

LenghuSky-8: An 8-Year All-Sky Cloud Dataset with Star-Aware Masks and Alt-Az Calibration for Segmentation and Nowcasting

Supplementary Material

Table 7. Statistics of failure case of all-sky camera. The events that occurs within 12 hours are merged together as one event.

Reason	# occur	# frame	Down time [days]
cover	181	30111	169.79
stronglight	388	5275	23.54
cameradown	56	2167	20.75
object	40	735	4.66
Total	665	38288	218.75

Table 8. Optional projection types of full-sky camera, which is provided by [19].

Name	Formula
Gnomonic projection	$r = f \tan \theta$
Stereographic projection	$r = 2f \tan \frac{\theta}{2}$
Equidistant projection	$r = 2f \theta$
Equisolid angle projection	$r = 2f \sin \frac{\theta}{2}$
Orthographic projection	$r = f \sin \theta$

A. Failure cases of the all-sky camera

Failure of all-sky camera refers to being unable to recognize a large proportion of the sky condition manually. Typical cases of failure is shown in Fig.4. We manually inspect the whole dataset and mark out the failure cases. Case 1~4 in Fig.4 are marked as cover; Case 5 are marked as object; Case 6 and 7 are marked as strong light; Case 8 is marked as Camera malfunction. 665 failures are recognized by human in the dataset. The start and the end of these events are also recorded. In segmentation task, the regions that cannot decide whether there is cloud or not are annotated as “contamination” class. An example of “cover” is the third column of Fig. 1. A statistics of the failure time is shown in Table 7. Mud and dew are the most frequent causes of failure, particularly from November through May. The complete table is available in the online repository. More detailed weather monitoring and camera failure information is available in <https://huggingface.co/datasets/ruiyicheng/LenghuSky-8/tree/main/data>. The region that would affect the determination of the local weather are annotated as contamination, e.g. the region that is covered by dew.

B. Projection type of all-sky camera

In this work, several projection types are used for distortion correction. These types are listed in table 8.

C. Manual annotation samples and annotation process details for cloud segmentation task

Manual annotation samples for cloud segmentation task is shown in Fig. 5. The 1,111-image reference set was annotated by 9 trained students/engineers with astronomy background under an astronomer lead. We stratified sampling by time-of-day, season, moon phase, and cloud coverage to ensure balanced coverage; rare cases largely determine the final size. Each image is labeled in LabelMe using written guidelines with conservative partial labeling: only high-confidence regions are labeled; ambiguous pixels are left unlabeled and ignored in loss/metrics. Every image then undergoes a second-pass expert review/edit to enforce cross-annotator consistency; disagreements are resolved by adjudication in this pass.

Some bright structures in examples Fig. 5 arise from scattering/stray light (dust/dew/scratches/saturation) and can resemble thin clouds in a single frame. Our guideline treats these as contamination when they obstruct sky/cloud attribution; otherwise they are annotated as sky/cloud, or left unlabeled if ambiguous. Annotators also consult adjacent frames to distinguish evolving clouds from static artifacts.

D. Fitting and residual of astrometric calibration

The Altitude-Azimuth fitting map for each time slot is shown in Fig. 6. Residual of fitting results are shown in Fig. 7.

E. Annotation of background

Annotation of the background as listed in Table 3 is shown in Fig. 8.

F. Experimental details of baseline models

F.1. Encoder-Decoder for cloud segmentation

We use polygon annotations produced in LabelMe JSON format. For each sample, the RGB image is recovered from the embedded base64-encoded payload and converted to a three-channel array. Semantic masks are rasterized by filling the annotated polygons per class on an empty canvas that shares the image resolution. Pixels not covered by any polygon are assigned an “ignore” label. The class mapping is sky→0, cloud→1, contamination→2, and

