LATENT DIFFUSION MODELS FOR GENERATIVE PRECIPITATION NOWCASTING WITH ACCURATE UNCERTAINTY QUANTIFICATION

Jussi Leinonen, Ulrich Hamann, Daniele Nerini, Urs Germann
Federal Office of Meteorology and Climatology MeteoSwiss
Locarno-Monti, Switzerland
{jussi.leinonen,ulrich.hamann,daniele.nerini,urs.germann}@meteoswiss.ch

Gabriele Franch
Fondazione Bruno Kessler
Trento, Italy
franch@fbk.eu

April 26, 2023

ABSTRACT

Diffusion models have been widely adopted in image generation, producing higher-quality and more diverse samples than generative adversarial networks (GANs). We introduce a latent diffusion model (LDM) for precipitation nowcasting — short-term forecasting based on the latest observational data. The LDM is more stable and requires less computation to train than GANs, albeit with more computationally expensive generation. We benchmark it against the GAN-based Deep Generative Models of Rainfall (DGMR) and a statistical model, PySTEPS. The LDM produces more accurate precipitation predictions, while the comparisons are more mixed when predicting whether the precipitation exceeds predefined thresholds. The clearest advantage of the LDM is that it generates more diverse predictions than DGMR or PySTEPS. Rank distribution tests indicate that the distribution of samples from the LDM accurately reflects the uncertainty of the predictions. Thus, LDMs are promising for any applications where uncertainty quantification is important, such as weather and climate.

1 Introduction

Sudden onset of precipitation frequently endangers human lives and causes damage and disruption to infrastructure through flooding and landslides, and is often accompanied by other hazardous weather phenomena such as hail, lightning and windstorms. Precipitation is also a fundamental driver of agriculture and hydroelectric power generation. Consequently, short-term precipitation forecasts are important tools that can benefit infrastructure managers, emergency services and the general public if provided in a timely manner.

Numerical weather prediction (NWP) models can typically forecast the probability and general intensity of precipitation occurring in a wider area, but they struggle at short spatial and temporal scales [1] because of the long running time and the time needed to assimilate data, i.e. to incorporate observational data used as the initial conditions. This problem is particularly severe with convective precipitation, which is associated with the highest rainfall rates, and originates from cells with a spatial scale on the order of a few tens of kilometers, making the exact location of the precipitation difficult to predict with NWP [2]. Experience over decades has shown that at lead times of minutes to a few hours, statistical and data-driven models that make optimal use of the latest available observations are useful tools for the short term prediction, or *nowcasting*, of precipitation. Such models have been widely deployed by meteorological agencies.

A common way to implement precipitation nowcasting is *Lagrangian extrapolation*: using motion-detection algorithms to derive motion vectors from consecutive measurements of rainfall by weather radar, then advecting the precipitation field using these vectors to predict its future movement [3, 4]. The skill of Lagrangian extrapolation

decreases rapidly with lead time because of the growth and decay of precipitation, in particular in convective situations. Multiple approaches have been proposed to overcome this limitation, including seamless blending with NWP forecasts (e.g. [5, 6]) and incorporating information about orographic forcing [7, 8, 6]. Advanced nowcasting methods also augment the Lagrangian extrapolation framework with features that aim to preserve the structure of precipitation and generate a set of multiple predictions (called an *ensemble* nowcast), where different ensemble members represent possible scenarios of future rainfall and their diversity can be used to quantify the forecast uncertainty. Prominent among such methods is the Short-Term Ensemble Prediction System (STEPS) [5, 9], implemented in the PySTEPS open-source library [10].

Numerous studies have also used various architectures of deep neural networks (DNNs) for nowcasting (e.g. [11, 12, 13, 14]), typically training the network to optimize a metric such as mean squared error (MSE) of the predicted precipitation. DNN-based nowcasting can learn to predict growth and decay, but suffers from blurring of the predictions, where the predicted fields become weaker and more widespread with increasing lead time. This reflects the increasing uncertainty of the prediction resulting from the low predictability of weather. Although such blurred predictions represent the mean expected rainfall, they are not realistic future scenarios. This hinders uncertainty quantification, which is an important aspect of a reliable forecast for downstream applications such as hydrological simulations.

