A Appendix

A.1 Discrete CRONOS Details

A.1.1 Grid Embedding

Assume we want to embed data on a grid with spacing $\Delta > 0$, and a maximum size of $K \in \mathbb{N}_+$, where

$$g_k = g_1 + (k-1)\Delta, \quad k \in \{1, \dots, K\}. \tag{12}$$

In total, we define the grid as $\mathbf{g} = [g_1, \dots, g_k]$. We define the grid quantizer as

$$q(t_i) = \text{clip}\left(1 + \left\lfloor \frac{t_i - g_1}{\Delta} + \tfrac{1}{2} \right\rfloor, 1, K\right), \tag{13}$$

where $\text{clip}(a, b, c) := \min(\max(a, b), c)$. I.e. we clip the value t to the closest grid point g_k. We define the Kronecker delta for indices

$$\delta_{k,q(t_i)} = \begin{cases} 1, & q(t_i) = k \\ 0, & \text{else.} \end{cases} \tag{14}$$

Since for some grid sizes Δ, there can be too many items, we define the occupancy:

$$m_k = \min\left(1, \sum_{i=1}^{T} \delta_{k,q(t_i)}\right) \in \{0, 1\}. \tag{15}$$

We use the last available index for filling:

$$i^*(k) = \text{argmax}_i(\delta_{k,q(t_i)} t_i), \tag{16}$$

i.e. the index which is *latest* while still falling into the index k. Finally, our embedded images are

$$\tilde{I}_k = \begin{cases} I_{i^*(k)}, & m_k = 1 \\ 0, & \text{else.} \end{cases} \tag{17}$$

Lastly we define the grid embedding as

$$\mathcal{E}^{\text{grid}}_{\mathbf{g}}\left(\{(I_i, t_i)\}_{i=1}^{T}\right) = [\tilde{I}_1, \dots, \tilde{I}_K]. \tag{18}$$

A.1.2 Last Observed Carry Forward.

Longitudinal imaging sequences are often incomplete, with missing acquisitions at certain timestamps. To ensure consistent tensor inputs, we adopt the last-observed carry-forward procedure: missing frames are initialized as zero images and then replaced by the nearest available context scan in time. We iteratively define the

$$\hat{I}_1 = \tilde{I}_{k_0}, \quad \hat{I}_k = m_k \tilde{I}_k + (1 - m_k)\hat{I}_{k-1} \quad k \in \{2, \dots, K\}, \tag{19}$$

where k_0 defines the first observed image in the grid

$$k_0 = \min\{k \in \{1, \dots, K\}\} | m_k = 1\}. \tag{20}$$

We define shorthand:

$$\mathcal{F}^{\text{LOCF}}\left([\tilde{I}_1, \dots, \tilde{I}_K]\right) = [\hat{I}_1, \dots, \hat{I}_K]. \tag{21}$$

This guarantees that the model always observes a full sequence $\{I_1, \dots, I_T\}$ while still preserving the true irregularity of acquisition. Empirically, sparsity filling stabilizes training and improves performance compared to leaving missing slots empty, since it prevents the network from confusing acquisition gaps with valid image content. This can be seen in the ablations, see Table 4.

A.1.3 Practical Implementation

For datasets which are regular, this grid embedding is simply achieved by randomly masking some entries. The occupancy operator also highlights one key trade-off issue: if Δ is too large, we possibly overwrite some images. If Δ is too small, our context window is also very small if we keep within a budget of K tensor size.

Algorithm 2 CRONOS: Discrete training and Inference

Require: Patients $\mathcal{P} = \{[\mathcal{I}_1, I_{\text{target},1}], \ldots, [\mathcal{I}_p, I_{\text{target},p}]\}$ and initial network v_θ

