SciVid: Cross-Domain Evaluation of Video Models in Scientific Applications

Supplementary Material

In this supplementary material, we provide additional information on baselines (Sec. A), implementation details (Sec. B), further experiments (Sec. C), and qualitative results (Sec. D). We also encourage the reader to watch our narrated video, 5334.mp4. Please refer to the main paper for bibliography references.

A. Baselines

Weather forecasting. We compare our best models to the following weather forecasting baselines:

- Eulerian Persistence is commonly used in the weather forecasting literature. It produces constant forecasts, where each frame is equal to the last input frame (also called the initial condition).
- IFS HRES is the main baseline considered when evaluating deterministic data-driven models. It comes from operational forecasts created with ECMWF's IFS model, typically regarded as the best global, medium-range weather forecasts. Following GraphCast, for the validation set, we evaluate forecasts initialized at 06 and 18 UTC, to ensure equal look ahead (of 3h) for HRES and data-driven models. However, as HRES forecasts are only run for up to a lead time of 3.75 days, we use HRES forecasts initialized at 00 and 12 UTC (which are run for 10 days) for longer lead times. For the test set, following Weatherbench 2 [48], we ignore the difference of look ahead times and use forecasts initialized at 00 and 12 UTC for all methods, including HRES. Furthermore, following common practice, we evaluate IFS forecasts against operational analyses as ground truth, rather than ERA5, to avoid putting IFS at an unfair disadvantage compared to data-driven models.
- GraphCast [32] is a state-of-the-art deterministic model for weather forecasting. For the validation set (2018), we use a version trained from 1979 to 2017 (included); whereas for the test set (2020), we use a version trained from 1979 to 2019 (included).
- GenCast [45] further improves upon GraphCast with a probabilistic diffusion-based approach. Like for GraphCast, in each setting, we use a version trained up until the year before the evaluation set. 50 members are used for the ensemble on the validation set, and 56 members on the test set. EDA initialization is used on the validation set and normal initial conditions are used when running on the test set. Due to the important inference cost involved in evaluating GenCast, we leave these small inconsistencies unaddressed, as they have been observed in practice to have negligible impact on performance.

Results for these baselines were communicated to us privately by the GenCast and Weatherbench 2 authors. Each model is *evaluated* at its native *training* resolution, i.e. 0.25°. In practice, we found that when *evaluating* on 1° resolution the model *trained* at 0.25° resolution, the results are indistinguishable from the ones we report. When *training* and *evaluating* on 1° resolution, performance for GraphCast and GenCast models are very similar to 0.25° results, except for very short lead times. For additional details on the evaluation on IFS HRES and GraphCast baselines, we refer the reader to the Weatherbench 2 paper [48]. For additional details on GenCast, we refer the reader to [45].

B. Implementation details

In this section, we describe our readouts (Sec. B.1) and evaluation metrics for each task (Sec. B.2) as well as our Weatherbench 2 conditioning setup (Sec. B.3).

B.1. Readout design and training details

We train all readouts with a batch size of 32. For weather forecasting where we observe overfitting for time steps

above 10k in most experimental setups. Therefore, unless explicitly mentioned, models for FlyVsFly, CalMS21, STIR and Digital Typhoon are trained for 40k steps following [11] while models trained on Weatherbench 2 are trained for 10k steps. We use the AdamW optimizer, with weight decay of $1e^{-4}$, a learning rate cosine schedule from $3e^{-4}$ to $1e^{-7}$, and a linear learning rate warmup for 1k steps. Please refer to Tab. B.1 for details on readout architectures.

Classification. The readout module consists of 16 heads, with 64 and 8 feature channels per head for CalMS21 and FlyVsFly respectively. We also apply data augmentations on the input videos which we found beneficial. In particular, we resize FlyVsFly and CalMS21 images by a factor of 1.4 and 2 times the model input resolution respectively and randomly extract a crop of the model input resolution. Note that weight decay is set to 0 for FlyVsFly only and that for CalMS21, we additionally apply random left-right flipping as well as Gaussian blurring with a kernel size of 36 during clip preprocessing.

Tracking. The readout module consists of 8 heads, with 128 feature channels per head. We trained it on 16-frame windows randomly selected from the 24-frame videos in the Kubric MOVi-E dataset [22], which features static and dynamic objects rendered on photorealistic backgrounds. Supervision was derived by sampling 64 point tracks per window, prioritizing foreground objects.

Weather forecasting. We use a standard four-stage DPT design, with all intermediate feature dimensions of 1024. We refer the reader to [47] for in-depth details on the DPT architecture. We use gradient clipping of 1 which we found to prevent training instabilities. To account for the fact that different channels have values of different orders of magnitude, we follow [32] by weighting the loss terms by the inverse standard deviation of the time difference residuals. As input to the readout, we provide the backbone features corresponding to the last time step. We observed that additionally using features from other prior frames to be detrimental to the results. As opposed to the original DPT implementation [47], which inputs different features depths at different upsampling stages, we provide features from a single backbone level to all upsampling stages, as we found this to work better in our setting. We further find it beneficial to remove the penultimate convolution layer and increase the intermediate feature dimensions.

Pressure forecasting. The readout module consists of 16 heads, with 64 feature channels per head. The evaluation is performed on the first 24 frames of the typhoon sequence. While the original training [29] uses only the first 24 frames of the sequence, we empirically find that randomly shifting the start of the 24-frame sequence between frames 0 to 8 improves performance. Our readout regresses the difference to the average train pressure, of 983.9 hPa, computed across the train set.

Eval	Architecture

FlyVsFly	`CrossAttention(`
	`  qkv_size=128,`
	`  num_heads=16)`
	
	`Linear(output_size=7)`

CalMS21	`CrossAttention(`
	`  qkv_size=1024,`
	`  num_heads=16)`
	
	`Linear(output_size=4)`

STIR	`CrossAttention(`
	`  qkv_size=1024,`
	`  num_heads=8)`
	
	`Linear(output_size=4)`

Weatherbench 2	`DensePredictionTransformer(`
	`  num_blocks=4,`
	`  feature_dim=1024)`
	
	`Conv(`
	`  kernel_size=3 × 3,`
	`  feature_dim=512)`
	
	`Conv(`
	`  kernel_size=1 × 1,`
	`  output_size=48 (16 × 3))`

Digital Typhoon	`CrossAttention(`
	`  qkv_size=1024,`
	`  num_heads=16)`
	
	`Linear(output_size=12)`

Table B.1. Architectures of readout modules for different tasks. For Weatherbench 2, `num blocks` refers to the number of reassamble and fusion operations. Please refer to [47] for further details on DPT components.

B.2. Evaluation metrics

Classification. We follow the multilabel average precision implementation from VideoPrism [70]. In particular, the background class is excluded by computing the average precision after removing the *not interacting* results from both the ground truth and predictions.

Tracking. We compute the percentage of predicted track endpoints that fall within specific distance thresholds of their corresponding ground truth locations. These thresholds are set at several pixel distances (4, 8, 16, 32, and 64 pixels) to assess accuracy at varying levels of precision. An average accuracy across these thresholds, denoted as δ^x_{avg}, is then computed to provide a single, overall measure of tracking performance. On the validation and test sets, the query and target points are extracted separately, with no matching available. Following [1], a simple closest-target

point heuristic is used to assign initial query points to their corresponding final location. The computed accuracy is therefore an estimate of the performance, as some tracked queries and target points can be erroneously matched.

Pressure forecasting. The paper introducing Digital Typhoon [29] reports root mean squared errors for the following time steps: 1, 2, 3, 6, 12. For comparing model performance using a single value, we compute the error averaged across these time steps. RMSE for Digital Typhoon is therefore defined as following:

$$\text{RMSE} = \frac{1}{T=5} \sum_{t \in \{1,2,3,6,12\}} \sqrt{\frac{1}{K} \sum_{k=1}^{K} (f_{k,t} - o_{k,t})^2} \quad (1)$$

where k is the sample index among K samples, f_i is the forecasted central pressure at time step i and o_i is the ground truth typhoon central pressure for this time step.