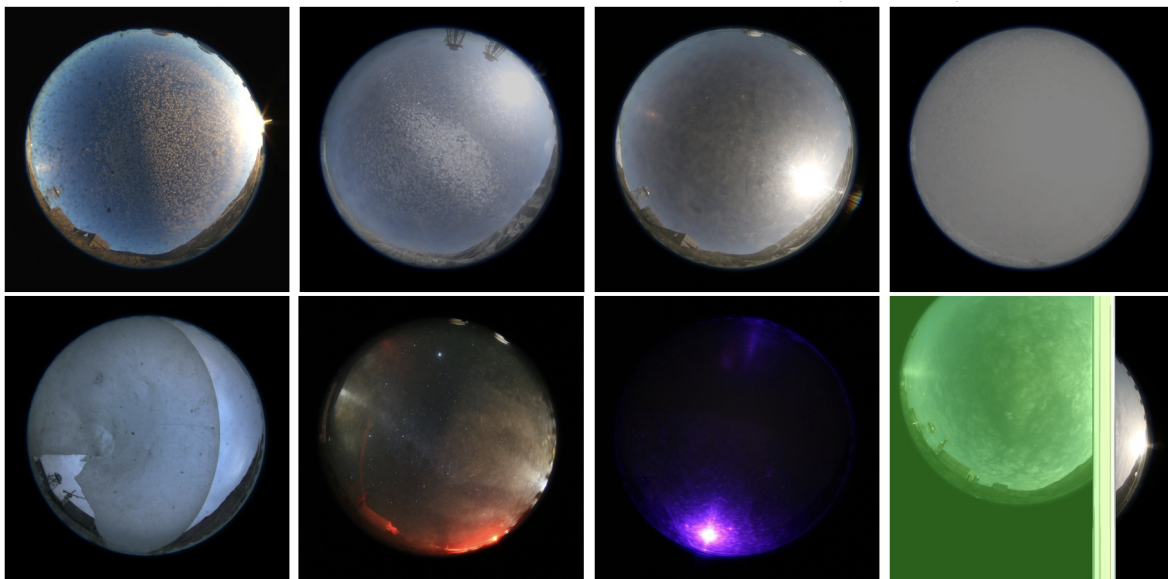


Figure 4. Failure cases of the all-sky camera. Top row (left to right): (1) Covered by dust or sand; (2) Covered by dew or ice; (3) Scattered light caused by mud coverage; (4) Covered by snow. Bottom row (left to right): (5) Obstruction by an external object; (6) Strong nearby light source; (7) Strong distant light source; (8) Camera malfunction.

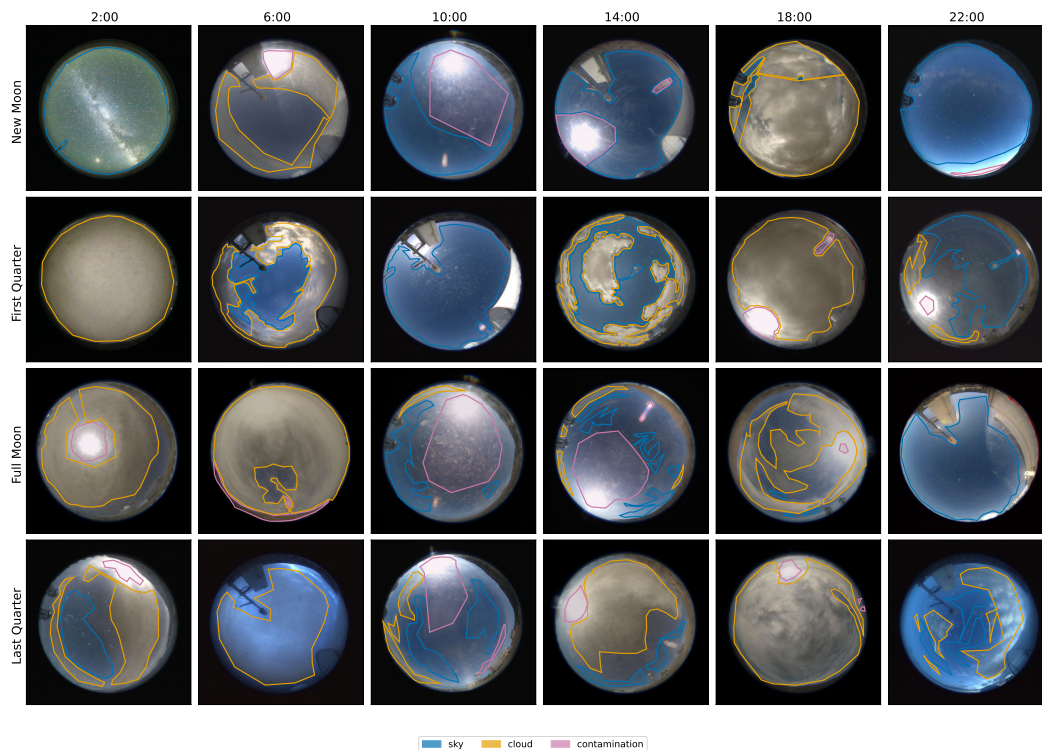


Figure 5. Manual annotation for cloud segmentation. Blue represents cloud regions; orange represents sky regions; Pink represents contamination regions; Columns represents images taken around given time in UTC+8; Rows represents images taken in different moon phase condition. Nearby frames are used by human to determine whether some regions are scatter light or cloud.

