DiffCast: A Unified Framework via Residual Diffusion for Precipitation Nowcasting

Supplementary Material

7. Datasets Detials

SEVIR, the Storm EVent ImagRy (SEVIR) [33] is an annotated, curated and spatio-temporally aligned dataset across five multiple data types including visible satellite imagery, infrared satellite imagery (mid-level water vapor and clean longwave window), NEXRAD radar mosaic of VIL(vertically integrated liquid mosaics) and ground lightning events. In this paper, we focus on the short term weather forecasting task and select all the radar mosaics of VIL as the main data. The dataset contains 20393 weather events from multiple sensors in 2017-2020. Each event consists of a 4-hour length sequence of images sampled in 5 minute steps covering 384 km×384 km patches sampled at locations throughout the continental U.S.. As our task is to predict the future VIL up to 20 frames (100 min) given 5 observed frames (25 min), we follow [8] to sample the 25 continuous frames with stride = 12 in every event and split the dataset into training, validation and test sets with the time point January 1, 2019 and June 1, 2019, respectively. The frames are rescaled back to the range 0-255 and binarized at thresholds [16,74,133,160,181,219] to calculate the CSI and HSS following original settings in [33].

MeteoNet [17] is a multimodel dataset including full time series of satellite and radar images, weather models and ground observations. It covers geographic areas of 550 km×550 km in the northwestern quarter of France and a span over three years, and records every 6 min from 2016 to 2018. Like the SEVIR, we split the radar sequence from 2016 to 2018 into training, validation and test sets with the time point January 1, 2018 and June 1, 2018, respectively. Then, we apply Algorithm 1 to filter precipitation events with a stride-20 sliding window to reduce the noise in the data. Note that a mean pixel threshold T_pixel is used as a filter to precipitation events. The data range of frames in MeteoNet is set to [0-70] and the thresholds are set to [12, 18, 24, 32] following [17] for the CSI and HSS evaluation.

Shanghai_Radar [5] is a dataset contains continuous radar echo frames generated by volume scans in intervals of approximately 6 minute from October 2015 to July 2018 in Pudong, Shanghai. Every radar echo map covers 501 km×501 km area. We follow [5] to preprocess the echo sequence and also apply Algorithm 1 to filter 25-frame weather event datasets. The data range of frames in Shanghai Radar is set to [0-70] and the thresholds are set to [20, 30, 35, 40] following [17] for the CSI and HSS computation.

CIKM is a radar dataset from CIKM AnalytiCup 2017 Competition, recording precipitation samples in 101 km×101km area of Guangdong, China. Each sample settles 15 historical radar echo maps as a sample in which the time interval between two consecutive maps is 6-minute. We follow [21] to process the dataset to pad each echo map into 128×128 and follow the original setting to split training, validation and test sets. We transform the pixel in each frame to the reflectivity of [0,76] dBZ and use the thresholds [20,30,35,40] to compute the CSI and HSS.

The lengths of event sequences in each dataset are set to 25 frames except for the CIKM dataset with 15 frames. Compared to most of the existing studies, which aim to make an hour prediction (*e.g.*, 10 frames with a 6-minute interval), our tasks (except for CIKM dataset) are for the forecast in two hours (*i.e.* 20 frames) in this paper, which are more challenging. Although some recent studies attempt to achieve two hours prediction by frame interpolation (*e.g.*, predicting 10 frames with a 12-minute interval), this trick simplifies the complexity of precipitation dynamics and results in a degrading temporal resolution for prediction.

Algorithm 1 Weather Event Filtering

Given continuous frames s, pixel threshold T_{pixel}
$i \leftarrow 10$
$L_{in}, L_{out} \leftarrow 5, 20$
$event\_set \leftarrow \{\}$
while $i + L_{out} < \text{Len}(s)$ **do**
  if $\text{Mean}(s[i]) > T_{pixel}$ **then**
    $event \leftarrow s[i - L_{in} : i + L_{out}]$
    $event\_pixel \leftarrow \sum_{frame \in event} \text{Mean}(frame)$
    if $event\_pixel >= (L_{in} + L_{out}) T_{pixel}/2$ **then**
      Add event to $event\_set$.
      $i \leftarrow i + L_{out}$
      Continue
    end if
  end if
  $i \leftarrow i + 1$
end while
Return $event\_set$

8. DiffCast: Implementation Details

In this section, we will give a detailed description of the implementation for DiffCast's main architecture and its training and inference process, as well as our experimental settings.