while training **do**
  $[\mathcal{I}, I_{\text{target}}] \sim \mathcal{P}(\mathcal{X})$ ▷ pick a random patient
  $\tau \sim \mathcal{U}(0, 1)$ ▷ pick a random flow step
  $\mathcal{I}_{\text{target}} \leftarrow [I_{\text{target}}, \ldots, I_{\text{target}}]$ ▷ Extend the dimension of I_{target} T times
  $\mathcal{I}' \leftarrow \text{LOCF}(\mathcal{I})$ ▷ Fill empty images
  $X_\tau \leftarrow (1 - \tau)\,\mathcal{I}' + \tau\,\mathcal{I}_{\text{target}} + \sigma$ ▷ Calculate the linear interpolation
  $\mathcal{L}_{\text{FM}} \leftarrow \left\| v_\theta(\tau,\, X_\tau) \;-\; (\mathcal{I}_{\text{target}} - \mathcal{I}') \right\|^2$ ▷ Calculate the velocity loss
  Update $\theta \;\leftarrow\; \text{AdamW}\,(\nabla_\theta \mathcal{L}_{\text{FM}})$
return v_θ
if inference **then**
  Initialize $X_0 \leftarrow \mathcal{I}'$
  Define integration grid $\{\tau_0 = 0, \ldots, \tau_N = 1\}$ with n steps
  $\hat{X}_{0:N} \leftarrow \text{ODEInt}\big(v_\theta, X_0,\ \{\tau_0, \ldots, \tau_N\}\big)$ ▷ numerically integrate the network
  return $\hat{X}_N$

Table 4: **Ablation Results for discrete CRONOS on ACDC:** This table compares TFM under different design changes, showing the performance under each scenario. The ablations were done on an ACDC validation set. We evaluate the effect of using a more lightweight version of the U-Net which does not use attention('No Att'). Instead, τ and image embeddings are merged via concatenation in the bottleneck. We also compare aggregating via the mean and the last image, but these results are only for inference. Training is still done the same way. Third, we compare LOCF with the alternative of using the image sequences $\mathcal{I}$ as they are given, which notably reduces performance. *Limiting the model to only see LCI during training and perform FM on this is unstable, which highlights the importance of temporal context.

Change	NRMSE (↓)	SSIM (↑)	PSNR (↑)
Att U-Net & Mean	0.0261	96.04	32.30
No Att: Mean	0.0270	95.77	31.88
No Att: Last	0.0271	95.77	31.87
No LOCF + Masking	0.0380	95.55	31.20
No LOCF	0.0444	90.92	27.30
LCI	0.0380	93.50	29.49

B Further Ablations

B.1 Critical Ablations

In Table 4 we see the different design choices of the discrete CRONOS. Notably, LOCF or masking (which is training only on the given frames), archives the highest boost versus the baseline. We note that the masked training takes longer to converge, but yields a similar, albeit not the same performance.

B.2 Zero-shot robustness to masking order

In Figure 7, we evaluate how prediction quality changes when masking context frames from early to late. As expected, increasing the temporal gap between the last available frame and the target leads to a monotonic decrease in SSIM. Interestingly, when the final frame remains present and only earlier frames are removed, the curve becomes U-shaped: performance peaks when roughly five context frames are masked. This aligns with the masking distribution used during training, where such levels of sparsity were most common. This also suggests that the model is optimized for the typical irregular pattern seen during training, and the fully unmasked case is not the easiest *in the zero-shot* evaluation.

