TRACE: LIGHTWEIGHT INFRARED SMALL MOVING TARGET DETECTION UNDER HEAVY NOISE WITH TEMPORAL TEMPLATES AND NEIGHBORHOOD EVIDENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Infrared small moving targets occupy only a few pixels, lack texture and shape information, and are often buried in strong noise and complex backgrounds, making reliable detection highly challenging. Existing multi-frame methods improve detection by exploiting inter-frame motion, and per-pixel temporal methods among them achieve leading results. However, these methods generally process multi-frame features throughout the network, resulting in slow inference and high memory consumption, and they struggle to distinguish targets from noise fluctuations under heavy noise, where the temporal change of a target pixel is often weaker than that of a noise pixel. We observe that a moving target changes neighboring pixels one after another, whereas a noise fluctuation stays on a single pixel, and propose TRACE, a lightweight detection network built on this difference. After standardizing each pixel against its own intensity history in the temporal window, TRACE compresses the multi-frame sequence into two-dimensional features using shared learnable temporal templates in the first layer, accumulating weak target signals while avoiding layer-by-layer multi-frame processing. It then aggregates evidence over a small spatial neighborhood, following the target track and suppressing false alarms caused by noise. Experiments on public datasets show that TRACE outperforms representative multi-frame methods, particularly under heavy noise, and that, compared with the leading per-pixel temporal method, it requires fewer parameters and less memory while running faster. This work offers a new perspective on efficient and robust infrared small moving target detection.

1 INTRODUCTION

Infrared early-warning and surveillance systems must detect a moving target while it is still far away, when it covers only a few pixels and has neither shape nor texture, and locate it in every frame at a fixed false-alarm rate. We study this task on NUDT-MIRSDT (Li et al., 2023b) and on its public heavy-noise version NUDT-MIRSDT-HiNo (Li et al., 2026), hereafter HiNo. Under heavy noise, the target signal can be weaker than the noise. Its contrast in one frame and the temporal profile of its pixel then often fail to separate it from noise, and ranking pixels by their own temporal change places noise above the target.

Single-frame detectors rely on the contrast of the target with its surround. Much of their design serves to keep a target of a few pixels from vanishing during downsampling, as in the exchange between shallow and deep features of ACM (Dai et al., 2021a) or the dense nested connections of DNA-Net (Li et al., 2022). Yet heavy noise can remove this contrast altogether. Multi-frame methods add temporal evidence, since a moving target changes each pixel it crosses only briefly, while the background drifts slowly. Learned multi-frame networks such as DTUM (Li et al., 2023b) fuse the features of several frames, yet those compared here miss many heavy-noise targets.

Temporal-profile methods (Gao et al., 2023; Li et al., 2026) detect the target as an anomaly in the profile of each pixel. Under heavy noise, however, the target pixel is often not the strongest anomaly in its frame: Appendix D shows that on the lowest-contrast frame of each HiNo test sequence, the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

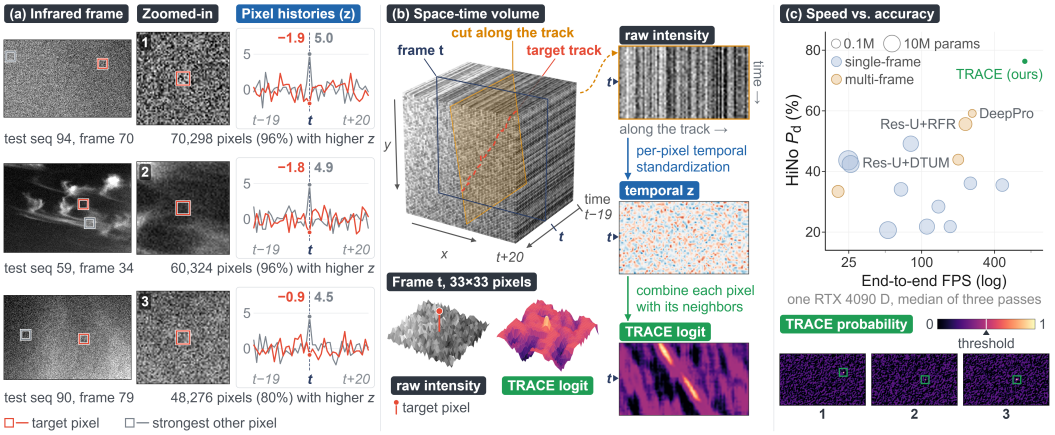


Figure 1: TRACE matches the profile of every pixel with shared temporal templates in one layer and aggregates the responses over a neighborhood that contains the target track. (a) Three HiNo test frames; red: target pixel; gray: the other pixel with the largest unclipped temporal z at t; counts: pixels whose z at t exceeds that of the target pixel. (b) Space-time volume of frames $t-19,\ldots,t+20$ of HiNo test sequence 101 around $t=45$: a cut along the target track (raw intensity, temporal z, TRACE logit) and a 33×33-pixel patch of frame t (raw intensity, TRACE logit). (c) HiNo detection rate P_d against end-to-end FPS, TRACE as in Table 1; TRACE probability maps (seed 1103) of the frames of (a), thresholded at 0.5, as in DeepPro (Li et al., 2026); green box: detected target.

strongest noise pixel exceeds the target pixel in temporal excursion by a median factor of 3.4, never less than 1.2. Figure 1(a) shows three low-contrast HiNo test frames, selected as in Appendix D; in each, most pixels are brighter relative to their profile than the target, so that ranking pixels by their own temporal change would report many noise pixels before the target. Yet the target differs from noise in its track: it changes neighboring pixels one after another, while a noise fluctuation stays on one pixel.

To exploit this difference, we propose TRACE, a small network with two trained parts: temporal template matching (TTM) and local spatial aggregation (LSA). After a parameter-free per-pixel temporal standardization (PTS), TTM matches the profile of every pixel over a 40-frame window with the same 32 learned temporal templates. This matching also reduces the sequence to two-dimensional maps in a single step, leaving no time axis in any later layer. LSA then aggregates the template responses over a neighborhood that contains the track of a target over the frames around the output frame. Along this track, the responses of neighboring pixels support one another, whereas an isolated fluctuation finds no such support. Heavy noise can thus hide the target in the profile of its own pixel, but not in the pattern that its track leaves on the template responses of the neighborhood. Figure 1(b) cuts the frames around t along the track of a target: the track cannot be seen in the raw intensity or in the temporal z after PTS, but it stands out in the TRACE logit. On the frames of Figure 1(a), TRACE detects every target, and among the compared deep methods it is the fastest and detects the most heavy-noise targets (Figure 1(c)). Our contributions are threefold:

- We present TRACE, a lightweight multi-frame infrared small target detector that enables accurate, real-time target detection under heavy noise, which substantially outperforms all 24 compared methods in detection accuracy.
- We propose temporal template matching (TTM), which models the full temporal window at every pixel within a single layer. TTM applies learned templates shared across all pixels to the entire window in one layer; since no subsequent layer carries a time axis, TRACE remains compact and efficient.
- We propose local spatial aggregation (LSA), which suppresses noise by scoring each pixel from the template responses of its neighborhood along the target track. LSA plays a key role in enabling robust detection under heavy noise without retraining across diverse noise models.

2 RELATED WORK

Single-frame detection. Classical detectors suppress the background with a max-median filter (Deshpande et al., 1999) or a two-dimensional adaptive prediction filter (Soni et al., 1993), or they enhance the target through its local contrast, as in LCM (Chen et al., 2014). Others separate a sparse target from a low-rank background, as in IPI (Gao et al., 2013). Learned detectors replace these models with a segmentation network. ACM (Dai et al., 2021a) passes point-wise detail from shallow layers to deep ones and global context in the opposite direction. DNA-Net (Li et al., 2022) extends this idea with dense nested connections between all depths, and UIUNet (Wu et al., 2022) with small U-Nets nested inside a large one. Other networks incorporate classical priors into their layers. ALCNet (Dai et al., 2021b) recasts the local contrast measure as a parameter-free layer, and RPCANet (Wu et al., 2024) unfolds robust principal component analysis into network stages. Beyond such priors, ISNet (Zhang et al., 2022) adds edge blocks to recover the shape of the target, and SCTransNet (Yuan et al., 2024) relates target and background through cross-attention on the skip connections. In contrast, MSHNet (Liu et al., 2024) retains a plain U-Net and trains it with a scale- and location-sensitive loss. All of these detectors read a single frame, and on HiNo every single-frame network in Table 1 misses more targets than it detects.

Multi-frame detection. Track-before-detect methods (Barniv, 1985; Reed et al., 1988) accumulate the signal of a dim target along hypothesized paths, and tensor methods split a stack of frames into a low-rank background and a sparse target. MSLSTIPT (Sun et al., 2020) learns multiple subspaces of a spatial-temporal patch tensor, SRSTT (Li et al., 2023a) regularizes a twisted spatial-temporal tensor for sparsity, and 4D-TR and 4D-TT (Wu et al., 2023) unfold a four-dimensional tensor with tensor ring and tensor train decompositions. However, the eight traditional methods in Table 1 detect only a small fraction of the heavy-noise targets. Learned multi-frame networks add temporal modeling to a segmentation backbone. DTUM (Li et al., 2023b) is a direction-coded temporal module attached to single-frame networks, and STDMANet (Yan et al., 2023) applies differential attention at several scales. RFR (Ying et al., 2025a) refines features recurrently over frames, and DQAligner (Deng et al., 2026) learns global dynamic queries for large-motion targets. On clean data, all four detect more targets than any single-frame network in Table 1, yet on HiNo the detection rate of each drops sharply. Recent networks outside Table 1 include TDCNet (Fang et al., 2026), which re-parameterizes temporal differences and 3D convolution into single convolutions, MOCID (Zhang et al., 2025), which pairs Fourier-based spatio-temporal attention with a displacement-aware Mamba module, CHAL (Duan et al., 2026), which treats targets as anomalies in a learned background, and TEP-SAM (Xing et al., 2026), which prompts the Segment Anything Model with temporal cues. Outside infrared detection, video recognition networks can fuse frames in their first layer (Karpathy et al., 2014) or through temporal convolutions factorized from spatial ones (Tran et al., 2018). TRACE also fuses the 40 frames in its first layer, but separately for each pixel and before any spatial layer.

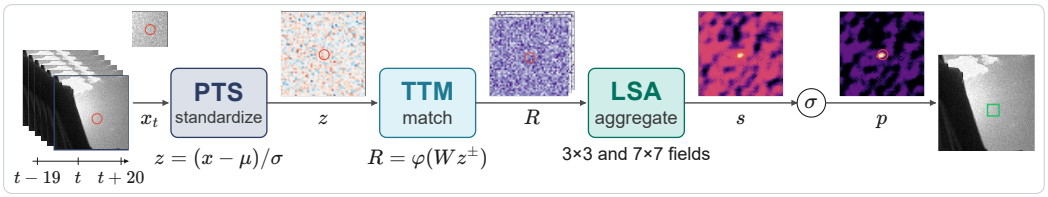
Temporal-profile detection. Gao et al. (2023) standardize the temporal profile of each pixel and train a one-dimensional network to detect the brief disturbance that a passing target causes at a low signal-to-noise ratio (SNR). DeepPro (Li et al., 2026), the current state of the art, recasts detection as anomaly detection in this profile. Its learned kernels span time only, and max pooling adds coarser spatial scales. Neither method learns a spatial filter. TRACE learns one over its template responses.

3 METHOD

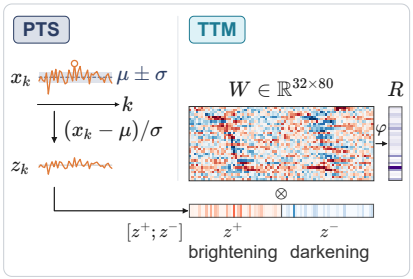
Given single-channel infrared frames scaled to $[0, 1]$, TRACE reports the image positions of the small targets in every frame. Following DeepPro (Li et al., 2026), for output frame t, TRACE reads the 40 frames $\mathcal{W}_t = \{t - 19, \ldots, t + 20\}$. It outputs a score map s_t at native resolution. Let $\mathbf{z}_t(\mathbf{q}) \in \mathbb{R}^{40}$ denote the standardized profile of pixel $\mathbf{q}$ over the window. The score of pixel $\mathbf{p}$ is then

$$s_t(\mathbf{p}) = f_\theta\big(R_t\big|_{\mathcal{N}(\mathbf{p})}\big), \qquad R_t(\mathbf{q}) = \psi_\theta\big(\mathbf{z}_t(\mathbf{q})\big), \tag{1}$$

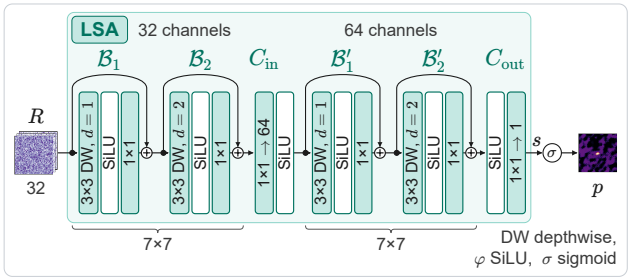
where ψ_θ is temporal template matching (TTM), the only trained map that reads the time axis, and f_θ is local spatial aggregation (LSA), which scores $\mathbf{p}$ from its neighborhood $\mathcal{N}(\mathbf{p})$. If $\mathcal{N}(\mathbf{p})$ contained only $\mathbf{p}$, each pixel would be scored from its own profile alone. TTM and LSA have 2,592 and



(a) Overall pipeline



(b) Inside PTS and TTM



(c) Inside LSA

Figure 2: TRACE. (a) Pipeline on HiNo test frames $t-19, \ldots, t+20$; red circle: target; green box: detection. (b) Inside PTS and TTM; heat map: templates $\mathbf{W}$ of HiNo seed 1103. (c) Inside LSA.

14,529 parameters, 17,121 in total, and require 16,640 multiply-adds per pixel and output frame. As Figure 2 shows, they follow a parameter-free per-pixel temporal standardization (PTS), and the probability map $p = \sigma(s)$, with σ the sigmoid, is thresholded to give the detections.

