FROM CONTINUOUS FIELDS TO DISCRETE TOKENS: PROBABILISTIC PRECIPITATION NOWCASTING WITH CONTROLLED-PATH DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Precipitation nowcasting requires accurate forecasts of uncertain rainfall evolution at low computational cost. Diffusion models generate diverse forecasts, but their training and iterative sampling can limit practical use. We introduce Discrete Probabilistic Precipitation Nowcasting (DPPN), to our knowledge the first discrete diffusion framework for probabilistic precipitation nowcasting. DPPN encodes continuous radar fields into compact discrete tokens that form a learned vocabulary of precipitation patterns. By modeling their conditional evolution, DPPN offers a perspective on nowcasting as learning a weather language. DPPN connects continuous Gaussian relaxations with discrete token states to reduce training variance and improve computational efficiency. The framework supports two forecasting formulations: direct generation conditioned on historical observations and cascaded generation conditioned on a deterministic prediction. Both generate distributions over future tokens, with supervision on decoded precipitation fields guiding forecast reconstruction. Experiments on SEVIR demonstrate competitive forecasting performance relative to continuous diffusion and flow-matching baselines, with fewer trainable parameters and lower reported wall-clock costs under the evaluated implementations. The generated results highlight discrete diffusion as an efficient approach to probabilistic nowcasting and demonstrate its capacity for controllable generation through diffusion path design.

1 INTRODUCTION

Precipitation nowcasting predicts future precipitation fields from recent radar observations and supports decisions about heavy rainfall and storms (Gong et al., 2024). Accurate forecasts must capture both the movement of precipitation systems and changes in local intensity. Deterministic models learn these patterns from historical sequences, but a single forecast provides an incomplete description when several futures remain plausible. Pointwise regression losses can smooth uncertain structures and underestimate local intensity (Gao et al., 2023). Probabilistic models address this uncertainty by generating ensembles of possible future precipitation fields.

Diffusion models generate probabilistic precipitation forecasts through repeated denoising, making computational cost a consideration for ensemble generation. PreDiff generates future sequences through conditional latent diffusion, while CasCast conditions generation on a deterministic forecast (Gao et al., 2023; Gong et al., 2024). Both compress precipitation fields to reduce the cost of each denoising evaluation. FlowCast uses conditional flow matching in a continuous latent space to improve the efficiency of forecast generation (Perrone Ribeiro & Faganeli Pucer, 2026). These methods motivate studying alternative generative formulations that balance predictive skill with training and inference cost.

Discrete tokens provide a vocabulary for representing precipitation patterns in radar fields. GPTCast and BlockGPT demonstrate how token representations support autoregressive precipitation forecasting (Franch et al., 2025; Meo et al., 2025). We investigate conditional discrete diffusion as a mechanism for generating future precipitation tokens. Duo connects continuous Gaussian relaxations with discrete token states, allowing training and sampling techniques from Gaussian diffusion to transfer to discrete generation (Sahoo et al., 2025). This connection provides the basis for our formula-

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.

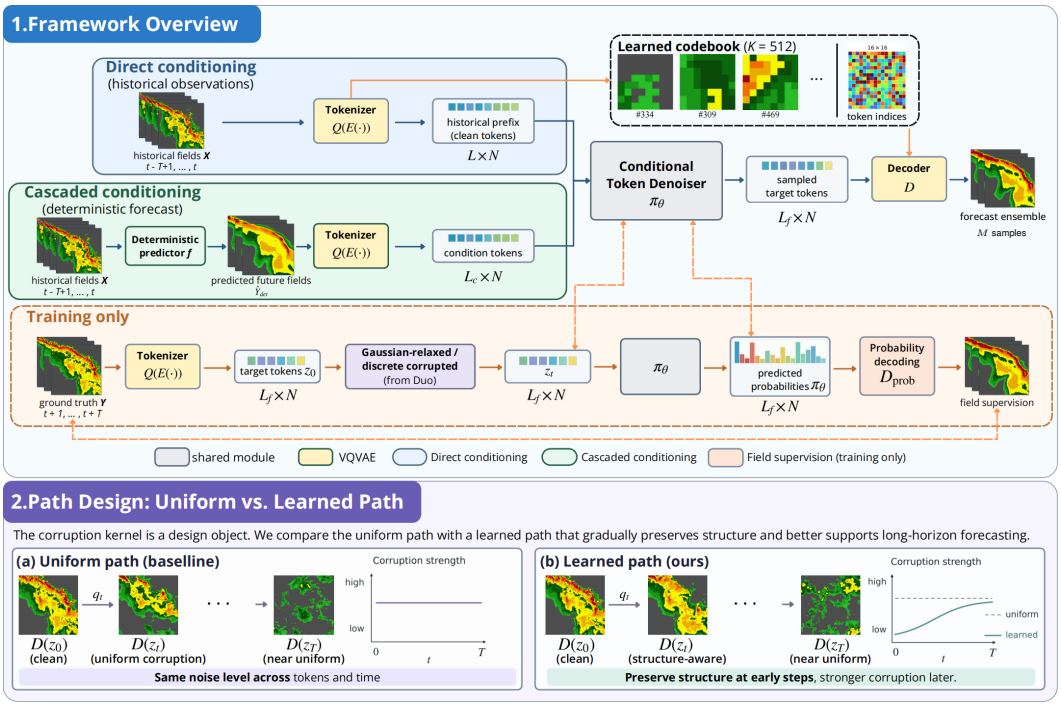


Figure 1: (1) Framework: A shared codebook ($K = 512$) tokenizes radar fields. The denoiser π_θ generates future tokens conditioned either on historical observations (direct, blue) or on a deterministic forecast $\widehat{Y}_{\text{det}} = f(X)$ (cascaded, green), and the sampled tokens are decoded into a forecast ensemble. During training (orange), corrupted target tokens are reconstructed under both token-level and decoded radar-space supervision. **(2) Path design:** Uniform corruption treats all incorrect tokens equally, whereas the data-informed path concentrates transitions on plausible forecast–target confusions, enabling control over the generation trajectory. The resulting trajectories are visualized in radar space for intuitive comparison.

tion, which aims to combine efficient probabilistic forecasting with flexibility in the design of the generation process.

We introduce DPPN, the first application of discrete diffusion to probabilistic precipitation nowcasting, to our knowledge. A vector-quantized autoencoder maps continuous radar fields to compact discrete tokens, and conditional discrete diffusion generates future tokens for decoding into precipitation fields (van den Oord et al., 2017). The learned vocabulary and its conditional evolution offer a weather-language perspective on forecasting, extending the token-based view explored in prior work (Franch et al., 2025). DPPN supports direct generation conditioned on historical observations and cascaded generation conditioned on a deterministic forecast. Supervision on decoded precipitation fields connects token prediction to forecast reconstruction, with intensity weighting in the cascaded configuration. Figure 1 summarizes the direct and cascaded conditioning regimes and contrasts the uniform and prescribed diffusion paths.

On the SEVIR benchmark, the DPPN model provides competitive native-resolution performance and achieves the highest pooled CSI among the evaluated methods. It uses fewer trainable parameters than the evaluated continuous diffusion and flow-matching baselines and can be trained on a single GPU. We further study how the discrete corruption path can be constructed from empirical token-transition statistics. This experiment shows that discrete diffusion exposes a mechanism for controlling the preferred directions of state changes during generation. Our contributions are:

1. **Discrete diffusion for probabilistic precipitation nowcasting**. We formulate probabilistic nowcasting as conditional generation over a learned vocabulary of radar tokens. The resulting model performs diffusion over compact categorical states rather than continuous radar fields or dense continuous latents.

2. **A unified framework for direct and cascaded conditioning**. DPPN supports generation conditioned either on historical radar observations or on a deterministic future prediction. Both formulations use the same token diffusion framework and incorporate differentiable radar-space supervision by decoding predicted token distributions.
3. **Competitive forecasting performance with reduced computational requirements**. On SEVIR, the cascaded model achieves the strongest pooled CSI at longer lead time of 1 hour among the evaluated methods with a much smaller model size. It has much shorter training and inference time although direct wall-clock comparisons should be interpreted cautiously because the baseline results use different hardware and implementations.
4. **Data-informed discrete diffusion path**. We construct a corruption path from empirical token-transition statistics that concentrates corruption on plausible forecast errors. Our experiments demonstrate that this path design controls both the timing and preferred direction of token transitions, steering generation toward realistic error patterns rather than uniform noise, and yields improved inference efficiency through fewer required sampling steps.

2 Related Work

2.1 Probabilistic Precipitation Nowcasting

Deterministic nowcasting models learn spatial and temporal dependencies in precipitation sequences. Convolutional long short-term memory networks (ConvLSTM) incorporate spatial structure into recurrent updates, while Earthformer exchanges information through attention within spatiotemporal cuboids and global vectors (Shi et al., 2015; Gao et al., 2022). Generative models extend forecasting to distributions over possible futures. The Deep Generative Model of Radar (DGMR) uses adversarial training for probabilistic radar forecasting, and NowcastNet combines physical evolution with conditional generation for extreme precipitation (Ravuri et al., 2021; Zhang et al., 2023). Diffusion models learn to reverse a gradual noising process (Ho et al., 2020), while score-based models generate samples using learned gradients of noise-perturbed log densities (Song & Ermon, 2019). Latent diffusion compresses observations before generative modeling to reduce the spatial cost of denoising (Rombach et al., 2022). PreDiff applies this approach to precipitation sequences and introduces knowledge alignment for constrained generation (Gao et al., 2023). CasCast conditions latent diffusion on a deterministic forecast, separating the prediction of larger-scale evolution from the generation of local precipitation patterns (Gong et al., 2024). Iterative generation can be computationally expensive for operational nowcasting; accelerated samplers such as DDIM offer a trade-off between sampling cost and sample quality (Song et al., 2021). Flow matching learns continuous transport vector fields along prescribed probability paths (Lipman et al., 2023), while rectified flow uses straight-line interpolation and iterative reflow to improve transport efficiency (Liu et al., 2023). FlowCast applies conditional flow matching in a continuous latent space for efficient probabilistic precipitation nowcasting (Perrone Ribeiro & Faganeli Pucer, 2026). FREUD + LSM-L (Schusterbauer et al., 2026) uses rectified flow transformers to capture atmospheric uncertainty.

2.2 Discrete Representations for Weather Forecasting

Vector-quantized autoencoders represent continuous observations using a finite learned codebook (van den Oord et al., 2017). For precipitation forecasting, this representation turns radar fields into token grids that a generative model can predict. GPTCast combines radar tokenization with a generative pre-trained transformer and interprets precipitation forecasting as weather-language modeling (Franch et al., 2025). BlockGPT develops frame-level autoregression, using spatial attention within frames and causal attention across frames (Meo et al., 2025). These studies establish discrete representations as a basis for forecasting precipitation evolution. DPPN builds on this representation perspective and generates future token grids through conditional discrete diffusion.

2.3 Discrete Diffusion and Controllable Generation

Discrete denoising diffusion probabilistic models (D3PMs) define corruption processes through transition matrices, including structured transitions and absorbing states (Austin et al., 2021). Masked diffusion language models use corruption that replaces tokens with a mask symbol (Sahoo et al., 2024). Duo establishes a connection between Gaussian diffusion and uniform-state discrete

diffusion, enabling the transfer of training and sampling techniques between these settings (Sahoo et al., 2025). DPPN adopts this connection for conditional precipitation generation, combining Gaussian-relaxed and discrete token states with temporal conditioning and decoded-field supervision. Discrete flow matching also relates probability-path construction to sampling through learned posteriors (Gat et al., 2024); subsequent work provides continuous-time Markov-chain constructions for general discrete probability paths (Shaul et al., 2025). Controllable forecasting can act on the forecast content or on the generation process. PreDiff uses a knowledge-alignment network to guide generation toward prescribed constraints, including anticipated precipitation intensity (Gao et al., 2023). Our path-design experiments instead investigate how fine-tuning adapts the diffusion generation process to a prescribed path. This control concerns the evolution of generative states during sampling; physical forecast time indexes the future precipitation sequence. The field-space training loss serves a separate purpose by supervising decoded predictions against target precipitation fields.

3 METHOD

Let $X \in [0,1]^{T_{\text{in}} \times H \times W}$ denote a sequence of normalized historical radar observations and $Y \in [0,1]^{T_{\text{out}} \times H \times W}$ the corresponding future fields. The goal of probabilistic nowcasting is to learn the conditional distribution of Y given X and to draw multiple plausible forecasts from it. The index of a radar frame represents physical forecast time, whereas $t \in [0,1]$ below denotes diffusion time.

Let Z_V denote the discrete representation of a radar sequence V produced by the tokenizer introduced in Section 3.1.1. DPPN supports two conditional forecasting formulations:

$$p_\theta^{\text{direct}}(Z_Y \mid Z_X), \qquad p_\theta^{\text{cascade}}(Z_Y \mid Z_{\widehat{Y}_{\text{det}}}), \qquad \widehat{Y}_{\text{det}} = f(X), \tag{1}$$

where f is a deterministic forecasting model. The direct formulation generates future tokens from observed history, while the cascaded formulation generates them conditioned on a deterministic future prediction.