Table 5. Detailed implementation of our Temp-Attn Block and GlobalNet.

Temp-Attn Block		
ResBlock×2	2×[Conv3x3 + GroupNorm8+ SiLU] + Conv3x3	Res Operator
Temporal Attention	Conv5x5 (Spatial) + Conv1x1 (Temporal)+FC	Attention Operator
Down/Upsampler	Conv1x1	
GlobalNet		
ResBlock×4	2×[Conv3x3 + GroupNorm8+ SiLU] + Conv3x3	Res Operator
ConvGRU×4	Conv3x3 + Conv3x3; HiddenState	GRU Operator
Downsampler×4	Conv1x1	

Architecture of DiffCast. We have described the main architecture of the DiffCast model in section 4.3. Here, we present our detailed implementation of the Temp-Attn Block and GlobalNet, which are mainly composed of temporal attention operator[3, 30] and ConvGRU operator[27, 37], respectively, as summarized in Table 5.

Algorithm 2 Training of The Framework

while not converged **do**
  Sampling a sequence $(x, y) \sim \mathcal{D}$, where $len(x) = L_{in}, len(y) = L_{out}$
  Making basic prediction $\mu = \mathcal{P}_{\theta_1}(x)$, where $len(\mu) = L_{out}$
  Building residual sequence r following Eq. (7)
  Grouping segments s_j from r following Eq. (17)
  Extracting global hidden state h following Eq. (14)
  Sampling diffusion step $t \sim \mathcal{U}(0, ..., T)$
  $\mathcal{L}_\epsilon \leftarrow 0$
  while $j < \lceil \frac{L_{\text{out}}}{K} \rceil$ **do**
    $\epsilon \sim \mathcal{N}(0, I)$
    Disturbing s_j to s_j^t following Eq. (1)
    Getting denoising loss $\mathcal{L}_\epsilon^j$ following Eq. (19)
    $\mathcal{L}_\epsilon = \mathcal{L}_\epsilon + \mathcal{L}_\epsilon^j$
  end while
  Computing deterministic loss $\mathcal{L}_\mathcal{P}$ following Eq. (6)
  Computing final loss $\mathcal{L}$ following Eq. (12) given α
  $(\theta_1, \theta_2, \theta_3) \leftarrow (\theta_1, \theta_2, \theta_3) - \nabla_{(\theta_1, \theta_2, \theta_3)} \mathcal{L}$
end while

Training and Inference. The DiffCast is trained with an end-to-end manner as shown in Figure 2 (b), where the base deterministic predictor and residual diffusion model are optimized within the same training iteration. The complete training procedure is summarized in Algorithm 2. In the inference phase, the framework also utilizes the base predictor to estimate the global trend and then apply the diffusion model to generate the residual segments autogressively. The final prediction is obtained by combining the two components. The inference procedure is summarized in Algorithm 3.

Experimental details All experiments are conducted on a computer with NVIDIA A6000 GPU (48G memory) and all models, including DiffCast equipped with various backbones and single backbones, can fit in a single GPU. As for the implementation of various backbones, we easily rebuild the most backbones from OpenSTL [31] library to adapt with DiffCast. We construct the GTUNet with a hierarchical UNet architecture with temporal attention blocks. This structure is composed of four up/down layers, with a hidden size of 64, and is subsequently upscaled/downscaled by factors of 1,2,4,8, respectively. Despite the increased number of parameters (*e.g.*, from 20.13MB to 66.40 MB for DiffCast_MAU) and higher training costs (*e.g.*, from 18 hours at 12 batch size to 23 hours at 6 batch size for 30K iterations) associated with the DiffCast framework, we can attain a great paramount for modeling the distribution of stochastic temporal evolution. Furthermore, the DiffCast framework can expedite the forecast by utilizing optimization techniques such as DDIM, DPM-solver *etc*. This capability can fully meet the requirements of short-term precipitation forecasting task in real-world scenarios (*e.g.*, 17 seconds to produce 20-frame forecasts), enabling real-time predictions that are both more accurate and realistic.