3.1 Per-pixel temporal standardization (PTS)

PTS standardizes each pixel over time, against the other frames of its window. This step removes the part of the pixel value that is constant over the window, such as a fixed offset, which Appendix C.1 measures in the heavy noise of HiNo. For a pixel $\mathbf{p}$, let x_k be its value in frame k, and let μ_t and σ_t be the mean and standard deviation of x_k over the frames of $\mathcal{W}_t \setminus \{t\}$ that lie inside the sequence. Excluding frame t keeps a target at t out of its own reference. The standardized profile is then $z_k = \operatorname{clip}\big((x_k - \mu_t)/\bar{\sigma}_t, -5, 5\big)$ for $k \in \mathcal{W}_t$, where $\bar{\sigma}_t = (\sigma_t^2 + \epsilon^2)^{1/2}$ and the floor $\epsilon = 1/255$ is one gray level.

3.2 Temporal template matching (TTM)

TTM reduces the standardized profile of each pixel to 32 template responses in a single step. Specifically, it first splits the profile $\mathbf{z} = (z_k)_{k\in\mathcal{W}_t}$ by sign into brightening and darkening channels, as in concatenated rectified linear units (Shang et al., 2016). This split gives $\mathbf{z}^{\pm} = [\mathbf{z}^+; \mathbf{z}^-] \in \mathbb{R}^{80}$ with $\mathbf{z}^+ = \max(\mathbf{z}, 0)$ and $\mathbf{z}^- = \max(-\mathbf{z}, 0)$ elementwise, and frames outside the sequence enter it as zeros. Next, TTM compares $\mathbf{z}^{\pm}$ with a bank of 32 learned templates that all pixels share,

$$R = \varphi\big(\mathbf{W}\mathbf{z}^{\pm} + \mathbf{b}\big), \qquad \mathbf{W} \in \mathbb{R}^{32\times 80}, \tag{2}$$

where φ is the SiLU (Elfwing et al., 2018) and the affine map is a 1×1 convolution over the 80 channels. Because of the sign split, each template $(\mathbf{w}^+, \mathbf{w}^-)$, one row of $\mathbf{W}$, computes $\mathbf{w}^{+\top}\mathbf{z}^+ + \mathbf{w}^{-\top}\mathbf{z}^- = \frac{1}{2}(\mathbf{w}^+ - \mathbf{w}^-)^\top\mathbf{z} + \frac{1}{2}(\mathbf{w}^+ + \mathbf{w}^-)^\top|\mathbf{z}|$, a signed linear response plus a term in $|\mathbf{z}|$. This term lets a template respond to a brightening and a darkening with the same sign. Setting $\mathbf{w}^- = -\mathbf{w}^+$ removes it and recovers TTM without the split.

3.3 Local spatial aggregation (LSA)

The template responses of neighboring pixels rise successively along the track of a moving target, a pattern that an isolated noise fluctuation does not produce. To collect this evidence, LSA aggregates neighboring responses in two depthwise stages of residual blocks $\mathcal{B}_d(X) = X + P\big(\varphi(D_d(X))\big)$, where D_d is a 3×3 depthwise convolution (Chollet, 2017) with dilation d and P is a 1×1 convolution. Each pair of blocks with dilations 1 and 2 covers 7×7. The first stage applies $\mathcal{B}_1$ and $\mathcal{B}_2$

to the 32 template maps R. A 1×1 convolution C_{in} expands its output to 64 channels, the second stage applies $\mathcal{B}'_1$ and $\mathcal{B}'_2$, and a 1×1 convolution C_{out} reduces the result to one score per pixel,

$$s = C_{\text{out}}\Big(\varphi\big(\mathcal{B}'_2(\mathcal{B}'_1(\varphi(C_{\text{in}}\,\mathcal{B}_2(\mathcal{B}_1(R)))))\big)\Big). \tag{3}$$

The neighborhood $\mathcal{N}(\mathbf{p})$ contains the track of a target over several frames around t, since NUDT-MIRSDT test targets move 0.65 pixels per frame at the median and 0.77 at the 90th percentile. Because the network is fully convolutional and has no normalization layers, a score depends only on its neighborhood and window, and the same weights apply at any resolution.

3.4 Training objective

The loss treats each target individually, so that an easy target cannot offset a weak one. For every target, $\mathcal{L}_{\text{tgt}}$ raises a soft maximum of s over the 5×5 window around its annotated pixel, in the multiple-instance sense (Dietterich et al., 1997), and $\mathcal{L}_{\text{core}}$ raises the score at that pixel. This supervision marks where each target is, not its outline, in line with the benchmarks, which score detections by position. A localization term $\mathcal{L}_{\text{loc}}$ favors the core over the rim, and pixels more than five pixels from every target receive a binary cross-entropy term $\mathcal{L}_{\text{bg}}$. In addition, the loss holds each group of sequences to a false-alarm budget, expressed in false alarms per 10^5 pixels. To this end, two budget-aware terms use the highest background scores of a batch, one for each false-alarm pixel that the budget allows on its frames. The term $\mathcal{L}_{\text{tail}}$ pushes their mean below zero, and $\mathcal{L}_{\text{rank}}$ pushes each target's soft maximum above that mean by a margin. In total,

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{tgt}} + 0.15\,\mathcal{L}_{\text{core}} + 0.1\,\mathcal{L}_{\text{loc}} + 0.25\,\mathcal{L}_{\text{bg}} + 0.5\,\mathcal{L}_{\text{tail}} + 0.25\,\mathcal{L}_{\text{rank}}. \tag{4}$$

Appendix A defines the full loss, and Appendix B lists the optimizer and schedule.

4 Experiments

4.1 Setup

NUDT-MIRSDT (Li et al., 2023b) has 120 infrared sequences whose targets are annotated with masks and with one-pixel centroids, and we train and evaluate on the centroids. Its official split has 80 training and 20 test sequences and leaves the other 20 unused. The eight test sequences with $\text{SNR} \leq 3$ also form a separate group. NUDT-MIRSDT-HiNo (Li et al., 2026) adds heavy noise to the same sequences. We call NUDT-MIRSDT and its $\text{SNR} \leq 3$ subset the clean groups and HiNo the heavy-noise group. With the same architecture and optimizer, we train one model on NUDT-MIRSDT for both clean groups and one on HiNo. We also evaluate on IRDST-simulation (Sun et al., 2023) with its official split, training the same network on the released masks.

We report the detection rate P_d, the fraction of targets with a detection within two pixels, and the false-alarm rate F_a, the number of false alarms per 10^5 pixels, counting only those more than five pixels from every target. So that every method is scored in the same way, all benchmark results in this section, unless stated otherwise, use the raw readout of the quoted rows of Table 1. This readout thresholds every pixel of the map and counts each false pixel, and the quoted rows apply it at threshold 0.5. A point readout, which keeps only the 5×5 local maxima of the map and so counts each response once, is used only where stated. On NUDT-MIRSDT, HiNo and IRDST-simulation, every DeepPro result is quoted from its paper or computed from its released weights. At threshold 0.5, these weights reproduce all six P_d and F_a cells and both AUCs of DeepPro's row in Table 1.

We take as budgets the false-alarm rates that DeepPro reports: 0.52, 0.72 and 1.76 per 10^5 pixels on $\text{SNR} \leq 3$, NUDT-MIRSDT and HiNo, respectively. At these budgets, we read TRACE, its ablation arms and the released DeepPro weights. In every group, each model uses the lowest threshold whose test false-alarm rate stays within the budget. Each result of TRACE is the mean over its five training seeds, and each ablation arm is retrained from scratch with the recipe and seeds of its reference model. Appendix C states the protocol in full, and Appendix E.1 reports TRACE with its point readout at thresholds fixed on the training sequences. These thresholds keep every group within its budget, and TRACE then detects 79.09% of the heavy-noise targets at $F_a = 0.92$.

We time TRACE and the 15 compared deep methods with public code end to end on one RTX 4090 D at 256×256 pixels in FP32 with batch size 1, taking the median of three passes, and count

Table 1: Comparison with 24 published methods on NUDT-MIRSDT, its $\mathrm{SNR} \leq 3$ test sequences and NUDT-MIRSDT-HiNo: P_d (%), F_a (10^{-5} per pixel), AUC, #Params (FP32 model size in MB) and FPS. Bold: best; underline: second best; *SF*/*MF*: single-/multi-frame.

		Methods	$\mathrm{SNR} \leq 3$		NUDT-MIRSDT			NUDT-MIRSDT-HiNo			#Params (MB)	FPS
			P_d	F_a	P_d	F_a	AUC	P_d	F_a	AUC		
Traditional Methods	*MF*	MSLSTIPT (Sun et al., 2020)	4.16	21.70	18.97	15.37	0.9404	3.93	73.76	0.9185	–	0.17
		IMNN-LWEC (Luo et al., 2022)	0.00	7.22	26.43	10.74	0.6734	4.97	83.28	0.5394	–	30.31
		SRSTT (Li et al., 2023a)	69.94	6.12	90.63	3.35	**0.9989**	4.34	55.04	0.5358	–	0.06
		4DST-BTMD (Luo et al., 2023a)	41.58	23.45	44.77	74.95	0.8488	4.80	77.29	0.6651	–	26.09
		STRL-LBCM (Luo et al., 2023b)	5.48	85.53	19.03	34.05	0.5972	2.55	77.78	0.5238	–	0.87
		4D-TR (Wu et al., 2023)	55.77	2.55	55.70	3.19	0.9946	4.63	120.16	0.6633	–	0.36
		4D-TT (Wu et al., 2023)	24.95	1.67	30.89	3.21	0.8287	6.94	73.18	0.5347	–	10.82
		NFTDGSTV (Liu et al., 2023)	1.51	32.31	13.77	35.32	0.8613	11.56	43.16	0.9524	–	0.58
Deep Learning Methods	*SF*	ACM (Dai et al., 2021a)	7.75	22.88	51.53	17.52	0.9298	0	–	–	1.592	300.2
		ALCNet (Dai et al., 2021b)	3.97	37.10	52.57	25.50	0.8435	36.09	91.99	0.9326	3.457	252.7
		Res-UNet (Xiao et al., 2018)	15.83	30.32	63.27	40.83	0.9198	35.51	22.55	0.9391	3.656	<u>463.5</u>
		DNA-Net (Li et al., 2022)	23.74	19.23	67.38	15.07	0.8843	49.16	60.98	0.9373	18.787	81.3
		ISNet (Zhang et al., 2022)	17.96	8.53	65.99	19.25	0.9123	28.40	90.17	0.9224	4.363	137.6
		UIUNet (Wu et al., 2022)	15.12	17.46	61.25	14.42	0.9436	43.67	28.87	0.9246	202.152	24.8
		AGPCNet (Zhang et al., 2023)	31.76	176.38	55.47	85.56	0.9443	42.452	13655.20	0.7348	49.442	25.9
		MSHNet (Liu et al., 2024)	2.46	78.07	36.78	41.91	0.7966	21.81	44.92	0.6748	16.262	110.6
		SCTransNet (Yuan et al., 2024)	23.63	122.24	62.81	74.20	0.9320	20.65	35.37	0.8283	45.304	52.7
		RPCANet (Wu et al., 2024)	30.06	81.21	61.13	41.28	0.8694	21.81	198.14	0.8786	2.720	172.2
		ILNet (Li et al., 2025)	17.39	55.89	57.90	34.09	0.7572	34.11	56.53	0.6551	6.322	67.5
	MF	Res-U+DTUM (Li et al., 2023b)	91.68	2.37	97.46	3.00	0.9967	43.90	4.86	0.9413	1.193	200.5
		STDMANet (Yan et al., 2023)	92.82	2.88	96.59	3.40	0.9908	51.65	<u>1.95</u>	0.8766	47.518	18.66
		Res-U+RFR (Ying et al., 2025a)	79.39	4.15	93.35	1.95	0.9657	55.64	111.23	0.7839	4.096	230.1
		DQAligner (Deng et al., 2026)	81.29	**0.37**	94.22	**0.15**	0.9419	33.41	2.95	0.8022	2.410	20.4
		DeepPro (Li et al., 2026)	<u>95.84</u>	<u>0.52</u>	<u>98.50</u>	<u>0.72</u>	0.9973	<u>59.17</u>	**1.76**	<u>0.9638</u>	<u>0.197</u>	261.6
		TRACE (ours)	**95.99**	<u>0.52</u>	**99.36**	<u>0.72</u>	<u>0.9979</u>	**76.34**	**1.76**	**0.9677**	**0.068**	**711.5**

Table 2: Comparison with five published multi-frame methods on the 66 test sequences of IRDST-simulation: P_d (%), F_a (10^{-5} per pixel) and AUC. Compared rows are quoted from Table III of Li et al. (2026). Bold: best; underline: second best.

Method	P_d (%)	F_a	AUC
Res-U+DTUM *(TNNLS'23)*	99.48	**0.00**	0.9730
STDMANet *(TGRS'23)*	98.88	**0.00**	0.9978
Res-U+RFR *(TGRS'25)*	98.90	9.36	0.9950
DQAligner *(TGRS'26)*	96.86	**0.00**	0.9742
DeepPro *(TPAMI'26)*	<u>99.85</u>	<u>0.13</u>	<u>0.9993</u>
TRACE (ours)	**99.99**	**0.00**	**0.9996**

their parameters as described in Appendix F. Table 1 quotes the frame rates of the eight traditional methods and STDMANet.

4.2 Comparison with published methods

Table 1 compares TRACE with 24 published methods, whose detection cells and model sizes are quoted from Li et al. (2026). On the two clean groups, TRACE detects about as many targets as the best compared method, and the released DeepPro weights, read at the same budgets with the same readout, detect 96.03% and 98.61% (Table 4). On HiNo, TRACE detects 76.34% of the targets at $F_a = 1.76$, substantially more than every compared method. This lead is 17.2 percentage points over DeepPro, the strongest compared method, which detects 59.17% at the same rate. All five seeds of TRACE exceed DeepPro by 15.1 to 18.1 points. The other 23 compared methods detect at most 55.64%, and every F_a reported for them exceeds 1.76. TRACE thus dominates every compared method with a reported F_a in the Pareto sense: none detects as many targets, and none reports a lower F_a.