3.1 DISCRETE PROBABILISTIC NOWCASTING

DPPN combines a frozen discrete radar representation with conditional token diffusion for precipitation nowcasting, as illustrated in Figure 1. A pretrained vector-quantized autoencoder first maps radar fields to compact token grids drawn from a shared vocabulary of precipitation patterns. A conditional denoiser then models the joint distribution of future tokens, which are decoded into continuous radar forecasts. DPPN supports two conditioning regimes. The direct formulation generates future tokens from the observed radar history, whereas the cascaded formulation conditions generation on a deterministic future prediction. During training, only the target tokens are corrupted, and the denoiser is supervised in both token and decoded radar space. During inference, independently initialized reverse trajectories produce a forecast ensemble. We adopt the Gaussian-relaxed and uniform-state discrete diffusion formulation of Sahoo et al. (2025) for standard training and sampling. Our task-specific design comprises the conditional nowcasting architecture, rainfall-aware radar-space supervision, and a data-informed alternative to the uniform corruption path.

3.1.1 DISCRETE RADAR REPRESENTATION

A pretrained and frozen hierarchical VQ-VAE provides an encoder E, a quantizer Q, and a decoder D (van den Oord et al., 2017). Applied frame by frame to a radar sequence V, it produces

$$Z_V = Q(E(V)) \in \{1, \dots, K\}^{T_V \times h \times w}, \qquad \widetilde{V} = D(Z_V), \tag{2}$$

where T_V is the number of frames in V, K is the vocabulary size, and $h \times w$ is the token-grid resolution. We use the same tokenizer for observations, deterministic forecasts, and future targets, providing a shared vocabulary across both conditioning regimes. DPPN operates on the top-level 16×16 token grid, with $K = 512$, so a target sequence contains $N = T_{\text{out}}hw$ tokens. We flatten these tokens as $z_0 \in \{1, \dots, K\}^N$ for diffusion and decode the generated indices using the frozen decoder. The complete tokenizer architecture, reconstruction quality, and vocabulary analysis are provided in Appendices A.1 and A.2

3.1.2 CONDITIONAL TOKEN DENOISING

The denoiser predicts a clean-token distribution $\pi_\theta(\cdot \mid z_t, t, c)$ at every target position, where z_t denotes the corrupted future tokens and c the conditioning tokens. The two formulations in Equation (1) use the same diffusion process but differ in how the condition is represented. For direct generation, historical and future tokens are concatenated, with the historical tokens retained as a clean prefix and corruption applied only to the future positions. The denoiser attends jointly to the history and corrupted future, while the losses and reverse updates are restricted to the future positions. For cascaded generation, we tokenize the deterministic Earthformer forecast $\widehat{Y}_{\mathrm{det}}$ and use it as the condition, following the cascaded forecasting strategy of Gong et al. (2024). At each spatial location j, embeddings from all forecast times are concatenated:

$$h_j = W_{\mathrm{in}}\left[e_{\mathrm{tok}}(z_t^{1,j}); \ldots; e_{\mathrm{tok}}(z_t^{T_{\mathrm{out}},j}); e_{\mathrm{tok}}(c^{1,j}); \ldots; e_{\mathrm{tok}}(c^{T_{\mathrm{out}},j})\right] + b_{\mathrm{in}}. \tag{3}$$

Here, e_{tok} is the token-embedding map, b_{in} is a learned bias, and $W_{\mathrm{in}} \in \mathbb{R}^{d \times 2T_{\mathrm{out}}d}$ projects the concatenated representation to the Transformer dimension d. This produces one feature per spatial location, with forecast-time information packed into the feature dimension. A diffusion-time-conditioned Transformer processes the resulting hw features and predicts $T_{\mathrm{out}}K$ logits at each location, which are reshaped into one K-category distribution for each forecast time. During cascaded training, condition dropout replaces the deterministic condition embeddings with zeros, enabling conditional and unconditional predictions for guided sampling. Model architecture and optimization details are provided in Appendix C.2.

3.1.3 RAINFALL-AWARE TRAINING AND STANDARD SAMPLING

Uniform-state diffusion. We adopt the Gaussian-to-discrete diffusion correspondence of Sahoo et al. (2025). For a clean future token $z_0^i \in \{1, \ldots, K\}$, the categorical forward marginal is

$$q_t(z_t^i = k \mid z_0^i) = \alpha_t \mathbf{1}[k = z_0^i] + (1 - \alpha_t)/K, \tag{4}$$

where the retention probability α_t decreases with diffusion time. Corruption is independent across target positions, while the denoiser predicts their clean states jointly using spatial and temporal context. Conditioning tokens remain clean. Following Sahoo et al. (2025), the same categorical process can be induced by projecting a Gaussian perturbation of the one-hot clean token:

$$u_t^i = a_t e_{z_0^i} + b_t \epsilon^i, \qquad \epsilon^i \sim \mathcal{N}(0, I_K), \qquad a_t = \sqrt{\mathrm{sigmoid}(-\gamma_t)}, \qquad b_t = \sqrt{\mathrm{sigmoid}(\gamma_t)}. \tag{5}$$

Here, $\gamma_t = \log(b_t^2/a_t^2)$ is the Gaussian log noise-to-signal ratio, and $\tau > 0$ below is the softmax relaxation temperature. The discrete state is obtained as $z_t^i = \arg\max_k u_{t,k}^i$, with retention probability

$$\alpha_t = \left(K \Pr_{\epsilon^i}\left[\arg\max_k u_{t,k}^i = z_0^i\right] - 1\right)/(K-1). \tag{6}$$

The probability is taken over the Gaussian noise ϵ^i in Equation (5). During the Gaussian-relaxed stage, the denoiser receives embeddings of $\mathrm{softmax}(u_t/\tau)$; during the discrete stage, it receives embeddings of categorical samples from Equation (4). Both stages predict the same clean tokens. We use *dual-state diffusion* to refer to this inherited Gaussian-relaxed and categorical representation of the forward process; the forecasting target itself remains discrete.

Token objective. We train clean-token prediction using the continuous-time uniform-state objective of Sahoo et al. (2025):

$$\mathcal{L}_{\mathrm{token}} = \mathbb{E}_{(z_0,c),\, t,\, z_t \sim q_t(\cdot|z_0)}\left[\sum_{i=1}^{N} \ell_{\mathrm{UD}}\big(\pi_\theta^i, z_t^i, z_0^i; \alpha_t, \dot{\alpha}_t\big)/N\right], \tag{7}$$

where $\dot{\alpha}_t = d\alpha_t/dt$. The integrand ℓ_{UD} derived by Sahoo et al. (2025), and the complete dual-state training schedule are given in Appendix B.1.

Radar-space supervision. Token accuracy alone does not measure the meteorological consequence of a prediction error. Confusing two tokens with similar decoded rainfall patterns is less consequential than confusing background with heavy precipitation. We therefore complement the

token objective with differentiable supervision in radar space. Let D_{prob} denote the frozen decoder interface that maps a token distribution to codebook embeddings and then to a radar field. Writing $\Pi_\theta = (\pi_\theta^i)_{i=1}^N$ for the predicted clean-token distributions at all target positions, we define $\widehat{Y}_t^{\mathrm{soft}} = D_{\mathrm{prob}}(\Pi_\theta)$ and optimize

$$\mathcal{L}_{\mathrm{radar}} = \mathbb{E}_{X,Y,t,\xi}\left[\left(\sum_r w(Y_r^{\mathrm{ref}})\left|\widehat{Y}_{t,r}^{\mathrm{soft}} - Y_r^{\mathrm{ref}}\right|\right) / \left(\sum_r w(Y_r^{\mathrm{ref}})\right)\right], \tag{8}$$

where r indexes forecast times and radar pixels, and ξ collects the random corruption and, when applicable, condition-dropout draws used to construct the denoiser input. We use the observed target as Y^{ref} when it is available and the frozen tokenizer reconstruction $D(Z_Y)$ when training from token-only caches. The direct formulation uses $w(y) = 1$. For cascaded generation, we increase the contribution of rainy and heavy-rain pixels:

$$w(y) = 1 + \lambda_{\mathrm{rain}}\mathbf{1}[y > \eta_{\mathrm{rain}}] + \lambda_{\mathrm{heavy}}\mathbf{1}[y > \eta_{\mathrm{heavy}}], \tag{9}$$

where λ_{rain} and λ_{heavy} are nonnegative intensity weights, and η_{rain} and η_{heavy} are thresholds on the normalized radar-intensity scale. This weighting reduces the dominance of background pixels and aligns token prediction more closely with precipitation-event reconstruction. The full objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{token}} + \lambda_{\mathrm{radar}}\mathcal{L}_{\mathrm{radar}}. \tag{10}$$

Although the tokenizer remains frozen, gradients from $\mathcal{L}_{\mathrm{radar}}$ propagate through its probability-input decoder to the denoiser. The resulting probability-decoded field is used only for differentiable supervision and differs from a forecast obtained by sampling and decoding discrete token indices.

Standard sampling. For the uniform path in Equation (4), we use the ancestral reverse transition of Sahoo et al. (2025), substituting the predicted clean-token probabilities into the uniform-state posterior. For cascaded forecasting, conditional guidance combines the conditional and unconditional reverse distributions:

$$R_{\mathrm{guided}}(k) \propto \exp\left[(1-g)\log R_{\mathrm{uncond}}(k) + g\log R_{\mathrm{cond}}(k)\right], \tag{11}$$

where $g = 1$ recovers the conditional transition. Repeating the reverse process from independent categorical initializations produces the ensemble $\{\widehat{Y}^{(m)}\}_{m=1}^M$, where M is the ensemble size. The complete training and sampling procedures are provided in Algorithms 1 and 2 in the Appendix.

3.1.4 Controllable Diffusion Paths

The standard formulation above adopts the uniform-state path of Sahoo et al. (2025), which assigns equal corruption probability to all incorrect token categories. This symmetry does not reflect the empirical structure of the learned precipitation vocabulary. Training-set statistics show that forecast-target token confusions are sparse and asymmetric. Deterministic forecasts commonly replace moderate or strong precipitation tokens with weaker patterns, while only a small subset of the vocabulary represents plausible alternatives. We therefore construct a prescribed corruption kernel $p_\rho(k \mid z_0^i)$ from empirical forecast–target token co-occurrence. The kernel preferentially moves a clean token toward categories that represent plausible forecast errors rather than distributing corruption uniformly over the vocabulary. The path parameter $\rho \in [0, 1]$ progresses from the categorical prior at $\rho = 0$ to the clean data at $\rho = 1$.

Because this non-uniform kernel no longer satisfies the assumptions of the uniform-state objective, we fine-tune the pretrained denoiser to estimate the clean-token posterior induced by the prescribed path. Given this posterior, the forward path determines the corresponding continuous-time reverse rates and path-following sampler. The pairwise cost and path schedule therefore control which token transitions are preferred and when they occur during generation. Appendix B.2 provides the complete kernel construction, fine-tuning objective, and reverse-rate derivation. Figure 7 there illustrates the empirical forecast-confusion structure, while Section 4.3 evaluates the realized sampling trajectories.

4 Experiments

We evaluate DPPN variants on SEVIR, focusing on the forecast quality and computational efficiency. DPPN uses a frozen tokenizer that maps each 128×128 radar frame to a 16×16 grid of

Table 1: SEVIR forecast quality. HSS averages six thresholds at native resolution. Bold and underline mark the best and second-best available scores.