Algorithm 3 Inference of The Framework

Given initial frames x
Making basic prediction $\mu = \mathcal{P}_{\theta_1}(x)$
Extracting global hidden state h following Eq. (14)
$j \leftarrow 0, \hat{s}_{j-1} \leftarrow 0$
while $j < \lceil \frac{L_{\text{out}}}{K} \rceil$ **do**
  $s_j^T \sim \mathcal{N}(0, 1)$
  while Reverse diffusion from $t = T$ to $t = 1$ **do**
    $\epsilon \sim \mathcal{N}(0, I)$
    Estimating target noise ϵ_{θ_2} following Eq. (16)
    Recovering s_j^{t-1} from s_j^t following Eq. (5)
  end while
  Getting current residual segment $\hat{s}_j$
end while
Computing target frames $\hat{y}$ following Eq. (13)

9. Additional Analysis

In this section, we give an extra analysis on the design of framework loss, the computational complexity and the hyperparameter K.

w/o stochastic Loss. We decompose the determinism and local stochastics in precipitation evolution and model them with a deterministic component and a residual diffusion component, respectively. Different from other two-stage frameworks, we train the overall framework in an end-to-end manner to simulate the interplay between the determinism and uncertainty, which indicate that the gradient from stochastic diffusion denoising loss can also optimize the deterministic backbone. In Table 6, we compare the performance between the pure deterministic backbones and the intermediate output μ from deterministic component in

Table 6. Analysis of performance for backbones with different optimization strategies on SEVIR.

Deterministic Method	CSI	CSI-pool4	CSI-pool16	HSS	LPIPS	SSIM
SimVP	0.2662	0.2844	0.3452	0.3369	0.3914	0.6570
DiffCast_Simvp-μ	**0.2690**	**0.2828**	**0.3134**	**0.3320**	**0.3961**	**0.6728**
Earthformer	**0.2513**	**0.2617**	**0.2910**	**0.3073**	**0.4140**	**0.6773**
DiffCast_Earthformer-μ	0.2490	0.2579	0.2834	0.3040	0.4391	0.6685
MAU	0.2463	0.2566	0.2861	0.3004	**0.3933**	0.6361
DiffCast_MAU-μ	**0.2542**	**0.2848**	**0.3157**	**0.3361**	0.3929	**0.7028**
ConvGRU	0.2560	0.2685	0.3005	0.3124	0.3785	**0.6764**
DiffCast_ConvGRU-μ	**0.2635**	**0.2873**	**0.3197**	**0.3350**	**0.3860**	**0.6818**
PhyDNet	0.2560	0.2685	0.3005	0.3124	0.3785	0.6764
DiffCast_PhyDNet-μ	**0.2659**	**0.2785**	**0.3105**	**0.3252**	**0.3748**	**0.6811**

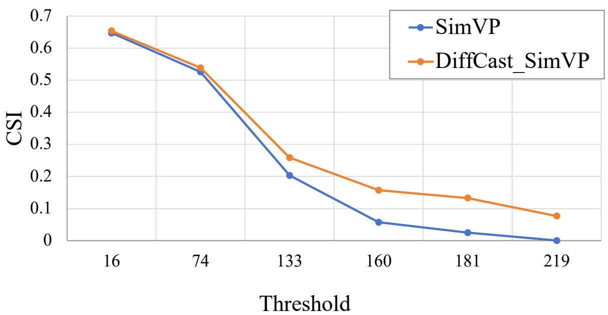

Figure 8. CSI with threshold

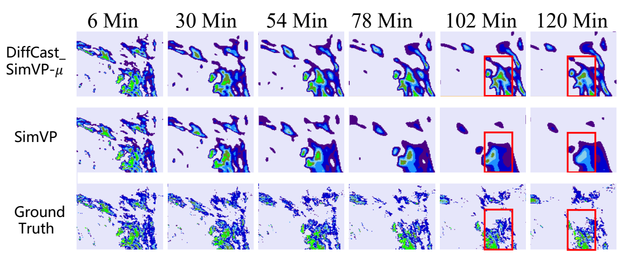

Figure 9. Qualitative results of SimVP w/o stochastic loss.

DiffCast. The results show that the stochastic loss indeed leads to a positive optimization on most of the deterministic backbones.

We point out that the conventional deterministic methods always under-estimate the high-value echoes with the increasing lead time (shown in Figure 1). To further investigate this, we show in Figure 8 the performance of DiffCast_SimVP, in terms of different thresholds. We observe that with the stochastic modeling the high-value echoes indeed can be more accurately maintained and predicted. Additionally, we show the qualitative results of SimVP w/o stochastic diffusion loss in Figure 9. The results indicate that DiffCast_SimVP-μ alleviates the echo value fading away issue compared to SimVP, which implies that stochastic objectives indeed help optimize the deterministic model.