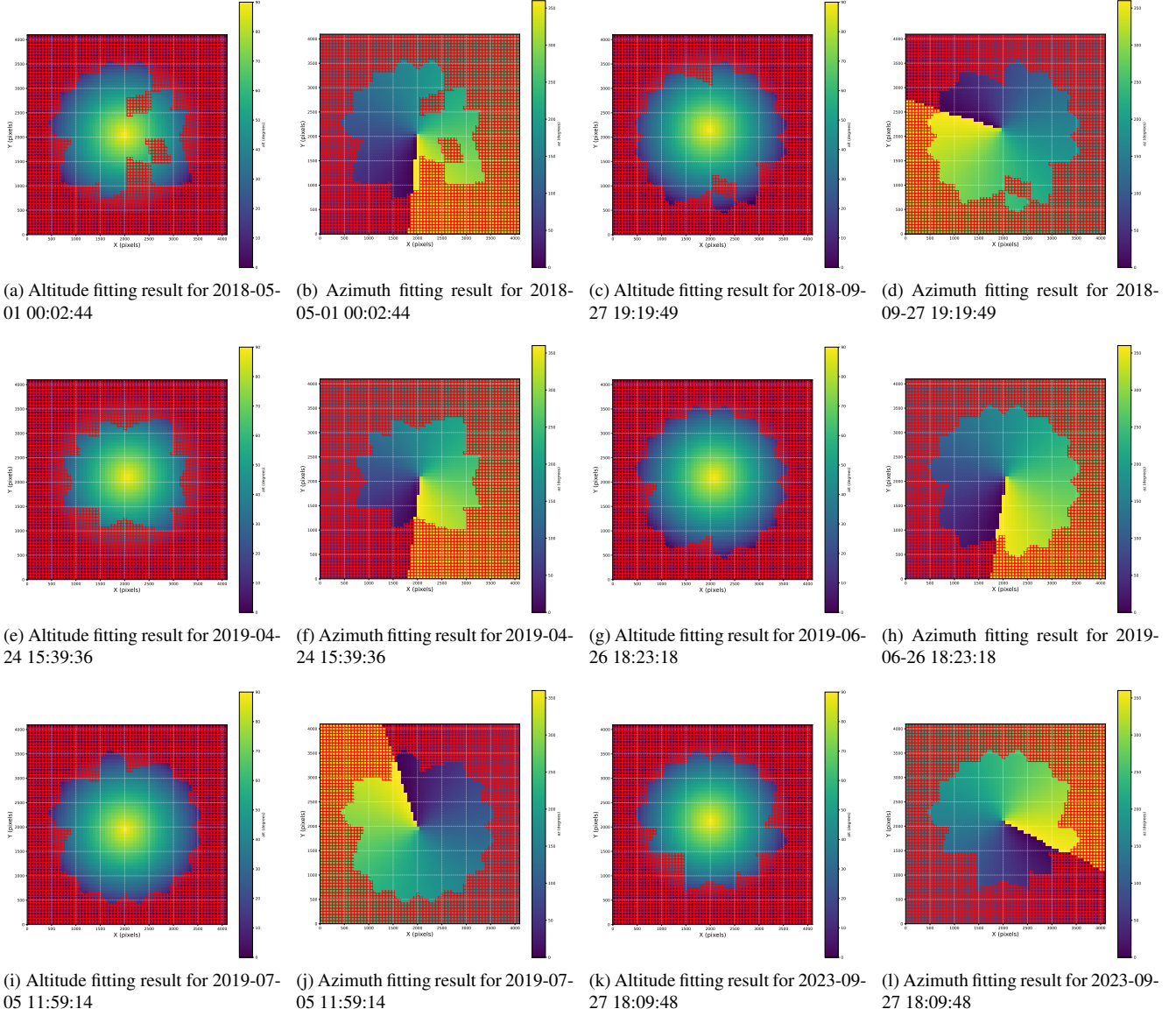


Figure 6. Altitude–Azimuth fitting results for different time slots. Red boxes means that the WCS of the corresponding HEALPix cell is not resolved by ASTROMETRY.NET for any image in the ensemble, and the corresponding altitude and azimuth is obtained by fitting results.

`ignore→3`. Polygons are rounded to integer coordinates and clipped to image bounds before rasterization to avoid off-by-one artifacts.

CloudSegNet is a compact encoder–decoder CNN with two downsampling stages and two symmetric upsampling stages. A layer-wise specification of this model is shown in Table 9.

We train with pixel-wise cross-entropy using `ignore_index`, and uniform class weights. The optimizer is Adam with learning rate $1e-4$. We train for up to 500 epochs and apply early stopping on validation loss

with a patience of 5 epochs, retaining the checkpoint with the best validation loss.

F.2. U-Net (CloudU-Net) for cloud segmentation

The pre-processing steps and training parameters of U-Net is similar to encoder-decoder as above. The layer-wise specification of the applied model in this paper is shown in Table 10.