Model	Size(M)	SSIM ↑	CRPS ↓	HSS ↑	mCSI ↑			mCSI @ 60 min ↑		
					POOL1	POOL4	POOL16	POOL1	POOL4	POOL16
Persistence	–	–	0.0526	–	0.2613	0.3702	0.4690	–	–	–
UNet	16.6	–	0.0353	–	0.3593	0.4098	0.4805	–	–	–
ConvLSTM	14.0	–	0.0332	–	0.4185	0.4452	0.5135	–	–	–
PredRNN	46.6	–	0.0306	–	0.4080	0.4497	0.5005	–	–	–
PhyDNet	13.7	–	0.0319	–	0.3940	0.4379	0.4854	–	–	–
E3D-LSTM	35.6	–	0.0297	–	0.4038	0.4492	0.4961	–	–	–
Rainformer	184.0	–	0.0357	–	0.3661	0.4232	0.4738	–	–	–
Earthformer	15.1	0.7632	0.0304	0.5390	0.4163	0.4309	0.4672	0.2974	0.3034	0.3270
DGMR	71.5	–	0.0435	–	0.2675	0.3431	0.4832	–	–	–
VideoGPT	99.6	–	0.0381	–	0.3653	0.4349	0.5798	–	–	–
LDM	438.6	–	0.0280	–	0.3580	0.4022	0.5522	–	–	–
PreDiff	220.5	–	0.0246	–	0.4100	0.4624	0.6244	–	–	–
CasCast	309.4	0.7587	0.0210	<u>0.5538</u>	0.4014	<u>0.4885</u>	<u>0.6280</u>	0.3137	0.3568	0.4554
FREUD + LSM-L	473.0	0.7373	<u>0.0201</u>	0.5276	0.4141	0.4635	0.5395	0.3003	0.3322	0.3935
FlowCast	230.4	0.7866	**0.0186**	**0.5607**	**0.4417**	0.4812	0.5609	<u>0.3276</u>	0.3572	0.4332
DPPN-direct	92.6	**0.8348**	0.0203	0.5351	0.4114	0.4683	0.6095	0.3165	<u>0.3671</u>	**0.5295**
DPPN-Cas	102	<u>0.8331</u>	0.0216	0.5466	<u>0.4186</u>	**0.5049**	**0.6348**	**0.3288**	**0.3783**	<u>0.4742</u>

indices from a 512 codebook. Six-frame Earthformer predictions are encoded by the same tokenizer and used as the conditioning sequence. The denoiser comprises 12 Transformer blocks with a hidden dimension of 768 and 12 attention heads, totaling 102M trainable parameters. We train DPPN with a batch size of 512. Training takes approximately six hours on one 32 GB RTX 4080S GPU. At inference, 100 reverse steps and a guidance weight of 2 are used. Complete training, sampling, and checkpoint-selection details are provided in Appendix C.2.

4.1 Experimental Setting

We follow the SEVIR data split and preprocessing protocol of PreDiff (Gao et al., 2023). Each sample contains seven context frames and six target frames at a 10-minute intervals, using one hour of observations to forecast the next 10 to 60 minutes, with further details provided in Appendix C.1. Radar intensities are represented on a $[0, 255]$ scale. We compute the Critical Success Index (CSI) at thresholds $\{16, 74, 133, 160, 181, 219\}$ as $\mathrm{CSI} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP} + \mathrm{FN})$, where true positives (TP), false positives (FP), and false negatives (FN) are accumulated over all pixels and forecast lead times. We report the mean CSI (mCSI) across the six thresholds at native resolution and after max pooling over 4×4 and 16×16 neighborhoods (Pool4 and Pool16). We report the Continuous Ranked Probability Score (CRPS), for which lower values indicate better probabilistic forecasts and which reduces to the mean absolute error for deterministic predictions. Following (Gong et al., 2024; Perrone Ribeiro & Faganeli Pucer, 2026), CRPS is computed using the Gaussian closed form with the predictive mean and standard deviation estimated from the ensemble members. Heidke Skill Score (HSS) and structural similarity index (SSIM) are reported. For fair comparison, 8 ensembles are generated for all diffusion models.

Diffusion models (Gong et al., 2024; Perrone Ribeiro & Faganeli Pucer, 2026) fuse ensemble forecasts by averaging their continuous representations and decoding the mean to obtain the final prediction. In contrast, each DPPN realization is generated as a sequence of categorical token indices, so arithmetic averaging is not defined in token-index space. Although the decoded precipitation fields can be averaged, pixel-wise means attenuate localized intensity when ensemble members contain spatially displaced structures. We therefore use an intensity-adaptive percentile fusion: the ensemble median for intensities below 74 and the 60th percentile for intensities above with details in Appendix C.3.

4.2 Comparison to the State of the Art

We compare DPPN-cas with persistence; deterministic models including UNet (Ronneberger et al., 2015), ConvLSTM (Shi et al., 2015), PredRNN (Wang et al., 2017), PhyDNet (Le Guen & Thome,

Table 2: SEVIR CSI at thresholds 74, 181, and 219. Each threshold includes three spatial scales. Bold and underline mark the best and second-best scores.

	CSI-74 ↑			CSI-181 ↑			CSI-219 ↑		
Model	POOL1	POOL4	POOL16	POOL1	POOL4	POOL16	POOL1	POOL4	POOL16
ConvLSTM	<u>0.6846</u>	0.6694	0.7039	0.2381	0.2864	0.3679	0.1220	0.1850	0.2651
Earthformer	0.6773	0.6818	0.6994	0.2448	0.2507	0.2875	0.1243	0.1305	0.1561
DGMR	0.5500	0.6302	0.7465	0.0537	0.1194	0.2804	0.0151	0.0414	0.1218
VideoGPT	0.6062	0.6529	0.7583	0.1997	0.2825	0.4543	0.1029	0.1691	0.3101
LDM	0.5848	0.6139	0.7229	0.1894	0.2420	0.4247	0.1019	0.1439	0.2896
PreDiff	0.6740	0.6691	0.7789	0.2357	0.3130	0.5127	0.1154	0.2065	0.3865
CasCast	0.6402	0.6880	0.7777	0.2441	<u>0.3545</u>	<u>0.5241</u>	0.1438	<u>0.2404</u>	<u>0.3967</u>
FREUD+LSM-L	0.6791	0.7060	0.7493	0.2407	0.2907	0.3724	0.1296	0.1785	0.2484
FlowCast	**0.6889**	<u>0.7064</u>	0.7606	**0.2822**	0.3329	0.4237	**0.1617**	0.2040	0.2761
DPPN-direct	0.6622	0.6901	**0.8037**	0.2525	0.3220	0.4905	0.1327	0.1855	0.3194
DPPN-cas	0.6590	**0.7087**	<u>0.7895</u>	<u>0.2625</u>	**0.3692**	**0.5276**	<u>0.1545</u>	**0.2549**	**0.4182**

Table 3: Model size and computational efficiency on SEVIR

		Training		Inference		
Model	#Param.(M)	Hardware	Time/epoch	Steps	Hardware	Time
PreDiff	220.5	RTX4080s/32GB	62mins	1000 (DDPM)	RTX4090/24GB	118 h
CasCast	309.4	RTX4080s/32GB	20mins	20 (DDIM)	RTX4080s/32GB	1 h
FREUD+LSM-L	473.0	RTX4080s/32GB	141mins	10 + 25 (Euler)	RTX4090/24GB	3.5 h
FlowCast	230.4	RTX4080s/32GB	50mins	10 (Euler)	RTX4090/24GB	2 h
DPPN	102	RTX4080s/32GB	6 mins	100	RTX4080s/32GB	1 h

2020), E3D-LSTM (Wang et al., 2019), Rainformer (Bai et al., 2022), and Earthformer (Gao et al., 2022); non-diffusion generative models DGMR (Ravuri et al., 2021) and VideoGPT (Yan et al., 2021); continuous diffusion models LDM (Rombach et al., 2022), PreDiff (Gao et al., 2023), and CasCast (Gong et al., 2024); and the continuous flow-matching model FlowCast (Perrone Ribeiro & Faganeli Pucer, 2026) and FREUD+LSM-L (Schusterbauer et al., 2026). Table 1 uses our local evaluations for DPPN, CasCast, Earthformer, FlowCast and FREUD, while the remaining scores are reported in the PreDiff benchmark (Gao et al., 2023) where available.

DPPN achieves the best pooled-scale CSI, with DPPN-cas reaching 0.5049 and 0.6348 at pool4 and pool16, respectively and for lead time of 60 mins. It actually provides stronger performance at longer lead times, with results from 30 to 60 minutes showing a growing advantage over the competing methods in Appendix C.4. The gains are particularly pronounced after spatial pooling and at higher precipitation intensities in Table 2 where DPPN progressively narrows the performance gap with FlowCast as the threshold increases, and achieves substantially higher pooled-scale CSI at thresholds 181 and 219. Compared with PreDiff, CasCast and FlowCast, these improvements are obtained with only 102M trainable parameters, less than half their model sizes. DPPN also requires less training and inference wall-clock on a single RTX4080s.

4.3 Controllable Generation through Path Design

We evaluate whether the prescribed diffusion path produces a measurably different generation trajectory. Starting from the pretrained cascaded denoiser, we finetune the model under the data-informed corruption path described in Appendix B.2. We compare its realized reverse trajectory with that of the uniform ancestral sampler using the same conditioning input, initialization, and sampling steps. Our analysis focuses on when token transitions occur and the decoded forecast becomes stable.

Figure 2 shows selected states from the two reverse trajectories. The prescribed-path sampler forms its main precipitation structure earlier and exhibits little visible change during the later sampling steps. In contrast, the uniform-path sampler continues to revise the forecast through the final step. Figure 3 quantifies the difference in terms of the token jumping rate in each sampling step. The prescribed path concentrates token updates earlier, with 95% of its observed transitions completed by step 32, compared with step 64 for the uniform path. It produces fewer transitions overall, averaging 3 jumps per token compared with 4 under the uniform path. The prescribed sampler

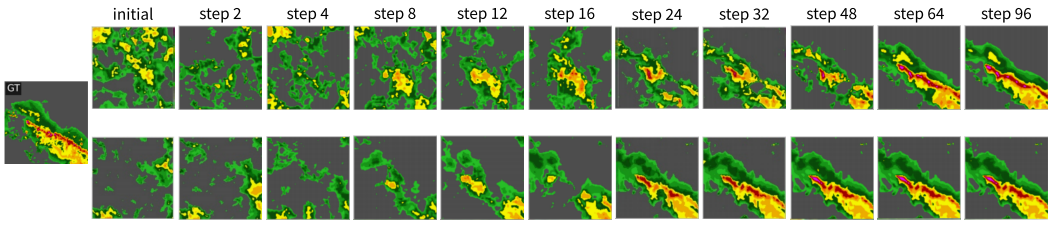
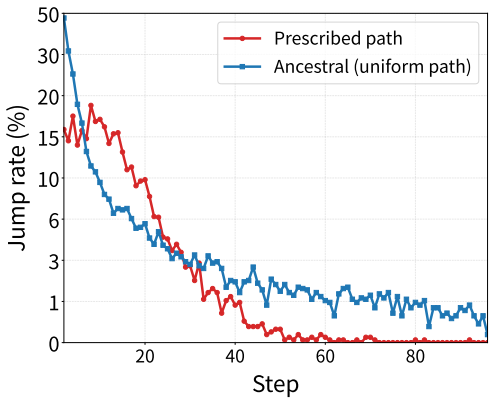


Figure 2: Decoded reverse trajectories. Rows show the uniform-path (first) and prescribed-path (second) samplers at selected sampling steps, indicating earlier stabilization under prescribed path.



		Single ensemble			Fusion
Sampler	Metric	30 steps	48 steps	64 steps	100 steps
Uniform	mCSI	0.2222	0.3007	0.3460	0.4186
Uniform	mCSI Pool4	0.2643	0.3498	0.4015	0.5049
Uniform	mCSI Pool16	0.3610	0.4513	0.5084	0.6348
Uniform	CSI-219	0.0301	0.0673	0.0967	0.1545
Uniform	PSNR ↑	21.60	24.01	25.38	27.52
Uniform	SSIM ↑	0.6422	0.7018	0.7659	0.8331
Prescribed	mCSI	0.3698	0.3873	0.3894	0.4177
Prescribed	mCSI Pool4	0.4203	0.4433	0.4462	0.4790
Prescribed	mCSI Pool16	0.5254	0.5562	0.5590	0.5715
Prescribed	CSI-219	0.0985	0.1216	0.1232	0.1511
Prescribed	PSNR ↑	25.95	26.10	26.11	26.66
Prescribed	SSIM ↑	0.7797	0.7895	0.7901	0.8224

Figure 3 and Table 4: Token jump rates and forecast quality under the uniform and prescribed diffusion paths. Figure 3 shows that the uniform-path sampler continues revising tokens throughout sampling, whereas the prescribed-path sampler concentrates transitions earlier. Table 4 reports quality at different sampling budgets.

therefore enters a stable regime earlier, whereas the uniform sampler distributes token revisions over a larger portion of the reverse trajectory. Together, these diagnostics show that path design provides measurable control over the timing of token updates during generation. The uniform and prescribed paths achieve comparable final forecast quality in Table 4, indicating that the primary effect of path design is to reorganize the sampling trajectory rather than improve forecast accuracy. The prescribed path reaches a stable state earlier, suggesting substantial smaller number of sampling steps.

More ablations including evaluation of the efficiency of the feature layout in Equation (3), the radar-field supervision loss (8) are given in Appendix C.5 and visualizations of the final predictions of different models are given in Appendix C.6. Discussion of the limitations are given in Appendix D.