Complexity and Hyperparameter K. In Table 7, we report the tradeoff between model size, memory and time cost conditioned on different segment length based on our

Table 7. Complexity analysis and hyperparameter K.

			Training		Inference	
	Model Size	CSI	Memory	Time Cost	Memory	Time Cost
MAU	20.13M	0.2463	21759MB	14.5h	2297MB	0.29s
DiffCast_MAU(K=2)	66.38M	0.2638	43815MB	22.8 h	3881MB	42s
DiffCast_MAU(K=4)	66.39M	0.2697	35471MB	20.5 h	3731MB	21s
DiffCast_MAU(K=5)	66.40M	**0.2716**	33791MB	19.5 h	3815MB	16s
DiffCast_MAU(K=10)	66.43M	0.2548	30443MB	18.5h	4065MB	8s

experimental setting with batchsize=4 for 30K training iterations. K is selected from $\{2, 4, 5, 10\}$ on validation set and K=5 delivers the best tradeoff. There are more requirements for memory compared with base predictor but it is acceptable in our practical application. It is notable that the model size is not influenced by K.

10. More Qualitative Results

In this section, we show more illustrative examples on different datasets to compare our DiffCast with baseline methods. As shown in Figure 10, 12, 11, 13, all deterministic backbones deliver blurry results after 60 minutes, with a phenomenon of high-value echoes and details fading away. However, when equipped into our DiffCast framework, all the prediction results of the backbones are consistently enhanced, where the forecast images are not blurry anymore, and the high-value echoes and details are carefully preserved. All the observations validate the effectiveness of the proposed DiffCast framework.

