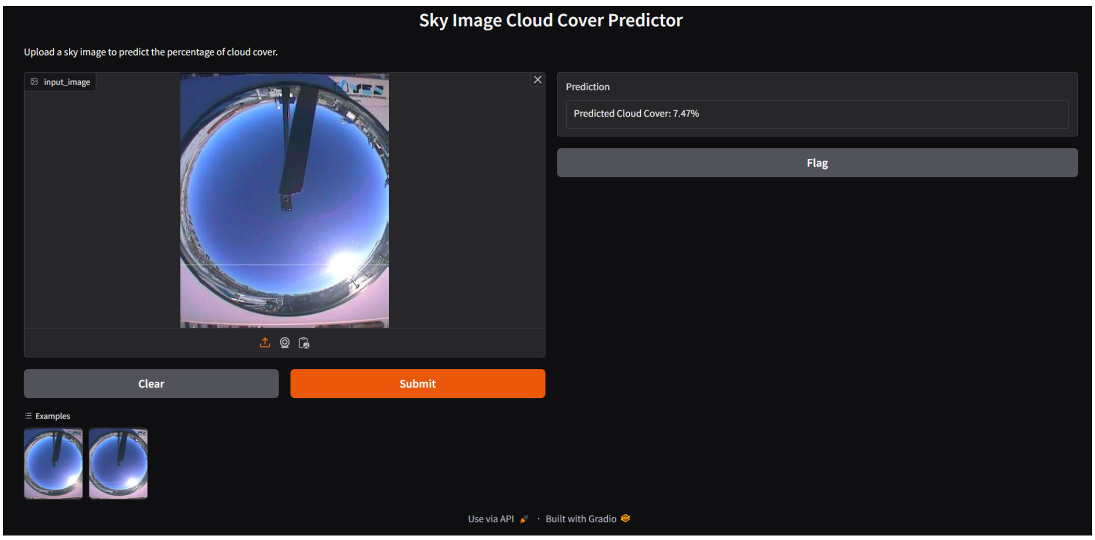

Figure 1: Web Interface for Cloud Coverage Prediction

3 Performance Metrics

Model Performance Results

```
Train Metrics:
    MAE: 3.06
    RMSE: 4.52
    R2: 0.975

Validation Metrics:
    MAE: 5.80
    RMSE: 9.62
    R2: 0.888

Test Metrics:
    MAE: 5.81
    RMSE: 9.69
    R2: 0.886
```

3.1 Project Objectives

- Develop an accurate cloud coverage prediction system
- Implement efficient data preprocessing pipeline
- Create memory-optimized model training framework
- Deploy an accessible web interface for predictions

4 Dataset

4.1 Data Sources

The project utilizes two main data components:

- `cloud_data_cleaned1.csv`: Contains metadata and labels
- Sky camera images dataset: Collection of hemispheric sky images

4.2 Dataset Structure

Column	Type	Description
image_name	string	Unique identifier for each image
label	string	Descriptive cloud coverage label
opaque_clouds	float	Percentage of cloud coverage

Table 1: Dataset Schema

5 Data Preprocessing

5.1 Image Conversion Pipeline

Algorithm 1 Parallel Image Conversion

```
Input: Directory containing RAW images
Output: Converted JPEG images
for each image in directory do
    Create ThreadPoolExecutor
    Submit conversion task
    if conversion successful then
        Remove original RAW file
        Log success
    else
        Log error
    end if
end for
```

5.2 Dataset Cleaning

The dataset cleaning process involves:

- Removing invalid image entries
- Standardizing image names
- Validating image-label pairs
- Handling missing values