The heavy-noise lead over DeepPro is neither specific to the budget nor confined to a few test sequences. TRACE leads at every false-alarm rate of the threshold sweeps in Appendix E.2, from 0.12 to 20 per 10^5 pixels. It detects at least one percentage point more in 11 of the 20 test sequences of Table 8, and the other nine differ by less than one point. Resampling these sequences gives a 95% interval of 8.4 to 26.8 points for the 17.2-point lead.

Figure 3 illustrates how heavy noise changes the profile of the target pixel. In the clean rows, this profile shows a clear pulse, even where the target is barely visible in its frame. In the HiNo rows, however, the target pixel's value in the frame shown is no higher than the noise peaks of its profile. The two HiNo frames are among the 13 frames of Appendix D on which TRACE detects the target and the strongest compared method does not. A red circle marks the target in each input frame, and

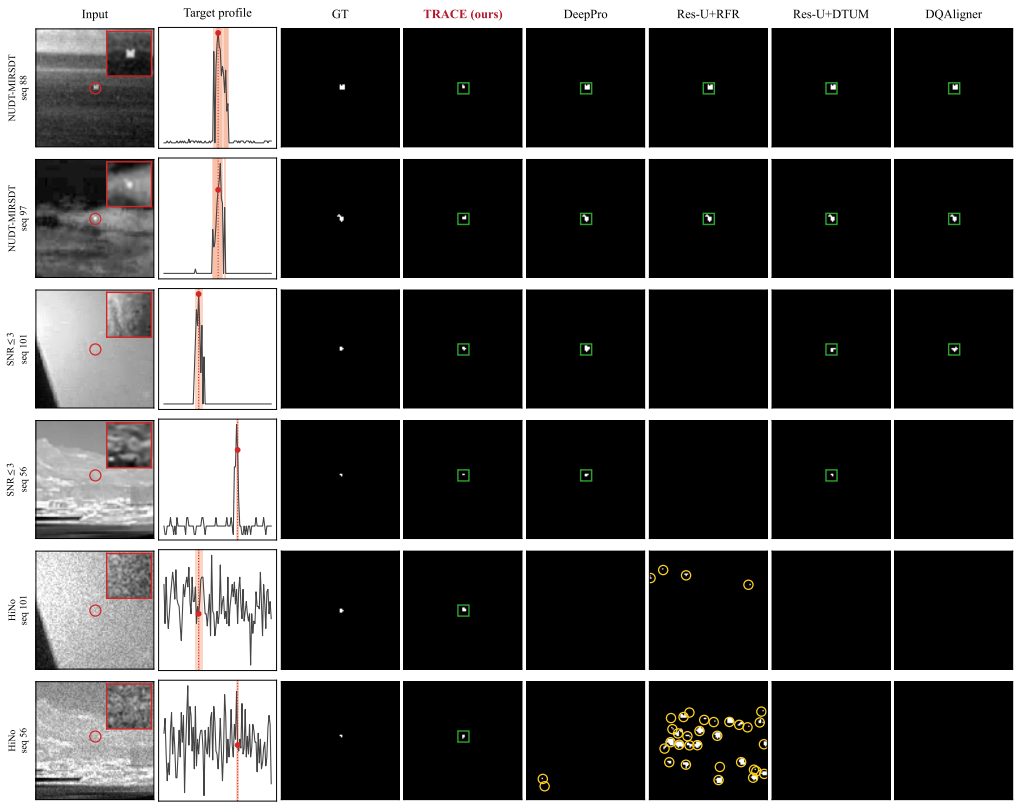


Figure 3: Detection maps of TRACE and four multi-frame methods on six test frames: two from NUDT-MIRSDT sequences with $\mathrm{SNR} > 3$, two from sequences with $\mathrm{SNR} \leq 3$, and the same two with HiNo noise. Second column: the target pixel's value over the sequence (shaded: frames in which the target covers it; dot: frame shown). Green box: detected target; yellow circle: false alarm.

an inset enlarges its surroundings. All methods, TRACE (seed 1103) included, are thresholded at 0.5, as in DeepPro (Li et al., 2026). GT shows the annotated masks, whereas TRACE marks target positions. Appendix E.6 describes the compared models that we retrained for this figure.

Table 2 extends the comparison to IRDST-simulation, which differs from NUDT-MIRSDT in its scenes and targets and has no added heavy noise. On this dataset, every compared multi-frame method detects at least 96.86% of the targets, whereas the ten single-frame methods reported by Li et al. (2026) detect 41.55% to 88.06%. We train TRACE with as many updates as DeepPro and read it at threshold 0.5, as in the quoted rows. In each of its five seeds, TRACE then detects 19,184 of the 19,185 targets with no false pixel, at least as many as every compared method. Finally, following DeepPro, we also validate TRACE on real thermal-infrared sequences of RGBT-Tiny (Ying et al., 2025b), selected by an annotation-based rule for small, moving and sparse targets. Trained on the same sequences and evaluated in the same way as DeepPro, with five seeds each, TRACE detects more of the test targets than DeepPro in every seed. Its lead thus extends from the benchmarks above to real thermal-infrared data. Appendix E.8 gives the rule, the split and the results.

4.3 Other Noise Models

Figure 4 compares TRACE and DeepPro, both with their HiNo weights and without retraining, under five further noise models generated from the clean frames of the 20 test sequences. H8 has the form of the HiNo noise, and W8 keeps only its white noise. H4 and H12 weaken and strengthen the white noise of H8, respectively, and D8 lets the per-pixel offsets of H8 drift over the sequence. Appendix E.4 defines these models and gives the full results. Across the generated noise models, TRACE leads by 16.8 to 29.0 percentage points, compared with 17.2 under the official HiNo noise.

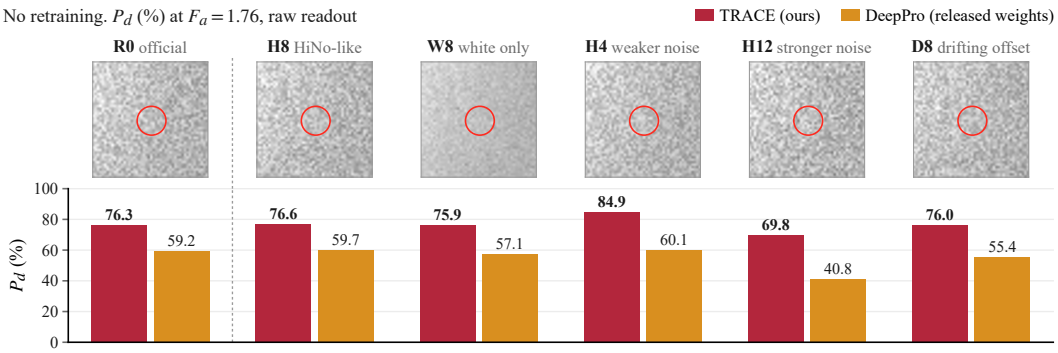


Figure 4: TRACE and DeepPro under HiNo noise (R0) and five generated noise models. Top: target (red circle) in frame 50 of test sequence 101, common gray scale. Bottom: P_d (%) at $F_a = 1.76$.

Table 3: Ablations on TRACE (top) and on an earlier version of TRACE (bottom): P_d (%) at the false-alarm budget of each group, as mean ± SD over five seeds and, in parentheses, the mean difference from the first row of each block, paired by seed.

Variant and modification	#Params	SNR ≤ 3	NUDT-MIRSDT	HiNo
TRACE (full model)	17,121	95.99 ± 0.25	99.36 ± 0.14	76.34 ± 1.27
TTM: output frame only templates read z_t^+ and z_t^-	14,625	87.79 ± 5.03 (-8.20)	91.08 ± 3.83 (-8.28)	31.61 ± 2.02 $(-44.73)^\dagger$
LSA: no neighborhood every depthwise kernel is 1×1	15,585	95.50 ± 0.25 $(-0.49)^\circ$	96.85 ± 0.25 (-2.51)	20.38 ± 11.36 $(-55.96)^\dagger$
Earlier version of TRACE, with one more scoring head and one more input channel of LSA				
32 templates, 40 frames	18,274	96.33 ± 0.34	99.33 ± 0.18	76.61 ± 1.19
16 templates	13,554	95.80 ± 0.41 $(-0.53)^*$	99.18 ± 0.28 $(-0.15)^\circ$	75.47 ± 1.18 (-1.15)
64 templates	30,786	96.41 ± 0.38 $(+0.08)^\circ$	99.39 ± 0.05 $(+0.06)^\circ$	76.58 ± 0.72 $(-0.03)^\circ$
20 frames window $t-9, \ldots, t+10$	16,994	93.08 ± 0.75 (-3.25)	97.42 ± 0.41 (-1.91)	72.60 ± 0.65 (-4.01)

Unmarked: mean drop in P_d of at least 1.0 point, with at least four of five seeds below the first row. $^\circ$ Mean change below 0.5 points. * Mean drop of 0.5 to 1.0 points. † Training loss at its initial value in every seed.

Its lead is largest under the strongest white noise, H12, with 69.83% against 40.83%. When the offsets drift over the sequence, in D8, TRACE detects 76.01% of the targets, about as many as under H8, whereas DeepPro drops from 59.75% to 55.41%, consistent with PTS recomputing the statistics of every pixel in each window. These results show that TRACE keeps its advantage under noise it was not trained on.

4.4 Ablation

TTM. In the arm with the output frame only, the templates read only z_t^+ and z_t^-, so that the other frames of the window enter only through the statistics of PTS. This arm lowers P_d by 8.20 and 8.28 points on SNR ≤ 3 and NUDT-MIRSDT. On HiNo its training fails, and P_d falls by 44.73 points. The templates thus need the profile of each pixel over the whole window, not only its value at t. Without the split of $\mathbf{z}$ by sign, TTM reads $\mathbf{z}$ instead of $\mathbf{z}^\pm$, and P_d changes by -0.19, -0.97 and $+0.01$ points.

Templates and window. We chose the number of templates and the window length on an earlier version of TRACE, which differs only by one more scoring head and one more input channel of LSA. Halving the templates to 16 lowers P_d by up to 1.15 points. Doubling them to 64 changes P_d by at most 0.08 points, but costs 68% more parameters and slows the network at 480×720 from

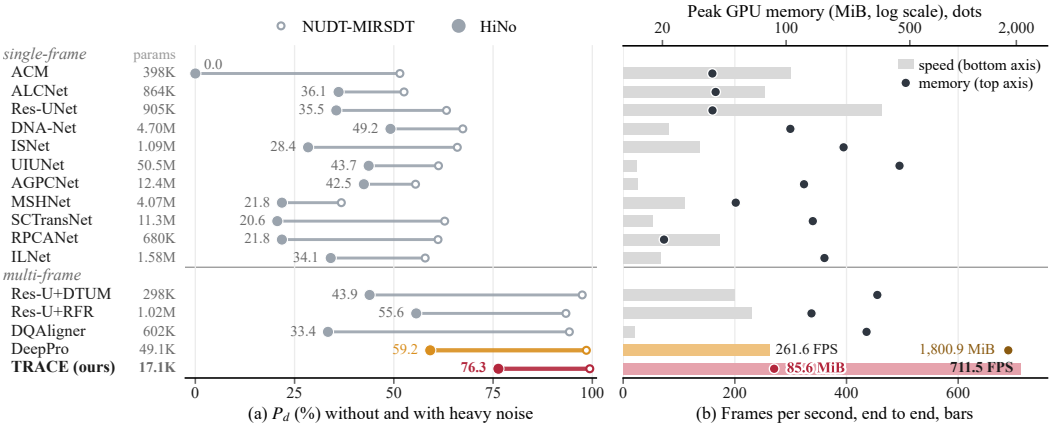


Figure 5: The 16 deep methods of Table 1 other than STDMANet, with their parameter counts. (a) P_d (%) on NUDT-MIRSDT (open dot) and HiNo (filled dot, value printed). (b) End-to-end frames per second (bars) and peak GPU memory (dots, log scale).

136 to 108 frames per second, so we keep 32 templates. Halving the window to 20 frames lowers P_d by 1.91 to 4.01 points in every seed, whereas 60 frames raise it by at most 0.89 points with 50% more frames, so 40 frames balance detection against cost.

LSA. The no-neighborhood arm reduces every depthwise kernel of LSA to 1×1, so that no unit reads a neighboring pixel. Its training fails on HiNo, where P_d falls by 55.96 points, whereas it falls by 0.49 and 2.51 points on $\mathrm{SNR} \leq 3$ and NUDT-MIRSDT. Without LSA, each pixel is judged from its own profile alone, which the heavy noise of HiNo makes unreliable. In the probe of Appendix E.7, a faint target is detected in 85% of trials when it moves through the neighborhood, against 27% amid random flashes and 24% with empty neighbors.

4.5 Speed and Memory

As Figure 5 shows, TRACE processes 711.5 frames per second with 85.6 MiB of peak memory, because TTM reads the time axis only once and every later layer operates on two-dimensional maps of at most 64 channels. With 17,121 parameters, it is the smallest of the 16 timed deep methods, the next being DeepPro with 49,129, and the fastest, ahead of Res-UNet at 463.5 frames per second. DeepPro processes 261.6 frames per second with 1,800.9 MiB of peak memory, so TRACE runs 2.7 times as fast with about a twentieth of its memory.

5 Conclusion

Under heavy noise, a small moving target is hardly visible in any single frame, and a noise pixel often changes more over time than the target pixel does. TRACE matches the profile of each pixel with shared temporal templates in one layer and scores the pixel from the responses of its neighborhood, where a target track gathers support and an isolated noise fluctuation does not. On HiNo, TRACE detects substantially more targets than each of the 24 compared methods, about 17 percentage points more than the strongest of them at the same false-alarm rate. Without retraining, it keeps a lead of 16.8 to 29.0 points over that method under five further noise models. On the clean data, it detects about as many targets as the best compared method, while it is the smallest and fastest of the timed deep methods. On IRDST-simulation, it detects at least as many targets as every compared method, and on real thermal-infrared sequences of RGBT-Tiny, it detects more targets than DeepPro in every seed. The ablations show that the time profile in TTM and the neighborhood of LSA each matter; without either, TRACE detects over 40 points fewer heavy-noise targets. TRACE keeps every map at native resolution, so a target of a few pixels is never lost to downsampling. Reading the time axis once with shared templates and deciding each pixel from its neighborhood thus makes infrared small moving target detection both robust to heavy noise and efficient.