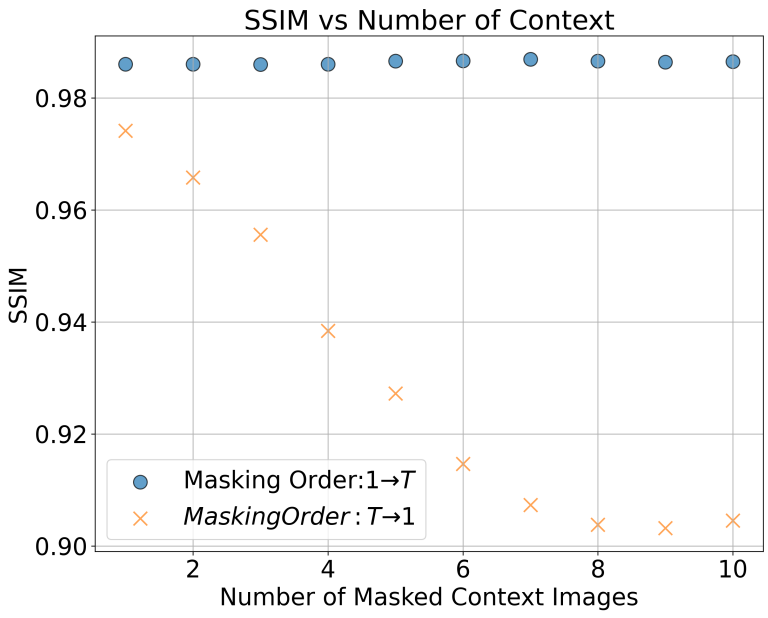


Figure 7: **Zero-shot masking of early vs. late context frames on ACDC.** We compare CRONOS's prediction quality when masking context frames at inference time. In the early masking protocol ($1 \rightarrow T$), we remove the earliest context frames first, in the late masking protocol ($T \rightarrow 1$), we remove the frames closest to the target. The x-axis denotes the total number of masked frames. Masking frames near the target causes a clear, monotonic drop in SSIM, confirming that CRONOS relies on temporally proximal information. Masking from earlier frames leads only to a slight degradation, and shows a weak U-shaped behavior at larger mask counts.

B.3 Hyperparameter Ablations

We did ablation studies on one ACDC validation subset.

Table 5: Ablation on feature size (FS) for the proposed method, compared to the last context image (LCI) baseline. Values in parentheses indicate the difference relative to LCI. Larger FS values generally improve NRMSE, SSIM, and PSNR, with FS = 32.0 achieving the best overall performance.

Feature Size	NRMSE (↓)	SSIM (↑)	PSNR (↑)	Max Mem[GB]
LCI	0.038	93.50	29.49	-
FS=8.0	0.027 (-0.012)	95.94 (+2.38)	32.07 (+2.74)	2.70
FS=16.0	0.026 (-0.013)	95.99 (+2.43)	32.17 (+2.83)	3.27
FS=32.0	0.026 (-0.013)	**96.06 (+2.49)**	32.24 (+2.90)	7.70
FS=64.0	**0.026 (-0.014)**	95.97 (+2.41)	**32.35 (+3.01)**	11.21

Table 6: **Evaluating Metrics vs. Number of Function Evaluations:** We evaluate how the number of function evaluations (NFEs) affects $SSIM$ performance on one ACDC validation set. $SSIM$ increases with more evaluations and peaks at $5 - 10$ NFEs, after which it plateaus. However, the improvement becomes marginal after beyond just 5 NFEs. As trade-off we use 10 NFE throughout.

NFE	NRMSE (↓)	SSIM (↑)	PSNR (↑)
LCI	0.038	93.50	29.49
NFE=1.0	0.030 (-0.009)	95.43 (+1.87)	31.47 (+2.13)
NFE=5.0	**0.025 (-0.014)**	**96.18 (+2.62)**	**32.47 (+3.13)**
NFE=10.0	0.026 (-0.013)	96.06 (+2.49)	32.24 (+2.90)
NFE=100.0	0.026 (-0.013)	96.04 (+2.48)	32.30 (+2.97)
NFE=200.0	0.025 (-0.014)	96.18 (+2.62)	32.47 (+3.13)

Table 7: **Training noise amount** validation performance ablation on training noise (TN) levels for the proposed method, compared to the last context image (LCI) baseline. Values in parentheses indicate the difference relative to LCI. Moderate TN levels (0.0–0.05) consistently improve NRMSE, SSIM, and PSNR, with TN = 0.0 yielding the best overall performance.

Training Noise	NRMSE (↓)	SSIM (↑)	PSNR (↑)
LCI	0.039	93.53	29.42
TN=0.0	0.026 (-0.013)	96.06 (+2.49)	32.24 (+2.90)
TN=0.01	**0.026 (-0.014)**	**96.12 (+2.56)**	32.32 (+2.98)
TN=0.025	0.026 (-0.013)	96.06 (+2.49)	**32.34 (+3.00)**
TN=0.05	0.026 (-0.013)	96.04 (+2.47)	32.26 (+2.92)
TN=0.1	0.026 (-0.013)	96.06 (+2.49)	32.24 (+2.90)
TN=0.3	0.026 (-0.013)	95.78 (+2.21)	32.26 (+2.92)

In Table 7, we see the different effects of training noise on the performance. While we find that 0.01 seems to yield the best performance, we kept the noise at 0, in order to compare to other baselines, which were not trained in a noisy fashion.