5 Conclusion and Limitations

We introduced DPPN the first discrete diffusion framework for probabilistic precipitation nowcasting, which recasts forecasting as generation in a learned discrete weather representation. By bridging Gaussian-relaxed and discrete token states, DPPN combines compact discrete representations with diffusion-based probabilistic generation, while supporting both direct forecasting and cascaded refinement. On SEVIR, DPPN achieves the strongest pooled-scale event detection among the evaluated continuous diffusion and flow-matching baselines especially for longer lead times, using less than half the trainable parameters of its continuous counterparts. Beyond forecast accuracy, we show that the corruption path can be estimated from data and paired with an exactly derived reverse sampler, making the generation trajectory an explicit and controllable design component. We explore replacing the standard uniform-state corruption path with a prescribed, empirically-derived kernel that concentrates corruption on plausible forecast-error token transitions, fine-tuning the denoiser to estimate the clean-token posterior and thereby steering generation toward realistic error patterns, yielding improved inference efficiency through fewer sampling steps. These results demonstrate the potential of discrete diffusion as a compact and controllable framework for probabilistic precipitation nowcasting.

AI USE STATEMENT

We used generative AI tools to assist with writing and polishing the manuscript, plotting figures, and implementing code for the experiments. The authors reviewed and verified the AI-assisted content presented in this paper and take responsibility for the final text, claims, results, and artifacts.

REPRODUCIBILITY STATEMENT

The method and experimental protocol are described in Sections 3 and 4. Appendix C.1 details dataset processing, and Appendix C.2 provides implementation, training, checkpoint-selection, and sampling settings. An anonymous code repository is available at `https://anonymous.4open.science/r/DPPN-B2C0`.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL `https://papers.nips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html`.

Cong Bai, Feng Sun, Jinglin Zhang, Yi Song, and Shengyong Chen. Rainformer: Features extraction balanced network for radar-based precipitation nowcasting. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022. doi: 10.1109/LGRS.2022.3162882.

Gabriele Franch, Elena Tomasi, Rishabh Wanjari, Virginia Poli, Chiara Cardinali, Pier Paolo Alberoni, and Marco Cristoforetti. GPTCast: a weather language model for precipitation nowcasting. *Geoscientific Model Development*, 18:5351–5371, 2025. doi: 10.5194/gmd-18-5351-2025. URL `https://gmd.copernicus.org/articles/18/5351/2025/`.

Zhihan Gao, Xingjian Shi, Hao Wang, Yi Zhu, Yuyang Wang, Mu Li, and Dit-Yan Yeung. Earthformer: Exploring space-time transformers for earth system forecasting. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL `https://arxiv.org/abs/2207.05833`.

Zhihan Gao, Xingjian Shi, Boran Han, Hao Wang, Xiaoyong Jin, Danielle Maddix, Yi Zhu, Mu Li, and Yuyang Wang. PreDiff: Precipitation nowcasting with latent diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL `https://papers.nips.cc/paper/2023/hash/f82ba6a6b981fbbecf5f2ee5de7db39c-Abstract-Conference.html`.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/hash/f0d629a734b56a642701bba7bc8bb3ed-Abstract-Conference.html`.

Junchao Gong, Lei Bai, Peng Ye, Wanghan Xu, Na Liu, Jianhua Dai, Xiaokang Yang, and Wanli Ouyang. CasCast: Skillful high-resolution precipitation nowcasting via cascaded modelling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15809–15822. PMLR, 2024. URL `https://proceedings.mlr.press/v235/gong24a.html`.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL `https://papers.nips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html`.

Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11474–11484, 2020. URL `https://openaccess.thecvf.com/content_CVPR_2020/html/Le_`

Guen_Disentangling_Physical_Dynamics_From_Unknown_Factors_for_Unsupervised_Video_Prediction_CVPR_2020_paper.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Cristian Meo, Varun Sarathchandran, Avijit Majhi, Shao Hung, Carlo Saccardi, Ruben Imhoff, Roberto Deidda, Remko Uijlenhoet, and Justin Dauwels. BlockGPT: Spatio-temporal modelling of rainfall via frame-level autoregression. *arXiv preprint arXiv:2510.06293*, 2025. URL https://arxiv.org/abs/2510.06293.

Bernardo Perrone Ribeiro and Jana Faganeli Pucer. Flowcast: Advancing precipitation nowcasting with conditional flow matching. In *International Conference on Learning Representations*, 2026.

Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, Rachel Prudden, Amol Mandhane, Aidan Clark, Andrew Brock, Karen Simonyan, Raia Hadsell, Niall Robinson, Ellen Clancy, Alberto Arribas, and Shakir Mohamed. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597:672–677, 2021. doi: 10.1038/s41586-021-03854-z. URL https://www.nature.com/articles/s41586-021-03854-z.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Abstract.html.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb0b13cc515724ab8015bc978fdde0ad-Abstract-Conference.html.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52584–52619. PMLR, 2025. URL https://proceedings.mlr.press/v267/sahoo25a.html.

Johannes Schusterbauer, Jannik Wiese, Nick Stracke, Timy Phan, and Björn Ommer. Probabilistic precipitation nowcasting with rectified flow transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://compvis.github.io/weather-rf/.

Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky T. Q. Chen. Flow matching with general discrete paths:

A kinetic-optimal perspective. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/29cd7f8331d13ede6dc6d6ef3dfacb70-Abstract-Conference.html.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://papers.nips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6306–6315, 2017.

Mark Veillette, Siddharth Samsi, and Chris Mattioli. SEVIR: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fa78a16157fed00d7a80515818432169-Abstract.html.

Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/e5f6ad6ce374177eef023bf5d0c018b6-Abstract.html.

Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Fei-Fei Li. Eidetic 3D LSTM: A model for video prediction and beyond. In *International Conference on Learning Representations*, 2019. URL https://faculty.ucmerced.edu/mhyang/papers/iclr2019_eidetic3d.pdf.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers, 2021. URL https://arxiv.org/abs/2104.10157.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=pfNyExj7z2.

Yuchen Zhang, Mingsheng Long, Kaiyuan Chen, Lanxiang Xing, Ronghua Jin, Michael I. Jordan, and Jianmin Wang. Skilful nowcasting of extreme precipitation with NowcastNet. *Nature*, 619: 526–532, 2023. doi: 10.1038/s41586-023-06184-4. URL https://www.nature.com/articles/s41586-023-06184-4.

A Tokenizer and Architecture Details

The frame-wise tokenizer introduced in Section 3.1.1 is a hierarchical VQ-VAE. It encodes each 128×128 vertically integrated liquid (VIL) radar frame into a top-level 16×16 grid of indices from a codebook of $K = 512$ eight-dimensional atoms, while a bottom-level 32×32 grid refines the reconstruction. DPPN operates only on the frozen top-level codebook, which provides a shared vocabulary for the conditioning and target sequences; the bottom-level tokens remain internal to the tokenizer. This representation preserves the overall structure of the radar fields with high peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) in Section A.1. Individual codebook atoms also show consistent associations with background, boundary, and compact

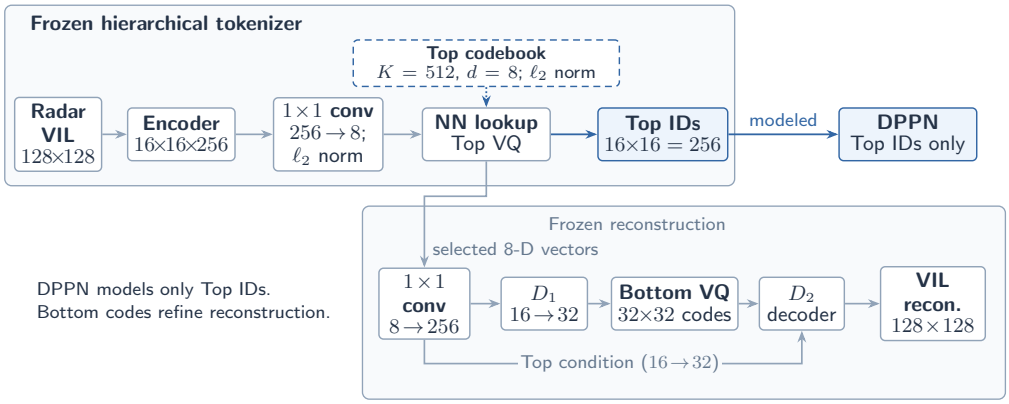


Figure 4: Frozen hierarchical tokenizer and DPPN interface. DPPN models the 256 top-level token IDs; the top-conditioned bottom branch refines reconstruction.

high-intensity precipitation patterns, supporting their interpretation as a learned vocabulary of radar structures in Section A.2.

A.1 Dictionary Size, Utilization, and Reconstruction Quality

Tokenizer architecture. The frozen pretrained tokenizer follows a two-level hierarchical architecture adapted from VQ-VAE-2 Razavi et al. (2019). Each 128×128 radar VIL frame is represented at two spatial resolutions: a top-level 16×16 grid that captures coarse precipitation structures and a bottom-level 32×32 grid that provides finer spatial refinement. The bottom-level representation is decoded conditionally on the top-level representation. Both levels are used by the tokenizer to reconstruct the radar field; DPPN models only the top-level tokens. Each radar frame is therefore represented in DPPN by $16 \times 16 = 256$ discrete tokens. At the top level, the encoder produces a 16×16 feature map with 256 channels. Before vector quantization, a learned 1×1 convolution projects each 256-dimensional encoder feature into an eight-dimensional lookup space. Each projected feature is then assigned to its nearest entry in a codebook containing 512 eight-dimensional vectors, following the nearest-neighbor vector quantization procedure of VQ-VAE van den Oord et al. (2017). After quantization, a second learned 1×1 convolution projects the selected 8-dimensional codebook vectors back to 256 channels for hierarchical decoding. The low-dimensional lookup space makes codebook learning and nearest-neighbor matching more tractable while preserving the higher channel capacity used by the encoder and decoder. Both the projected encoder features and the codebook vectors are ℓ_2-normalized before nearest-neighbor matching. The combination of low-dimensional codebook lookup and ℓ_2 normalization follows the codebook-learning techniques used in improved VQGAN Yu et al. (2022). These techniques are intended to promote broader and more stable codebook utilization by reducing sensitivity to feature magnitude and limiting the concentration of token assignments on a small subset of codebook entries. The encoder, the codebooks at both hierarchical levels, and the decoder remain frozen throughout DPPN training and all analyses reported in this appendix. Figure 4 summarizes the tokenizer and its interface with DPPN.

Dictionary size and utilization. Unless otherwise specified, the term *dictionary* in this work refers to the top-level codebook used by DPPN rather than to the complete set of representations in the hierarchical tokenizer. This dictionary contains $K = 512$ entries, each represented by an 8-dimensional lookup vector. Because the top-level representation has a spatial resolution of 16×16, each radar frame is represented by 256 categorical variables, with each variable taking one of 512 possible token values. The same top-level dictionary serves three purposes in our analysis. First, its token indices define the discrete vocabulary modeled by DPPN. Second, the radar fields and local rainfall structures associated with individual dictionary entries are examined to visualize the precipitation patterns represented by the learned tokens. Third, the empirical token frequencies are used to evaluate dictionary utilization. All 512 top-level entries occur in both the training and validation sets, corresponding to 100% dictionary coverage on the validation set. The validation perplexity is

Table 5: Pixel-level reconstruction of the tokenizer on validation frames. CSI is reported per VIL threshold; SSIM and PSNR are overall means computed across all frames.

Metric	16	74	133	160	181	219	Mean
CSI	0.8583	0.8036	0.6132	0.5272	0.4896	0.3568	0.6081
SSIM	–	–	–	–	–	–	0.8667
PSNR (dB)	–	–	–	–	–	–	28.96

400.52, computed as

$$\mathrm{PPL} = \exp\left(-\sum_{k=1}^{K} p_k \log p_k\right), \tag{12}$$

where p_k denotes the empirical frequency of dictionary entry k across all top-level token positions in the validation set. This perplexity corresponds to an effective vocabulary size of approximately 401 entries. The combination of complete coverage and high perplexity indicates that the dictionary is broadly utilized, although the usage frequencies are non-uniform because some learned precipitation patterns occur more frequently than others. Dictionary coverage and perplexity characterize complementary aspects of utilization: coverage determines whether each entry is used at least once, whereas perplexity measures the diversity and balance of token usage across the dictionary.

Reconstruction protocol and quality. Broad dictionary utilization alone does not establish that the discrete representation preserves the precipitation information required for nowcasting. We therefore evaluate the reconstruction quality of the complete frozen hierarchical tokenizer on all validation samples, using all six target frames from each sample. Each normalized radar field is encoded into the top- and bottom-level representations and then reconstructed directly through the tokenizer, without DPPN forecasting or diffusion. The reported metrics therefore isolate the information loss introduced by tokenization and decoding from errors introduced by the forecasting model. For each intensity threshold, the CSI is computed from hits, misses, and false alarms accumulated over all evaluated frames and pixels. The mean CSI (mCSI) is obtained by averaging the CSI values at the six evaluation thresholds. The SSIM is computed for each reconstructed frame and then averaged over the validation set. The PSNR is computed from the aggregate mean squared reconstruction error on the normalized radar fields. As reported in Table 5, the tokenizer achieves an mCSI of 0.6081, an SSIM of 0.8667, and a PSNR of 28.96 dB on the validation set. Reconstruction performance decreases at the highest precipitation intensities. In particular, CSI@219 is 0.3568, compared with an mCSI of 0.6081 across all six thresholds. These results indicate that the hierarchical tokenizer preserves the overall spatial structure and moderate-intensity precipitation patterns while providing a broadly utilized top-level dictionary. However, accurately reconstructing the most intense and comparatively sparse precipitation events remains more challenging.