Deep-learning models have also been used to generate more realistic precipitation fields than allowed by simple loss functions, predicting the conditional distribution of the future state of the weather instead of its conditional mean only. This has most often been achieved with Generative Adversarial Networks (GANs; [15]), which consist of two simultaneously-trained neural networks: a discriminator that is trained to distinguish real samples that belong to the training dataset from generated samples, and a generator that is trained to produce samples that "fool" the discriminator, thus learning to produce samples that resemble those in the training set. GANs have been used to create precipitation fields in applications such as postprocessing and downscaling [16, 17, 18], precipitation estimation from remote sensing measurements [19, 20] and disaggregation [21]. The state of the art in generative nowcasting is, to our knowledge, presently Deep Generative Models of Rainfall (DGMR) [22], which uses a conditional GAN with a regularization term to incentivize the model to produce forecasts close to the true precipitation. DGMR is able to create realistic rainfall predictions that are also numerically accurate, and it can create multiple predictions for each input, enabling ensemble nowcasting.

While GANs are conceptually quite simple, their adversarial training tends to make training them costly and difficult [23]. The shifting objectives often cause the convergence to be unstable or slow, and it is necessary to expend training resources to train the discriminator, which is not needed after training in most GAN applications. GANs can also be prone to *mode collapse* [24], where a generator learns to output just one or a few different examples. In conditional GANs this can manifest as the generator ignoring its noise input, always generating identical outputs for a given input.

Denoising diffusion models (DMs), also called score-based generative models, have recently emerged as an alternative to GANs in generative modeling [25, 26]. Their mathematical formulation is based on a forward process that gradually degrades an N-dimensional sample with increasing amounts of added noise until the sample is indistinguishable from random noise. The neural network is trained to perform one step in an iterative denoising process that reverses the forward process. When the denoising is performed starting from a sample containing only random noise, the reverse process converges to a sample in the training data distribution. DMs have been shown to outperform GANs in terms of sample quality and diversity [27], and can be conditioned to specific inputs similarly to GANs. In image processing tasks, they have excelled at tasks such as text-to-image generation, inpainting, uncropping and superresolution [28, 29, 30, 31]. DMs are trained to optimize a relatively simple loss function, avoiding the complications of adversarial training and thus making them easier and less computationally expensive to train than GANs. They are also not susceptible to mode collapse. A downside of DMs compared to GANs is the higher cost of generation: since the reverse diffusion process is iterative, the model has to be evaluated several times. Early DMs such as Denoising Diffusion Probabilistic Models (DDPM [32]) could require thousands of iterations; this was brought down by alternate process models such as the Denoising Diffusion Implicit Models (DDIM [33]) to the order of 100 iterations. Recently, samplers based on pseudo-linear multistep (PLMS [34]) differential equation solvers have decreased the number of required iterations further, producing good samples with 30–50 iterations and acceptable ones with as few as 10.

The ability of DMs to generate diverse samples suggests that they are potentially useful in applications where modeling the uncertainty of predictions is important, such as weather, climate and hydrology. The ability of DMs to generate precipitation fields was recently demonstrated [35]. In this work, we introduce the use of DMs for ensemble precipitation nowcasting. To reduce the computational cost, we utilize the latent diffusion model (LDM) concept used by Stable Diffusion [36], where the diffusion process is run in a latent variable space mapped to the physical pixel space by an autoencoder. There are three main components of the model, which we call LDCast:

1. **Forecaster stack**: To condition the model, we introduce a novel spatiotemporal prediction architecture based on Adaptive Fourier Neural Operators (AFNOs) [37, 38], with temporal cross attention to map between the input and output time coordinates.
2. **Denoiser stack**: We adapt the network used by [36], using 3D convolutions to model spatiotemporal differences, and an AFNO-based module used in place of cross attention to couple the network to the conditioning.
3. **Variational autoencoder** (VAE): We use simple 3D convolutional neural networks (CNNs) as the encoder and the decoder in a VAE with a continuous latent space to reduce the number of data points by a factor of 64.