Weather forecasting. Following common practice [48], we measure performance using the area-weighted root mean squared error (wRMSE). Area-weighting is used because on an equiangular latitude-longitude grid, grid cells at the poles have a much smaller area compared to grid cells at the equator. For a grid cell with latitude index i, the latitude weight $w(i)$ is computed as:

$$w(i) = \frac{\sin \theta_i^u - \sin \theta_i^l}{\frac{1}{I} \sum_i^I (\sin \theta_i^u - \sin \theta_i^l)}, \quad (2)$$

where θ_i^u and θ_i^l indicate upper and lower latitude bounds, respectively. For each variable (corresponding to a specific pressure level), the wRMSE is defined as follows:

$$\text{wRMSE} = \sqrt{\frac{1}{TIJ} \sum_t^T \sum_i^I \sum_j^J w(i)(f_{t,i,j} - o_{t,i,j})^2}, \quad (3)$$

where $f_{t,i,j}$ and $o_{t,i,j}$ are respectively the forecasted and the ground truth values at spatio-temporal position (t, i, j).

B.3. Weatherbench 2 conditioning

When conditioning on all other upper-level variables (*w. cond.*) for the Weatherbench 2 task, we use atmospheric variables proposed in GraphCast [32]: geopotential, specific humidity, temperature, u component of wind, v component of wind and vertical velocity. The 13 vertical levels are folded into the channel dimension for atmospheric variables resulting in 78 channel. In order to adapt the models to the larger number of channels, we reinitialize the first input layer to accommodate the larger number of channels, using the same random initialization as in the original implementation for VideoMAEv2-g [62] and 4DS-e [11]. In Sec. C.6 we discuss the results *w. cond.* for which we reinitialize the embedding layer to provide more than three variables as input to these two models.

C. Additional experiments

Here, we provide additional results on backbone finetuning (Sec. C.1), efficient backbone low-rank adaptation [24] (Sec. C.2), data-efficiency (Sec. C.3), per-threshold SOTA comparison on STIR (Sec. C.4), SOTA comparison for Digital Typhoon (Sec. C.5), conditioning, readout comparison and prediction strategy on Weatherbench 2 (Sec. C.6, C.7, C.8), backbone feature depth (Sec. C.9), backbone and readout training length (Sec. C.10, C.11), single input frame performance (Sec. C.12) and a noise study of our evaluations (Sec. C.13).

C.1. Backbone finetuning

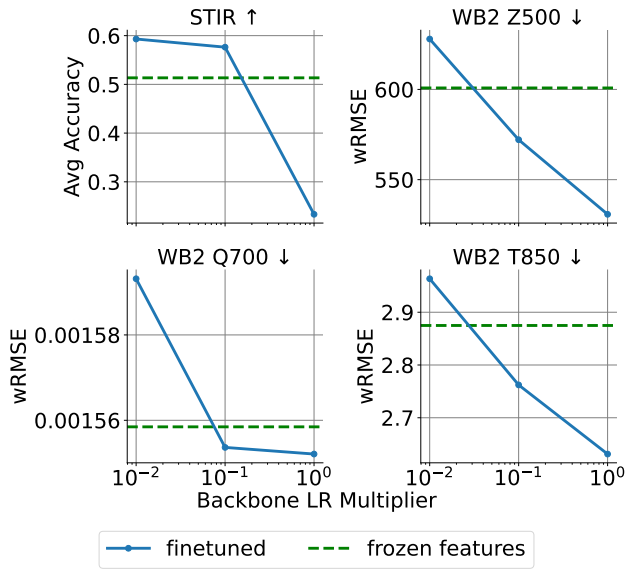


Figure C.1. **Backbone finetuning STIR & WeatherBench 2.** Finetuning the backbone helps improve the final performance for both Weatherbench 2 and STIR provided an appropriate task-dependant learning rate is applied to the backbone. A 100x smaller backbone learning achieves the best performance for STIR while applying the same learning rate to both the readout and the backbone yields the best results for Weatherbench 2.

We reach state-of-the-art performance by training readouts on top of frozen features for three out of five tasks (CalMS21, FlyVsFly and Digital Typhoon). We investigate whether finetuning the backbone improves the final performance on STIR and Weatherbench 2, the two tasks for which an important gap compared to the state of the art

Benchmark % of params	Frozen ❄ 0	LoRA r=4 0.3	LoRA r=32 2.3	Finetuned 🔥 100
STIR ↑	51.3	57.4	58.4	**59.3**
WB2 T850 ↓	2.87	2.74	2.69	**2.63**

Table C.1. **Comparison of different adaptations**. With the 4DS-e backbone, LoRA adaptation with rank (r) 32 approaches the performance of full finetuning at a fraction of the parameter cost.

remains. For this analysis, we finetune the 4DS-e backbone in addition to training the readout, using different fractions of the readout learning (1, 0.1 or 0.01) to account for the fact that the pretrained backbone might benefit from smaller updates, while larger updates could be needed for the readout which is trained from random initializations. For all other settings, we follow the protocol described in Sec. 4 of the main paper and Sec. B of supp. mat. In both cases we observe that performance can be improved at the additional computation cost of finetuning the backbone. In Fig. C.1, we see that updates of the backbone with the same learning rate as the readout yields best results on Weatherbench 2, which has a large domain gap with the original video input distribution. Using a learning rate two orders of magnitude smaller for the backbone compared to the readout results in the best performance on STIR. Variations in the backbone learning rate multiplier incur large variation in performance. We note that the optimal learning rate for one task (0.01 for STIR, 1 for Weatherbench 2) yields results that are *worse* than frozen features on the other task, indicating that this parameter needs to be carefully tuned for each task. Finetuning the backbone with appropriate learning rates applied to the backbone updates helps significantly, but a performance gap compared to the state of the art remains for both tasks, which we detail in Fig. D.3 for STIR and Fig. 4 for Weatherbench 2.

C.2. Low rank adaptation

To investigate lightweight backbone adaptation techniques, in Fig. C.2, we compare three training strategies on STIR and Weatherbench 2: training the readout with (i) a frozen ❄4DS-e backbone, (ii) full backbone finetuning 🔥and (iii) finetuning with low rank adaptation (LoRA) [24]. We observe that LoRA finetuning strikes an effective balance and that setting the LoRA rank r to 32 yields some improvement over a more lightweight adapter (r=4). When setting r to 32, results approaches the full finetuning performanche, while requiring training only 2.3% parameters compared to the backbone full finetuning (see Tab. C.1).

C.3. Data-efficiency

While SciVid offers a diverse suite of scientific applications, it does not focus on the development of *data-efficient* adaptation methods. Indeed, datasets within SciVid contain

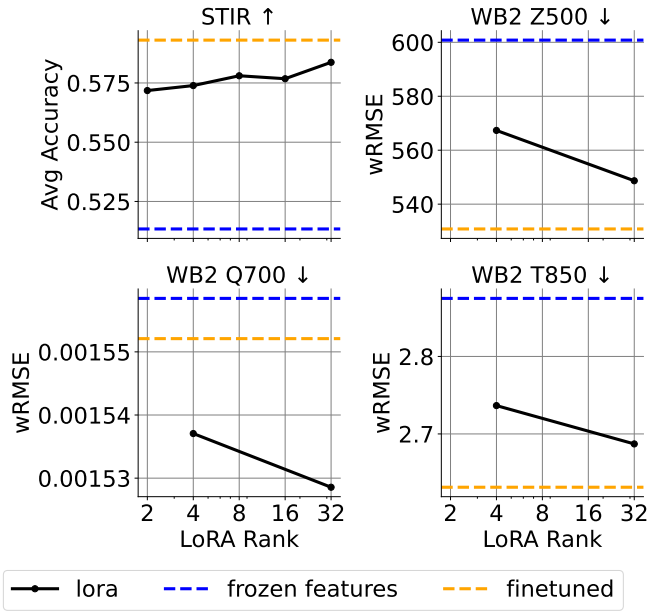


Figure C.2. **Comparison of different adaptation strategies**. With the 4DS-e backbone, LoRA adaptation with rank (r) 32 approaches the performance of full finetuning at a fraction of the parameter cost.

tens of thousands of training samples, with the exception of Digital Typhoon which contains 696. Collecting such datasets requires considerable effort, which might be prohibitive for some applications.