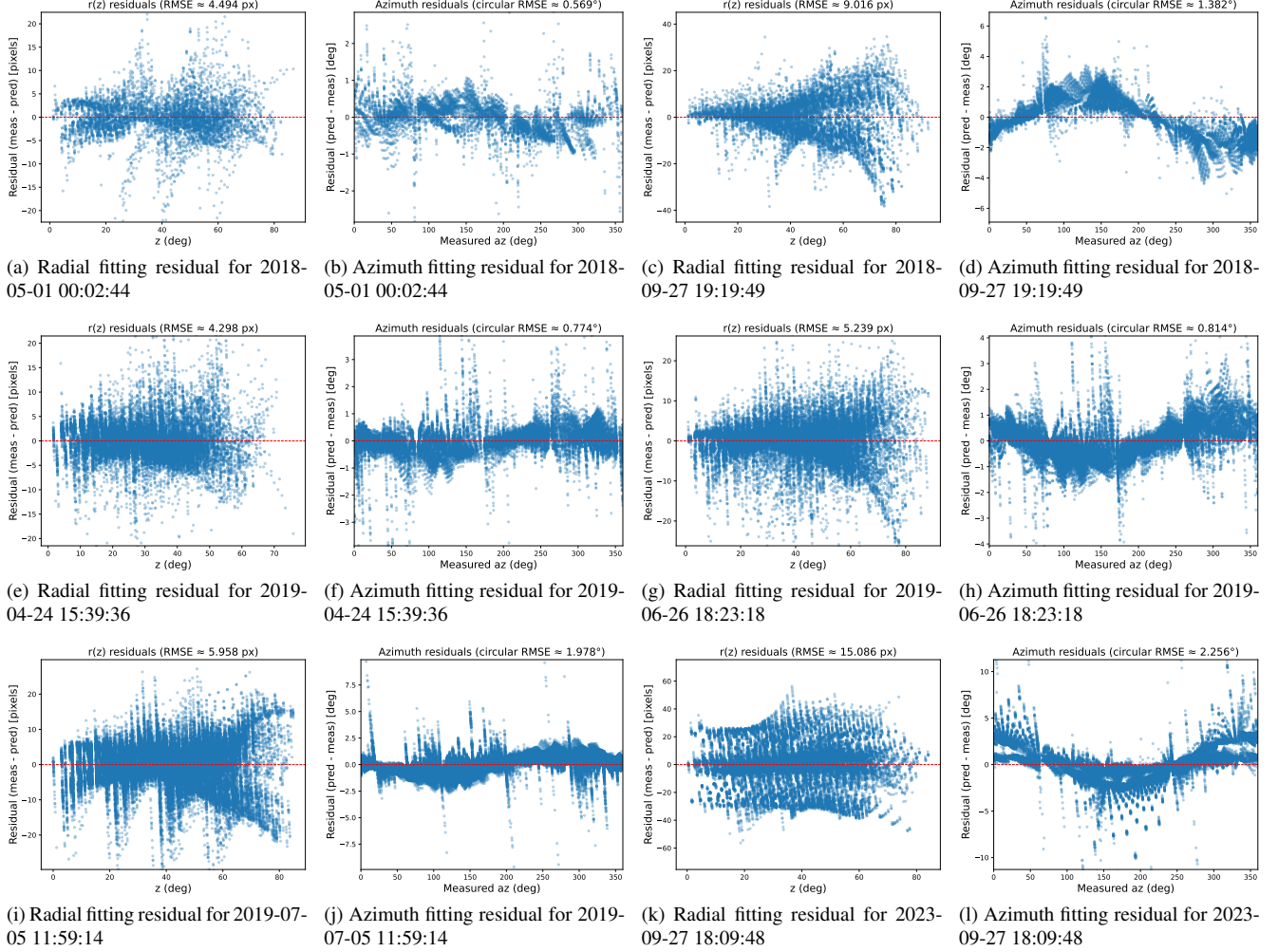


Figure 7. Astrometric calibration fitting residual for different time slots in radial and azimuthal direction.

Table 9. Layer-wise specification of Encoder-Decoder (CloudSegNet) for a $3 \times 512 \times 512$ input. “k/s/p” denotes kernel/stride/padding.