Figure 5 presents the reconstruction for one diagnostic case. The reconstructed field closely matches the overall spatial structure of the input, while the highest-intensity regions exhibit some attenuation.

A.2 Dictionary Atoms as a Weather Vocabulary

We examine whether the discrete dictionary represents recurring precipitation-related radar patterns, supporting its use as a vocabulary of *weather-language* tokens. An atom is a learned codebook vector; a token is its discrete index at a spatial grid position. All analyses use the same tokenizer described above (Section A.1).

Associating atoms with radar patterns. For each token, we accumulate an 11-dimensional descriptor over its spatially aligned 8×8 radar crops (intensity statistics, spatial gradient, and coverage at the six VIL thresholds $\{16, 74, 133, 160, 181, 219\}$), computed from 32,709 training sequences (50.2M token observations). Figure 6 shows 16 illustrative tokens in two rows of eight, selected from the previous 64-token atlas to display background, boundary, broad-smooth, compact high-intensity, and mixed patterns (e.g., atoms 150, 334, 309, 229, and 492, respectively). This display subset is not a random or frequency-representative sample. These associations remain stable across the six forecast lead positions, with a median stability score of 0.986 across all tokens.

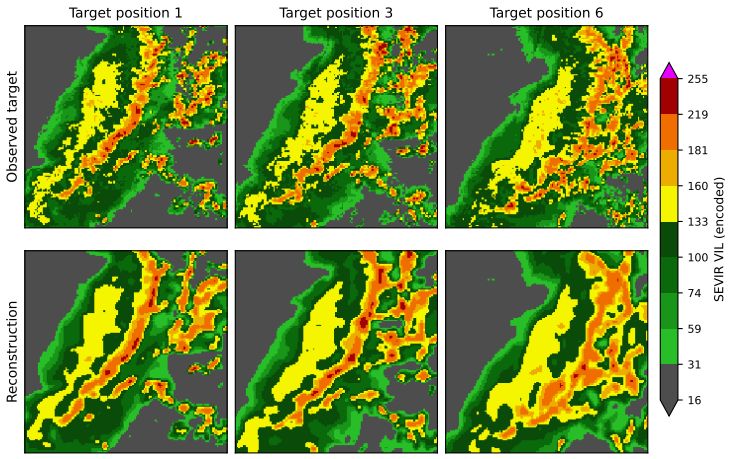


Figure 5: Observed targets and their reconstructions.

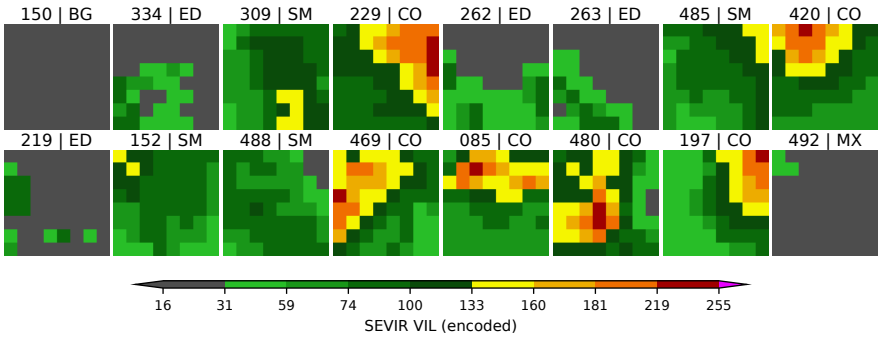


Figure 6: Observed radar crops associated with 16 selected tokens, arranged in two rows of eight. Numbers are zero-based implementation token IDs; the mathematical indexing convention in the main text is one-based. Each tile is an observed 8×8 crop nearest to that token's full-archive mean descriptor, shown with the official SEVIR encoded-VIL palette and bin boundaries. BG, ED, SM, CO, and MX denote background/low echo, echo boundary, broad smooth echo, compact high-intensity core, and mixed/context-dependent heuristic profiles, respectively; they are not independently annotated weather classes.

A weather-language vocabulary. The following findings support taking the learned codebook as the token vocabulary of a weather language. First, individual atoms correspond to recurring, lead-time-stable radar morphology rather than arbitrary indices. Second, the vocabulary is fully used, at a 100% utilization rate across training and validation data in Section A.1. Third, reconstruction through this vocabulary preserves the observed field with high PSNR and structural similarity in Section A.1, so the tokens carry the information DPPN's diffusion process needs to reconstruct precipitation fields. We therefore treat the codebook as a weather-language vocabulary and DPPN as modeling its conditional evolution, while the atom-to-morphology mapping itself remains statistical rather than a one-to-one meteorological labeling.

B Training Objectives and Sampling Algorithms

B.1 Uniform-State Diffusion Objective and Ancestral Reverse Transition

Uniform-State Diffusion Objective. We use the per-token objective implemented by Duo (Sahoo et al., 2025). For one position, let i be the clean token, j the corrupted token, and p_k the predicted

probability of clean category k. With $\alpha = \alpha_t$ and $\dot{\alpha} = \dot{\alpha}_t$, define

$$B_k = K\alpha p_k + 1 - \alpha, \qquad d = \mathbf{1}[i = j], \qquad b = 1 - \alpha + K\alpha d, \qquad r = \frac{1-\alpha}{K\alpha + 1 - \alpha}. \tag{13}$$

The per-token integrand in Equation (7) is

$$\ell_{\mathrm{UD}} = \frac{\dot{\alpha}}{K\alpha}(T_1 - T_2), \tag{14}$$

$$T_1 = K\left(\frac{1}{b} - \frac{1}{B_j}\right), \tag{15}$$

$$T_2 = -(dr + 1 - d)\left(\sum_{k=1}^{K} \log B_k - K \log B_j\right) - \frac{K\alpha}{1-\alpha}(1-d)(\log B_i - \log B_j) + \left((K-1)rd - \frac{1-d}{r}\right)\log r. \tag{16}$$

Training evaluates this expression at interior noise levels and averages it over target positions. The same expression applies during Gaussian relaxation using the projected corrupted token and the corresponding discrete schedule.

Ancestral Reverse Transition For $s < t$, let $a = \alpha_s$, $b = \alpha_t$, and $j = z_t^i$ at one target position. Duo's clean-distribution substitution yields the reverse transition

$$R_{s \leftarrow t}(k \mid j, p) = \frac{(ap_k + (1-a)/K)\,((b/a)\mathbf{1}[k=j] + (1 - b/a)/K)}{bp_j + (1-b)/K}. \tag{17}$$

The sampler draws the next token from this distribution, using the denoiser's clean-token probabilities p. For cascaded guidance, Equation (11) combines two such distributions before sampling.

B.2 Controllable Diffusion Paths: Construction and Prescribed Sampling

This appendix provides the complete construction and reverse-sampling procedure for the prescribed-path variant introduced in Section 3.1.4. The tokenizer, denoiser architecture, and conditioning mechanism remain unchanged. Only the corruption kernel, fine-tuning objective, and reverse update differ from the uniform-state formulation in Section 3.1.

Motivation. The uniform-state path assigns the same corruption probability to every incorrect token. Although analytically convenient, this assumption does not reflect the structure of the learned precipitation vocabulary. Training-set statistics show that the deterministic forecast confuses a target token with only a small subset of the remaining vocabulary. These confusions are also asymmetric. For a moderate-intensity target token, the deterministic forecast commonly produces tokens associated with weaker or empty precipitation, whereas confusions with stronger precipitation tokens are rare. Our guiding principle is therefore that corruption should move a clean token toward plausible forecast errors rather than toward all vocabulary entries with equal probability.

Figure 7 illustrates this observation for one representative clean token. At the same retention probability, the uniform kernel distributes its off-diagonal probability mass equally over all incorrect tokens, whereas the prescribed kernel concentrates that mass on forecast–target confusions observed in the training data. This empirical structure motivates a sparse, direction-dependent corruption path derived from paired forecast and target tokens.

Empirical example. Table 6 reports the source values used in Figure 7. The uniform and prescribed kernels are compared at the same clean-token retention probability of 0.501, so they differ only in how the remaining probability mass is distributed among incorrect tokens. The uniform kernel assigns each incorrect token a probability of 9.8×10^{-4}. In contrast, the prescribed kernel assigns more mass to tokens that frequently occur in the deterministic forecast when the target token is $z_0^i = 241$. The lowest-cost candidate receives 12.7 times its uniform probability, whereas the highest-cost displayed candidates receive only 0.10 times their uniform probability. This redistribution reduces the entropy-based effective support from 512 to 31 tokens, indicating that the

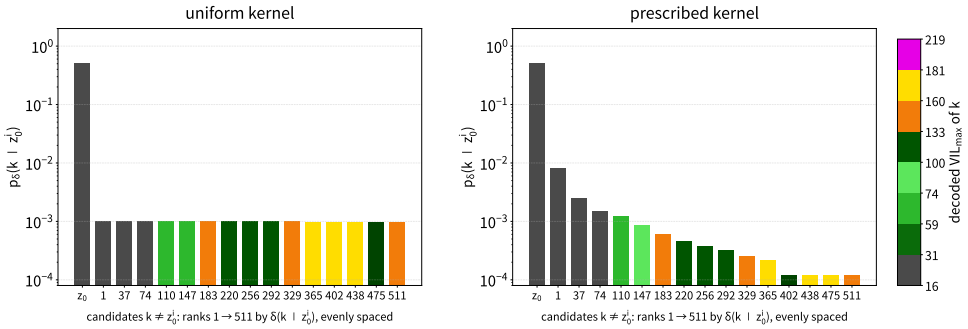


Figure 7: Uniform (left) and prescribed (right) corruption kernels for a representative moderate-intensity token. At matched retention probability, the uniform kernel assigns equal mass to all incorrect tokens, whereas the prescribed kernel concentrates corruption on a small set of forecast–target confusions observed in the training data. Candidate colors indicate decoded maximum VIL intensity. The prescribed kernel reduces the effective support from 512 to 31 tokens and preferentially assigns mass to weaker precipitation patterns.

prescribed corruption is concentrated on a small subset of empirically plausible confusions. For this representative moderate-intensity token, the most strongly favored candidates decode mainly to weak or empty precipitation patterns, consistent with the tendency of the deterministic forecast to underestimate precipitation intensity. The relationship is not determined by intensity alone, however, because the cost is estimated from paired forecast–target occurrences and therefore captures the broader empirical structure of forecast errors.

Cost and corruption kernel. Let C_{k,z_0^i} denote the number of training-set positions at which the ground-truth token is z_0^i and the paired deterministic forecast produces token k. We convert these counts into a smoothed conditional distribution and define the corresponding normalized negative log-frequency as

$$\widetilde{P}(k \mid z_0^i) = \frac{C_{k,z_0^i} + \frac{1}{2}}{\sum_{k'} C_{k',z_0^i} + \frac{K}{2}}, \qquad \ell_{\mathrm{raw}}(k \mid z_0^i) = \frac{-\log \widetilde{P}(k \mid z_0^i)}{\mathrm{median}_{(k,z_0):\, k \neq z_0} \left[-\log \widetilde{P}(k \mid z_0)\right]}. \tag{18}$$

The additive smoothing assigns a finite cost to unobserved pairs, while the negative logarithm gives lower costs to more frequently observed forecast–target confusions. Because C_{k,z_0^i} is direction-dependent, the resulting cost distinguishes corruption from z_0^i to k from the reverse relation. For the path to terminate at the clean-data endpoint, the clean token must be the unique minimum-cost state in every column. The raw empirical cost does not always satisfy this condition. We therefore repair it as

$$\delta(k \mid z_0^i) = s\left(\ell_{\mathrm{raw}}(k \mid z_0^i) - \ell_{\mathrm{raw}}(z_0^i \mid z_0^i) + \left(m - \min_{\mathrm{off}}\right)\mathbf{1}[k \neq z_0^i]\right), \tag{19}$$

where $m = 1$, $\min_{\mathrm{off}}$ is the smallest off-diagonal value after subtracting the diagonal, and s restores the median scale. Before this repair, only approximately half of the columns have the clean token as their minimum-cost entry. The repair preserves the ordering among off-diagonal entries while ensuring that the kernel concentrates on z_0^i at the clean-data endpoint. Using the repaired cost, we define the prescribed corruption kernel as

$$p_\rho(k \mid z_0^i) = \frac{\exp\left[-\beta(\rho)\delta(k \mid z_0^i)\right]}{\sum_{k'} \exp\left[-\beta(\rho)\delta(k' \mid z_0^i)\right]}, \qquad \beta(\rho) = \beta_{\max}\rho^2. \tag{20}$$