To take a step towards addressing this limitation, we provide initial comparisons of models in data-constrained settings. Fig. C.3 illustrates the relationship between frozen (❄) backbone performance and the fraction of training data used for two models, 4DS-L and 4DS-e. For Digital Typhoon, we divide the dataset size by powers of 2, with 1/8 corresponding to 100 samples. For CalMS21, which has 27k training clips, we divide the dataset size by powers of 4, with 1/256 corresponding to 100 samples. For ERA5, we use powers of 3 to preserve forecast times spanning a full day (00/06/12/18), with 1/81 corresponding to 700 samples. We observe that 4DS-e, which performs better in the full data regime than 4DS-L, generally maintains its advantage when we reduce the amount of training data. Finally, this analysis also highlights room for improvement in low data regimes, and provides a more challenging setting that can potentially better differentiate model capabilities.

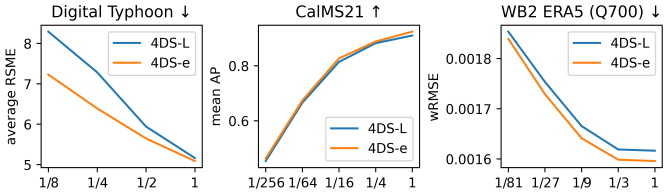


Figure C.3. ❄ **Model performance vs. fraction of training data.**

	fraction	4DS-L	4DS-e
Digital Typhoon (RMSE ↓)	1	5.16	5.09
	1/2	5.94	5.65
	1/4	7.28	6.39
	1/8	8.29	7.23
CalMS21 (mAP↑)	1	91.0	92.4
	1/4	88.2	88.8
	1/16	81.4	82.6
	1/64	66.5	67.4
	1/256	45.3	46.3
WB2 ERA5 Q700 (wRMSE ↓)	1	1.62e-03	1.60e-03
	1/3	1.62e-03	1.60e-03
	1/9	1.67e-03	1.64e-03
	1/27	1.75e-03	1.73e-03
	1/81	1.85e-03	1.84e-03

Table C.2. ❄ Model performance vs. fraction of training data.

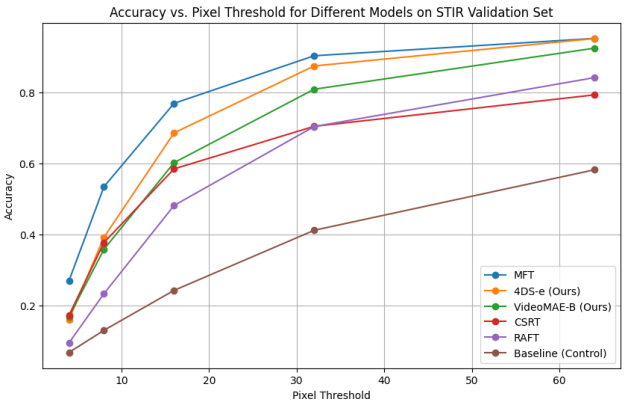


Figure C.4. **SOTA comparison on STIR.** We indicate for different models the percentage of predicted track endpoints that fall within specific distance thresholds (4, 8, 16, 32, and 64 pixels) of their corresponding ground truth locations. *Ours* refer to our readout with 4DS-e or VideoMAE-B backbones with finetuning (🔥).

C.4. SOTA comparison on STIR

Fig. C.4 illustrates tracking performance on STIR at various threshold distances (4, 8, 16, 32, and 64 pixels). MFT [41] surpasses other methods at all thresholds except the largest (64 pixels), where it performs comparably to our readout on 4DS-e with finetuning. The performance gap between methods are generally consistent across thresholds. A notable exception is CSRT [35], which outperforms RAFT [59] at smaller distances but exhibits lower performance at larger distances.

C.5. SOTA comparison Digital Typhoon

In Fig. C.5, we qualitatively compare our readout training with the 4DS-L and V-JEPA-H backbones to the method from [29], the *mean train pressure* baseline and the *copy last pressure* oracle. Our method outperforms both [29] and the *mean train pressure* baseline on both the validation and test sets for all time steps. We observe that the *copy last pressure* oracle, which uses the last input central pressure value, achieves a lower RMSE at smaller time steps, benefitting from temporal continuity. At later time steps, our readout with the 4DS-e backbone consistently outperforms all approaches.

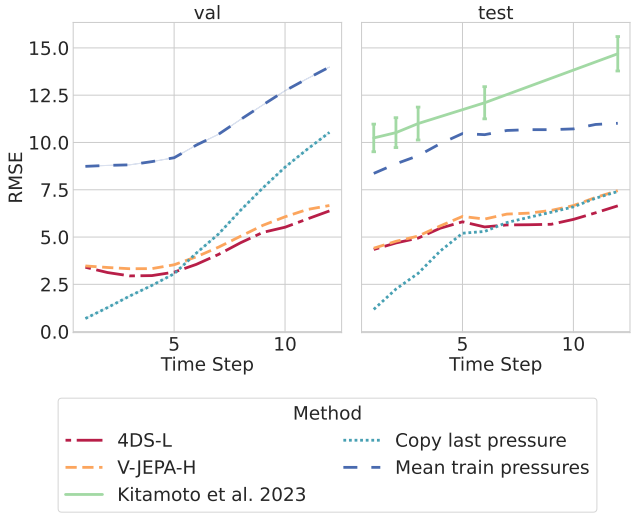


Figure C.5. **SOTA comparison Digital Typhoon.** We significantly outperform the method from Kitamoto et al. [29] (numbers obtained from their paper). While the *copy last pressure* baseline achieves a lower RMSE at smaller time steps, our readout training using frozen features (❄) from 4DS-L performs best overall.

C.6. Weatherbench 2 *w. cond.*

We compare finetuning *w. cond.* and *w.o. cond.* (see Sec. B.3 for more details) for the two backbones which perform best on Weatherbench 2: 4DS-e and VideoMAEv2-g. We observe that training *w. cond.* yields small improvements on geopotential forecasting (Z500) while *w.o. cond.* works better for temperature and specific humidity forecasting for the two backbones. These initial results highlight the need to develop adaptation approaches which can efficiently leverage the extra input information while efficiently leveraging representations learnt during pretraining.

Model	*w. cond.*	Z500 ↓	T850 ↓	Q700 ↓
4DS-e	✗	531	2.63	1.55e-03
4DS-e	✓	549	2.71	1.54e-03
VideoMAEv2-g	✗	**512**	**2.57**	1.51e-03
VideoMAEv2-g	✓	539	2.65	**1.49e-03**

Table C.3. **Weatherbench 2 results with and without conditioning.** We compare finetuning (🔥) *w. cond.* and *w.o. cond.* for the two strongest backbones on the Weatherbench 2 task on the validation set.

C.7. Readout comparison on Weatherbench 2

We experimented with a pure attention-based readout for Weatherbench 2, compared to the DPT-based [47] readout used. We follow the attention readout design in [11] for depth prediction and adapt it by predicting a 3 channel output for all target frames. The results are shown in Tab. C.4. We evaluate the performance both when the backbone model is frozen and when it is finetuned together with the readout. We observe that in both scenarios, the DPT readout outperforms the attention one by a large margin.