To produce samples of forecast future precipitation, the past precipitation field is first encoded with the encoder part of the VAE. Then, the forecaster is used to produce a prediction of the future precipitation; this prediction is used to condition the denoiser, which is run in a loop with the PLMS sampler [34] to produce samples in 50 iterations. Finally, the predicted latent rainfall field is decoded with the VAE decoder. Further details are given in Sect. 4.2.

We observe that LDCast creates predictions of the future evolution of precipitation that are visually realistic and highly consistent with the inputs. We compare the outputs to DGMR and PySTEPS benchmarks using two datasets, described in 4.1: the test set from the Swiss radar-based precipitation dataset on which the model was trained, and a German dataset that was used for evaluation only, providing a test where both LDCast and DGMR are outside the regions of their respective training datasets. With quantitative ensemble forecast accuracy metrics, LDCast outperforms PySTEPS and DGMR, although DGMR sometimes achieves better scores in forecasting whether the precipitation exceeds given thresholds. The clearest advantage of LDCast is in accurate uncertainty quantification. We show that DGMR produces overconfident predictions, i.e. the ensemble members are too close to each other both quantitatively and in terms of the amount of diversity of precipitation patterns produced, while LDCast achieves a realistic assessment of the uncertainty of the forecast.

2 Results

In Fig. 1, we show four examples of precipitation predicted with LDCast. In each case, we show the actual precipitation on the top and one LDCast prediction on the bottom. We display the first ensemble member for each prediction, although any member would be equally valid. The first two cases are from the test set of the Swiss dataset while the last two are from the German dataset.

The first case contains intense convective rainfall. The LDCast prediction contains precipitation cells with a correct intensity and degree of organization, producing line-like structures and clusters similar to those found in the observed precipitation. Not every detail is correctly predicted; however, this cannot be expected at long lead times from a single ensemble member.

The second case shows an organized convective system at the top and more isolated cells on the bottom left. LDCast again reproduces the correct spatial patterns; the precipitation intensity of the cells on the bottom appears to be roughly correctly predicted while the intensity at the top is somewhat underestimated especially in the 40 min and 60 min frames. Interestingly, LDCast correctly predicts the separation of the convective cores on the top, although it forecasts a more complete separation than actually occurs.

The third case shows larger-scale rainfall with embedded convection at more moderate rain rate compared to the first two cases. LDCast again reproduces the degree of spatial organization well and correctly detects the relatively fast motion of the rainfall field from the bottom left towards the top right.

In the fourth case, linear precipitation structures move rapidly towards the top and somewhat towards the right of the images. LDCast correctly predicts the motion and maintains the linear shape until 60 min, after which the predicted rainfall loses cohesion faster than that observed. There is a high variability among the other ensemble members, indicating a low predictability in this case; none of the ensemble members preserve the linear shape quite as strongly as the observation.

In all four cases, it can be seen that the prediction is initially close to the observation and then diverges gradually. This demonstrates that the forecaster stack effectively conditions the prediction to the observed past rainfall.

2.1 Prediction accuracy

We used the continuous ranked probability score (CRPS; Sect. 4.3.1) as the quantitative metric for assessing the accuracy of the precipitation rate predictions. CRPS takes into account the distribution of the 32 ensemble members, making it suitable for ensemble forecast verification. CRPS is evaluated pixelwise and thus does not reflect the