AI USE STATEMENT

We used large language models to polish and help draft the text and to draft experiment and analysis code, which the authors revised and verified. The authors defined the research questions, the method and the experimental protocol, checked all code, results and claims, and take full responsibility for the content of the paper.

REPRODUCIBILITY STATEMENT

Section 3 and Appendices A and B specify the model, the loss and the training recipe. Section 4.1 and Appendix C describe the datasets, the splits and the evaluation protocol, including those of IRDST-simulation, and Appendix E.1 describes the point readout of TRACE and the procedure that sets its thresholds. Appendix D defines the hardest frames and the temporal excursions, and Appendix F gives the timing protocol and the changes we made to the released code of the compared methods. Appendix E reports the results of every seed. The supplementary material contains the model, inference and evaluation code, the trained weights of all five seeds on NUDT-MIRSDT and HiNo, and videos of the detections of TRACE on test sequences; the training code will be released upon acceptance.

REFERENCES

Yair Barniv. Dynamic programming solution for detecting dim moving targets. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21(1):144–156, 1985. doi: 10.1109/TAES.1985.310548.

C. L. Philip Chen, Hong Li, Yantao Wei, Tian Xia, and Yuan Yan Tang. A local contrast method for small infrared target detection. *IEEE Transactions on Geoscience and Remote Sensing*, 52(1): 574–581, 2014. doi: 10.1109/TGRS.2013.2242477.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, 2017.

Yimian Dai, Yiquan Wu, Fei Zhou, and Kobus Barnard. Asymmetric contextual modulation for infrared small target detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 950–959, 2021a.

Yimian Dai, Yiquan Wu, Fei Zhou, and Kobus Barnard. Attentional local contrast networks for infrared small target detection. *IEEE Transactions on Geoscience and Remote Sensing*, 59(11): 9813–9824, 2021b.

Chuiyi Deng, Yanyin Guo, Xiang Xu, Zhuoyi Zhao, Yixin Xia, Runxuan An, Junwei Li, and Antonio Plaza. Learning global dynamic query for large-motion infrared small target detection. *IEEE Transactions on Geoscience and Remote Sensing*, 64:5002016, 2026. doi: 10.1109/TGRS.2026.3657842.

Suyog D. Deshpande, Meng Hwa Er, Ronda Venkateswarlu, and Philip Chan. Max-mean and max-median filters for detection of small targets. In *Signal and Data Processing of Small Targets*, volume 3809, pp. 74–83, 1999.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.

Weiwei Duan, Luping Ji, Shipeng Lei, Sicheng Zhu, Jianghong Huang, and Mao Ye. CHAL: Causal-guided hierarchical anomaly-aware learning for moving infrared small target detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21357–21366, 2026.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Houzhang Fang, Shukai Guo, Qiuhuan Chen, Yi Chang, and Luxin Yan. Spatio-temporal context learning with temporal difference convolution for moving infrared small target detection. In *AAAI Conference on Artificial Intelligence*, volume 40, pp. 3840–3848, 2026. doi: 10.1609/aaai.v40i5.37385.

Chenqiang Gao, Deyu Meng, Yi Yang, Yongtao Wang, Xiaofang Zhou, and Alexander G. Hauptmann. Infrared patch-image model for small target detection in a single image. *IEEE Transactions on Image Processing*, 22(12):4996–5009, 2013.

Weihua Gao, Wenlong Niu, Pengcheng Wang, Yanzhao Li, Chunxu Ren, Xiaodong Peng, and Zhen Yang. Moving point target detection based on temporal transient disturbance learning in low SNR. *Remote Sensing*, 15(10):2523, 2023. doi: 10.3390/rs15102523.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014. doi: 10.1109/CVPR.2014.223.

Boyang Li, Chao Xiao, Longguang Wang, Yingqian Wang, Zaiping Lin, Miao Li, Wei An, and Yulan Guo. Dense nested attention network for infrared small target detection. *IEEE Transactions on Image Processing*, 32:1745–1758, 2022.

Haoqing Li, Jinfu Yang, Runshi Wang, and Yifei Xu. ILNet: Low-level matters for salient infrared small target detection. *IEEE Transactions on Aerospace and Electronic Systems*, 61(4):8306–8318, 2025. doi: 10.1109/TAES.2025.3544613.

Jie Li, Ping Zhang, Lingyi Zhang, and Zhiyuan Zhang. Sparse regularization-based spatial–temporal twist tensor model for infrared small target detection. *IEEE Transactions on Geoscience and Remote Sensing*, 61:5000417, 2023a. doi: 10.1109/TGRS.2023.3234608.

Ruojing Li, Wei An, Chao Xiao, Boyang Li, Yingqian Wang, Miao Li, and Yulan Guo. Direction-coded temporal U-shape module for multiframe infrared small target detection. *IEEE Transactions on Neural Networks and Learning Systems*, 36(1):555–568, 2023b.

Ruojing Li, Wei An, Yingqian Wang, Xinyi Ying, Yimian Dai, Longguang Wang, Miao Li, Yulan Guo, and Li Liu. Probing deep into temporal profile makes the infrared small target detector much better. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(8):10157–10175, 2026. doi: 10.1109/TPAMI.2026.3683258.

Qiankun Liu, Rui Liu, Bolun Zheng, Hongkui Wang, and Ying Fu. Infrared small target detection with scale and location sensitivity. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17490–17499, 2024.

Ting Liu, Jungang Yang, Boyang Li, Yingqian Wang, and Wei An. Infrared small target detection via nonconvex tensor Tucker decomposition with factor prior. *IEEE Transactions on Geoscience and Remote Sensing*, 61:5617317, 2023. doi: 10.1109/TGRS.2023.3298192.

Yuan Luo, Xiaorun Li, Shuhan Chen, Chaoqun Xia, and Liaoying Zhao. IMNN-LWEC: A novel infrared small target detection based on spatial–temporal tensor model. *IEEE Transactions on Geoscience and Remote Sensing*, 60:5004022, 2022. doi: 10.1109/TGRS.2022.3230051.

Yuan Luo, Xiaorun Li, Shuhan Chen, and Chaoqun Xia. 4DST-BTMD: An infrared small target detection method based on 4-D data-sphered space. *IEEE Transactions on Geoscience and Remote Sensing*, 62:5000520, 2023a. doi: 10.1109/TGRS.2023.3347420.

Yuan Luo, Xiaorun Li, Yunfeng Yan, and Chaoqun Xia. Spatial-temporal tensor representation learning with priors for infrared small target detection. *IEEE Transactions on Aerospace and Electronic Systems*, 59(6):9598–9620, 2023b. doi: 10.1109/TAES.2023.3320635.

Irving S. Reed, Robert M. Gagliardi, and Larry B. Stotts. Optical moving target detection with 3-D matched filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 24(4):327–336, 1988. doi: 10.1109/7.7174.

Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning*, pp. 2217–2225, 2016.

Tushar Soni, James R. Zeidler, and Walter H. Ku. Performance evaluation of 2-D adaptive prediction filters for detection of small objects in image data. *IEEE Transactions on Image Processing*, 2(3): 327–340, 1993.

Heng Sun, Junxiang Bai, Fan Yang, and Xiangzhi Bai. Receptive-field and direction induced attention network for infrared dim small target detection with a large-scale dataset IRDST. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–13, 2023.

Yang Sun, Jungang Yang, and Wei An. Infrared dim and small target detection via multiple subspace learning and spatial-temporal patch-tensor model. *IEEE Transactions on Geoscience and Remote Sensing*, 59(5):3737–3752, 2020.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018. doi: 10.1109/CVPR.2018.00675.

Fengyi Wu, Hang Yu, Anran Liu, Junhai Luo, and Zhenming Peng. Infrared small target detection using spatiotemporal 4-D tensor train and ring unfolding. *IEEE Transactions on Geoscience and Remote Sensing*, 61:5002922, 2023. doi: 10.1109/TGRS.2023.3288024.

Fengyi Wu, Tianfang Zhang, Lei Li, Yian Huang, and Zhenming Peng. RPCANet: Deep unfolding RPCA based infrared small target detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4809–4818, 2024.

Xin Wu, Danfeng Hong, and Jocelyn Chanussot. UIU-Net: U-Net in U-Net for infrared small object detection. *IEEE Transactions on Image Processing*, 32:364–376, 2022.

Xiao Xiao, Shen Lian, Zhiming Luo, and Shaozi Li. Weighted Res-UNet for high-quality retina vessel segmentation. In *International Conference on Information Technology in Medicine and Education*, pp. 327–331, 2018. doi: 10.1109/ITME.2018.00080.

Yinghui Xing, Donghao Chu, Shizhou Zhang, and Di Xu. Temporal-emerged prompting for Segment Anything in multiframe infrared small target detection. In *International Conference on Machine Learning*, 2026. arXiv:2606.27655.

Puti Yan, Runze Hou, Xuguang Duan, Chengfei Yue, Xin Wang, and Xibin Cao. STDMANet: Spatio-temporal differential multiscale attention network for small moving infrared target detection. *IEEE Transactions on Geoscience and Remote Sensing*, 61:5602516, 2023. doi: 10.1109/TGRS.2023.3241311.

Xinyi Ying, Li Liu, Zaiping Lin, Yangsi Shi, Yingqian Wang, Ruojing Li, Xu Cao, Boyang Li, Shilin Zhou, and Wei An. Infrared small target detection in satellite videos: A new dataset and a novel recurrent feature refinement framework. *IEEE Transactions on Geoscience and Remote Sensing*, 63:5002818, 2025a. doi: 10.1109/TGRS.2025.3542368.

Xinyi Ying, Chao Xiao, Wei An, Ruojing Li, Xu He, Boyang Li, Xu Cao, Zhaoxu Li, Yingqian Wang, Mingyuan Hu, Qingyu Xu, Zaiping Lin, Miao Li, Shilin Zhou, Li Liu, and Weidong Sheng. Visible-thermal tiny object detection: A benchmark dataset and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):6088–6096, 2025b. doi: 10.1109/TPAMI.2025.3544621.

Shuai Yuan, Hanlin Qin, Xiang Yan, Naveed Akhtar, and Ajmal Mian. SCTransNet: Spatial-channel cross transformer network for infrared small target detection. *IEEE Transactions on Geoscience and Remote Sensing*, 62:5002615, 2024. doi: 10.1109/TGRS.2024.3383649.

Mingjin Zhang, Rui Zhang, Yuxiang Yang, Haichen Bai, Jing Zhang, and Jie Guo. ISNet: Shape matters for infrared small target detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 877–886, 2022.

Mingjin Zhang, Yuanjun Ouyang, Fei Gao, Jie Guo, Qiming Zhang, and Jing Zhang. MOCID: Motion context and displacement information learning for moving infrared small target detection. In *AAAI Conference on Artificial Intelligence*, volume 39, pp. 10022–10030, 2025. doi: 10.1609/aaai.v39i10.33087.

Tianfang Zhang, Lei Li, Siying Cao, Tian Pu, and Zhenming Peng. Attention-guided pyramid context networks for detecting infrared small target under complex background. *IEEE Transactions on Aerospace and Electronic Systems*, 59(4):4250–4261, 2023. doi: 10.1109/TAES.2023.3238703.

A TRAINING LOSS

The loss of Section 3.4 is computed on the score map s of each supervised output frame. For a frame, let $M_1, \dots, M_J$ be the 8-connected components of the centroid annotation on NUDT-MIRSDT and HiNo, one pixel per target, and of the released target masks on IRDST-simulation. For target j, the hit region H_j is M_j dilated by a 5×5 square (Chebyshev radius 2, the hit tolerance of the evaluation). The core C_j contains the pixels of M_j within Chebyshev distance 1 of the pixel of M_j closest to its centroid, and the rim is $E_j = H_j \setminus M_j$. Negatives $\mathcal{N}$ are the pixels outside the union of all M_j dilated by an 11×11 square (Chebyshev radius 5, the exclusion radius of the evaluation). The soft maximum of s over a set S is

$$m(S) = \kappa\Big(\log \sum_{q \in S} \exp\big(s_q/\kappa\big) - \log|S| \Big), \qquad \kappa = 0.25, \tag{5}$$

which equals s_q when all values in S are equal and approaches the maximum as $\kappa \to 0$. With $\mathrm{sp}(x) = \log(1 + e^x)$ and means taken over all targets of the batch,

$$\mathcal{L}_{\mathrm{tgt}} = \mathrm{mean}_j\ \mathrm{sp}\big(-m(H_j)\big), \qquad \mathcal{L}_{\mathrm{core}} = \mathrm{mean}_j\ \mathrm{sp}\big(-m(C_j)\big), \tag{6}$$

$$\mathcal{L}_{\mathrm{loc}} = \mathrm{mean}_j\ \mathrm{sp}\big(m(E_j) - m(C_j) + 0.25\big), \qquad \mathcal{L}_{\mathrm{bg}} = \mathrm{mean}_{q \in \mathcal{N}}\ \mathrm{sp}(s_q), \tag{7}$$

where $\mathcal{L}_{\mathrm{loc}}$ runs over targets with a nonempty rim and $\mathcal{L}_{\mathrm{bg}}$ over the negatives of all frames of the batch. The budget terms are computed per evaluation group g. On NUDT-MIRSDT a training sequence belongs to the NUDT-MIRSDT group and, when its proxy signal-to-noise ratio is at most 3, also to the $\mathrm{SNR} \le 3$ group. The proxy is the median, over the annotated target pixels $\mathbf{p}$ in all frames of the sequence, of $|I(\mathbf{p}) - \mathrm{med}_A| / (1.4826\ \mathrm{MAD}_A + 1)$, where I is the frame in gray levels and A is the ring of pixels at Euclidean distance 4 to 8 from $\mathbf{p}$. It selects 34 of the 80 training sequences, numbered 6, 13, 14, 16, 19, 22, 23, 24, 25, 26, 28, 29, 31, 33, 34, 35, 37, 42, 45, 49, 50, 51, 54, 60, 62, 64, 66, 69, 70, 77, 80, 81, 82 and 83. The same 34 sequences form the $\mathrm{SNR} \le 3$ group when the frozen training threshold of Appendix E.1 is set. On HiNo every training sequence belongs to the single HiNo group. Let β_g be the budget of group g in false-alarm pixels per 10^5 pixels, n_g the number of pixels of the batch frames in g, and $\mathcal{N}_g$ their negatives. With $k_g = \lceil \beta_g n_g / 10^5 \rceil$ and η_g the mean of the k_g largest scores in $\mathcal{N}_g$,

$$\mathcal{L}_{\mathrm{tail}} = \mathrm{mean}_g\ \mathrm{sp}(\eta_g), \qquad \mathcal{L}_{\mathrm{rank}} = \mathrm{mean}_g\ \mathrm{mean}_{j \in g}\ \mathrm{sp}\big(\eta_g - m(H_j) + 0.5\big). \tag{8}$$

Here $j \in g$ runs over the targets in the batch frames of g. The total loss is the weighted sum in equation 4. The network has no auxiliary head, and the loss has no other term.