Model	Readout	Z500 ↓	T850 ↓	Q700 ↓
VideoMAEv2-g ❄	Attn.	892	4.09	2.09e-03
VideoMAEv2-g ❄	DPT	594	2.82	1.56e-03
VideoMAEv2-g 🔥	Attn.	853	3.98	2.09e-03
VideoMAEv2-g 🔥	DPT	**587**	**2.79**	**1.54e-03**

Table C.4. **Attention vs. DPT-based readout on Weatherbench 2.** We compare the two readouts for the strongest backbone model on this task. In both the frozen (❄) and finetuning (🔥) settings, the DPT readout outperforms the attention one by a large margin.

C.8. Prediction strategy on Weatherbench 2

We ablate the prediction strategy on the Weatherbench 2 by either predicting the target variables directly or residuals with respect to the last input weather state. Results in Tab. C.5 show that when finetuning *w. cond.* predicting the residual yields the best quantitative performance. Interestingly, increasing training steps from 10k to 40k marginally improves the results depending on the variable when predicting residuals but significantly helps in the direct target prediction setting. We posit that residual training allow to reach reasonable performance fast, but limits potential further gains. In contrast, direct target prediction is more challenging, but may allow to reach a superior optimum with sufficient training. We also compare the two approaches qualitatively in Fig C.6 and observe that residual prediction yields high-frequency artefacts, in particular on the humidity Q700. Direct prediction of the targets removes these artefacts, but tends to over-smooth the results, leading to lower overall accuracy.

Model	Steps	Res.	Z500 ↓	T850 ↓	Q700 ↓
VideoMAEv2-g	10k	✗	571	2.78	1.52e-03
VideoMAEv2-g	10k	✓	**539**	2.65	1.49e-03
VideoMAEv2-g	40k	✗	563	2.67	**1.45e-03**
VideoMAEv2-g	40k	✓	540	**2.64**	1.48e-03

Table C.5. **Residual vs. direct prediction on Weatherbench 2.** We compare directly predicting the target variables or a residual (res.) with respect to the last input weather state. We show results for our readout trained on top of the VideoMAEv2-g backbone with finetuning (🔥) and *w.o. cond.*. While at short training lengths, predicting the residual yields the best quantitative results, at longer ones, the performance gap is marginal.

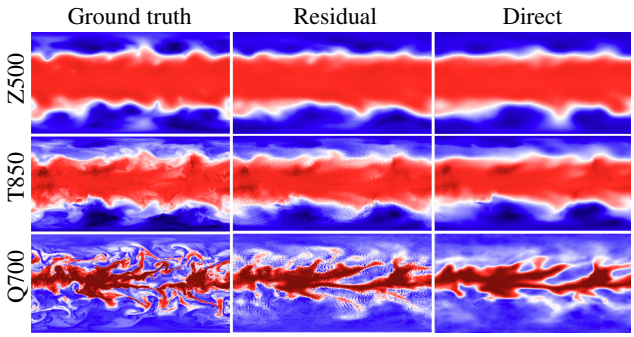


Figure C.6. **Residual vs. direct prediction on Weatherbench 2.** We compare directly predicting the target variables or a residual with respect to the last input weather state. We show results on the last predicted frame for our readout on top of the VideoMAEv2-g backbone with finetuning (🔥) and *w. cond.*. We observe that residual prediction yields high-frequency artefacts, in particular on the humidity Q700. Direct prediction of the target variable removes these artefacts, but potentially leads to over-smoothing, highlighting different failure modes of the two approaches.

C.9. Backbone feature depth

We investigate where in the model are the best representations for each task by extracting features at different model depths from the 4DS-e backbone – 10%, 25%, 50%, 75% and 95% (for example, 75% corresponds to layer 42 out of 56 for 4DS-e). We observe in Fig. C.7 that the features from the layer at 95% depth reaches a good compromise across tasks. One notable exception is Weatherbench 2 where for all forecasted variables, early features perform better, and performance degrades progressively with model depth for each of the three forecasted variables.

C.10. Backbone training length

In Fig. C.8, we investigate the importance of pretraining duration, measured by the number of examples the 4DS-e backbone sees during training. We observe that longer pretraining helps cross tasks, with peak performance around

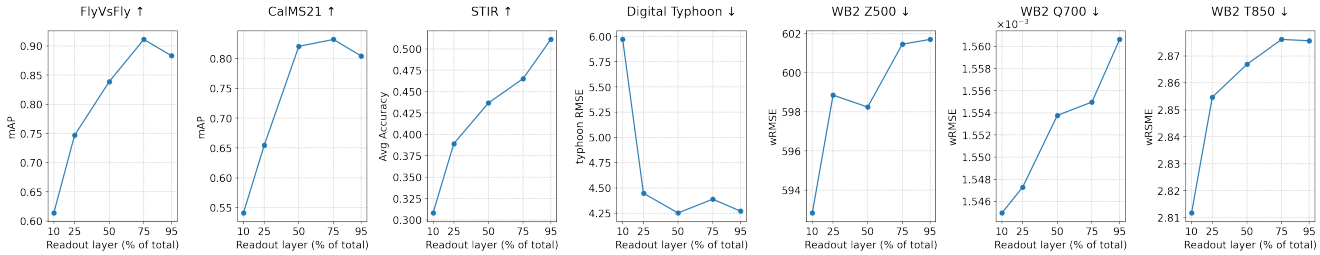


Figure C.7. **Feature depth.** We show the performance of the frozen (❄) 4DS-e model when we attach the readouts at different layers of the model (as a percentage of the total number of self-attention blocks, 56 for 4DS-e). We observe that features from the layer at 95% depth offers a good performance across tasks, similarly to findings in [11], except for Weatherbench 2 where early features perform better.

1B seen examples for most tasks. On Digital Typhoon, the observed irregular variations are likely caused by the noise levels for this evaluation (see C.13 for more details).

C.11. Readout training length

We investigate whether training the readouts for longer improves performance across tasks. In Fig. C.9, we observe that this is the case for STIR and FlyVsFly. For CalMS21 and Digital Typhoon, we observe best performance at 80K steps. We find our selected 40k step training schedule to be a good compromise across all tasks for performance and evaluation speed. Further reducing the training to 20k steps negatively affects performance – for example, on CalMS21 and FlyVsFly, reducing the number of training steps from 40k to 20k results in a 3% absolute mAP decrease – while increasing the training to 80k yields at best a +1% mAP increase. The readout training length has little impact on Weatherbench 2, within noise levels (see Tab. C.6).

C.12. Temporal dynamics for frame-based model

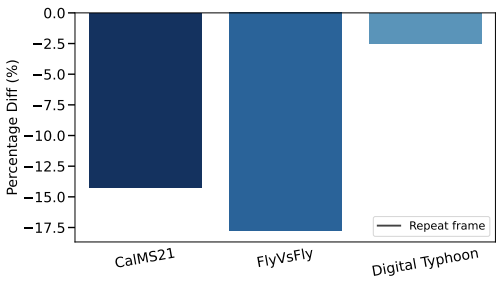


Figure C.10. **Performance change when relying on a single frame for the readout training.** In this figure we show the (percentage) performance when training a readout with a frozen DINOv2-g backbone (❄) and a single video frame repeated along the temporal dimension, compared to using all original video frames. We observe a consistent performance drop when using a single frame, highlighting the importance of temporal understanding in our benchmarks.

In this section, we investigate frame-based performance on a subset of SCIVID tasks which are easily amenable to this setup. For this, compare how performance changes when training a readout with a frozen DINOv2-g backbone when repeating a single frame along the temporal dimension with respect to its counterpart trained on the original videos. We repeat the middle frame for CalMS21 and FlyVsFly datasets, and use the last frame for Digital Typhoon, which should be the most informative with respect to the task. We apply the same logic on both training and evaluation sets for consistency. We show the results in Fig. C.10. We observe that training our readout with a frame-based model using a single frame compared to all frames consistently degrades performance, with the largest drop on FlyVsFly – this highlights the importance on temporal understanding in our selected benchmarks.