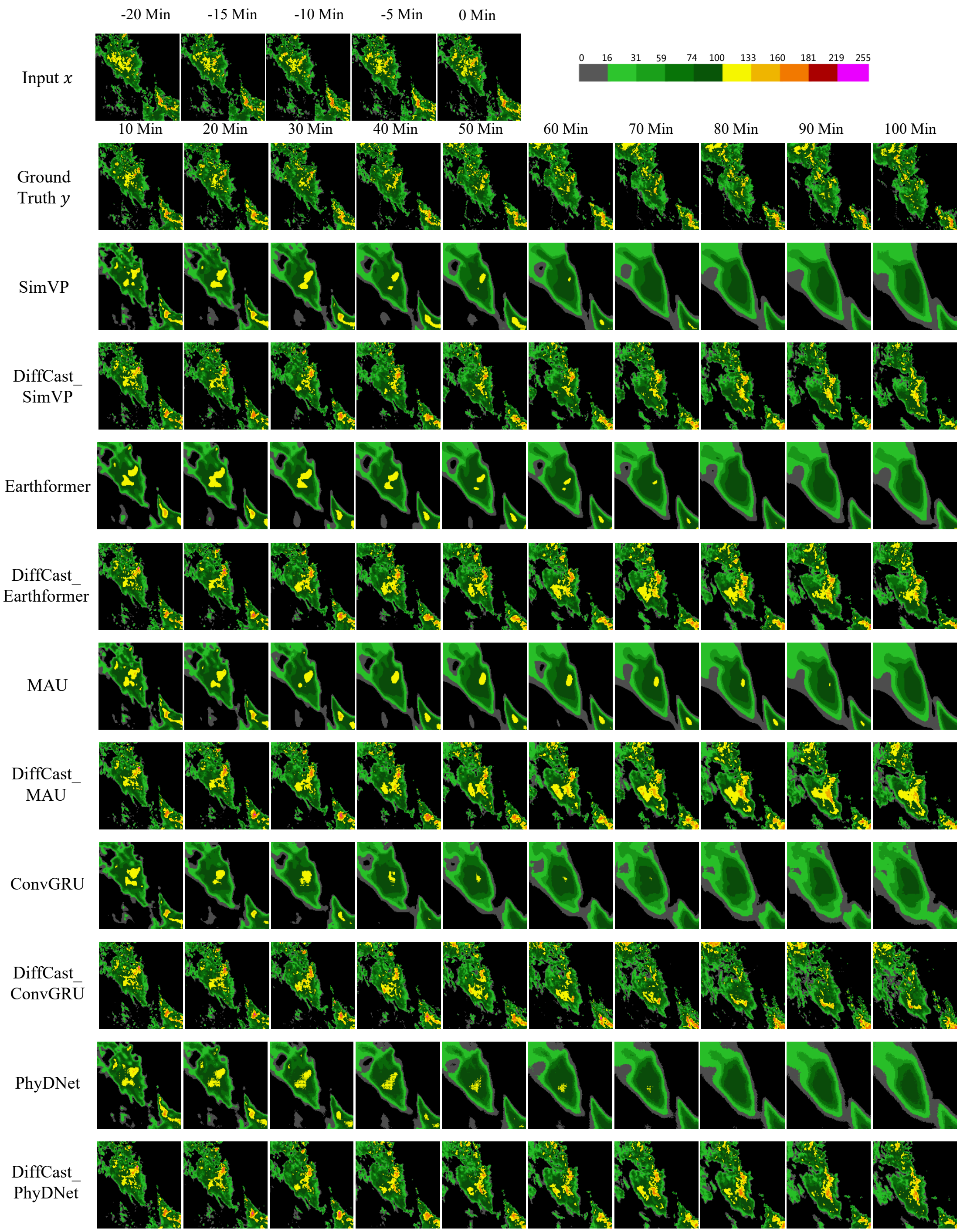

Figure 10. Prediction examples on the SEVIR.