Stage	Layer (in→out)	Operation	k/s/p	Channel out	Size (H×W)
Enc-1	3→64	Conv2d + ReLU + BN	3/1/1	64	512×512
Enc-1	64→64	Conv2d + ReLU + BN	3/1/1	64	512×512
Enc-1↓	–	MaxPool2d	2/2/0	64	256×256
Enc-2	64→128	Conv2d + ReLU + BN	3/1/1	128	256×256
Enc-2	128→128	Conv2d + ReLU + BN	3/1/1	128	256×256
Enc-2↓	–	MaxPool2d	2/2/0	128	128×128
Dec-1↑	128→64	ConvTranspose2d + ReLU	2/2/0	64	256×256
Dec-1	64→64	Conv2d + ReLU + BN	3/1/1	64	256×256
Dec-2↑	64→32	ConvTranspose2d + ReLU	2/2/0	32	512×512
Head	32→3	Conv2d (logit)	1/1/0	3	512×512

F.3. SegMAN for cloud segmentation

The pre-processing steps and training parameters of U-Net is similar to encoder-decoder as above. SegMAN is an encoder–decoder segmentation network with scalable vari-

ants (Tiny/Small/Base/Large). The parameters used in this work for these scale is exactly the same with the original paper [12]. All variants share the same data interface and segmentation head: the model outputs dense logits that are bilinearly resized to the mask resolution when needed. Vari-

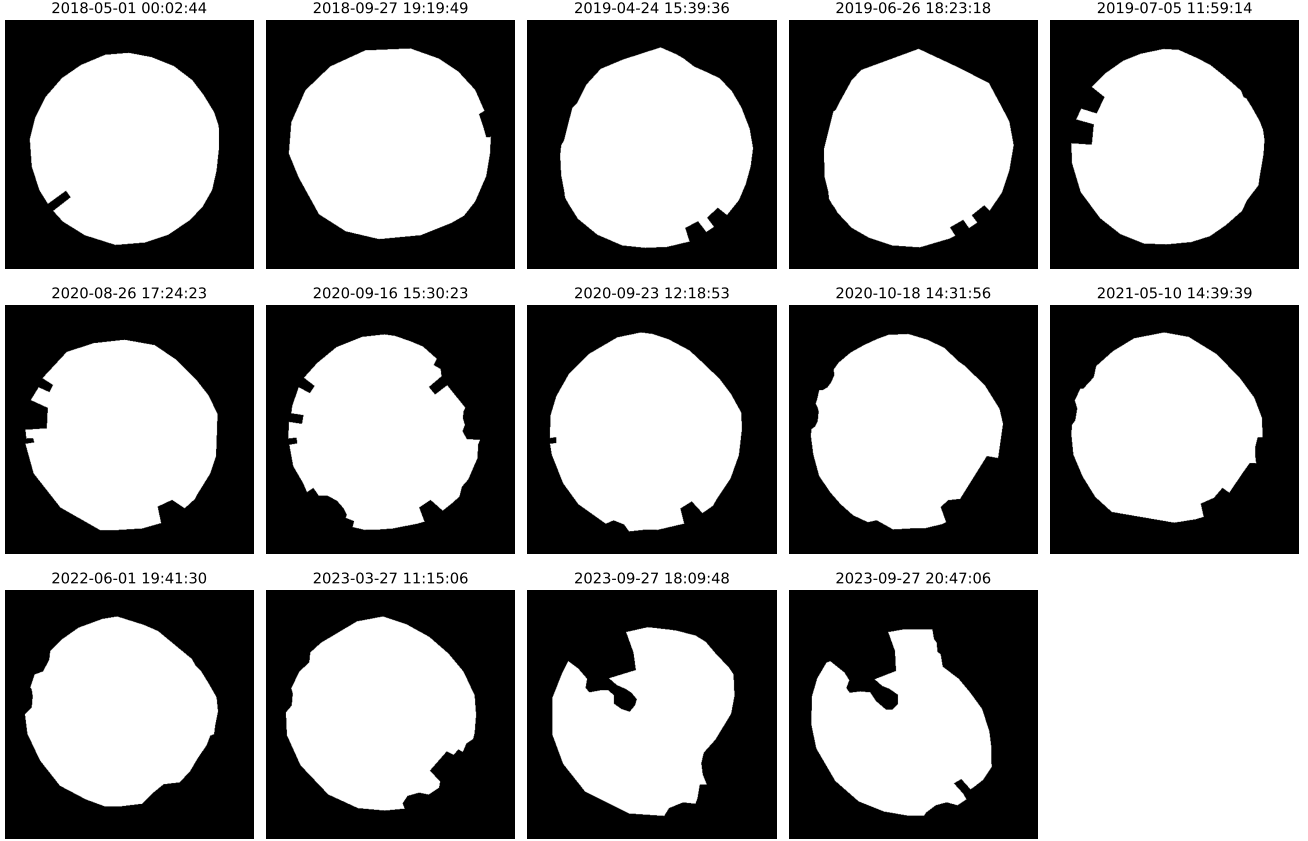


Figure 8. Annotation of all background

ants differ only in encoder capacity and decoder width; we keep training/evaluation identical across variants.