C.13. Evaluation noise

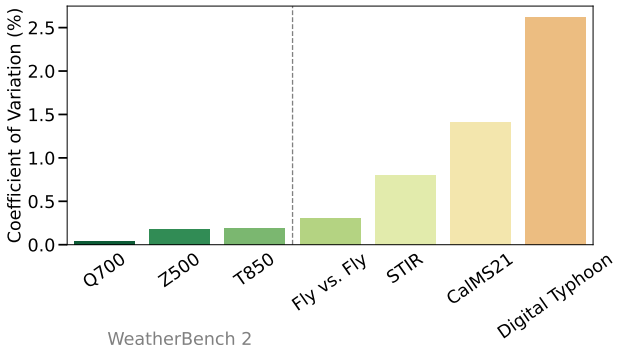


Figure C.11. **Evaluation noise.** We compare the noise levels for each task, with our readout on top of the frozen 4DS-e backbone (❄), across different value ranges by computing the coefficient of variations. We observe varying levels of relative noise across tasks, with the highest levels for Digital Typhoon [29], which has the smallest evaluation set.

Each dataset and evaluation comes with a different level of stochasticity across experiments which are identical up to random seed initialization. We evaluate the noise of each

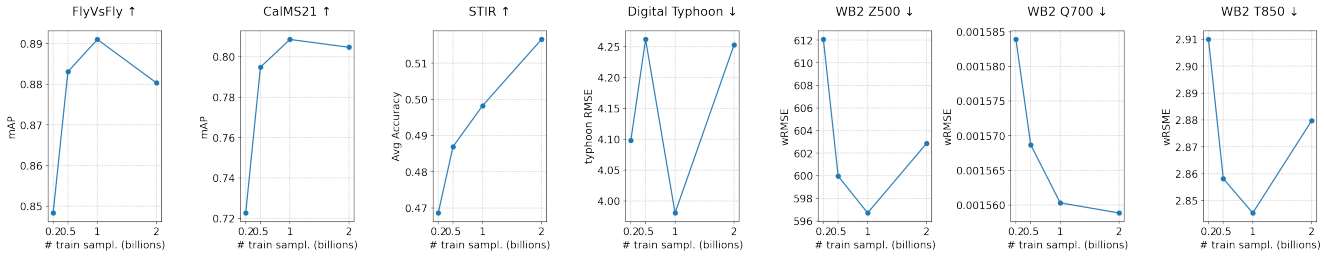


Figure C.8. **Backbone training length.** We show the performance of the readout training on top of the frozen 4DS-e model (❄), when the backbone has seen more or less examples during pretraining. We observe that longer pretraining helps overall across tasks, with peak performance around 1B seen examples for all tasks, except for STIR and Weatherbench 2 Q700.

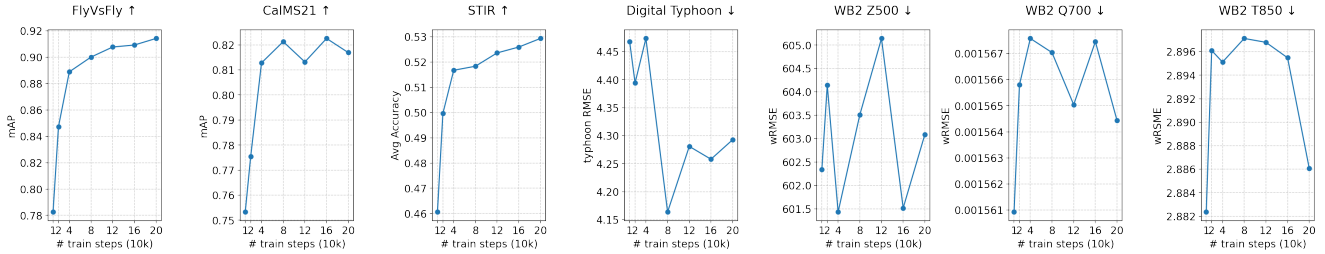


Figure C.9. **Readout training steps.** We show the performance of the readout trainings on top of the frozen 4DS-e model (❄) when varying the training duration. We observe that longer training helps continuously for STIR and FlyVSFly. For CalMS21 and Digital Typhoon, we observe peak performance around 80K steps. For Weatherbench 2, the best overall results are at 10k steps.

					WeatherBench 2		
	CalMS21	FlyVsFly	STIR	Digital Typhoon	Z500	T850	Q700
Mean	0.791	0.889	0.746	4.32	603	2.88	1.56e-03
Std	1.12e-02	2.68e-03	5.97e-03	0.113	1.04	5.35e-03	6.37e-07

Table C.6. We assess the stochasticity of our evaluations by computing the standard deviation for each task across 5 random seeds. We train the readout with frozen features from the 4DS-e model (❄) as input in the same setting as Tab. 5 of the main paper.

evaluation for our readout with the frozen 4DS-e backbone in the setup described Sec. 5 of the main paper. In Fig. C.11, we observe varying levels of relative noise across tasks, with the highest levels for Digital Typhoon [29], which has the smallest evaluation set. To complement Fig. C.11, we provide in see Tab. C.6 the corresponding mean and standard deviations computed across 5 seeds. In particular, on Digital Typhoon we observe a standard deviation of 0.113 for a mean value of 4.32 – this confirms the important noise levels observed in Fig. C.11.

D. Additional qualitative results

Here, we first qualitatively analyse the input distributions across tasks (Sec. D.1). We then show additional qualitative results, for FlyVsFly and CalMS21 (Sec. D.2), STIR (Sec. D.3) and Weatherbench 2 (Sec. D.4).

D.1. Distribution of input variables

To highlight some of the challenges associated with our benchmark, we plot the histograms of input values in the different tasks of SCIVID in Tab. D.1, and compare them to the histograms of pixel values in natural image videos which are typically used for ViFM pretraining. While the pretraining distribution of ViFMs is often not released, it is frequently mined from YouTube. As a proxy, we use the public Kinetics dataset [27] and visualize the histogram of the pixel values in that dataset in Tab. D.1 top. In the 5 tasks that we consider, we interpret the input values either as greyscale or RGB channels for tasks that involve frames with single-scalar and multi-scalar data input respectively. As can be observed in Tab. D.1 the distribution of pixel values in our tasks differs significantly from that observed in a typical video datasets, which consists of "natural" online videos, e.g., depicting people performing various actions.

This shift in data distribution highlights the unique characteristics in the tasks and data in SCIVID, and underscores

the need to develop methods that can accommodate such variability. We also point out that in natural videos there is often a strong correlation between the values in the three RGB channels, which might be captured by a trained ViFM. On the other hand, as can be seen for Weatherbench 2 in Tab. D.1, the three variables that we model have very different correlation properties, which might make it difficult to exploit the pretrained model. In this plot, the RGB channels correspond to, respectively, Geopotential@500, Temperature@850 and Specific Humidity@700. This shift, again, highlights the need for generalizable models that can adapt to potentially significantly different new data, by efficiently taking advantage of the available supervision.

D.2. FlyVsFly and CalMS21

We illustrate the animal behavior classification performance for our readout with the frozen 4DS-e model by displaying the confusion matrices on FlyVsFly in Fig. D.1 and CalMS21 in Fig. D.2. In both cases we observe instance of ambiguity and possible labelling errors in Fig. D.2.

D.3. STIR

In Fig. D.3 we compare the performance of our readout using the 4DS-e and DinoV2-g backbones with finetuning. We observe significant tracking errors when using the image-based DinoV2-g features while 4DS-e demonstrates stronger performance and successful tracking across challenging scenarios, including camera motion and occlusion events.