Here $\beta_{\max} > 0$ sets the terminal sharpness of the path. At $\rho = 0$, $\beta(0) = 0$ and the kernel is uniform over the vocabulary. As ρ approaches one, the increasing sharpness concentrates the kernel on its unique minimum, $k = z_0^i$. The prescribed path therefore has the same categorical-prior and clean-data endpoints as the uniform-state path, while following a sparse, data-informed trajectory between

Table 6: Source values for Figure 7. The clean token $z_0^i = 241$ has decoded VIL intensity 16 and empirical frequency 0.00371, where decoded VIL is measured from a constant-token plate. The displayed alternatives are sampled at evenly spaced cost ranks from 1 to 511. Uniform p^{unif} and prescribed probabilities $p_\rho(k \mid z_0^i)$ are evaluated at the same clean-token retention probability of 0.501. The ratio is $p_\rho(k \mid z_0^i)/p^{\mathrm{unif}}(k \mid z_0^i)$.

rank	token k	cost $\delta(k \mid z_0^i)$	decoded VIL	uniform prob.	prescribed prob.	ratio
clean	241	0.000	16	0.501	0.501	1.00×
1	468	0.587	0	9.77×10^{-4}	0.0124	12.69×
37	343	0.821	2	9.77×10^{-4}	2.84×10^{-3}	2.91×
74	225	0.892	0	9.77×10^{-4}	1.81×10^{-3}	1.86×
110	110	0.943	30	9.77×10^{-4}	1.31×10^{-3}	1.34×
147	119	1.006	22	9.77×10^{-4}	8.84×10^{-4}	0.91×
183	507	1.055	178	9.77×10^{-4}	6.48×10^{-4}	0.66×
220	382	1.101	84	9.77×10^{-4}	4.84×10^{-4}	0.50×
256	39	1.129	86	9.77×10^{-4}	4.05×10^{-4}	0.42×
292	247	1.164	66	9.77×10^{-4}	3.26×10^{-4}	0.33×
329	470	1.190	65	9.77×10^{-4}	2.76×10^{-4}	0.28×
365	221	1.233	165	9.77×10^{-4}	2.11×10^{-4}	0.22×
402	20	1.269	123	9.77×10^{-4}	1.68×10^{-4}	0.17×
438	368	1.349	97	9.77×10^{-4}	1.02×10^{-4}	0.10×
475	334	1.349	112	9.77×10^{-4}	1.02×10^{-4}	0.10×
511	26	1.349	135	9.77×10^{-4}	1.02×10^{-4}	0.10×

them. Let $\hat{p}(z_0^i)$ denote the empirical clean-token distribution. The marginal state distribution along the path is

$$\bar{p}_\rho(k) = \sum_{z_0^i} \hat{p}(z_0^i)\, p_\rho(k \mid z_0^i). \tag{21}$$

This is the state distribution observed by the denoiser during fine-tuning and the distribution that the reverse sampler is intended to reproduce. We estimate the full $K \times K$ co-occurrence table from 5.49×10^7 paired forecast–target token positions in the training set. For the representative token in Figure 7, the prescribed kernel reduces the effective support from 512 to 31 entries at matched retention probability.

Design space. The construction above is a family rather than a single path, and this family is what makes the generation process designable. Three ingredients are free. First, the cost δ may be any nonnegative, direction-dependent pairwise statistic that vanishes on the diagonal: empirical token co-occurrence as used here, codebook distances, intensity-level relations, or a hand-edited table that suppresses or promotes selected transitions. Second, the sharpness schedule $\beta(\rho)$ may be any monotone schedule with $\beta(0) = 0$ and $\beta(1)$ large enough to concentrate the kernel on the diagonal. Third, the prior $\hat{p}$ is the empirical clean-token marginal or a reweighted one that emphasizes selected entries, which shifts the population of entries the path can reach. The endpoints themselves are fixed by the tokenizer: the path always starts from the categorical prior and terminates at the data. Because the reverse rates are derived from the path rather than tuned, every admissible path keeps the sampler exact, and changing the path requires re-fitting the posterior through Equation (22); that re-fitting is the cost of this flexibility. We instantiate the cost from empirical co-occurrence and study that instance, while the remaining choices define the design space that path control can exploit.

Fine-tuning objective. We fine-tune the denoiser pretrained under Equation (4), continuing training under the prescribed path with the clean-token cross-entropy

$$\mathcal{L}_{\mathrm{ft}} = \mathbb{E}_{(z_0,c),\, \rho,\, z_\rho \sim \prod_i p_\rho(\cdot \mid z_0^i)} \left[-\frac{1}{N} \sum_{i=1}^{N} \log \pi_\theta^i\big(z_0^i \mid z_\rho, \rho, c\big) \right], \tag{22}$$

where ρ is drawn uniformly from $(0, 1)$, corruption is applied independently at each target position, and the pretrained tokenizer and decoder remain frozen. The path parameter enters the denoiser through its noise-level embedding, mapped so that the clean endpoint corresponds to zero noise.

The minimizer of Equation (22) is the posterior $q_\rho(z_0^i \mid k) \propto \hat{p}(z_0^i)\, p_\rho(k \mid z_0^i)$ implied by the prescribed path, which is exactly the quantity the reverse rates below require. We therefore replace the variational objective of Equation (7) during fine-tuning, since its derivation assumes the uniform-state path of Equation (4).

Replacing the objective is also supported numerically: on a three-code reduction of the path, the empirical minimizer of Equation (7) deviates from the path posterior by 0.13 in total variation, whereas the clean-token cross-entropy and the generalized KL objective of (Sahoo et al., 2025) both attain it to within 0.002, the resolution of the grid used for the comparison. The radar-space term of Equation (8) is retained, so the fine-tuning loss is $\mathcal{L} = \mathcal{L}_{\text{ft}} + \lambda_{\text{radar}}\mathcal{L}_{\text{radar}}$.

Sampling rule. Fix a hypothesis z_0^i for the clean entry at one target position, and write $j = z_\rho^i$ for its current state along the path, mirroring the notation $j = z_t^i$ used for the ancestral transition above. Differentiating Equation (20) with respect to ρ yields the smallest nonnegative rate that transports $p_\rho(\cdot \mid z_0^i)$ along the path,

$$\phi_\rho(k \mid z_0^i) = \nu(z_0^i)\, p_\rho(k \mid z_0^i) + \partial_\rho p_\rho(k \mid z_0^i), \qquad \nu(z_0^i) = \dot{\beta}(\rho)\left(\max_k \delta(k \mid z_0^i) - \bar{\delta}_\rho(z_0^i)\right) + \epsilon, \tag{23}$$

where $\bar{\delta}_\rho(z_0^i) = \sum_k p_\rho(k \mid z_0^i)\delta(k \mid z_0^i)$ and $\epsilon > 0$ is numerical. Since z_0^i is unknown at generation time, the sampler averages these per-hypothesis rates under the fine-tuned posterior,

$$W_\rho(j \to k) = \sum_{z_0^i} \pi_\theta^i(z_0^i \mid j, \rho, c)\, \phi_\rho(k \mid z_0^i), \qquad k \neq j, \tag{24}$$

which reproduces the prescribed marginals $\bar{p}_\rho$ exactly whenever π_θ matches the posterior targeted by Equation (22); replacing the posterior by a single hypothesis, such as its mode, does not preserve them. Freezing the rate over one step of the path parameter gives the update

$$\Pr[\text{jump}] = 1 - \exp\left(-\Delta\rho\, \nu_{\text{out}}(j)\right), \qquad \Pr[k \mid \text{jump}] = \frac{W_\rho(j \to k)}{\nu_{\text{out}}(j)}, \qquad \nu_{\text{out}}(j) = \sum_{k \neq j} W_\rho(j \to k), \tag{25}$$

which remains well defined when $\Delta\rho\, \nu_{\text{out}}$ is large, unlike a first-order rule that treats $\Delta\rho\, \nu_{\text{out}}$ as a probability. Let $\nu_{\text{grid},u}(j)$ be the exit rate at path position u obtained by using the path-implied posterior $q_u(z_0 \mid j) \propto \hat{p}(z_0) p_u(j \mid z_0)$ in Equation (24). Define the cumulative path speed $A(\rho) = \int_0^\rho \text{median}_j\, \nu_{\text{grid},u}(j)\, du$. We choose the $S + 1$ grid points so that $A(\rho_r) = rA(1)/S$ for $r = 0, \ldots, S$, where S is the number of reverse steps.

Because $\beta(\rho) = \beta_{\max}\rho^2$ has $\dot{\beta}(0) = 0$, the reverse rate vanishes at the prior endpoint, so the first step of the finite-$\Delta\rho$ Euler discretization moves no token; the effective chain is one step shorter than the nominal budget. Equation (24) requires only the clean-token posterior, so the path geometry is precomputed while the denoiser supplies one distribution per step, and conditional guidance is applied to π_θ inside Equation (24) rather than to the transition as in Equation (11).

C Additional Experiments

C.1 Data Alignment and Reproducibility

SEVIR. The Storm EVent ImageRy (SEVIR) dataset (Veillette et al., 2020) contains spatially and temporally aligned satellite, radar, and lightning observations of storm events over the United States. Each event covers a 384×384 km region over four hours. Following PreDiff (Gao et al., 2023), we use the NEXRAD Vertically Integrated Liquid (VIL) radar product and preprocess it from 384×384 pixels at five-minute intervals to 128×128 pixels at 10-minute intervals. Each sample uses seven context frames to predict six target frames, corresponding to one hour of observations and forecast lead times of 10 to 60 minutes. The processed split contains 35718 training, 9060 validation, and 12159 test sequences. Dataset statistics are summarized in Table 7.

C.2 Implementation Details

Architecture and optimization. The DPPN-cas denoiser consists of 12 Transformer blocks with a hidden dimension of 768, 12 attention heads, and a dropout rate of 0.1. The model contains 102M

Algorithm 1 DPPN Training

Require: Dataset $\mathcal{D}$ containing radar sequences or precomputed tokens; frozen tokenizer $\mathcal{T} = Q \circ E$ and decoder interfaces D, D_{prob}; denoiser π_θ; deterministic forecaster f for cascaded training.
Require: Gaussian schedule $(a_t, b_t, \alpha_t^{\mathrm{G}}, \dot{\alpha}_t^{\mathrm{G}})$; discrete schedule $(\alpha_t^{\mathrm{D}}, \dot{\alpha}_t^{\mathrm{D}})$; relaxation temperature τ and switch step n_{relax}; loss weight λ_{radar}; condition-drop probability p_{drop}.
Require: Training mode $\mathrm{mode} \in \{\textsc{uniform}, \textsc{path}\}$; for $\mathrm{mode} = \textsc{path}$, prescribed kernel $p_\rho(\cdot \mid \cdot)$ from Equation (20) and a pretrained π_θ (Appendix B.2).
Initialize denoiser parameters θ (random, or pretrained under $\mathrm{mode} = \textsc{uniform}$ when $\mathrm{mode} = \textsc{path}$).
while not converged **do**
  Sample a training batch from $\mathcal{D}$.
  Obtain target tokens Z_Y from the cache or set $Z_Y \leftarrow \mathcal{T}(Y)$.
  Obtain c from the cache or set $c \leftarrow \mathcal{T}(X)$ (direct) or $c \leftarrow \mathcal{T}(f(X))$ (cascaded).
  Set $Y^{\mathrm{ref}} \leftarrow Y$ when available; otherwise set $Y^{\mathrm{ref}} \leftarrow D(Z_Y)$.
  For cascaded training, set the condition embeddings to zero for the batch with probability p_{drop}.
  if $\mathrm{mode} = \textsc{uniform}$ **then**
    Sample diffusion time $t \sim \mathcal{U}(0, 1)$.
    if training step $n \le n_{\mathrm{relax}}$ **then**
      Sample $\epsilon \sim \mathcal{N}(0, I)$ and set $u_t \leftarrow a_t \,\mathrm{OneHot}(Z_Y) + b_t \epsilon$.
      Set $z_t \leftarrow \arg\max_k u_{t,k}$, $h_t \leftarrow e_{\mathrm{tok}}[\mathrm{softmax}(u_t/\tau)]$, and $(\bar{\alpha}_t, \dot{\bar{\alpha}}_t) \leftarrow (\alpha_t^{\mathrm{G}}, \dot{\alpha}_t^{\mathrm{G}})$.
    else
      Sample $z_t \sim q_t(\cdot \mid Z_Y; \alpha_t^{\mathrm{D}})$, set $h_t \leftarrow e_{\mathrm{tok}}(z_t)$, and $(\bar{\alpha}_t, \dot{\bar{\alpha}}_t) \leftarrow (\alpha_t^{\mathrm{D}}, \dot{\alpha}_t^{\mathrm{D}})$.
    end if
    Predict $\Pi_\theta \leftarrow \pi_\theta(\cdot \mid h_t, t, c)$ and compute $\mathcal{L}_{\mathrm{main}} \leftarrow \mathcal{L}_{\mathrm{token}}$ from $(\Pi_\theta, z_t, Z_Y, \bar{\alpha}_t, \dot{\bar{\alpha}}_t)$ via Equation (7).
  else *(*$\mathrm{mode} = \textsc{path}$*)*
    Sample path parameter $\rho \sim \mathcal{U}(0, 1)$ and set $t \leftarrow \rho$ for the noise-level embedding.
    Sample $z_\rho \sim \prod_i p_\rho(\cdot \mid Z_Y^i)$ from Equation (20) and set $h_t \leftarrow e_{\mathrm{tok}}(z_\rho)$.
    Predict $\Pi_\theta \leftarrow \pi_\theta(\cdot \mid h_t, \rho, c)$ and compute $\mathcal{L}_{\mathrm{main}} \leftarrow \mathcal{L}_{\mathrm{ft}}$ from (Π_θ, Z_Y) via Equation (22).
  end if
  Set $\widehat{Y}_t^{\mathrm{soft}} \leftarrow D_{\mathrm{prob}}(\Pi_\theta)$ and compute $\mathcal{L}_{\mathrm{radar}}$ using Equation 8.
  Set $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{main}} + \lambda_{\mathrm{radar}} \mathcal{L}_{\mathrm{radar}}$.
  Update θ by a gradient step on $\mathcal{L}$.
end while

trainable parameters, excluding the frozen tokenizer. We optimize the model using AdamW with a learning rate of 3×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and zero weight decay. The learning rate is linearly warmed up for 400 steps and then held constant. We use a global batch size of 512, gradient clipping at 1.0, and an exponential moving average with a decay rate of 0.9999. Condition dropout is applied with probability 0.1, and the radar-space loss is weighted by 0.1. Training proceeds for at most 8000 optimization steps.