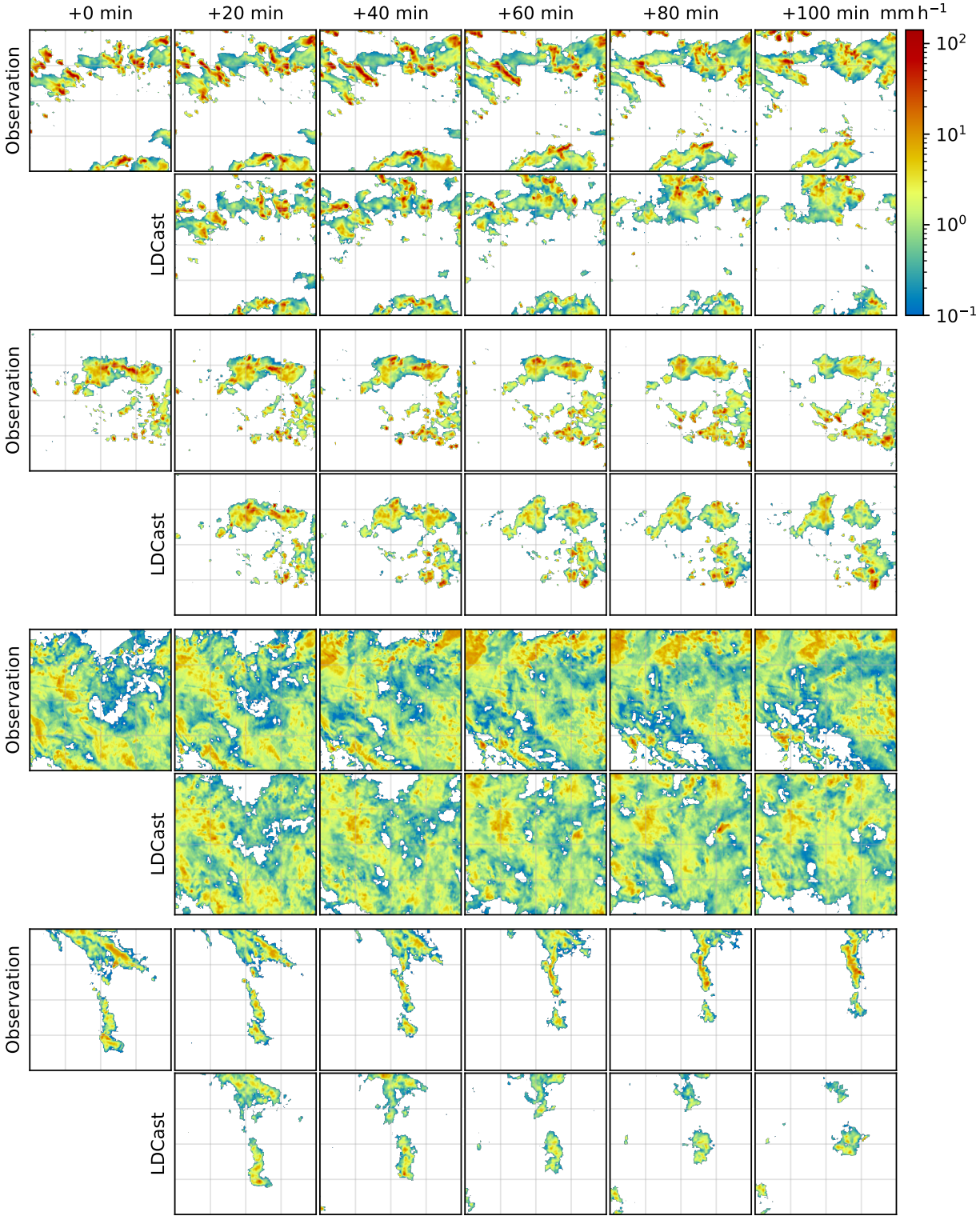

Figure 1: Sample cases of $256\ \mathrm{km} \times 256\ \mathrm{km}$ size comparing the precipitation rate observation and the prediction of the LDCast model. Time steps are produced by the model at $5\ \mathrm{min}$ resolution but they are visualized at $20\ \mathrm{min}$ intervals due to space constraints. The first ensemble member is shown in each case.