6 Model Architecture

6.1 CLIP-Based Feature Extraction

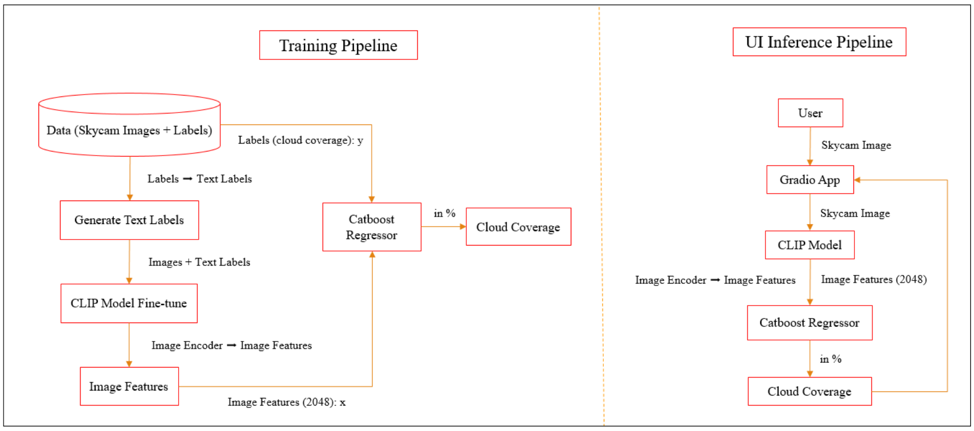

Figure 2: System Architecture Overview

The model consists of three main components:

6.1.1 Image Encoder

```python
class MemoryEfficientImageEncoder(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.model = timm.create_model(
            cfg.model_name,
            pretrained=cfg.pretrained,
            num_classes=0,
            global_pool='avg'
        )
```

6.1.2 Text Encoder

```python
class MemoryEfficientTextEncoder(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.model = DistilBertModel.from_pretrained(
            cfg.text_encoder_model
        )
```

6.1.3 Projection Head

```python
class ProjectionHead(nn.Module):
    def __init__(self, cfg, embedding_dim):
        super().__init__()
        self.projection = nn.Linear(
            embedding_dim,
            cfg.projection_dim
        )
```

7 Training Pipeline

7.1 Memory Optimization Techniques

- Gradient checkpointing
- Mixed precision training
- Efficient data loading with sharding
- Memory usage tracking

7.2 Training Configuration

```python
class Config:
    model_name = 'resnet50'
    image_embedding = 2048
    text_embedding = 768
    projection_dim = 256
    batch_size = 16
    epochs = 15
    temperature = 1.0
```

7.3 Training Process

Algorithm 2 Training Pipeline

```
Initialize model and optimizer
Load and preprocess data
for each epoch do
    Train model
    Validate performance
    if best performance then
        Save checkpoint
    end if
    Update learning rate
end for
```

8 CatBoost Model Training

8.1 Feature Engineering

The CatBoost model uses features extracted from:

- CLIP embeddings
- Image statistics
- Temporal information

8.2 Model Configuration

```
params = {
    'iterations': 1000,
    'learning_rate': 0.05,
    'depth': 6,
    'loss_function': 'RMSE'
}
```

8.3 Training and Evaluation

- Cross-validation setup
- Hyperparameter optimization
- Performance metrics monitoring

9 Results and Discussion

9.1 Performance Metrics

```
--------------------------------------------------
Train MAE: 3.063867080191343
Train RMSE: 4.519084504196453
Train MSE: 20.422124756068502
Train R2: 0.9753638347539638
--------------------------------------------------
```

```
--------------------------------------------------
Valid MAE: 5.798734045374874
Valid RMSE: 9.624912586359335
Valid MSE: 92.63894229505833
Valid R2: 0.8879491192051271
--------------------------------------------------
--------------------------------------------------
Test MAE: 5.812910249786526
Test RMSE: 9.69202086124033
Test MSE: 93.93526837471777
Test R2: 0.886006145423602
--------------------------------------------------
```

Model Performance Metrics

10 Future Work

10.1 Phase 3 Integration

The cloud coverage percentages obtained in this phase will serve as input features for Phase 3's temperature prediction model. Key integration points include:

- Cloud coverage pattern analysis
- Temperature correlation modeling
- Integration with meteorological data
- Enhanced prediction pipeline

11 Conclusion

Phase 2 successfully establishes a robust cloud coverage detection system, achieving an R^2 score of 0.886 on the test set. This foundation will be crucial for the temperature prediction implementation in Phase 3.