F.4. ConvLSTM for cloud nowcast

We implement a next-frame ConvLSTM baseline on sequences of logits with per-frame masks. Each training sample comes from `CloudLogitsDataset`, which yields a video tensor $x \in \mathbb{R}^{C \times T \times H \times W}$ together with a binary mask stack $m \in \{0, 1\}^{T \times H \times W}$; frames are built from timestamped files and normalised per sequence by $(x - \mu)/(\sigma \cdot 6)$ before batching. For learning, we enforce $T = n_{\text{input}} + 1$, feed the first n_{input} frames to the ConvLSTM, and regress the last frame with a masked MSE on the target mask m_T ; optimisation uses Adam. The configuration used in our experiments sets $n_{\text{input}}=2$, hidden dimensions $[64, 64]$, kernel size 3, and dropout 0, with data sampled at 60-min intervals from user-specified time ranges. Layer-wise specification of used ConvLSTM in the paper is shown in Table 11.

F.5. VideoGPT for cloud nowcast

Our VideoGPT nowcaster follows a two-stage discrete latent modelling pipeline. First, a VQ-VAE encodes videos

by time-slicing, then vector-quantises features with a codebook of size K ; training minimises masked reconstruction MSE plus standard commitment/codebook losses and tracks codebook perplexity. After training VQ-VAE, we freeze the best checkpoint and train a causal Transformer (GPT) as an autoregressive language model over the flattened VQ indices, using cross-entropy on next-token prediction. Key hyperparameters include $K=512$ codes for VQ-VAE and a GPT with $d_{\text{model}}=512$, $n_{\text{head}}=8$, and 6 layers. Architecture of the VideoGPT used in this work is shown in Table 12.

G. Per-class experimental results

The per-class nowcasting results are shown in Table 13; The per-class cloud segmentation results are shown in Table 14.

Table 10. Layer-wise specification of the U-Net used in our experiments (bilinear upsampling variant) for a $3 \times 512 \times 512$ input.

Stage	Layer (in→out)	Operation	k/s/p	Ch. out	Size (H×W)
Enc-0	3→64	Conv2d + ReLU + BN	3/1/1	64	512×512
Enc-0	64→64	Conv2d + ReLU + BN	3/1/1	64	512×512
Down-1↓	–	MaxPool2d	2/2/0	64	256×256
Down-1	64→128	Conv2d + ReLU + BN	3/1/1	128	256×256
Down-1	128→128	Conv2d + ReLU + BN	3/1/1	128	256×256
Down-2↓	–	MaxPool2d	2/2/0	128	128×128
Down-2	128→256	Conv2d + ReLU + BN	3/1/1	256	128×128
Down-2	256→256	Conv2d + ReLU + BN	3/1/1	256	128×128
Down-3↓	–	MaxPool2d	2/2/0	256	64×64
Down-3	256→512	Conv2d + ReLU + BN	3/1/1	512	64×64
Down-3	512→512	Conv2d + ReLU + BN	3/1/1	512	64×64
Bottleneck↓	–	MaxPool2d	2/2/0	512	32×32
Bottleneck	512→512	Conv2d + ReLU + BN	3/1/1	512	32×32
Bottleneck	512→512	Conv2d + ReLU + BN	3/1/1	512	32×32
Up-1↑	512→512	Upsample (bilinear)	2/–/–	512	64×64
Up-1	512+512	Concat (skip from Down-3)	–	1024	64×64
Up-1	1024→256	Conv2d + ReLU + BN	3/1/1	256	64×64
Up-1	256→256	Conv2d + ReLU + BN	3/1/1	256	64×64
Up-2↑	256→256	Upsample (bilinear)	2/–/–	256	128×128
Up-2	256+256	Concat (skip from Down-2)	–	512	128×128
Up-2	512→128	Conv2d + ReLU + BN	3/1/1	128	128×128
Up-2	128→128	Conv2d + ReLU + BN	3/1/1	128	128×128
Up-3↑	128→128	Upsample (bilinear)	2/–/–	128	256×256
Up-3	128+128	Concat (skip from Down-1)	–	256	256×256
Up-3	256→64	Conv2d + ReLU + BN	3/1/1	64	256×256
Up-3	64→64	Conv2d + ReLU + BN	3/1/1	64	256×256
Up-4↑	64→64	Upsample (bilinear)	2/–/–	64	512×512
Up-4	64+64	Concat (skip from Enc-0)	–	128	512×512
Up-4	128→64	Conv2d + ReLU + BN	3/1/1	64	512×512
Up-4	64→64	Conv2d + ReLU + BN	3/1/1	64	512×512
Head	64→3	Conv2d (logits)	1/1/0	3	512×512

Table 11. Layer-wise specification of the ConvLSTM next-frame baseline. Input is a $[B, C=3, T_{in}=2, H, W]$ clip; output is a single frame $[B, 3, H, W]$. “k/s/p” denotes kernel/stride/padding.