B IMPLEMENTATION AND TRAINING DETAILS

We train with AdamW at a learning rate of 3×10^{-4} and a weight decay of 10^{-4}, and clip gradients at norm 5. The learning rate of epoch $e = 1, 2, \dots$ is $10^{-5} + \frac{1}{2}(3 \times 10^{-4} - 10^{-5})\big(1 + \cos(\pi(e-1)/96)\big)$, a cosine decay toward 10^{-5} with a horizon of 96 epochs. Training runs for 8 epochs on NUDT-MIRSDT and for 48 on HiNo, one model per benchmark, so it covers only the first part of the decay, and the learning rate of the last epoch is 2.96×10^{-4} and 1.60×10^{-4}, respectively. Each epoch visits the 80 training sequences in an order drawn anew for that epoch and processes them in groups of four. Each group yields 13 updates, whose output frames are frames 1 to 8, 9 to 16 and so on up to 89 to 96 of each of the four 100-frame sequences, and frames 97 to 100 in the last update, so every frame of every training sequence is supervised once per epoch. An epoch thus has 260 updates, and training has 2,080 updates on NUDT-MIRSDT and 12,480 on HiNo. Every arm of Table 3 is trained with this schedule.

In every update each sequence of the group contributes its output frames together with the frames of their 40-frame windows that lie inside the sequence, all cropped to 128×128 pixels at one position per sequence. If the middle output frame, the fifth of eight or the third of four in the last update, contains a target, then with probability 0.75 the crop is centered on a target pixel of that frame drawn at random, shifted in each coordinate by an offset drawn uniformly between -21.3 and 21.3 pixels, a sixth of the crop size, and moved back inside the image where it extends beyond it. Otherwise its position is drawn uniformly among the positions at which it lies inside the image. We use no augmentation. The model of the last epoch is kept, and no epoch is selected. Every configuration

is trained with the seeds 1103, 2207, 3301, 4409 and 5501, and each run uses one RTX 4090 D, in FP32 with deterministic cuDNN, Python 3.12 and PyTorch 2.6.0 built for CUDA 12.4. Over the five seeds, training TRACE takes 236 to 248 s on NUDT-MIRSDT and 1,353 to 1,442 s on HiNo, with a peak of $1{,}358.0$ MiB of allocated GPU memory.

The window of 40 frames, the 32 templates and the 64 channels of the second stage of LSA were set when the first version of this model was designed and were kept in every later version. The numbers of epochs were chosen for that first version, which had 42,251 parameters and the same window, optimizer and schedule. On each of the four folds of the training sequences it was trained for 96 epochs and validated after epochs 8, 16, 24, 32, 48, 64 and 96 by the partial AUC at low false-alarm rates on the held-out sequences, weighted three to one between the data as given and the same data with Gaussian noise of standard deviation 4 gray levels added. We took the median of the four best epochs, rounded up. On NUDT-MIRSDT all four folds were best at the first check, epoch 8, and on HiNo the best epochs were 32, 48, 48 and 96, which gives 48. The horizon of the cosine decay remained at the 96 epochs of these runs when training was shortened. That first version, with its split of the window into 19 past frames, the current frame and 20 future frames, was one of eight variants trained together. On the training folds it outperformed the causal variant, which reads the frames $t-39,\dots,t$, in every fold of all three groups, and under heavy noise it was the best of the eight variants in every fold. Windows of 20 and 60 frames and 16 and 64 templates (Section 4.4) and second-stage widths of 16 and 32 were later explored on predecessors of the final model, and none of the values above was changed.

On IRDST-simulation TRACE is trained on 225 of the 250 official training sequences. Every tenth of the 250 in the order of their identifiers, 25 in all, is held out and used only to calibrate a threshold for the point readout, which Table 2 does not use. Training is counted in updates rather than epochs. The number of updates is computed from the defaults of DeepPro's released training script, 32 epochs and batch size 4, and the sampling rate of 0.03 that it documents for this dataset: 0.03 of the 73,585 training windows that its loader forms from the 225 sequences is 2,208 after rounding, and 32 epochs of 2,208 windows in batches of four give 17,664 updates. Each update draws four of the 225 sequences uniformly at random, with replacement, and places the eight output frames at a random position at which the 40-frame windows of all of them lie inside the sequence. Crops follow the rule above, and the learning rate follows the same cosine form, computed per update, from 3×10^{-4} toward 10^{-5} over the 17,664 updates. The loss has a single group with a false-alarm budget of 0.13 per 10^5 pixels, the rate DeepPro reports on this dataset. Its target components, and the target pixels on which crops are centered, come from the released target masks, while the evaluation reduces each component to its centroid pixel. A run takes 2,166 to 2,253 s over the five seeds.

C Datasets, splits and evaluation protocol

NUDT-MIRSDT and NUDT-MIRSDT-HiNo use the official split, with 80 training and 20 test sequences. The eight test sequences with $\mathrm{SNR}\le 3$ form a third evaluation group, and the $\mathrm{SNR}\le 3$ group of the training sequences is defined in Appendix A. In the sequence numbering of the dataset, the training sequences are 1 to 84 except 47, 56, 59 and 76, and the test sequences are 47, 56, 59, 76, 85 to 97, 101, 105 and 119. The test sequences hold 1,729 targets in each of NUDT-MIRSDT and HiNo. The $\mathrm{SNR}\le 3$ group consists of the test sequences 47, 56, 59, 76, 92, 101, 105 and 119 and holds 529 of the NUDT-MIRSDT targets. The frame size differs between the test sequences but not within one: the frames are 161 to 261 pixels high and 233 to 379 pixels wide and hold $66{,}155$ pixels on average, so the heavy-noise budget allows about 1.2 false alarms per frame. IRDST-simulation uses its official split, whose 66 test sequences hold 19,185 targets. In both datasets targets are the 8-connected components of the centroid annotation, one pixel per target. On the centroid annotation of the 20 test sequences, which HiNo shares, a target moves a median of 0.65 pixels per frame, measured as $\|\mathbf{c}_{t+6}-\mathbf{c}_{t-6}\|/12$ for its centroid $\mathbf{c}_t$, with a 90th percentile of 0.77 and a maximum of 0.89. Over the 40-frame window its centroid moves a median of 25 pixels.

A target is counted as a hit when a detection lies within Chebyshev distance 2 of one of its pixels. False alarms are counted only at pixels farther than Chebyshev distance 5 from every target pixel, and F_a is their number per 10^5 pixels. Under the raw readout every pixel that reaches the threshold is a detection. Under the point readout of Section 4.1 only the 5×5 local maxima of the probability map that reach the threshold are detections, so a cluster of false pixels is counted once, and 8-connected

local maxima of equal value are counted as one detection, placed at the one of them nearest their mean position. TRACE reports detections as points, not masks, so we report no IoU. The budgets are the false-alarm rates DeepPro reports: 0.52, 0.72 and 1.76 per 10^5 pixels on $\mathrm{SNR} \leq 3$, NUDT-MIRSDT and HiNo. A row read at the budget uses, for each model and seed, the lowest threshold that keeps the test false-alarm rate within DeepPro's rate for that group. A row read at the frozen training threshold uses the threshold of Appendix E.1 unchanged. AUC is computed over the 27 thresholds of DeepPro's evaluation code, namely 0, 10^{-20}, 10^{-10}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.1, 0.2, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.8, 0.85, 0.9, 0.95, 0.99 and 1. Res-U+DTUM and DQAligner read the current frame and the four before it, and Res-U+RFR carries its features forward in time, so these three methods are causal. DeepPro maps each 40-frame block to all of its outputs at once, so every output except the last of a block also depends on later frames.

The 24 compared methods of Table 1 are those of Table II of DeepPro, which reports each of them on NUDT-MIRSDT, on its $\mathrm{SNR} \leq 3$ test sequences and on HiNo. Their detection cells are copied from that table, and we did not retrain or rerun any of these methods to obtain them.

C.1 Noise statistics of NUDT-MIRSDT-HiNo

We measure the noise statistics of NUDT-MIRSDT-HiNo on the 80 training sequences, each of 100 frames. At a pixel of frame k, let x_k be the NUDT-MIRSDT value and $\tilde{x}_k$ the HiNo value, both in gray levels, and let $\Delta_k = \tilde{x}_k - x_k$. A pixel is treated as background when in every frame it lies farther than Chebyshev distance 3 from every annotated target pixel and its HiNo value is neither 0 nor 255. The 80 sequences hold 5,028,962 background pixels, and every statistic below is computed over them. Apart from quantization to integer gray levels, the noise model of NUDT-MIRSDT-HiNo (Li et al., 2026) gives $\Delta_k = (g-1)x_k + o + n_k$, with the gain g and the offset o of the pixel and the white noise n_k of frame k. The mean of Δ_k over the 100 frames is therefore $\bar{\Delta} = (g-1)\bar{x} + o + \bar{n}$, where $\bar{x}$ and $\bar{n}$ are the means of x_k and n_k. We call $\bar{\Delta}$ the fixed part and $\Delta_k - \bar{\Delta}$ the per-frame part.

Across the background pixels of a sequence, the standard deviation of $\bar{\Delta}$ has a median of 9.59 gray levels over the 80 sequences and ranges from 3.8 to 16.1. It increases with brightness. Pooled over all sequences, it is 3.9 gray levels at pixels with $30 \leq \bar{x} < 50$, 8.8 at $90 \leq \bar{x} < 110$ and 15.5 at $170 \leq \bar{x} < 190$. The fixed part is also constant in time. The means of Δ_k over the first and the last 50 frames correlate across pixels with a median of 0.986, and in every sequence this correlation lies within 0.001 of the value that a fixed part constant in time would produce, given the per-frame noise remaining in each 50-frame mean.

We estimate the spread of the per-frame part from the successive differences $(\Delta_{k+1} - \Delta_k)/\sqrt{2}$, which cancel $\bar{\Delta}$ and, unlike $\Delta_k - \bar{\Delta}$, are not reduced by the factor $\sqrt{99/100}$ that subtracting a 100-frame mean introduces. Their standard deviation is 8.00 gray levels over all background pixels and frames and lies between 7.97 and 8.01 in every sequence. The per-frame part is consistent with white Gaussian noise. Its correlation between successive frames has a median of -0.010 over the sequences, which matches the value $-1/99$ that subtracting a 100-frame mean produces in white noise, and its excess kurtosis has a median of -0.002, where a Gaussian has 0.

For g uniform on $(1-h, 1+h)$ and o of standard deviation σ_o, the variance of $\bar{\Delta}$ over pixels with the same $\bar{x}$ is $a\bar{x}^2 + b$, with $a = h^2/3$ and $b = \sigma_o^2 + 0.64$, where $0.64 = 8^2/100$ is the variance of $\bar{n}$. We pool the background pixels of all sequences into bins of five gray levels in $\bar{x}$ and fit a and b to the variance of $\bar{\Delta}$ in each bin, with $\bar{x}^2$ replaced by its mean over the bin, by least squares weighted by the pixel counts. The fit uses the 36 bins with $20 \leq \bar{x} < 200$, since outside this range the exclusion of pixels that reach 0 or 255 selects pixels by their gain and offset. It gives $h = \sqrt{3a} = 0.150$ and $\sigma_o = (b - 0.64)^{1/2} = 1.30$, against 0.15 and 1.3 in the noise model.

On the same background pixels of the clean sequences, the standard deviation of x_k over the 100 frames is below one gray level at 97.4% of the pixels and exactly zero at 59.1%.

D Hardest frames and temporal excursions

The hardest-frame statistics of Section 1 follow a rule that does not depend on the output of TRACE or DeepPro. For each of the 20 HiNo test sequences we consider the frames whose 40-frame window

lies inside the sequence and that hold exactly one target. The single-frame contrast of the target is $c = (I(\mathbf{p}) - \mathrm{med}_\Omega)/(1.4826\ \mathrm{MAD}_\Omega + 1/255)$, where I is the frame scaled to $[0, 1]$, $\mathbf{p}$ is the target pixel and Ω is its 11×11 neighborhood without the central 5×5 core. The hardest frame of a sequence is the eligible frame with the smallest c. In 17 of the 20 sequences this c is negative, so the target pixel lies below the median of its surround.

On each hardest frame we score every pixel by its largest temporal excursion over the 40-frame window, the maximum of $(x_k - \mathrm{med}_k\, x)/(1.4826\ \mathrm{MAD}_k\, x + 1)$ with x in gray levels. Pixels within a 15×15 box around the target are excluded, and so are pixels whose profile is flat, with a robust spread below one gray level. In all 20 frames the strongest remaining pixel scores higher than the target pixel. The ratio of the two scores has a median of 3.4, an interquartile range of 2.7 to 4.0 and a range of 1.2 to 45.8. Neither statistic uses a trained model.