Validation and checkpoint selection. We periodically evaluate the decoded forecasts on the validation set using mCSI. The checkpoint with the highest validation mCSI is selected for test-set evaluation.

Sampling. For cascaded inference, Earthformer first produces a deterministic six-frame forecast, which is encoded by the frozen tokenizer to obtain the conditioning tokens. Each realization is generated using 100 ancestral reverse steps with a guidance weight of 2.0, followed by greedy terminal noise removal and decoding through the frozen tokenizer. We generate eight independently initialized realizations for each input. Categorical metrics are computed from a radar-space ensemble summary; mean and p90 summaries are distinguished where reported. CRPS is computed from the individual ensemble members.

C.3 Ensemble Fusion for Discrete Diffusion

Each DPPN ensemble member is generated as a sequence of discrete codebook indices, for which arithmetic averaging is not meaningful. The decoded precipitation fields can be averaged in radar space, but pixel-wise averaging may smooth localized structures when the ensemble represents multiple plausible precipitation patterns. Motivated by probability-matching approaches used to summa-

Algorithm 2 DPPN Ensemble Sampling

Require: Historical radar sequence X; deterministic forecaster f for cascaded generation; frozen tokenizer $\mathcal{T} = Q \circ E$ and decoder D; trained denoiser π_θ.
Require: Sampling mode $\mathrm{mode} \in \{\textsc{uniform}, \textsc{path}\}$; ensemble size M; guidance scale g; reverse-time grid $1 = t_S > \cdots > t_0 = \varepsilon$ for $\mathrm{mode} = \textsc{uniform}$, or path grid $0 = \rho_0 < \cdots < \rho_S = 1$ whose points equalize increments of the cumulative path speed $A(\rho)$ for $\mathrm{mode} = \textsc{path}$ (Appendix B.2).
Ensure: Ensemble forecasts $\{\widehat{Y}^{(m)}\}_{m=1}^{M}$.

Set $c \leftarrow \mathcal{T}(X)$ for direct generation, or $c \leftarrow \mathcal{T}(f(X))$ for cascaded generation.
for $m = 1, \ldots, M$ **do**
  Sample $z_0^{(m),i} \sim \mathrm{Uniform}(\{1, \ldots, K\})$ independently for every future-token position i (initial state, indexed by t_S or ρ_0).
  if $\mathrm{mode} = \textsc{uniform}$ **then**
    for $r = S, \ldots, 1$ **do**
      $\pi_{\mathrm{cond}} \leftarrow \pi_\theta(\cdot \mid z_{t_r}^{(m)}, t_r, c)$.
      $R_{\mathrm{cond}} \leftarrow R_{t_{r-1} \leftarrow t_r}(z_{t_r}^{(m)}, \pi_{\mathrm{cond}})$.
      if cascaded generation and $g \neq 1$ **then**
        $\pi_{\mathrm{uncond}} \leftarrow \pi_\theta(\cdot \mid z_{t_r}^{(m)}, t_r, \varnothing)$.
        $R_{\mathrm{uncond}} \leftarrow R_{t_{r-1} \leftarrow t_r}(z_{t_r}^{(m)}, \pi_{\mathrm{uncond}})$.
        $R \leftarrow \mathrm{Normalize}(R_{\mathrm{cond}}^{g} R_{\mathrm{uncond}}^{1-g})$.
      else $R \leftarrow R_{\mathrm{cond}}$.
      end if
      Sample $z_{t_{r-1}}^{(m)} \sim \mathrm{Cat}(R)$.
    end for
    Set $\widehat{z}_0^{(m),i} \leftarrow \arg\max_k \pi_\theta^i(k \mid z_{t_0}^{(m)}, t_0, c)$ for every position i.
  else *(*$\mathrm{mode} = \textsc{path}$*)*
    for $r = 1, \ldots, S$ **do**
      Query $\pi_\theta(\cdot \mid z_{\rho_{r-1}}^{(m)}, \rho_{r-1}, c)$, applying conditional guidance as in Equation (11) when $g \neq 1$.
      Using this distribution, compute per-hypothesis rates $\phi_{\rho_{r-1}}(\cdot \mid z_0^i)$ from Equation (23) and averaged rates $W_{\rho_{r-1}}(z_{\rho_{r-1}}^{(m),i} \to \cdot)$ from Equation (24), for every position i.
      Sample the jump and destination from Equation (25) with $\Delta\rho = \rho_r - \rho_{r-1}$ to obtain $z_{\rho_r}^{(m)}$.
    end for
    Set $\widehat{z}_0^{(m),i} \leftarrow \arg\max_k \pi_\theta^i(k \mid z_{\rho_S}^{(m)}, \rho_S, c)$ for every position i.
  end if
  Decode $\widehat{Y}^{(m)} \leftarrow D(\widehat{z}_0^{(m)})$.
end for
return $\{\widehat{Y}^{(m)}\}_{m=1}^{M}$.

Table 7: Summary of the dataset used for evaluation.

Dataset	N_{train}	N_{val}	N_{test}	Dimensionality	Interval	L_{in}	L_{out}
SEVIR	35718	9060	12159	128×128	10 mins	7	6

rize NWP precipitation ensembles, we evaluate an intensity-adaptive quantile aggregation strategy. Unlike probability-matched means, which reorder pooled ensemble values according to the spatial ranking of an ensemble-mean field, our method operates independently at each pixel. We use the ensemble median for weaker precipitation and the 60th percentile for intensities at or above 74. This heuristic reduces the attenuation of intense precipitation relative to arithmetic averaging, but performs worse than ensemble-mean aggregation at lower intensity. It therefore represents a trade-off between preserving intense precipitation and accurately representing weaker events, rather than a uniformly superior ensemble summary. More sophisticated approaches that account jointly for intensity, spatial coherence, and ensemble uncertainty remain an important direction for future work. Figure 8 presents an illustrative example comparing four ensemble members with the final fused forecast. The fused forecast preserves the main precipitation structures across the ensemble while providing a single deterministic prediction with slight smoothing.

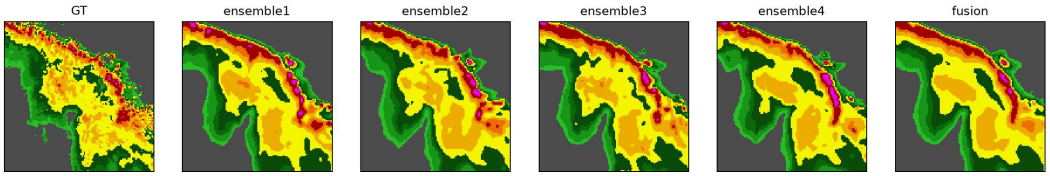


Figure 8: Ensembles and its final fusion prediction.

Table 8: Lead-time-dependent mean CSI (mCSI) and heavy-precipitation CSI at thresholds 181 and 219. Results are reported at native resolution (Pool1) and after 4×4 (Pool4) and 16×16 (Pool16) max pooling. Best results are shown in bold, and second-best results are underlined.

	mCSI				CSI-181				CSI-219			
Model	30mins	40mins	50mins	60mins	30mins	40mins	50mins	60mins	30mins	40mins	50mins	60mins
Pool1: native resolution												
Earthformer	0.4129	0.3642	0.3286	0.2974	0.2314	0.1727	0.1314	0.0936	0.0977	0.0606	0.0370	0.0186
CasCast	0.4256	0.3774	0.3428	0.3137	0.2644	0.2095	0.1725	0.1423	0.1506	0.1086	0.0834	0.0601
FREUD + LSM-L	0.4127	0.3642	0.3286	0.3003	0.2359	0.1766	0.1332	0.1015	0.1177	0.0732	0.0450	0.0257
FlowCast	**0.4381**	**0.3904**	**0.3555**	0.3276	**0.2723**	0.2169	0.1782	0.1464	0.1462	0.1023	0.0731	0.0517
DPPN-direct	0.4196	0.3759	0.3426	0.3165	0.2575	0.2073	0.1713	0.1444	0.1284	0.0930	0.0660	0.0496
DPPN-cas	0.4290	0.3866	0.3553	**0.3288**	0.2691	**0.2213**	**0.1881**	**0.1599**	**0.1556**	**0.1211**	**0.0984**	**0.0776**
Pool4: 4×4 max pooling												
Earthformer	0.4240	0.3728	0.3358	0.3034	0.2312	0.1700	0.1297	0.0916	0.1014	0.0635	0.0391	0.0205
CasCast	0.4906	0.4345	0.3937	0.3568	0.3470	0.2799	0.2328	0.1913	0.2257	0.1671	0.1331	0.0965
FREUD + LSM-L	0.4643	0.4073	0.3649	0.3322	0.2866	0.2130	0.1599	0.1234	0.1673	0.1052	0.0657	0.0398
FlowCast	0.4803	0.4301	0.3910	0.3572	0.3268	0.2681	0.2209	0.1817	0.1934	0.1405	0.1068	0.0748
DPPN-direct	0.4812	0.4344	0.3973	0.3671	0.3347	0.2796	0.2357	0.2013	0.1921	0.1458	0.1108	0.0874
DPPN-cas	**0.4990**	**0.4455**	**0.4102**	**0.3783**	**0.3634**	**0.3047**	**0.2620**	**0.2258**	**0.2526**	**0.2020**	**0.1681**	**0.1370**
Pool16: 16×16 max pooling												
Earthformer	0.4579	0.4021	0.3622	0.3270	0.2653	0.1964	0.1521	0.1106	0.1199	0.0798	0.0516	0.0268
CasCast	0.6022	0.5449	0.4996	0.4554	0.4801	0.4073	0.3503	0.2944	0.3470	0.2799	0.2321	0.1763
FREUD + LSM-L	0.5453	0.4806	0.4307	0.3935	0.3755	0.2848	0.2190	0.1721	0.2414	0.1598	0.1025	0.0672
FlowCast	0.5641	0.5160	0.4748	0.4332	0.4248	0.3643	0.3125	0.2592	0.2718	0.2158	0.1701	0.1228
DPPN-direct	**0.6226**	**0.5862**	**0.5562**	**0.5295**	**0.5055**	**0.4614**	**0.4249**	**0.3920**	0.3373	0.2908	0.2547	0.2235
DPPN-cas	0.5947	0.5431	0.5088	0.4742	0.4803	0.4254	0.3846	0.3440	**0.3798**	**0.3305**	**0.2914**	**0.2493**

C.4 Additional Experiments Results

Table 8 compares overall and heavy-precipitation performance from 30 to 60 minutes. At native resolution, FlowCast achieves the highest mCSI from 30 to 50 minutes, while DPPN-cas ranks first at 60 minutes. FlowCast also obtains the highest CSI-181 at 30 minutes, whereas DPPN-cas leads at 40–60 minutes and achieves the highest CSI-219 at every lead time. Under Pool4, DPPN-cas ranks first for mCSI, CSI-181, and CSI-219 at all four lead times. Under Pool16, DPPN-direct achieves the highest mCSI and CSI-181 throughout the 30–60-minute range, while DPPN-cas remains best for CSI-219. DPPN-direct becomes the second-best Pool16 method for CSI-219 at 40–60 minutes. Thus, DPPN-direct ranks more favorably as spatial tolerance increases, while the cascaded formulation retains a consistent advantage at the highest threshold. These results indicate that DPPN is particularly effective at maintaining the approximate location and spatial extent of intense precipitation at longer lead times. Its smaller advantage at native resolution suggests that fine-scale displacement and reconstruction errors remain more influential when no spatial tolerance is applied.