Stage	Layer (in→out)	Operation	k/s/p	Ch. out	Size (H×W)
LSTM-1	3→64	ConvLSTM2D + Dropout	3/1/1	64	$H \times W$
LSTM-2	64→64	ConvLSTM2D + Dropout	3/1/1	64	$H \times W$
Head	64→3	Conv2d (readout)	1/1/0	3	$H \times W$

Table 12. Architecture of the VideoGPT nowcaster (VQ-VAE + GPT). For a $[B, 3, T, 64, 64]$ input, the encoder downsamples to $[B, 256, T, 8, 8]$, which is vector-quantised (codebook size K). Tokens are modeled autoregressively by a Transformer and decoded back to frames.

VQ-VAE Encoder					
Enc-1	$3 \rightarrow 64$	Conv2d + BN + ReLU + ResidualBlock	4/2/1	64	32×32
Enc-2	$64 \rightarrow 128$	Conv2d + BN + ReLU + ResidualBlock	4/2/1	128	16×16
Enc-3	$128 \rightarrow 256$	Conv2d + BN + ReLU + ResidualBlock	4/2/1	256	8×8
Bottleneck	$256 \rightarrow 256$	Conv2d + BN + ReLU	3/1/1	256	8×8
Vector Quantiser					
VQ	$256 \rightarrow K$	VectorQuantizer ($K=512, D=256$)	–	K	$T \times 8 \times 8$
VQ-VAE Decoder					
Dec-3 \uparrow	$256 \rightarrow 128$	ConvTranspose2d + BN + ReLU + ResidualBlock	4/2/1	128	16×16
Dec-2 \uparrow	$128 \rightarrow 64$	ConvTranspose2d + BN + ReLU + ResidualBlock	4/2/1	64	32×32
Dec-1 \uparrow	$64 \rightarrow 64$	ConvTranspose2d + BN + ReLU + ResidualBlock	4/2/1	64	64×64
Head	$64 \rightarrow 3$	Conv2d + Tanh	3/1/1	3	64×64
GPT (autoregressive over tokens)					
TokEmb	$K \rightarrow d_{\text{model}}$	Embedding	–	512	–
PosEmb	–	Learned positional embedding (length 20,000)	–	512	–
Transf	$512 \rightarrow 512$	#Layers= 6, #Heads= 8 (TransformerEncoderLayer)	–	512	–
LN	$512 \rightarrow 512$	LayerNorm	–	512	–
Head	$512 \rightarrow K$	Linear (projection to vocab)	–	K	–

Table 13. Per-class metrics for nowcasting

Class	Baseline	Precision	Recall	F1-score
Sky	Trivial	0.914	0.913	0.914
	Optical Flow	0.913	0.913	0.913
	ConvLSTM	0.922	0.905	0.913
	VideoGPT-1	0.920	0.882	0.900
	VideoGPT-2	0.915	0.887	0.901
	VideoGPT-7	0.919	0.883	0.900
Cloud	Trivial	0.876	0.878	0.877
	Optical Flow	0.878	0.878	0.878
	ConvLSTM	0.860	0.907	0.883
	VideoGPT-1	0.833	0.904	0.867
	VideoGPT-2	0.841	0.897	0.868
	VideoGPT-7	0.830	0.903	0.865
Contamination	Trivial	0.787	0.782	0.785
	Optical Flow	0.784	0.788	0.786
	ConvLSTM	0.831	0.734	0.779
	VideoGPT-1	0.768	0.685	0.724
	VideoGPT-2	0.763	0.690	0.724
	VideoGPT-7	0.774	0.668	0.717

Table 14. Per-class metrics for cloud segmentation(median with 16th-83rd percentiles)