B.4 Runtime and Memory Usage

To provide a transparent comparison of computational efficiency, we report wall-clock training time and peak GPU memory for a single run of each baseline. All experiments were run on the same hardware with batch size 4. Table 8 shows that CRONOS achieves competitive runtime and memory usage compared to other spatio-temporal baselines, while diffusion-based approaches remain significantly more expensive (see main text).

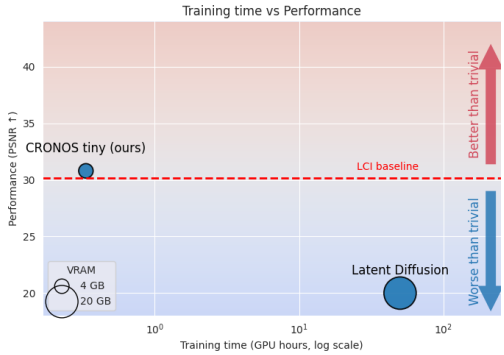
B.5 *Image to image comparison*

In Figure 8, we include a diffusion baseline adapted from Puglisi et al. (2025). Diffusion models are widely used for generative modeling, but their iterative denoising leads to orders of magnitude

Method	Runtime [10^3s]	Max Memory [GB]
ViViT	16	9.0
NODE+LSTM	60	6.6
SimVP	20	5.2
ConvLSTM	14	3.5
CRONOS irregular	18	7.7
CRONOS continuous	12	7.0

Table 8: Wall-clock runtime in thousands of seconds and maximum memory usage for a single run.

Figure 8: **Image to Image - ACDC.** Efficiency–performance trade-off on ACDC: PSNR (↑) vs. training time (GPU hours, log scale). The red dashed line marks the LCI baseline; red/blue shading indicates better/worse than baseline. Bubble size encodes VRAM needs. *Ours CRONOS (tiny)* achieves above-baseline PSNR with roughly two orders-of-magnitude less training than Latent Diffusion AE, which remains below the baseline despite substantially higher cost.



longer inference times and, in our setting, they do not outperform the simple LCI heuristic. *Most importantly*, to the best of our knowledge, **no prior** 3D continuous time diffusion approach in medical imaging **supports forecasting from multiple input volumes**. We therefore present diffusion here as a motivating image-to-image reference, but focus the remainder of our study on spatio-temporal learning (STL) methods, which are both more computationally feasible and explicitly designed for multi-context forecasting. So, the results from 8, the inference would take roughly 35 minutes, and auto-regressive diffusion would take $5 - 6\,h$ in total per sampling step

update figure

B.6 Continuous ablation

B.6.1 Main Results Ablation

For the headline comparison we further restrict to a *single* context frame (image-to-image forecasting), which is markedly harder for the discrete CRONOS than multi-context prediction, as the last-observed carry-forward is more challenging. We keep the same training objective and evaluation protocol; see Tab. 3 for metrics under this stricter setting. We observe that the continuous CRONOS is better than LCI, while the discrete one is not. Even stronger prediction can be seen in 9.

B.6.2 Additional Ablations for use of continuous time

To isolate the effect of explicit time, we construct a continuous setting where only the continuous variant has access to time information. We use a 4-frame grid, where we uniformly sample 2 images. From the missing future frames, we predict a single future frame. Here, the discrete model receives *no timestamps*. The continuous model receives the two context times and the target time during inference. All else is being equal (architecture, training regime, etc.). Results in Tab. 9 show that explicit timestamps are necessary in this protocol.

Table 9: **Continuous conditioning improves sequence-to-image forecasting.** Comparison of the LCI baseline with CRONOS under discrete (implicit time) and continuous (explicit timestamps) settings. Metrics are reported as SSIM↑, PSNR↑, NRMSE↓. The continuous variant attains the best scores across all metrics, indicating that real-valued time conditioning is effectively exploited.

