**loud\_forecast**

**Cloud Forecasting Project Report**

**Scope**: End-to-end pipeline to forecast next-hour cloud mask using EUMETSAT satellite imagery for June–July, with data ingestion, preprocessing, ConvLSTM model training, evaluation, and export for visualization.

**Data Ingestion and Preparation**

* **Data source**: http://203.135.4.150:3333/images/
* **Downloader**: data\_ext\_script.py
  + Crawls specific June 2025 date folders listed in TARGETS and mirrors directory structure into downloaded\_images/.
  + Streams file downloads with progress bars and robust error handling.
* **Folder discovery**: get\_summer\_folders() scans downloaded\_images/ for -06- and -07- dates.
* **File parsing**: Filenames are URL-decoded; hour extracted from patterns like [HH-mm] via regex.
* **Band selection**: Uses robust multi-band set for all-day detection: IR\_097, IR\_134, WV\_062.
* **Sequence building**: For each day:
  + Lists per-band files present for times with complete coverage.
  + Constructs sliding windows of 3 input timesteps to predict the 4th hour (3→1 forecasting).
  + Skips sequences with unreadable images or missing bands; logs warnings.
* **Image preprocessing**:
  + Convert to grayscale and resize to 256×256.
  + Normalize to [0,1]; fill black stripes (zeros) with mean of valid pixels.

**Target Construction (Cloud Mask)**

* Ground-truth mask from future hour using IR thresholds:
  + mask = (IR\_097 < 0.6) OR (IR\_134 < 0.6) producing a binary mask.

**Dataset API**

* Python generator → tf.data.Dataset pipelines with shuffle, batch (BATCH\_SIZE=8), and prefetch.
* Train/validation split by date (val\_days=3) to avoid leakage.

**Model Architecture**

* ConvLSTM2D based encoder with batch normalization:
  + ConvLSTM2D(16, 3×3, relu, return\_sequences=True)
  + BatchNorm
  + ConvLSTM2D(8, 3×3, relu, return\_sequences=False)
  + BatchNorm
  + Conv2D(1, 1×1, sigmoid)
* Input shape inferred from first valid sample: (time=3, height, width, channels=3).
* Loss: Dice loss for class imbalance; metric: accuracy.

**Training**

* Checkpointing: saves best model as cloud\_forecast\_best\_model.keras by lowest validation loss.
* Example training loop with epochs=30.

**Evaluation and Metrics**

* Visual diagnostics: Side-by-side plots of predicted vs true masks and last input IR frame.
* CSI (Critical Success Index) computed at configurable threshold (default 0.2) on binarized predictions.
* Console stats: min/max/mean of predictions and truths per batch.

**Export**

* export\_forecast() saves a predicted mask to forecast\_cloud\_mask.png for web/asset use.

**Repository Artifacts**

* cloud\_forecast\_pipeline.py: Full pipeline and training script.
* data\_ext\_script.py: Robust recursive downloader for date folders.
* Trained models: cloud\_forecast\_best\_model.keras (root and subfolders: model 3/, pehla model/).
* Notebooks: Cloud\_forecast.ipynb, dir.ipynb (exploration/experiments).
* Images: model 3 images/ holds sample outputs; hm.png, forecast\_cloud\_mask.png are visualization artifacts.

**How to Run**

1. Download data
   * Edit TARGETS in data\_ext\_script.py as needed.
   * Run: python data\_ext\_script.py (creates downloaded\_images/DATE/...).
2. Train model
   * Ensure downloaded\_images/ contains required bands for selected dates.
   * Run: python cloud\_forecast\_pipeline.py (saves best model to cloud\_forecast\_best\_model.keras).
3. Inspect results
   * Review plotted figures and forecast\_cloud\_mask.png export.

**Notes and Decisions**

* Chose IR-centric thresholds to define cloud masks, complemented by water vapor band context.
* Dice loss preferred due to sparse positive class in segmentation.
* Sliding window ensures temporal context without excessive memory footprint.
* Defensive preprocessing handles corrupted/striped imagery gracefully.

**Next Steps (Optional)**

* Add more channels (e.g., VIS\_006/008, HRV) and learnable fusion.
* Replace hand-crafted thresholds with supervised labels if available.
* Use focal loss or BCE+DICE hybrid; add Tversky index.
* Add data augmentation and mixed precision training.
* Evaluate with additional metrics (IoU, Precision/Recall) across thresholds.