Class	Baseline	Precision	Recall	F1-score
Sky	DINOv3 local	0.920 ^{+0.028} _{-0.022}	0.945 ^{+0.018} _{-0.009}	0.936 ^{+0.006} _{-0.015}
	DINOv3 local + CLS	0.917 ^{+0.037} _{-0.025}	0.941 ^{+0.019} _{-0.014}	0.932 ^{+0.012} _{-0.013}
	DINOv3 local + CLS + register	0.913 ^{+0.038} _{-0.031}	0.942 ^{+0.018} _{-0.025}	0.926 ^{+0.015} _{-0.012}
	DINOv3 local + mean(CLS + register)	0.921 ^{+0.031} _{-0.019}	0.944 ^{+0.018} _{-0.019}	0.936 ^{+0.009} _{-0.016}
	Encoder-Decoder	0.764 ^{+0.061} _{-0.027}	0.868 ^{+0.056} _{-0.034}	0.816 ^{+0.021} _{-0.009}
	U-Net	0.869 ^{+0.023} _{-0.039}	0.912 ^{+0.025} _{-0.047}	0.882 ^{+0.017} _{-0.016}
	SegMAN Tiny	0.881 ^{+0.027} _{-0.019}	0.926 ^{+0.022} _{-0.027}	0.906 ^{+0.008} _{-0.017}
	SegMAN Small	0.871 ^{+0.026} _{-0.031}	0.929 ^{+0.031} _{-0.019}	0.904 ^{+0.013} _{-0.010}
	SegMAN Base	0.874 ^{+0.017} _{-0.027}	0.927 ^{+0.025} _{-0.017}	0.901 ^{+0.010} _{-0.012}
	SegMAN Large	0.854 ^{+0.025} _{-0.014}	0.931 ^{+0.021} _{-0.026}	0.899 ^{+0.005} _{-0.021}
Cloud	DINOv3 local	0.961 ^{+0.013} _{-0.018}	0.966 ^{+0.011} _{-0.011}	0.961 ^{+0.013} _{-0.010}
	DINOv3 local + CLS	0.965 ^{+0.006} _{-0.024}	0.969 ^{+0.007} _{-0.016}	0.960 ^{+0.012} _{-0.012}
	DINOv3 local + CLS + register	0.954 ^{+0.019} _{-0.016}	0.965 ^{+0.013} _{-0.011}	0.960 ^{+0.009} _{-0.013}
	DINOv3 local + mean(CLS + register)	0.966 ^{+0.010} _{-0.021}	0.970 ^{+0.007} _{-0.017}	0.962 ^{+0.013} _{-0.010}
	Encoder-Decoder	0.823 ^{+0.052} _{-0.034}	0.823 ^{+0.050} _{-0.047}	0.822 ^{+0.028} _{-0.035}
	U-Net	0.863 ^{+0.035} _{-0.062}	0.921 ^{+0.017} _{-0.031}	0.880 ^{+0.020} _{-0.036}
	SegMAN Tiny	0.902 ^{+0.031} _{-0.027}	0.925 ^{+0.018} _{-0.042}	0.910 ^{+0.017} _{-0.019}
	SegMAN Small	0.905 ^{+0.031} _{-0.029}	0.912 ^{+0.036} _{-0.047}	0.904 ^{+0.020} _{-0.015}
	SegMAN Base	0.900 ^{+0.028} _{-0.029}	0.921 ^{+0.021} _{-0.038}	0.913 ^{+0.008} _{-0.024}
	SegMAN Large	0.899 ^{+0.025} _{-0.034}	0.900 ^{+0.032} _{-0.024}	0.899 ^{+0.018} _{-0.024}
Contamination	DINOv3 local	0.808 ^{+0.056} _{-0.092}	0.686 ^{+0.080} _{-0.121}	0.726 ^{+0.059} _{-0.055}
	DINOv3 local + CLS	0.783 ^{+0.068} _{-0.057}	0.644 ^{+0.135} _{-0.068}	0.715 ^{+0.053} _{-0.050}
	DINOv3 local + CLS + register	0.806 ^{+0.060} _{-0.084}	0.630 ^{+0.121} _{-0.113}	0.707 ^{+0.067} _{-0.059}
	DINOv3 local + mean(CLS + register)	0.793 ^{+0.073} _{-0.087}	0.675 ^{+0.126} _{-0.114}	0.715 ^{+0.047} _{-0.042}
	Encoder-Decoder	0.730 ^{+0.108} _{-0.124}	0.285 ^{+0.077} _{-0.071}	0.396 ^{+0.077} _{-0.056}
	U-Net	0.839 ^{+0.055} _{-0.278}	0.280 ^{+0.076} _{-0.089}	0.415 ^{+0.072} _{-0.130}
	SegMAN Tiny	0.637 ^{+0.113} _{-0.126}	0.437 ^{+0.112} _{-0.092}	0.496 ^{+0.100} _{-0.052}
	SegMAN Small	0.678 ^{+0.117} _{-0.153}	0.383 ^{+0.121} _{-0.077}	0.489 ^{+0.043} _{-0.057}
	SegMAN Base	0.683 ^{+0.069} _{-0.065}	0.376 ^{+0.089} _{-0.105}	0.483 ^{+0.057} _{-0.095}
	SegMAN Large	0.654 ^{+0.145} _{-0.112}	0.359 ^{+0.066} _{-0.087}	0.464 ^{+0.045} _{-0.065}