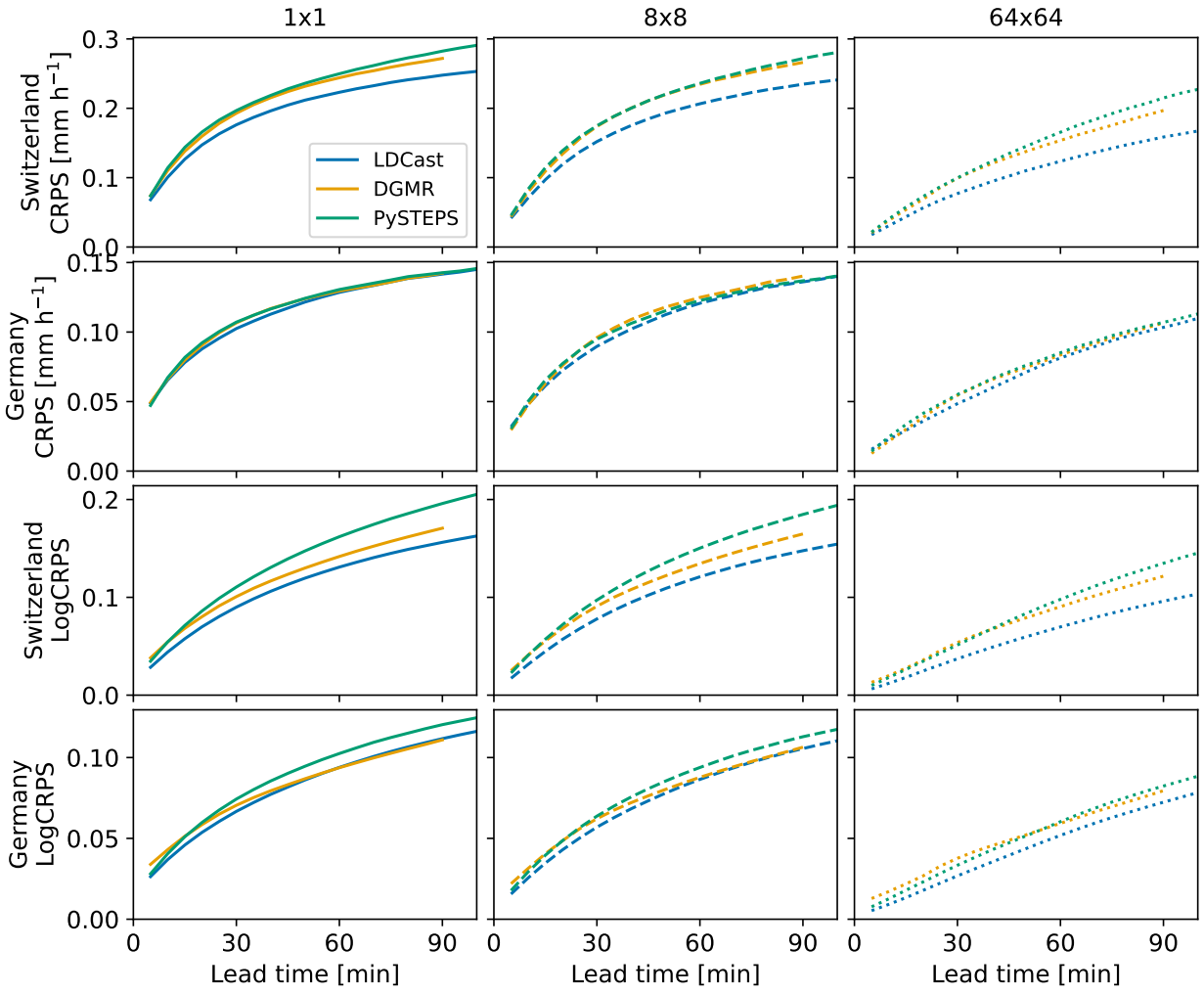

Figure 2: CRPS (lower is better) for the LDCast model as a function of the forecast lead time, compared to the DGMR and PySTEPS benchmarks. The two top rows show CRPS for the absolute precipitation R while the bottom rows show LogCRPS, i.e. CRPS for $\log_{10}(R)$. The three columns correspond to different amounts of averaging: no averaging (1 km scale) for the first column, 8 km $\times$ 8 km averaging for the second and 64 km $\times$ 64 km for the third.

accuracy of the spatial patterns in the prediction. To assess whether precipitation is correctly predicted over different spatial scales, we also calculated the CRPS for precipitation averaged over $8\,\mathrm{km} \times 8\,\mathrm{km}$ and $64\,\mathrm{km} \times 64\,\mathrm{km}$ windows. Furthermore, to give a metric of the relative error in addition to the absolute error, we computed the CRPS for the logarithm of the rainfall (LogCRPS) using a fill value of $0.02\ \mathrm{mm\,h^{-1}}$ for regions of zero rainfall (as used when training LDCast).