D.4. Weatherbench 2

We provide a qualitative comparison between the residual (i.e., predicting the differences between the target and the last seen frame) and direct target prediction in Fig. C.6. We plot the results of the best-performing backbone in the finetuning setting, and visualize the predictions on the last frame for a particular test sample. We observe that by predicting the residual (Fig. C.6 middle column), the model can forecast more detailed results but produces visible high-frequency artefacts. On the other hand, direct prediction (Fig. C.6 right column) does not exhibit these artefact but seems to over-smooth the results. These two complementary failure modes suggest that capturing high-frequency details using the readouts we experimented with without producing artefacts is non-trivial in this task.

We also show a visual comparison between our readout on top of a frozen ViFM (4DS-e) and an image-based backbone (DinoV2-g), both *w.o. cond.*, in Fig. D.4. We show the prediction of the first and the last (16th) future frame for a particular test sample. We note that both backbones lead to strong artefacts (discussed above) that are especially visible at the last predicted frame. Moreover, quantitatively, in this particular example, 4DS-e outperforms DinoV2-g by approximately 20%. However, it is difficult to note specific differences between the two predictions, highlighting that qualitative interpretation is not easy for an untrained eye.

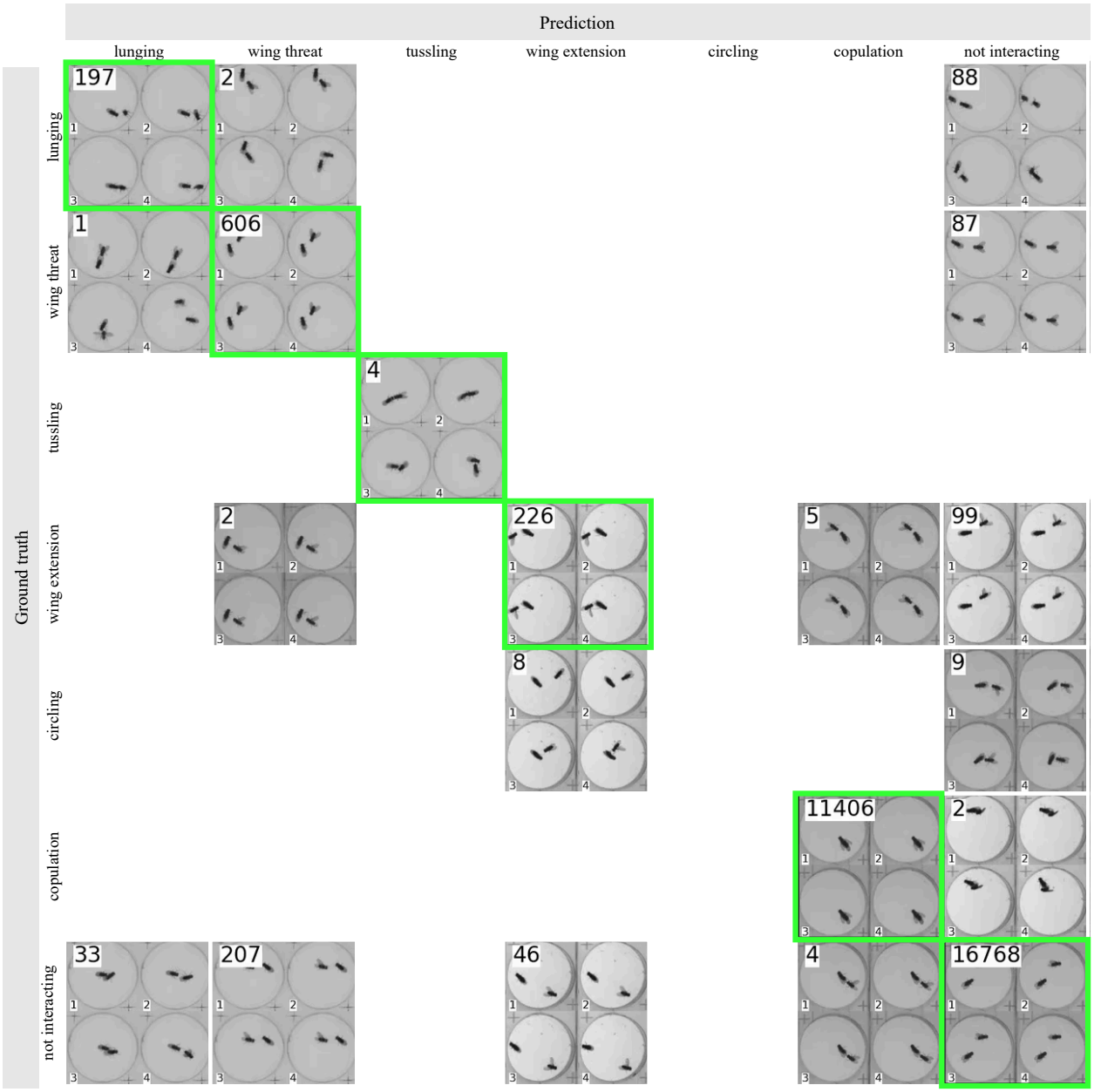


Figure D.1. **Visual confusion matrix on the FlyVsFly dataset.** We show results from our readout trained on top of the frozen 4DS-e backbone (❄). This matrix is generated from ∼30k samples of the validation set, after filtering out multi-label video clips. Each cell represents a ground truth/prediction pair and displays four representative frames (first, two middle, and last) from a corresponding video clip. The number in the top-left corner of each cell indicates the total count of videos for that specific pair. We observe instances of ambiguity, such as the video labeled as "circling" but predicted as "wing extension" where the behavior indeed transitions to include "wing extension" towards the end. This is due to the fact that the behavior active around the middle frame was selected during labeling, while the model may capture behavior prominent at other moments.