The frames of Figure 1(a) and of the heavy-noise rows of Figure 3 are selected by a second rule, which relies on trained models. In 13 of the 20 sequences some eligible frame has a TRACE probability at the target pixel, for seed 1103, that reaches the frozen training threshold of 0.3105, and a DeepPro probability, from its released weights, below 0.0140, the lowest threshold at which DeepPro's point readout on the HiNo test sequences stays within the budget of 1.76, as in its point-readout row of Table 4. In each of these sequences we select such a frame with the smallest c. Over all 1,072 eligible frames of the 20 sequences, with the same model and the same two thresholds, the probability at the target pixel reaches the threshold for both detectors in 612 frames, for TRACE alone in 236, for DeepPro alone in 6 and for neither in 218. Figure 1(a) shows the three of the 13 selected frames with the largest share of pixels whose standardized value at frame t, computed as in Section 3.1 but without the clip, exceeds that of the target pixel. These frames hold 72,870, 62,700 and 60,600 pixels, of which 70,298, 60,324 and 48,276 exceed the target pixel. At threshold 0.5, which Figures 1 and 3 use, the TRACE probability at the target pixel still reaches the threshold in 11 of the 13 selected frames, among them these three and the two of the heavy-noise rows of Figure 3.

E Additional results

E.1 Point readout and thresholds fixed on the training sequences

The point readout of Section 4.1 reduces the probability map of TRACE to its 5×5 local maxima. This section reads it at the budgets of Section 4.1, with thresholds set on the test sequences as there, and at thresholds fixed on the training sequences for these budgets, one per dataset: one threshold serves both NUDT-MIRSDT groups, and HiNo has its own. The 80 training sequences are divided into four fixed folds of 20, and both datasets use the same folds, so a sequence and its HiNo version always lie in the same fold. For each dataset and fold, a model is trained on the other 60 sequences with the recipe of Appendix B and the seed 1103 alone, for 8 epochs on NUDT-MIRSDT and 48 on HiNo, and after its last epoch it predicts every frame of the 20 held-out sequences. The out-of-fold predictions of the four fold models are pooled per evaluation group: all 80 sequences for NUDT-MIRSDT, the 34 low-SNR sequences among them for $\mathrm{SNR} \leq 3$, and all 80 sequences for HiNo. For each group we select the lowest threshold at which the false-alarm rate of the point readout on the pooled predictions stays within the budget of the group, and the threshold of a dataset is the largest of these values over its groups. Raising the threshold never adds a false alarm, so this value keeps every group of the dataset within its budget. On NUDT-MIRSDT the $\mathrm{SNR} \leq 3$ group determines the threshold, 0.0630, whereas the NUDT-MIRSDT group alone would allow 0.0395. HiNo has a single group, and its threshold is 0.3105. The final models of the five seeds, each trained on all 80 training sequences of its dataset, use the threshold of that dataset unchanged, and no test sequence enters the choice of either threshold. We call these values the frozen training thresholds. At these thresholds the out-of-fold predictions detect 99.46% of the targets at $F_a = 0.52$ on $\mathrm{SNR} \leq 3$, 99.64% at $F_a = 0.41$ on NUDT-MIRSDT and 83.51% at $F_a = 1.76$ on HiNo.

Table 1 quotes DeepPro's published results, read with the raw readout at threshold 0.5, and reads TRACE with the same raw readout at the budgets. Table 4 runs DeepPro, from its released weights, and TRACE through our evaluation code, so both are counted in the same way, and it shows both methods under both readouts, which separates the effect of the readout from that of the operating point. Its rows at the budget read each method at the false-alarm rate DeepPro reports for each group, 0.52, 0.72 and 1.76 per 10^5 pixels, with the lowest threshold that keeps the test false-alarm

Table 4: P_d (%) and F_a (10^{-5} per pixel) of DeepPro (released weights) and TRACE (mean over five seeds). Bold: higher P_d of each at-budget pair; lower F_a of the two frozen-threshold rows.

Method	Readout	Threshold	SNR ≤ 3		NUDT-MIRSDT		NUDT-MIRSDT-HiNo	
			P_d	F_a	P_d	F_a	P_d	F_a
DeepPro *(TPAMI'26)*	raw pixels	at the budget	**96.03**	0.52	98.61	0.72	59.17	1.76
TRACE (ours)	raw pixels	at the budget	95.99	0.52	**99.36**	0.72	**76.34**	1.76
DeepPro *(TPAMI'26)*	points	at the budget	93.95	0.52	97.11	0.72	58.70	1.76
TRACE (ours)	points	at the budget	**98.34**	0.52	**99.55**	0.72	**80.44**	1.76
TRACE (ours)	raw pixels	frozen on training data	96.29	0.73	98.07	0.29	78.16	3.44
TRACE (ours)	points	frozen on training data	96.86	**0.18**	98.48	**0.07**	79.09	**0.92**

rate within that value, and its TRACE row with the raw readout at the budget is the TRACE row of Table 1. Only the last two rows set their thresholds without the test sequences.

Our pipeline agrees with the one that produced the published numbers. Run on the released DeepPro weights at DeepPro's own threshold 0.5 with the raw readout, it returns 95.84 at $F_a = 0.52$, 98.50 at 0.72 and 59.17 at 1.76, or 95.8412, 98.4962 and 59.1671 at 0.5190, 0.7173 and 1.7580 before rounding. All six published cells match to the printed precision. Both published AUCs, 0.9973 on NUDT-MIRSDT and 0.9638 on HiNo, match to four decimals. On the same raw map TRACE has AUC 0.9979 and 0.9677. Under the point readout TRACE has 0.9969 and 0.9764, and DeepPro has 0.9854 and 0.9079. On $\mathrm{SNR} \leq 3$ the AUC is 0.9977 for TRACE and 0.9952 for DeepPro with the raw readout, and 0.9949 and 0.9755 with the point readout. At 0.5 DeepPro remains slightly below each budget, so reading it at the budget, as Table 4 does, lets its threshold fall slightly until F_a reaches 0.52, 0.72 and 1.76. Its raw-readout row is therefore equal to or slightly above the published cells. Its point-readout row is below them because the point readout, unlike the published one, credits a target only through a local maximum within two pixels.

Table 4 does not compare the two networks at threshold 0.5, because there they run at different false-alarm rates. With the raw readout, TRACE reaches 93.5, 96.6 and 78.0 at $F_a = 0.43$, 0.17 and 3.23, and DeepPro 95.8, 98.5 and 59.2 at 0.52, 0.72 and 1.76. TRACE thus operates at about five sixths of DeepPro's false-alarm rate on $\mathrm{SNR} \leq 3$, a quarter on NUDT-MIRSDT and nearly twice under heavy noise. At this threshold TRACE also varies across seeds, with a standard deviation of 5.1, 2.6 and 2.2 points on the three groups. Read at the budget with the same readout, the spread decreases to 0.25, 0.14 and 1.27.

The readout changes the comparison, so we report both. With the point readout at the budgets TRACE detects 98.34%, 99.55% and 80.44% of the targets. Under heavy noise it leads by 17.2 points with the raw readout and by 21.7 with the point readout. On $\mathrm{SNR} \leq 3$ the two methods perform comparably with the raw readout, while TRACE leads by 4.4 points with the point readout. The point readout is stricter about hits and more lenient about false alarms. A target is counted only when a local maximum lies within two pixels of it, but a cluster of false pixels is counted once. TRACE improves under it in all three groups and DeepPro declines in all three.

The last row reads TRACE with the point readout at the frozen training thresholds, and the row above it with the raw readout at thresholds frozen in the same way. We calibrate a second threshold per dataset on the same out-of-fold predictions with the same rule, 0.2021 for NUDT-MIRSDT, again set by the $\mathrm{SNR} \leq 3$ group, and 0.4728 for HiNo. At these thresholds the out-of-fold predictions detect 99.01%, 99.40% and 82.26% of the targets at $F_a = 0.52$, 0.41 and 1.76 on $\mathrm{SNR} \leq 3$, NUDT-MIRSDT and HiNo. On the test sequences the raw readout of TRACE stays within the budget on NUDT-MIRSDT, at $F_a = 0.29$, but exceeds it on $\mathrm{SNR} \leq 3$, at 0.73 with three of the five seeds above 0.52, and on HiNo, at 3.44 with every seed above 1.76. The point readout at its frozen training threshold stays within every budget in every seed. Against DeepPro under the same point readout, TRACE then detects more targets in all three groups and produces fewer false alarms, about a tenth of DeepPro's rate on NUDT-MIRSDT. Against DeepPro's raw-readout row it detects more targets in two groups and slightly fewer on NUDT-MIRSDT, 98.48 against 98.61, again at a lower false-alarm rate.

Figure 6 gives the point-readout curves of the three groups over a range of false-alarm rates, for TRACE, DeepPro and the no-neighborhood arm of Table 3, so the comparison can also be checked

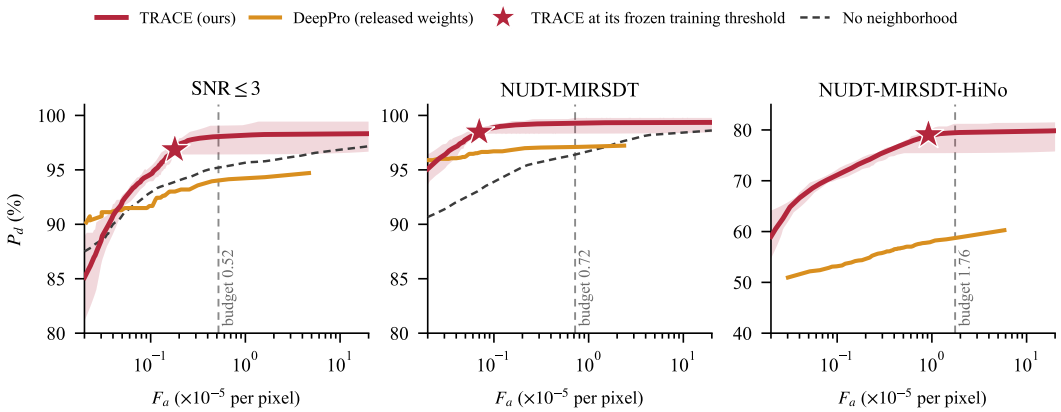


Figure 6: P_d against F_a under the point readout in the three groups: TRACE and the no-neighborhood arm (five-seed means; arm on the clean groups only) and DeepPro (released weights). Shaded: range of the five TRACE seeds; vertical dashed lines: budgets.

Table 5: P_d (%) and F_a (10^{-5} per pixel) of the five TRACE seeds, with their mean and sample standard deviation, under the point readout at the frozen training thresholds.

Seed	SNR ≤ 3		NUDT-MIRSDT		NUDT-MIRSDT-HiNo	
	P_d	F_a	P_d	F_a	P_d	F_a
1103	96.22	0.12	98.32	0.05	80.86	1.18
2207	98.49	0.24	99.13	0.11	78.89	0.64
3301	98.11	0.23	98.90	0.10	80.51	1.37
4409	97.73	0.22	98.96	0.09	75.01	0.42
5501	93.76	0.06	97.11	0.03	80.16	1.02
Mean	96.86	0.18	98.48	0.07	79.09	0.92
Standard deviation	1.94	0.08	0.83	0.03	2.39	0.39

at false-alarm rates other than the budgets. Table 3 reads the same arm with the raw readout. The figure is drawn from the stored threshold sweeps of the same runs without rerunning any model, and its stars mark the last row of Table 4. Under heavy noise TRACE detects more targets than DeepPro at every false-alarm rate shown. On the two clean groups the curves cross: DeepPro detects more only below about 0.04×10^{-5} on SNR ≤ 3 and 0.03×10^{-5} on NUDT-MIRSDT, and TRACE detects more at every higher rate shown. Each curve is interpolated between the thresholds of a fixed grid, so at a budget it lies slightly below the value read at the exact threshold, 98.34, 99.55 and 80.44 for TRACE in Table 4. Under heavy noise the no-neighborhood arm is not drawn. Its training failed there without a warm-up, and its nearly flat probability map, 0.605 to three decimals at every pixel of the six heavy-noise test frames whose maps we exported, falls between two thresholds of the grid, while Table 10 reports its point-readout value as 19.55.

Table 5 gives P_d and F_a of TRACE for each of the five seeds, read with the point readout at the frozen training thresholds, 0.0630 on both NUDT-MIRSDT groups and 0.3105 on HiNo. The mean and the standard deviation are computed from the unrounded values, and the means are the last row of Table 4. Every seed stays within the false-alarm budget of each group, and in each group the seed that detects the fewest targets, 5501 on both NUDT-MIRSDT groups and 4409 on HiNo, also has the lowest false-alarm rate.

Table 6 splits the point-readout rows of TRACE and DeepPro at the budget in Table 4 by test sequence, recounted from the stored outputs of the same runs at the same thresholds. A sequence counts as a tie when the two methods differ by less than one point before rounding, and the 95% interval of Δ resamples the sequences of a group 10,000 times with replacement, with the thresholds fixed. The row All pools the 1,729 targets of the 20 sequences and the 529 of the eight SNR ≤ 3 sequences. Under heavy noise TRACE detects more targets than DeepPro in 14 of the 20 sequences

Table 6: P_d (%) of TRACE and DeepPro on each HiNo and $\mathrm{SNR} \leq 3$ test sequence, point readout at the budget. TRACE: mean over five seeds; Δ: TRACE minus DeepPro; All: all targets pooled.

Sequence	Targets	NUDT-MIRSDT-HiNo			$\mathrm{SNR} \leq 3$		
		TRACE	DeepPro	Δ	TRACE	DeepPro	Δ
47	79	99.49	70.89	+28.6	99.75	96.20	+3.5
56	41	83.90	63.41	+20.5	94.63	92.68	+2.0
59	67	90.15	61.19	+29.0	92.84	95.52	−2.7
76	56	0.00	0.00	0.0	97.50	96.43	+1.1
85	100	100.00	100.00	0.0			
86	100	100.00	94.00	+6.0			
87	100	99.80	97.00	+2.8			
88	100	100.00	100.00	0.0			
89	100	100.00	100.00	0.0			
90	100	100.00	59.00	+41.0			
91	100	99.40	97.00	+2.4			
92	100	84.80	28.00	+56.8	100.00	98.00	+2.0
93	100	88.80	40.00	+48.8			
94	100	73.40	0.00	+73.4			
95	100	33.20	5.00	+28.2			
96	100	1.60	0.00	+1.6			
97	100	100.00	100.00	0.0			
101	90	62.00	1.11	+60.9	100.00	95.56	+4.4
105	53	100.00	100.00	0.0	99.62	92.45	+7.2
119	43	64.19	41.86	+22.3	100.00	74.42	+25.6
All		80.44	58.70	+21.7	98.34	93.95	+4.4
95% interval of Δ over resampled sequences				11.0 to 33.6			1.1 to 9.4
Sequences won, tied and lost by TRACE				14 / 6 / 0			7 / 0 / 1

and fewer in none. In five of the six ties both methods detect every target, and in sequence 76 neither detects any. Sequences 92, 94 and 101 account for 49% of the pooled lead of 21.7 points, and pooled over the other 17 sequences the lead is 13.3 points. On the eight $\mathrm{SNR} \leq 3$ sequences TRACE detects fewer targets than DeepPro in sequence 59, by 2.7 points, and sequence 119 alone accounts for 47% of the lead of 4.4 points. With eight sequences, the 95% interval of the $\mathrm{SNR} \leq 3$ group in Table 6 is only indicative. When the sequences are resampled for each seed separately, the interval of the heavy-noise lead stays above 10 points in every seed, while on $\mathrm{SNR} \leq 3$ it includes zero for seed 5501. DeepPro is a single released model, so its spread over training seeds is unknown.