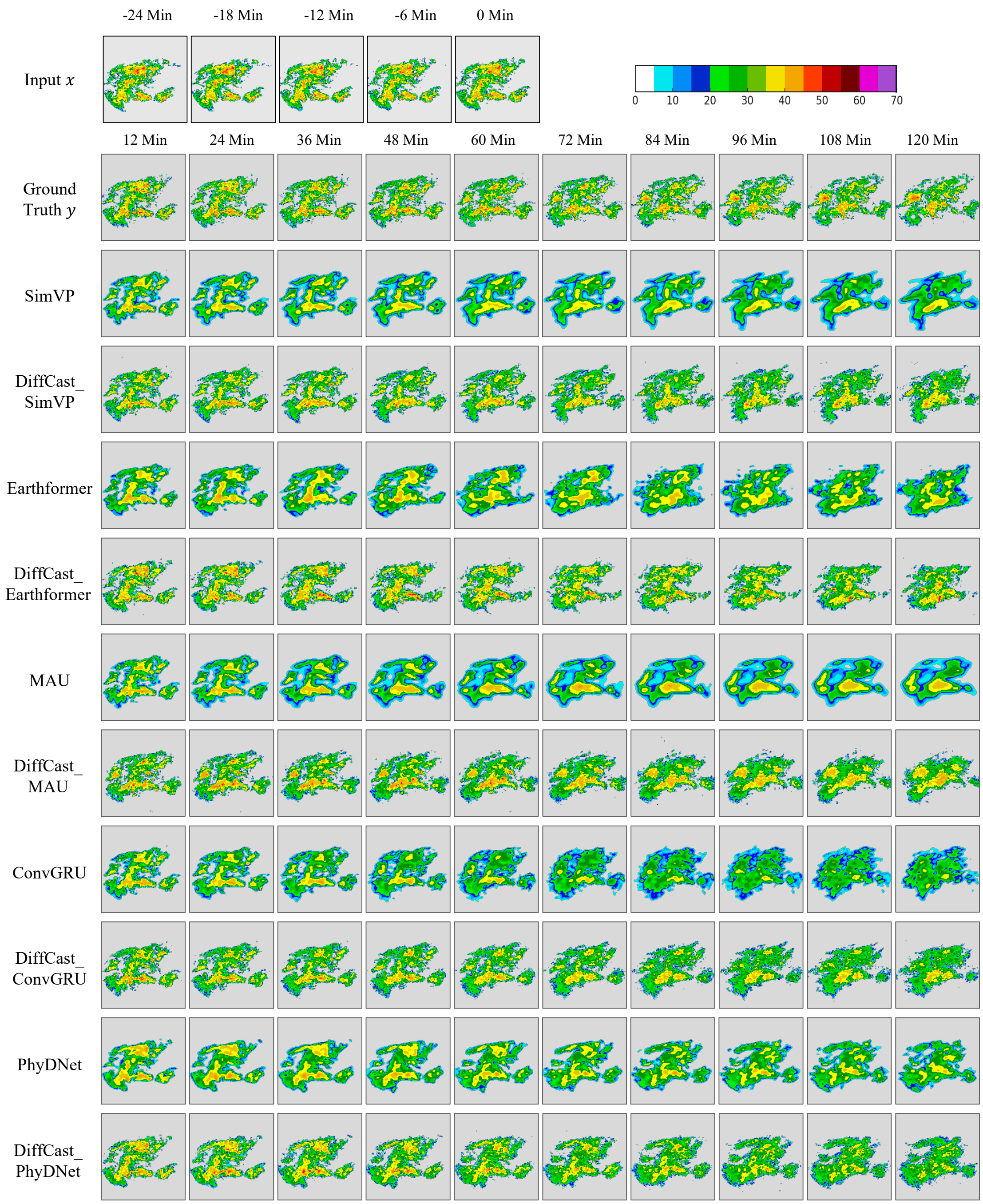

Figure 11. Prediction examples on the Shanghai Radar.

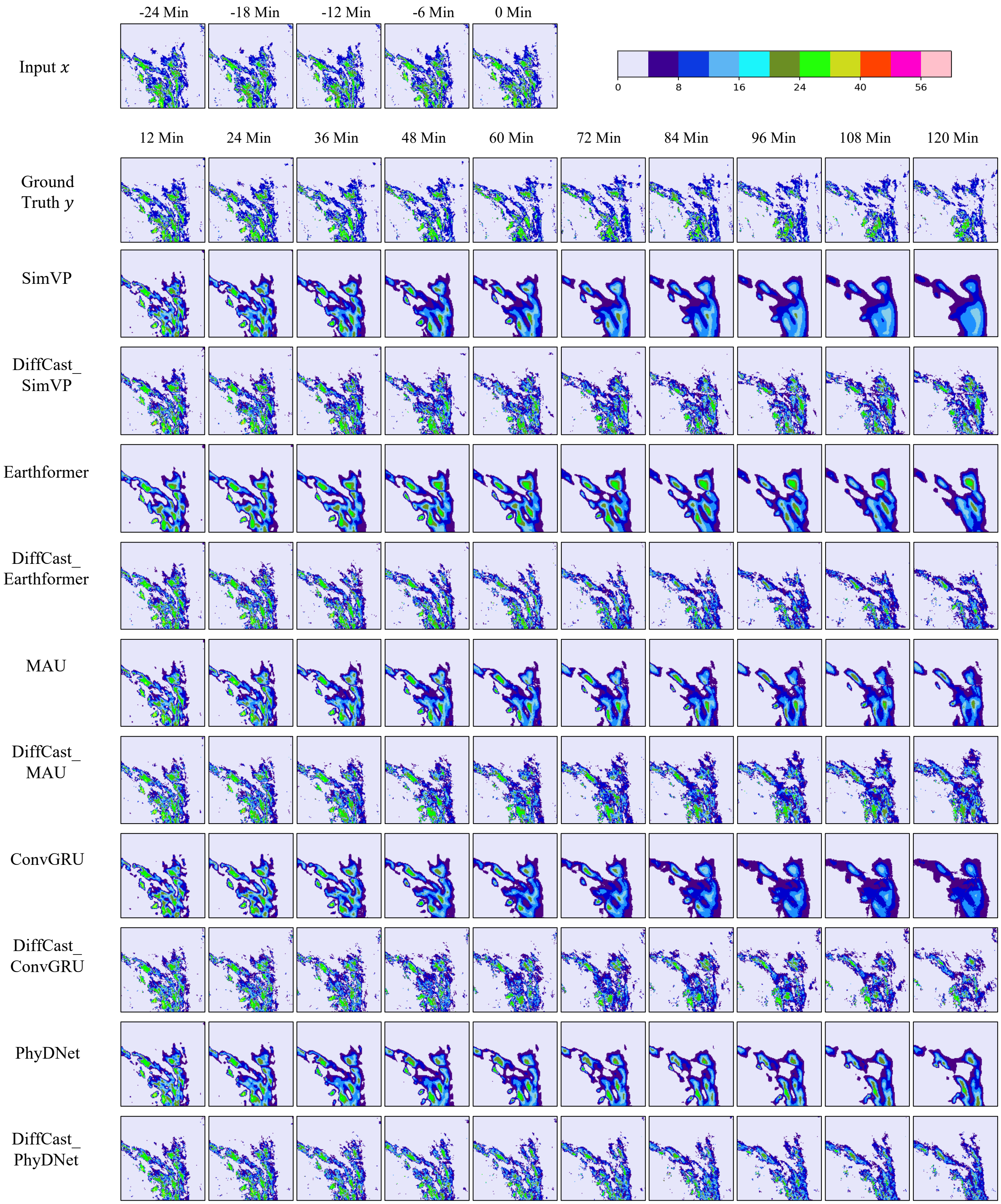

Figure 12. Prediction examples on the MeteoNet.