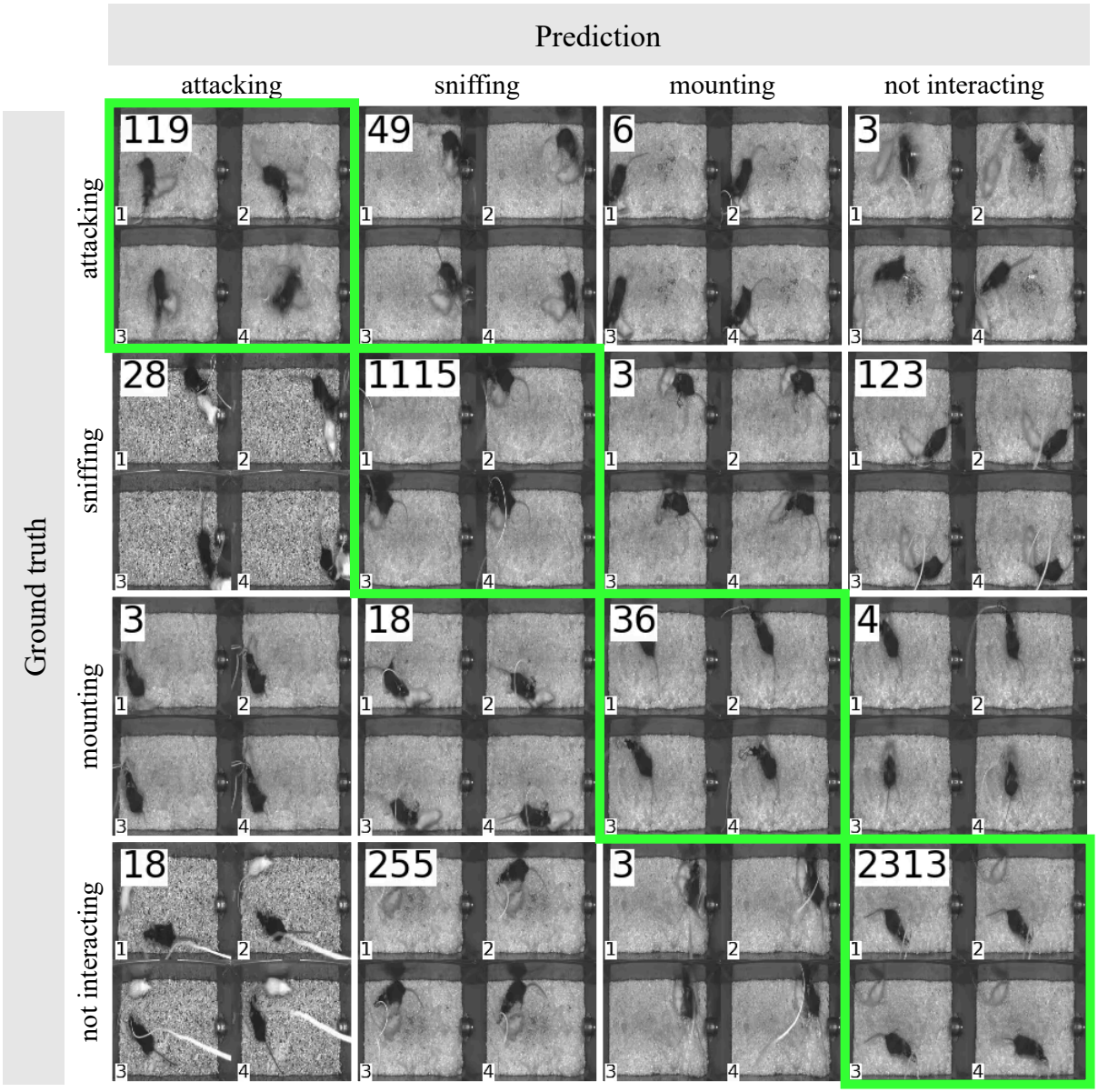


Figure D.2. **Visual confusion matrix on the CalMS21 dataset.** We show results from our readout trained on top of the frozen 4DS-e backbone (❄). This matrix is generated from ∼4k samples of the validation set, after filtering out multi-label video clips. Each cell represents a ground truth/prediction pair and displays four representative frames (first, two middle, and last) from a corresponding video clip. The number in the top-left corner of each cell indicates the total count of videos for that specific pair. Again, we observe instances of ambiguity, such as the video labeled as "not interacting" but predicted as "mounting", but where the animals are clearly on top of one another.

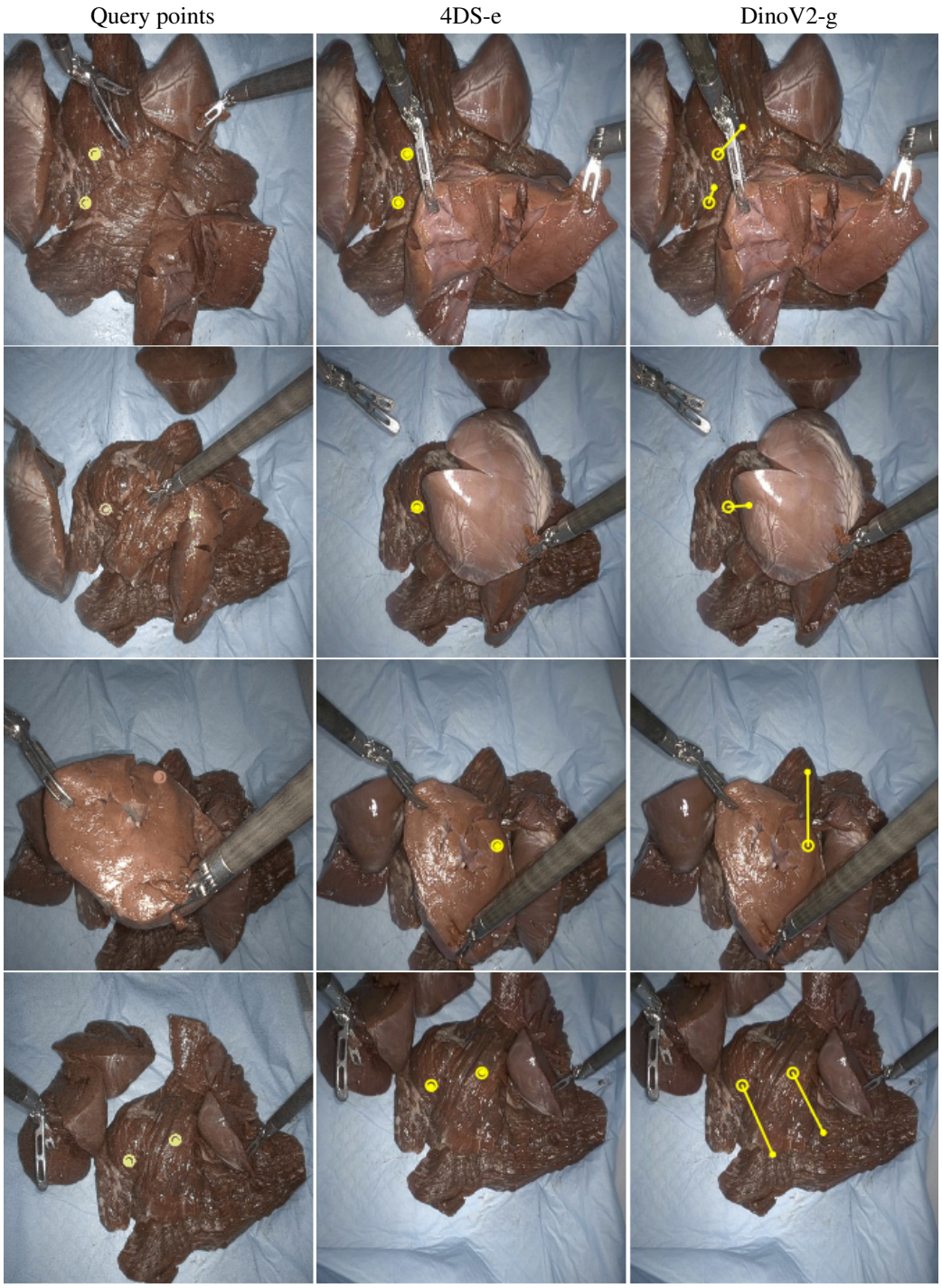
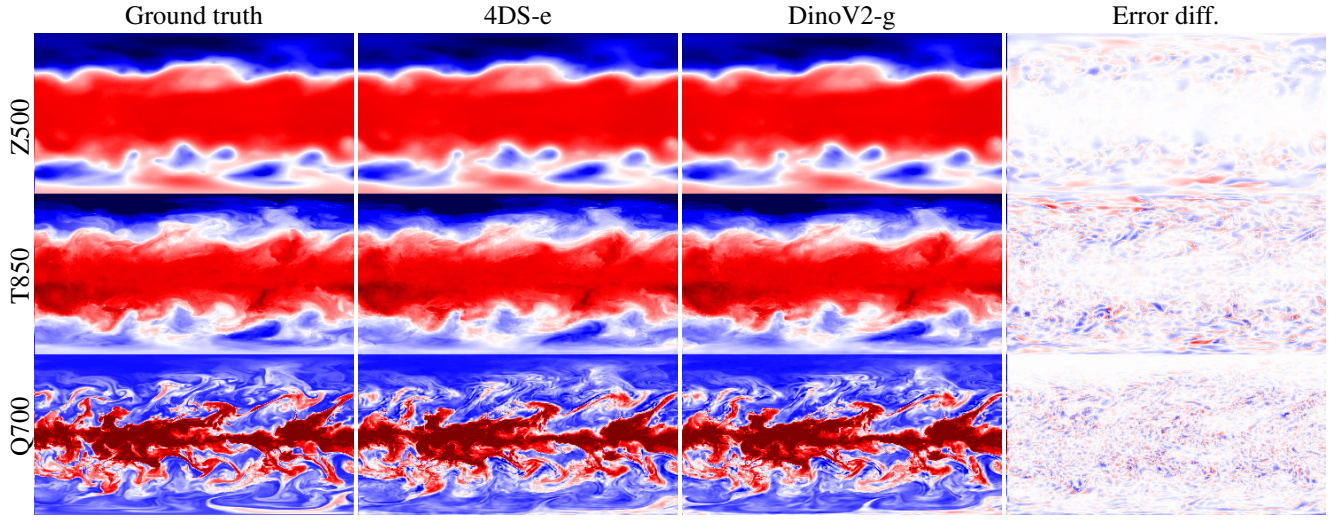
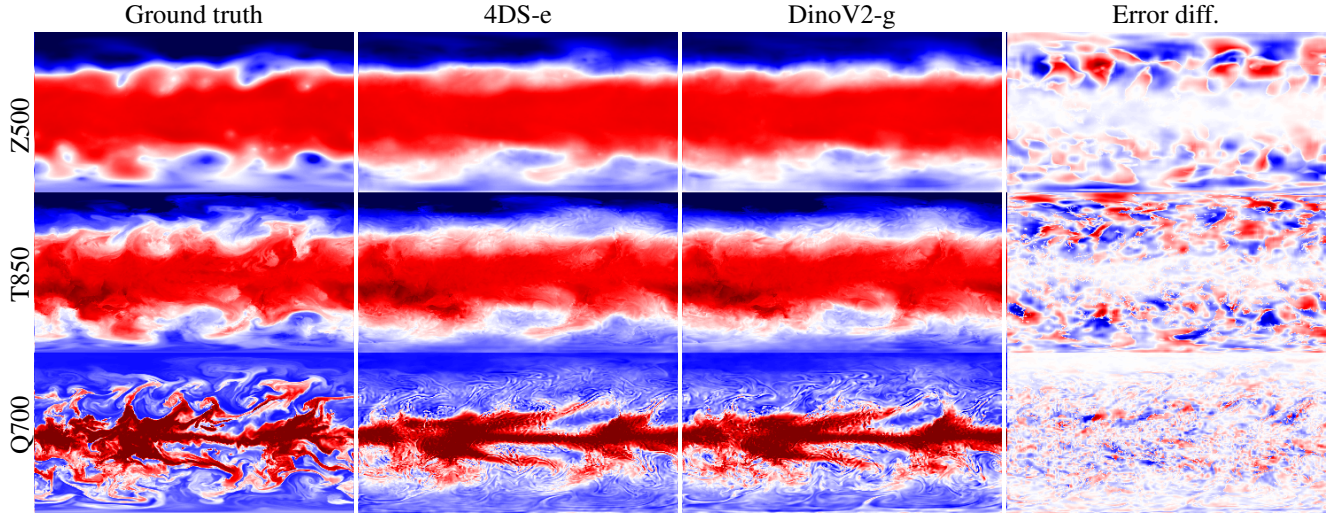