In 39 of the 100 frames of a test sequence, frames 1 to 19 and 81 to 100, the window extends beyond the sequence, and every result includes the targets of these frames. At the same thresholds TRACE detects 79.28% of the heavy-noise targets on these frames, 1.8 points fewer than on the other 61 frames and fewer in four of the five seeds, and DeepPro 58.91%, against 58.59% on the other frames. On the clean groups neither method changes by more than 1.4 points on average between the two sets of frames.

E.2 Per-seed and per-sequence results under the raw readout

Table 7 gives P_d of each seed of TRACE in the TRACE row of Table 1, read with the raw readout at the budget of each group, and the threshold at which it is read. Every seed has 273, 952 and 2,328 false pixels on $\mathrm{SNR} \leq 3$, NUDT-MIRSDT and HiNo, as many as DeepPro read in the same way, so F_a is 0.52, 0.72 and 1.76 in every seed. Under heavy noise every seed detects more targets than DeepPro, by 15.1 to 18.1 points.

Table 8 splits the raw-readout rows of Table 4 at the budget by test sequence, recounted from the stored outputs of the same runs at the same thresholds. Ties, the interval and the row All are defined as in Table 6. Under heavy noise TRACE detects at least one point more than DeepPro in 11 of the 20 sequences, and the other nine differ by less than one point and count as ties. In six of the ties both methods detect every target, and in sequence 76 neither detects any; in sequence 96 TRACE is ahead by 0.2 points, and in sequence 105 DeepPro detects 100% of the targets and TRACE 99.62%,

Table 7: P_d (%) and threshold of the five TRACE seeds, with the mean and sample standard deviation of P_d, under the raw readout at the budget.

Seed	$\mathrm{SNR} \leq 3$		NUDT-MIRSDT		NUDT-MIRSDT-HiNo	
	P_d	Threshold	P_d	Threshold	P_d	Threshold
1103	96.22	0.1196	99.48	0.0077	77.21	0.7275
2207	95.65	0.5176	99.48	0.0542	76.00	0.6731
3301	96.03	0.6142	99.19	0.0495	77.27	0.8620
4409	95.84	0.6602	99.42	0.0424	74.26	0.5398
5501	96.22	0.0202	99.25	0.0042	76.98	0.8128
Mean	95.99		99.36		76.34	
Standard deviation	0.25		0.14		1.27	

the only sequence in which DeepPro is higher. In sequence 97 the two methods differ by exactly one target of 100, which counts as a win. Sequences 92, 94 and 101 account for 49% of the pooled lead of 17.2 points, and pooled over the other 17 sequences the lead is 10.6 points. When the sequences are resampled for each seed separately, the interval of the heavy-noise lead stays above 7 points in every seed. On the eight $\mathrm{SNR} \leq 3$ sequences the two methods are comparable: TRACE detects at least one point fewer in sequences 47 and 76 and at least one point more in 101 and 119, and the other four differ by less than one point, with TRACE lower in three of them. In Table 8 the differences of sequences 47 and 56 both print as -1.0; before rounding, TRACE is at least one point lower in sequence 47, a loss, and less than one point lower in sequence 56, a tie. When the sequences are resampled for each seed separately, the interval of the difference includes zero in every seed, and with eight sequences the 95% interval of this group is only indicative. DeepPro is a single released model, so its spread over training seeds is unknown. Beyond the budget, the stored threshold sweeps of the same runs give TRACE the higher heavy-noise detection rate with the raw readout at every false-alarm rate from 0.12 to 20 per 10^5 pixels, by 15.8 to 20.2 points for the mean of the five seeds, and every seed leads at every rate.

E.3 IRDST-simulation

IRDST-simulation (Sun et al., 2023) differs from NUDT-MIRSDT in its scenes and targets and has no added heavy noise, and every compared multi-frame method of Table 2 detects at least 96.86% of its targets. We train TRACE from scratch on 225 of its 250 official training sequences, with the five seeds of the main experiments and 17,664 updates, a number computed from DeepPro's training recipe for these sequences. In Table 2 of the main text it is read in the same way as the rows quoted from DeepPro, on the raw per-pixel map at threshold 0.5. In every seed TRACE detects 19,184 of the 19,185 targets of the 66 test sequences without a false pixel, at least as many targets as every compared method. The best quoted rate is DeepPro's 99.85% at $F_a = 0.13 \times 10^{-5}$.

E.4 Other noise models

The five HiNo models of TRACE and the released HiNo weights of DeepPro, which we run for inference only, are applied without retraining to five noisy versions of the 20 test sequences, generated from the clean frames. H8 has the form of the HiNo noise, $y = gx + o + n$, with g uniform on $(0.85, 1.15)$ and o of standard deviation 2 gray levels fixed per pixel, and white noise n of standard deviation 8 gray levels in every frame. Its offsets spread slightly more than the 1.3 gray levels measured for HiNo in Appendix C.1. W8 keeps only the white noise. H4 and H12 are H8 with white noise of standard deviation 4 and 12. D8 lets the offset of H8 drift linearly over the sequence, $o_k = o + d\,(k - 50.5)/99$ for frames $k = 1, \ldots, 100$, with d of standard deviation 16 gray levels per pixel. Values are rounded and clipped to $[0, 255]$, with one random stream per sequence. Each condition is read with the raw and with the point readout at $F_a = 1.76$, with the threshold set on the test sequences of that condition as in Table 4, and R0, the official HiNo test set, reproduces the HiNo column of that table.

Table 8: P_d (%) of TRACE and DeepPro on each HiNo and $\mathrm{SNR} \leq 3$ test sequence, raw readout at the budget. TRACE: mean over five seeds; Δ: TRACE minus DeepPro; All: all targets pooled.

Sequence	Targets	NUDT-MIRSDT-HiNo			$\mathrm{SNR} \leq 3$		
		TRACE	DeepPro	Δ	TRACE	DeepPro	Δ
47	79	96.96	77.22	+19.7	98.99	100.00	−1.0
56	41	82.44	48.78	+33.7	89.27	90.24	−1.0
59	67	86.27	65.67	+20.6	97.91	98.51	−0.6
76	56	0.00	0.00	0.0	82.14	83.93	−1.8
85	100	100.00	100.00	0.0			
86	100	100.00	100.00	0.0			
87	100	100.00	100.00	0.0			
88	100	100.00	100.00	0.0			
89	100	100.00	100.00	0.0			
90	100	99.60	60.00	+39.6			
91	100	100.00	100.00	0.0			
92	100	74.40	23.00	+51.4	100.00	100.00	0.0
93	100	71.60	36.00	+35.6			
94	100	53.80	0.00	+53.8			
95	100	30.40	5.00	+25.4			
96	100	0.20	0.00	+0.2			
97	100	100.00	99.00	+1.0			
101	90	44.00	0.00	+44.0	100.00	98.89	+1.1
105	53	99.62	100.00	−0.4	99.62	100.00	−0.4
119	43	68.37	51.16	+17.2	89.77	86.05	+3.7
All		76.34	59.17	+17.2	95.99	96.03	0.0
95% interval of Δ over resampled sequences				8.4 to 26.8			−0.9 to 0.9
Sequences won, tied and lost by TRACE				11 / 9 / 0			2 / 4 / 2

Table 9: P_d (%) of TRACE and DeepPro (released HiNo weights) under the HiNo noise (R0) and five other noise models at $F_a = 1.76$ (10^{-5} per pixel), and P_d and F_a of TRACE (point readout) at its frozen training threshold. TRACE: mean $\pm$ SD over five seeds; Δ: TRACE minus DeepPro.

	Raw readout			Point readout		TRACE, frozen threshold	
Noise	TRACE	DeepPro	Δ	TRACE	DeepPro	P_d	F_a
R0	76.34 ± 1.27	59.17	+17.2	80.44 ± 1.18	58.70	79.09 ± 2.39	0.92 ± 0.39
H8	76.56 ± 1.10	59.75	+16.8	80.67 ± 1.11	58.53	78.81 ± 2.22	0.78 ± 0.35
W8	75.91 ± 1.21	57.14	+18.8	79.51 ± 1.09	56.62	77.74 ± 2.18	0.85 ± 0.34
H4	84.93 ± 1.40	60.09	+24.8	87.69 ± 0.35	64.49	89.55 ± 0.97	3.84 ± 0.96
H12	69.83 ± 0.82	40.83	+29.0	70.98 ± 1.19	40.95	68.58 ± 2.56	0.33 ± 0.18
D8	76.01 ± 1.14	55.41	+20.6	79.38 ± 0.72	55.70	78.59 ± 2.11	1.34 ± 0.56

Under each of the five conditions and with both readouts, TRACE detects more targets than DeepPro, by 16.8 to 29.0 points with the raw readout and by 22.1 to 30.0 points with the point readout, and every seed detects more targets than DeepPro under every condition. Table 9 gives the full numbers, including TRACE with the point readout at its frozen training threshold.

E.5 Warm-up, budget-aware terms and further ablation runs

We retrained TRACE and its no-neighborhood arm on HiNo with two changes to the loss, fixed in writing before the runs. In the warm-up, $\mathcal{L}_{\mathrm{tail}}$ and $\mathcal{L}_{\mathrm{rank}}$ have weight zero for the first 600 of the 12,480 updates. Without the budget terms, they have weight zero throughout, and $\mathcal{L}_{\mathrm{tgt}}$, $\mathcal{L}_{\mathrm{core}}$, $\mathcal{L}_{\mathrm{loc}}$ and $\mathcal{L}_{\mathrm{bg}}$ keep their weights. Everything else is the HiNo recipe of the full model, with its five seeds. A run is considered trained when its mean loss over the last 500 updates lies more than 0.05 below the loss of a constant output, 1.601 with the loss of the paper and 0.740 without the budget terms. The runs with the loss of the paper are the TRACE and no-neighborhood runs of Table 3; these logged only epoch means, so for them the criterion uses the mean loss of the last two

Table 10: HiNo P_d (%) of TRACE and its no-neighborhood arm, each trained in three ways: with the loss of the paper, with a warm-up, and without the two budget-aware terms. Mean $\pm$ SD over five seeds; Trained: seeds that meet the training criterion.

Model	Loss	Raw readout	Point readout	Trained
TRACE	Loss of the paper	76.34 ± 1.27	80.44 ± 1.18	5/5
	Warm-up	74.91 ± 1.13	79.14 ± 1.64	5/5
	No budget terms	70.08 ± 2.13	78.16 ± 1.05	5/5
No neighborhood	Loss of the paper	20.38 ± 11.36	19.55 ± 11.14	0/5
	Warm-up	32.01 ± 1.27	31.73 ± 1.22	5/5
	No budget terms	36.88 ± 0.59	36.61 ± 0.57	5/5
TRACE minus no neighborhood	Loss of the paper	55.96	60.89	
	Warm-up	42.90	47.41	
	No budget terms	33.20	41.55	

epochs. Table 10 counts the seeds that meet this criterion and reads the runs of TRACE and of the no-neighborhood arm at the budget with the raw readout, as in Table 3, and with the point readout, which the written protocol named as the primary readout. Without the warm-up the no-neighborhood arm detects no heavy-noise target at its frozen training threshold in any seed.

With the warm-up the training of this arm succeeds in every seed, and TRACE trained in the same way detects 42.9 points more under heavy noise, paired by seed, with 42.0, 43.6, 46.1, 41.6 and 41.2 points for the seeds 1103, 2207, 3301, 4409 and 5501, and 47.4 points more with the point readout. The warm-up itself lowers the detection rate of TRACE by 1.4 points, and all five seeds fall below their runs in Table 3. Without the two budget-aware terms TRACE detects 70.08% of the heavy-noise targets, 6.3 points below the full model, with 5.8, 8.3, 4.7, 6.2 and 6.4 points for the seeds in the same order. This is 10.9 points above DeepPro read in the same way, and every seed stays above DeepPro's 59.17%. With the point readout it detects 78.16%, 2.3 points below the full model and 19.5 points above DeepPro read with the point readout. Without these two terms, reading neighboring pixels adds 33.2 points, 30.8 to 36.1 over the seeds, and 41.6 points with the point readout. The first 600 updates of each arm without the budget terms coincide with those of its warm-up arm.

The TTM arm of Table 3 with the output frame only also failed to train on HiNo: in every seed, its mean loss over the last two epochs stayed within 0.05 of the loss of a constant output. After this result, we retrained it on HiNo with the warm-up above and the same five seeds. It then trains in every seed, with a mean loss of 1.41 to 1.42 over the last two epochs, and detects $34.04 \pm 1.19\%$ of the heavy-noise targets with the raw readout and 37.40% with the point readout. TRACE trained with the same warm-up detects 40.87 points more with the raw readout, 39.79 to 43.26 per seed, and 41.75 points more with the point readout, paired by seed. Without the sign split of TTM, which then reads $\mathbf{z}$ instead of $\mathbf{z}^{\pm}$ with 15,841 parameters, TRACE retrained with the recipe and seeds of the full model detects 95.80 ± 0.54, 98.39 ± 0.13 and $76.36 \pm 0.66\%$ of the targets on $\mathrm{SNR} \leq 3$, NUDT-MIRSDT and HiNo with the raw readout at the budgets.