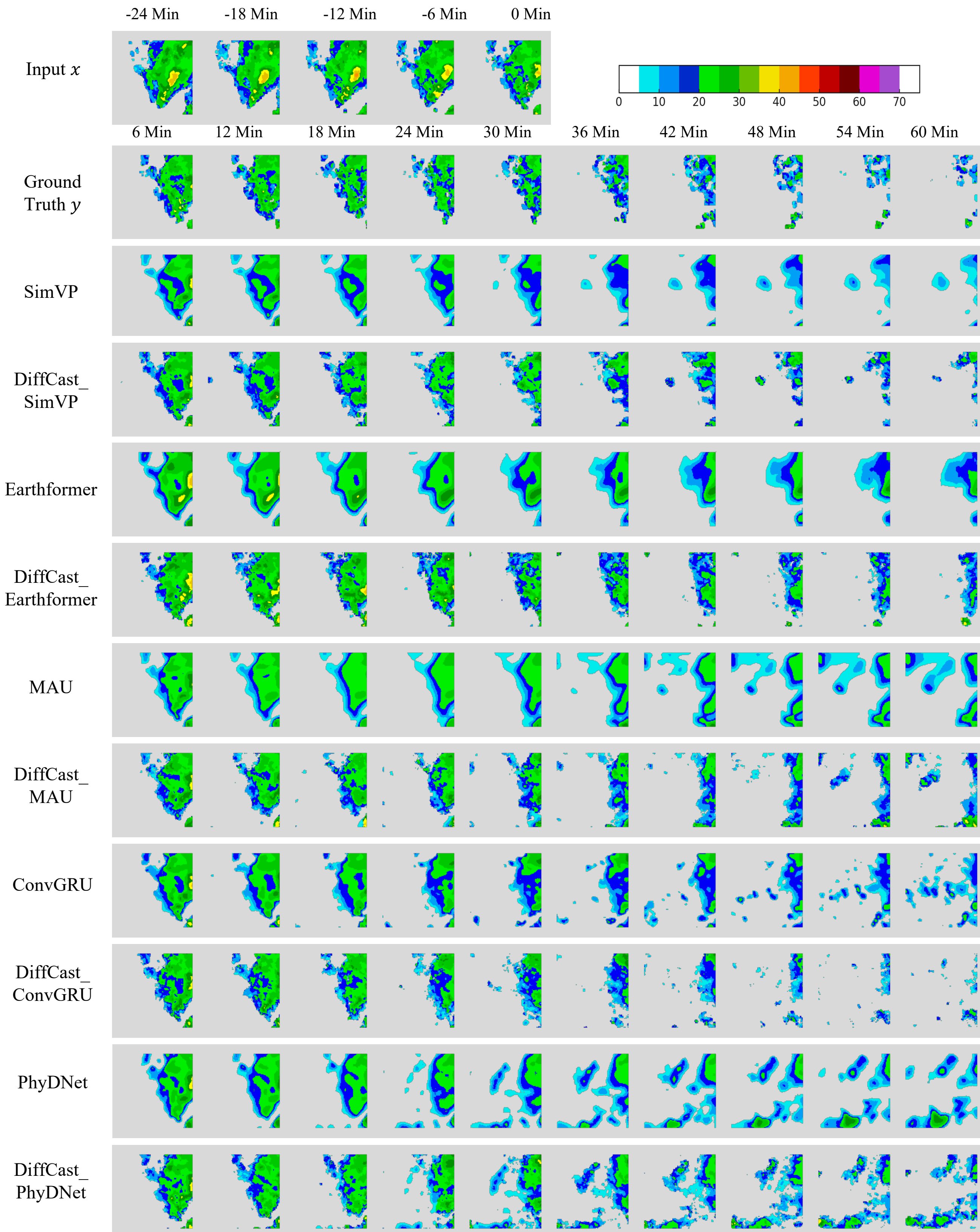

Figure 13. Prediction examples on the CIKM dataset.