Figure D.3. **Point tracking error comparison on the STIR dataset.** The first column shows the initial query points, while the subsequent columns display the predicted locations in the final frame for our readout with frozen 4DS-e and DinoV2-g backbones (❄). Ground truth points are represented by large circles, predicted locations by smaller circles, and the error between them by connecting lines. Notably, DinoV2-g exhibits significant tracking errors, even falling short of the control baseline. In contrast, 4DS-e demonstrates superior performance in diverse and challenging scenarios: points close to moving objects (row 1), occlusion events (row 2), points on moving objects (row 3), and camera motion (row 4).



(a) 1st future step.



(b) 16th future step.

Figure D.4. **Qualitative comparison on Weatherbench 2.** We present ground truth and predicted values for Geopotential at 500hPa (Z500), Temperature at 850hPa (T850), and Specific Humidity at 700hPa (Q700) at (a) the 1st and (b) the 16th future step (12-hour intervals between each step). We show predictions for our readout on top of frozen 4DS-e and DinoV2-g backbones (❄), both *w.o. cond.*. Ground truth and predictions are normalized using the training set's mean and twice the standard deviation for each variable. These are visualized using the "seismic" colormap. Error differences are also shown, normalized to highlight relative performance: blue indicates 4DS-e outperforms DinoV2-g, red indicates DinoV2-g outperforms 4DS-e, and white indicates comparable performance.

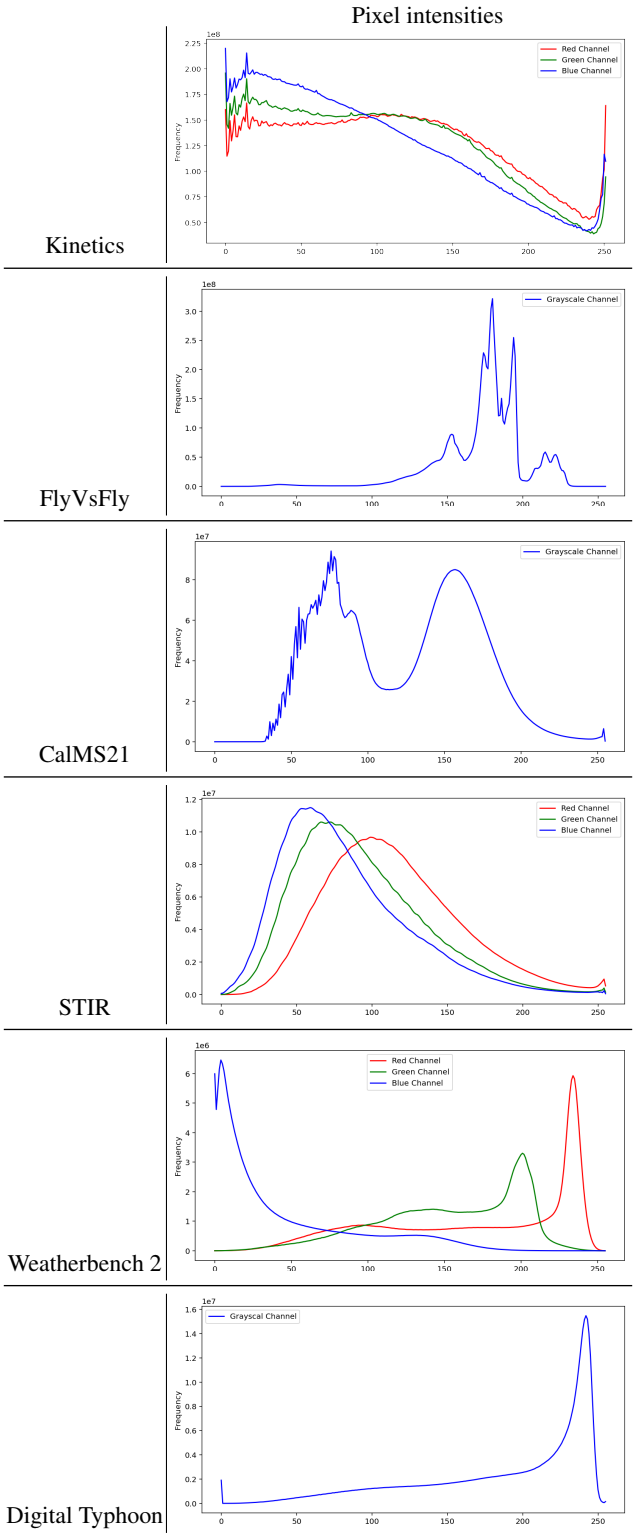


Table D.1. **Pixel histograms.** The histograms of pixel intensity values of input variables (shown as Greyscale and RGB channels) in SCIVID (last 5) exhibit significant deviations from the histograms of pixel values in datasets commonly used for pre-training ViFMs such as Kinetics [27] (top row), indicating a notable shift in the underlying data distributions.