E.6 Baselines retrained for Figure 3

No trained weights are released for Res-U+RFR on either dataset or for Res-U+DTUM and DQAligner on NUDT-MIRSDT-HiNo, so Figure 3 shows models that we trained once with the official code and hyperparameters, on the 80 training sequences and with seed 1103, keeping the final epoch. For DQAligner, whose official batch of two does not fit into the memory of one 24 GB GPU, each batch is processed as two single-sample passes with accumulated gradients and a single update; the loss and its gradient are those of the official batch, but batch normalization sees one sample at a time. Table 11 scores them with our evaluation code and the raw readout at threshold 0.5, the readout of the quoted rows, and places the rows quoted in Table 1 beside them. The retrained Res-U+RFR reaches a higher AUC than the quoted rows, but at threshold 0.5 more of its background pixels reach the threshold on NUDT-MIRSDT and HiNo, though not on $\mathrm{SNR} \leq 3$, so Figure 3 shows more false alarms for it on these two groups than the quoted rows suggest. The re-

Table 11: P_d (%), F_a (10^{-5} per pixel) and AUC of the baselines of Figure 3 that we retrained, and their rows quoted in Table 1. –: not reported.

		Retrained (ours)			Quoted		
Method	Group	P_d	F_a	AUC	P_d	F_a	AUC
Res-U+RFR	NUDT-MIRSDT	91.38	2.50	0.9828	93.35	1.95	0.9657
Res-U+RFR	SNR ≤ 3	74.10	2.83	0.9565	79.39	4.15	–
Res-U+RFR	HiNo	37.48	482.32	0.8955	55.64	111.23	0.7839
Res-U+DTUM	HiNo	28.57	1.63	0.6525	43.90	4.86	0.9413
DQAligner	HiNo	12.03	5.49	0.6831	33.41	2.95	0.8022

trained Res-U+DTUM detects fewer heavy-noise targets than the quoted row, at a lower false-alarm rate and with a lower AUC.

E.7 A PROBE OF THE NEIGHBORHOOD WITH SYNTHETIC TARGETS

We apply the five HiNo models of TRACE, those of Table 1, with frozen weights to synthetic clips of 40 frames of 64×64 pixels. The background is 140 gray levels plus a smooth texture, fixed over the frames, with a standard deviation of 15 gray levels, and every pixel of every frame carries independent Gaussian noise with a standard deviation of 7.8 gray levels. The target is a 3×3 square with edges softened by a Gaussian of standard deviation 0.5 pixels. It moves right by one pixel per frame and reaches the center of the clip at frame t, the frame that is scored. In three of the four conditions the central 5×5 pixels receive the same signal, and only the pixels around them differ. In the path condition the target moves through these surrounding pixels, and in the center-only condition they receive nothing. In the scrambled condition each frame instead flashes a target of the same contrast at a random position in the region within six pixels of the center and outside the central 5×5 pixels. As these positions lie off the pixel grid, each flash covers 2×2 pixels. Together the flashes still deliver 2.7 times as much target signal into this region as the path, and they reach it in all 40 frames, against at most 19 for the path. A fourth condition keeps the path and leaves the central 5×5 pixels empty. A trial is counted as a detection when p reaches the frozen training threshold of HiNo, 0.3105, at a pixel within one pixel of the target center at frame t. Unlike the point readout, this rule extracts no local maxima and counts no false alarms. Every condition and contrast has 160 trials: 32 noise backgrounds, the same in every condition, for each of the five models. Table 12 lists the detection rates.

At a contrast of 12 gray levels, 1.5 noise standard deviations, TRACE detects the moving target in 85.0% of the trials, 75.0% to 93.8% over the five models, with scrambled flashes around it in 26.9%, 15.6% to 34.4%, and with nothing around it in 23.8%, 15.6% to 28.1%. Every model detects the target in 16 to 21 more of its 32 trials with the path than with the scrambled flashes. The flashes deliver more signal into the neighborhood than the path but raise the detection rate only slightly above that of the center alone, so the neighbors add evidence when their signal follows a track. The neighbors alone do not suffice: with the path around an empty center TRACE detects no target at this contrast. The order of the three conditions is the same at 8, 10, 14 and 16 gray levels, and at 22 gray levels the target is detected in at least 96% of the trials in all three. The probe was run with one design, after the model was fixed, and reruns reproduce its numbers exactly; the design was not fixed in writing before the run.

E.8 REAL THERMAL-INFRARED SEQUENCES FROM RGBT-TINY

As in Li et al. (2026), we evaluate on real thermal-infrared sequences of RGBT-Tiny (Ying et al., 2025b) that contain infrared small targets. We select them from the sequences listed in the DeepPro code with a rule that reads only the annotations and matches the setting of this paper, small, moving and sparse targets. Specifically, a sequence is kept when the median area of its boxes is at most 400 square pixels, when at most 10% of its boxes are stationary and when the median number of objects per frame is at most five. A box is stationary when its center moves by less than 2 pixels within the 40-frame window of its frame. The rule keeps 44 sequences, and each stays in its original training or test set. Five of the training sequences, each from a recording that contributes no other

Table 12: Detection rate (%) of the five HiNo models of TRACE on synthetic clips of the four conditions, by target contrast in gray levels and in noise standard deviations.

Contrast		Path	Scrambled	Center only	Path,
Gray levels	Noise SD				empty center
8	1.03	21.9	6.3	4.4	0.0
10	1.28	46.3	13.8	12.5	0.0
12	1.54	85.0	26.9	23.8	0.0
14	1.79	96.9	41.9	41.3	0.6
16	2.05	100.0	67.5	63.8	1.3
22	2.82	100.0	96.3	99.4	4.4

Table 13: P_d (%) and F_a (10^{-5} per pixel) on the RGBT-Tiny test sequences, point readout, at a matched F_a and at a threshold fixed on five held-out calibration sequences. Mean $\pm$ SD over five seeds; bold: best.

	F_a matched to 4.47		Calibration threshold	
Method	P_d (%)	Seeds ahead of DeepPro	P_d (%)	F_a
DeepPro *(TPAMI'26)*	72.68 $\pm$ 2.24	–	74.11 $\pm$ 1.92	6.77
TRACE (ours)	**80.49 $\pm$ 0.31**	5/5	**81.11 $\pm$ 0.31**	**6.30**

retained sequence, are then held out to calibrate thresholds. The resulting split has 26 training, 5 calibration and 13 test sequences. The test sequences, which mostly show planes and drones, contain 5,476 frames and 8,256 annotated objects. Each target is represented by the center pixel of its box. Hits and false alarms are counted as in Appendix C, with the point readout of Section 4.1 for both methods. We train TRACE and DeepPro on the 26 training sequences with five seeds each and 6,784 updates per run. TRACE keeps the loss and optimizer of the main experiments, DeepPro keeps its own, and every seed is reported. The false-alarm rate is matched to 4.47×10^{-5} on the test pixels, and a second operating point uses the threshold fixed on the calibration sequences. Both methods are read with the point readout: the sigmoid output of DeepPro equals 1 in single precision at many background pixels, so that at this false-alarm rate the per-pixel readout is not defined for it, whereas TRACE detects 79.81% of the test targets with the per-pixel readout and 80.49% with the point readout.

Table 13 shows that, at the matched rate, TRACE detects 80.49% of the test targets and DeepPro 72.68%. TRACE is ahead in each of the five seeds, by 6.26 to 11.26 percentage points, and on 10 of the 13 test sequences. With the threshold fixed on the calibration sequences, TRACE reaches 81.11% at $F_a = 6.30 \times 10^{-5}$ and DeepPro 74.11% at $F_a = 6.77 \times 10^{-5}$. Trained without the two budget-aware terms of the loss, TRACE detects $73.68 \pm 1.47\%$ of the test targets, 6.81 points fewer, and is still ahead of DeepPro in each of the five seeds; at the threshold fixed on the calibration sequences it reaches $74.40 \pm 2.26\%$ at $F_a = 4.99 \times 10^{-5}$. On HiNo, too, TRACE detects fewer targets without these terms, with either readout (Table 10).

F TIMING PROTOCOL AND DEVIATIONS FROM THE RELEASED CODE

The 15 compared deep methods with public code and TRACE are timed on one RTX 4090 D at 256×256 pixels, in FP32 with batch size 1 and deterministic cuDNN, on a GPU that runs no other compute process. Table 14 lists the code timed for each of these 15 compared methods. ACM, ALCNet, DNA-Net, ISNet and UIUNet are timed with the implementations that the repository of Res-U+DTUM (Li et al., 2023b) provides for them, and Res-UNet with the one in the BasicIRSTD toolbox, so for these six methods we did not time the code released with the original paper. The other nine compared methods are timed with the repositories of their authors. The eight traditional methods and STDMANet are not timed, and Table 1 quotes their published rates. One timing wrapper serves every method and differs between methods only in the network, its input layout and the constants that scale its input. The end-to-end stage copies each new frame from host memory to the GPU, updates the frame window, scales it, and runs the network and the readout. The forward stage times the network alone, on inputs already on the GPU. Neither stage reads files or computes

Table 14: GitHub repository and commit of the code timed for each of the 15 compared deep methods with public code.

Methods	Repository	Commit
ACM, ALCNet, DNA-Net, ISNet, UIUNet, Res-U+DTUM	`TinaLRJ/Multi-frame-infrared-small-target-detection-DTUM`	`61053d8`
Res-UNet	`XinyiYing/BasicIRSTD`	`95650c3`
AGPCNet	`Tianfang-Zhang/AGPCNet`	`5671b81`
MSHNet	`ying-fu/MSHNet`	`46cdfd4`
SCTransNet	`xdFai/SCTransNet`	`e027628`
RPCANet	`fengyiwu98/RPCANet`	`1c32231`
ILNet	`Li-Haoqing/ILNet`	`82513dd`
Res-U+RFR	`XinyiYing/RFR`	`3451268`
DQAligner	`dengfa02/DQAligner_MIRSTD`	`c586f39`
DeepPro	`TinaLRJ/DeepPro`	`692b18f`

metrics. Rates are per output frame, so DeepPro is amortized over the 36 new outputs of each 40-frame block. The readout is the same code for every method: a sigmoid, the 5×5 local maxima that reach 0.5, and a copy of their coordinates to the host. Unlike our evaluation code, it does not merge adjacent local maxima of equal value into one point. The whole measurement is repeated in three separate passes, each over all methods, and we report the median of the three values, a rule registered before the measurements. Our values for the #Params column, computed with DeepPro's counting function, reproduce the quoted ones within 0.1%, except for Res-UNet, 1.0% lower, and DeepPro, 0.25% lower because the published 0.197 is rounded.

All timed inputs are random frames. DeepPro and TRACE run with their trained HiNo weights, and the other 14 methods run from a random initialization with seed 1103. In the code of these 14 methods we found no operation whose computational cost depends on the weights, so we expect their forward rates to change little with trained weights, but we did not verify this. Their end-to-end rates can depend on the weights, because the readout copies every detection to the host and the number of detections depends on the weights and on the input. In the untimed warm-up calls the readout returned no detections for ACM, ALCNet, DNA-Net, ISNet, DQAligner and TRACE, 110 to 140 per frame for UIUNet, 1,856 to 2,818 per frame for the other eight methods, and 542,589 to 635,882 per 40-frame block for DeepPro. These loads arise from random frames and, except for DeepPro and TRACE, from random weights, so they need not match the loads on the benchmark sequences. In both cuDNN modes the forward stage, which has no readout, ranks the 16 timed methods in the same order as the end-to-end stage.

We had also registered a limit of 10% on the spread between passes. In the deterministic mode, which Table 1 and Figure 5 report, two rows exceeded it: the forward stage of RPCANet at 19.2% and the end-to-end stage of TRACE at 11.6%, where one disturbed pass returned 635.7 frames per second. The median is unaffected. With autotuned cuDNN the same protocol gives 713.1 frames per second for TRACE and 266.0 for DeepPro, against 711.5 and 261.6 in the deterministic mode. We quote the autotuned mode for reference only. More of its rows exceed the limit, among them the forward stage of TRACE. In the deterministic mode the network forward alone runs at 793.0 frames per second for TRACE and 267.2 for DeepPro, a ratio of 2.97. For the other methods the forward rates are 513.4 for Res-UNet, 319.4 for ACM, 265.4 for ALCNet, 242.0 for Res-U+RFR, 209.3 for Res-U+DTUM, 178.9 for RPCANet, 141.0 for ISNet, 113.1 for MSHNet, 83.0 for DNA-Net, 67.9 for ILNet, 53.6 for SCTransNet, 25.6 for AGPCNet, 25.5 for UIUNet and 20.6 for DQAligner. Peak memory is the largest amount of GPU memory allocated to tensors during the timed end-to-end calls, as PyTorch reports it: 85.6 MiB for TRACE and $1{,}800.9$ MiB for DeepPro. It includes the weights, the window of 40 frames that both methods keep on the GPU and every intermediate tensor, but not the CUDA context or memory that the allocator reserves without using it. The no-neighborhood arm of Table 3 runs its forward pass 1.031 times as fast as TRACE, so access to neighboring pixels accounts for about 3% of the forward time at 256×256 pixels.

The released code was changed in three ways, which affect five methods. ILNet runs in its M configuration, the one whose parameter count matches the published table, and its hard-coded resizing to 512×512 is replaced by resizing to the input size. The compiled deformable-convolution extensions used by ISNet, Res-U+RFR and DQAligner are replaced

by `torchvision.ops.deform_conv2d`, and their Python model code is left unchanged. AGPCNet's backbone is not initialized from ImageNet, because all methods other than DeepPro and TRACE start from a random initialization. Two Python packages missing on the timing machine were supplied with the timing code: a copy of `einops` 0.8.2, used by RPCANet, SCTransNet and DQAligner, and, for SCTransNet, a plain attribute dictionary in place of the configuration class of `ml_collections`, which here only holds the hyperparameters of SCTransNet. The model files themselves are unchanged copies of those at the commits of Table 14, and the SHA-256 hash of each file is recorded. The ILNet and AGPCNet changes are applied when the networks are built.