Method	SSIM ↑	PSNR ↑	NRMSE ↓
LCI (baseline)	0.94997	31.727	0.03038
CRONOS discrete	0.96671	33.534	0.02211
CRONOS continuous	0.97814	36.007	0.01670

C Reproducibility

We will describe the experiments to further facilitate reproducibility.

C.1 Experimental Details

All methods were trained with AdamW and a cosine-annealed learning-rate schedule, using a batch size of 4. The learning rate was fixed at $1e{-}4$ for all experiments. For CRONOS, we used 10 integration steps during inference, see appendix (B) for ablations. To ensure fair comparison, we ran each experiment three times with different validation splits and the same random seed within each split. We trained each model for the same number of epochs, selecting the checkpoint with the best NRMSE. We report NRMSE, PSNR and SSIM for each experiment. Further details can be found C.2, such as how we fix the random masking for ACDC and ISLES. Furthermore, we use Last Context Image (LCI), a heuristic that serves as strong heuristic baseline. This is simply the last available image in the sequence. This baseline is medically motivated, as it serves as a part of medical decision making when looking at longitudinal series (see Therasse et al. (2000)). Furthermore, in a setting with slowly changing anatomy, this is surprisingly strong.

C.2 Random Masking

For ACDC and ISLES, we randomly omit context images during both training and validation, to simulate irregular sampling. Since we believe we are the first to benchmark methods in this very specific irregular setting, we **highlight a potentially grave pitfall**: If validation masks are resampled at each validation epoch, even with a fixed seed, the masking evaluation metrics change every time, which is exacerbated by our small validation set . This variability affects even the trivial LCI baseline and makes "best" epoch selection arbitrary. Since the validation set is small, context sequences can be extremely sparse or dense, causing the LCI baseline's performance to fluctuate drastically[3]. To avoid this issue, we generate one fixed set of masks per split (using a single seed) and reuse those exact masks for every model at every validation epoch. This ensures consistent validation conditions, meaningful epoch selection, and fully reproducible and interpretable validation results. In all cases, models were selected by the lowest validation MSE and then evaluated on the held-out test set.

C.3 Network architecture

Our default network v_θ is a four-scale 3D UNet with residual blocks and FiLM time conditioning. The input $X \in \mathbb{R}^{B\times(C\times T)\times D\times H\times W}$ (typically $C{=}1$) stacks T context volumes along channels and is linearly mixed by a $1{\times}1{\times}1$ Conv3D to $C_{\text{stem}}{=}32$. The encoder uses channel expansion rates $[1, 1, 2, 4]$ with one ResBlocks per scale (Conv3D($k{=}3$)–GN(8)–SiLU–Conv3D($k{=}3$)–GN(8), with a $1{\times}1{\times}1$ projection on the skip if needed); downsampling is Conv3D($k{=}3$, $s{=}2$). The bottleneck has one ResBlocks and optional windowed self-attention over flattened spatial tokens (window 8^3, $h{=}4$ heads, head dim 64), followed by a final ResBlock. The decoder upsamples trilinearly by 2, applies Conv3D($k{=}3$), concatenates the encoder skip, and mirrors the two-ResBlock pattern with channel width equal to the reversed down blocks.. FiLM modulation is injected after the first GN in every block: $\hat{h} = \alpha_\ell \odot \mathrm{GN}(h) + \beta_\ell + h$, where $(\alpha_\ell, \beta_\ell)$ come from an MLP over a fixed-dimensional

[3]In natural imaging, validation sets are much larger, so random fluctuations are less severe. In medical imaging, however, smaller validation sets make these fluctuations significant.

conditioning code; either Fourier features of the scalar flow step $\tau \in [0, 1]$ for the irregular setting, or the summed Fourier encodings of real timestamps $\sum_{i=1}^{T} \gamma(t_i)/T$ for the continuous setting. All convolutions use padding to preserve spatial sizes; normalization is GroupNorm(8), activation is SiLU, and skips are concatenative with a $1\times1\times1$ channel-align conv. The head is Conv3D($32\to32$, k=3)–SiLU–Conv3D($32\to1$, k=1) producing a single-channel voxelwise velocity field.

D ADDENDA

D.1 USE OF LLMS

An LLM was used to help rephrase and polish this work.