C.5 Ablation Studies

C.5.1 Radar-Space Supervision

We remove the radar-space loss in Equation (8) while retaining the token-level objective and the same eight-member fusion evaluation as the full DPPN-cas model. Table 9 compares both variants on the 12159-sample SEVIR test split. Removing radar-space supervision reduces native-resolution

Table 9: Effect of radar-space supervision on DPPN-cas.

	mCSI			CSI-219			
Variant	Pool1	Pool4	Pool16	Pool1	Pool4	Pool16	SSIM
DPPN-cas	**0.4186**	**0.5049**	**0.6348**	**0.1545**	0.2549	**0.4182**	**0.8331**
w/o radar loss	0.4063	0.5039	0.6291	0.1418	**0.2553**	0.4161	0.8196

Table 10: Performance in native resolution and computational efficiency of different feature layouts.

Model	mCSI ↑	CSI-160 ↑	CSI-181 ↑	CSI-219 ↑	SSIM ↑	CRPS ↓	Training time per epoch↓	Inference time ↓
Feature layout	0.4186	0.2972	0.2625	0.1545	0.8331	0.0216	6mins	1h
Sequence layout	0.4221	0.2999	0.2650	0.1569	0.8384	0.0218	20mins	3h

mCSI from 0.4186 to 0.4063 and SSIM from 0.8331 to 0.8196. The mCSI differences are smaller after spatial pooling: 0.5049 versus 0.5039 at Pool4 and 0.6348 versus 0.6291 at Pool16. At the highest precipitation threshold, CSI-219 decreases from 0.1545 to 0.1418 at Pool1 and from 0.4182 to 0.4161 at Pool16, while the ablated model is slightly higher at Pool4 (0.2553 versus 0.2549). Its MSE is 0.00566. Thus, radar-space supervision improves native-resolution reconstruction and event detection in this comparison, while its effect on pooled CSI is modest and not uniform across metrics.

C.5.2 Efficiency on Equation (3)

Equation 3 defines the efficient feature layout. It packs the embeddings of all future frames into the feature dimension and constructs one vector at each spatial location. We compare this layout with a sequence layout for representing future frames in the conditional diffusion Transformer. The *feature layout* performs attention over hw spatial tokens. The *sequence layout* retains each frame as a separate token, pairs each noisy target token with its corresponding deterministic prediction token, and performs attention over $T_{\text{out}}hw$ spatiotemporal tokens. Both layouts predict the same $T_{\text{out}} \times h \times w$ target tokens under the same diffusion objective.

Table 10 compares the current fused DPPN-cas scores for the feature layout with the supplied eight-member pixel-mean result for the sequence layout. The sequence configuration raises native-resolution mCSI by 0.0035 and SSIM by 0.0053. The feature configuration achieves higher pooled scores, with mCSI differences of 0.0202 at Pool4 and 0.0574 at Pool16. Its CSI-219 differences reach 0.0202 at Pool4 and 0.0738 at Pool16. With $T_{\text{out}} = 6$ and the same attention width, the feature layout uses one-sixth as many tokens and an attention map with one-thirty-sixth as many entries.

C.6 Visualization Cases

Visualizations of predictions from different diffusion models are provided in Figure 9 and 10. Earthformer produces relatively smooth forecasts, whereas the diffusion models recover substantially finer spatial details. CasCast generates detailed structures around the Earthformer prediction but also exhibits noticeable spatial discrepancies. FREUD shows degraded structure at longer lead times, while FlowCast produces detailed but sometimes misplaced precipitation patterns. DPPN provides a better balance between structural fidelity and fine-scale detail, maintaining the overall precipitation structure while recovering plausible localized features.

D Limitations

The proposed formulation has several identifiable limitations. First, forecast fidelity is constrained by the frozen tokenizer: quantization and reconstruction errors, particularly for high-intensity precipitation, establish an information ceiling for subsequent diffusion modeling. Second, cascaded conditioning introduces dependence on the deterministic predictor, as errors in $\widehat{Y}_{\text{det}}$ are inherited by the conditioning tokens and may propagate through the generative process. Third, performance is metric- and scale-dependent: although DPPN-cas provides stronger pooled-scale event detection, FlowCast achieves better native-resolution mCSI, high-intensity CSI, and CRPS in the evaluated

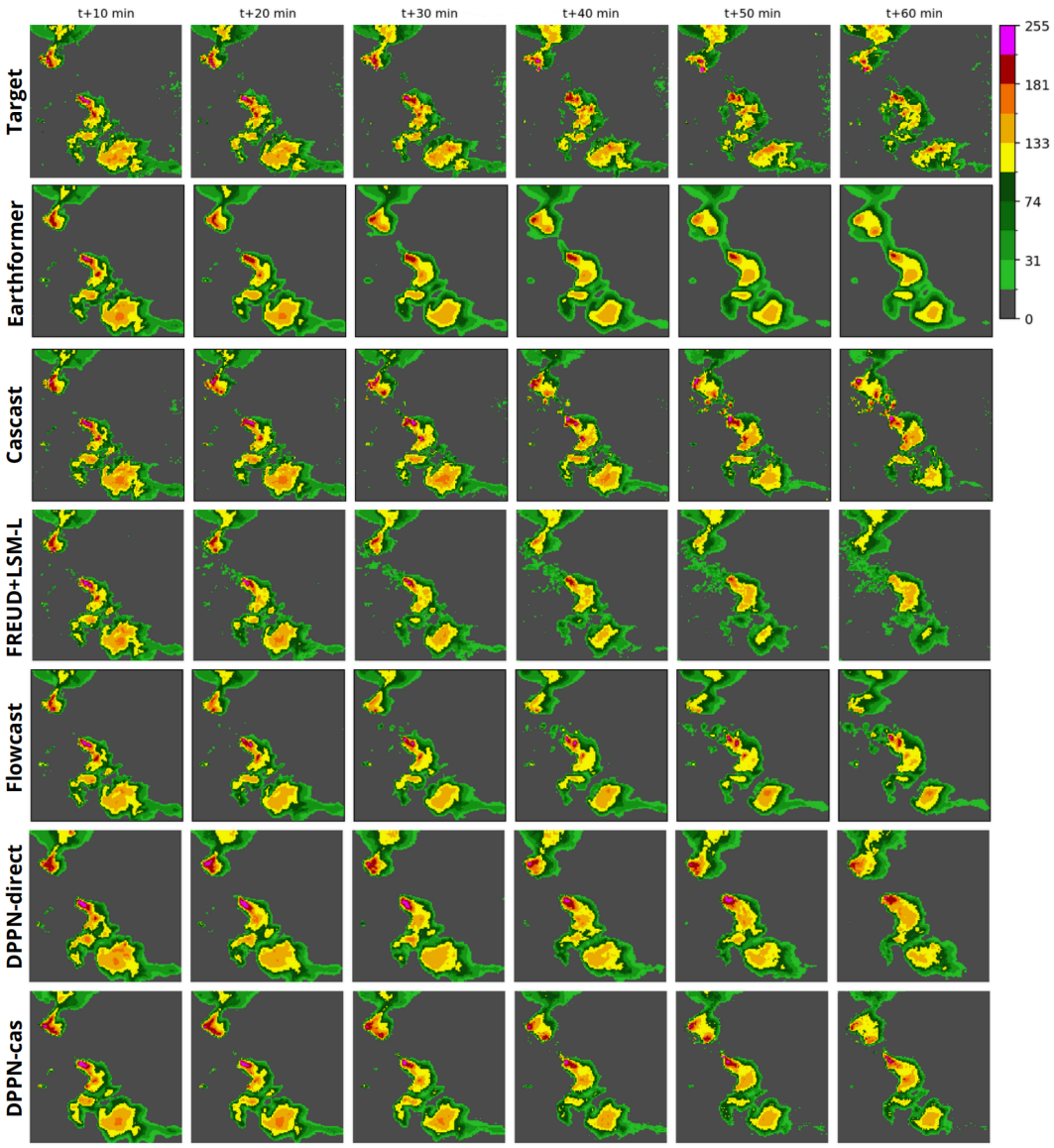


Figure 9: Qualitative comparison of precipitation forecasts produced by Earthformer, CasCast, FREUD+LSM-L, FlowCast, DPPN-direct and DPPN-cas at different forecast lead times.

comparison. Finally, the empirical study is restricted to SEVIR, and therefore does not establish the robustness of the learned representation, corruption path, or ensemble calibration across different radar domains and precipitation regimes.

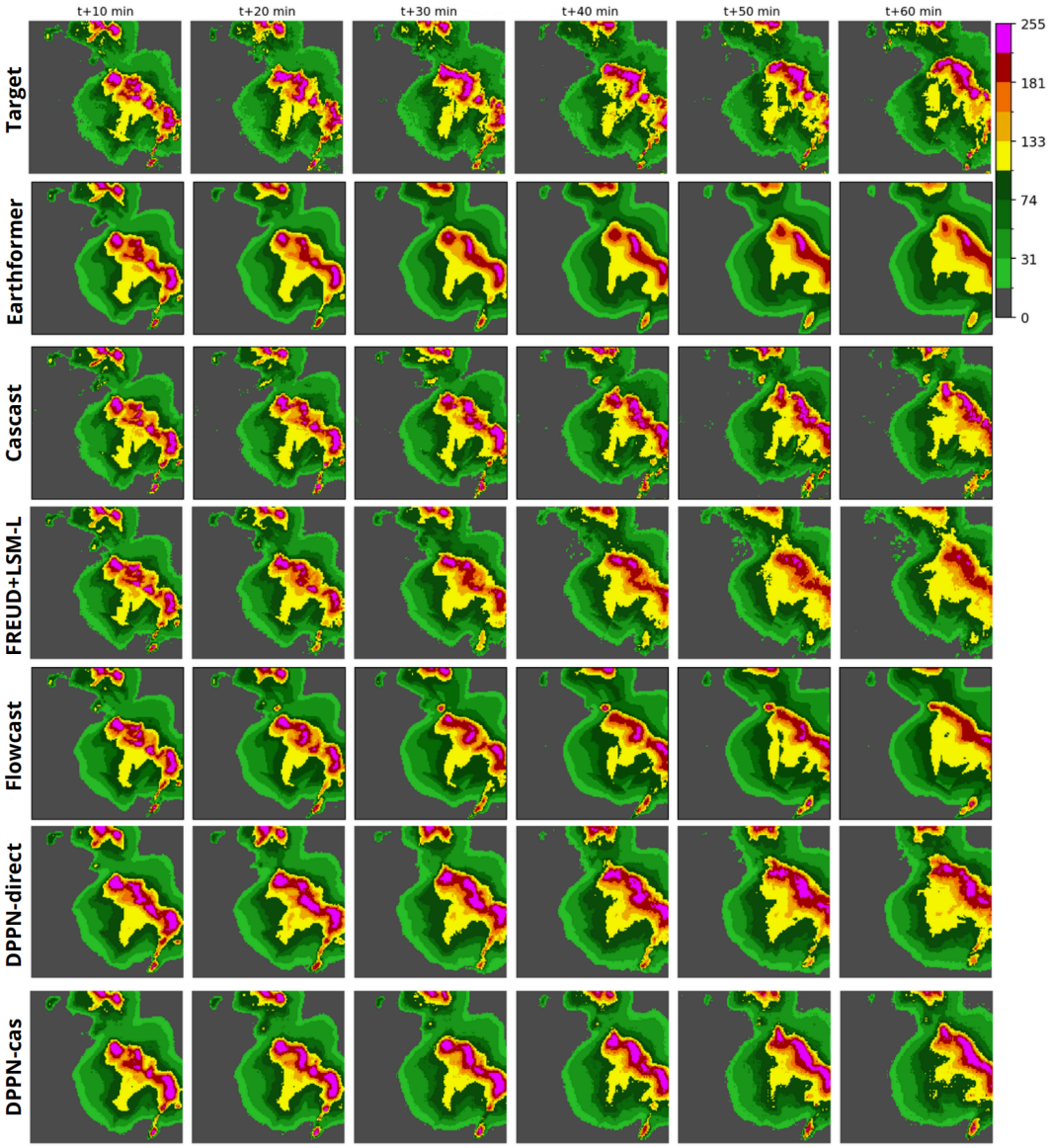


Figure 10: Qualitative comparison of precipitation forecasts produced by Earthformer, CasCast, FREUD+LSM-L, FlowCast, DPPN-direct and DPPN-cas at different forecast lead times.