The results of the CRPS calculation as a function of lead time are shown in Fig. 2. The CRPS from LDCast is compared to DGMR and PySTEPS ensemble predictions. With the Swiss dataset, LDCast clearly outperforms DGMR and PySTEPS at all scales in both CRPS and LogCRPS. The advantage of LDCast over the other models increases with longer lead times. With the German dataset, all three models are quite close to each other in CRPS, with LDCast achieving slightly better overall scores. There are somewhat larger differences between the models in LogCRPS of the German dataset, with LDCast the best model in most situations.

2.2 Representation of uncertainty

In Fig. 3, we show examples of the first five ensemble members of LDCast and DGMR at 90 min lead time. This is the maximum lead time of DGMR, and thus the prediction where the largest variability between ensemble members is

expected. As with Fig. 1, the first two examples are from the Swiss test dataset while the last two are from the German dataset. Visual comparison of the LDCast and DGMR outputs shows that the DGMR ensemble members are rather similar to each other, while the variability of the LDCast outputs is much greater. Notably, the mutual similarity of the DGMR outputs appears greater than their similarity to the observation, suggesting that DGMR produces overconfident predictions.

We can quantitatively examine the correctness of the uncertainty estimates using rank distributions (Sect. 4.3.2). These are shown in Fig. 4 for multiple scales and compared to DGMR and PySTEPS. Similar to Fig. 2, we also show the results for rainfall averaged over $8\ \mathrm{km} \times 8\ \mathrm{km}$ and $64\ \mathrm{km} \times 64\ \mathrm{km}$ windows. The LDCast results are closest to the ideal flat distributions. DGMR rank distributions are "U-shaped", that is, they contain many high and low ranks, corresponding to overconfident predictions in agreement with the qualitative comparison above. PySTEPS rank distributions at the $1\ \mathrm{km} \times 1\ \mathrm{km}$ and $8\ \mathrm{km} \times 8\ \mathrm{km}$ scales contain too many high ranks (but not too many low ones), indicating that PySTEPS produces many cases where all ensemble members underestimate the precipitation. At the $64\ \mathrm{km} \times 64\ \mathrm{km}$ scale PySTEPS also produces a U-shaped distribution, while that of LDCast is still relatively flat. The Kullback–Leibler divergence (KL) from the uniform distribution shows that LDCast achieves scores clearly closest to the optimum. The rank distribution results are very similar between the Swiss and German datasets.

2.3 Forecasting event occurrence

We used the fractions skill score (FSS; Sect. 4.3.3) to measure the skill of the models at predicting whether the precipitation exceeds certain threshold values. We computed the FSS at scales of $2^N\ \mathrm{km}$ (with N an integer) up to $256\ \mathrm{km}$. The results are shown in Fig. 5 for thresholds of $0.1\ \mathrm{mm\,h^{-1}}$, $1\ \mathrm{mm\,h^{-1}}$ and $10\ \mathrm{mm\,h^{-1}}$, averaged over all lead times. With the Swiss test dataset, LDCast performs approximately equally to DGMR and better than PySTEPS at all scales for the $R \geq 0.1\ \mathrm{mm\,h^{-1}}$ and $R \geq 1\ \mathrm{mm\,h^{-1}}$ thresholds; for $R \geq 10\ \mathrm{mm\,h^{-1}}$, the results are similar except LDCast achieves better scores at the 32–128 km scales. With the German dataset, LDCast is slightly better than DGMR at the $0.1\ \mathrm{mm\,h^{-1}}$ threshold, while being slightly behind at $1\ \mathrm{mm\,h^{-1}}$ and considerably behind at $10\ \mathrm{mm\,h^{-1}}$. The generative models based on deep learning perform better than PySTEPS in all cases except $R \geq 10\ \mathrm{mm\,h^{-1}}$ at long scales for the Swiss dataset and $R \geq 10\ \mathrm{mm\,h^{-1}}$ at short scales for the German dataset.

3 Discussion

In this article, we have introduced the use of latent diffusion models for generative nowcasting of precipitation measured by weather radars. Our model, LDCast, generates ensembles of realistic precipitation fields, using 4 time steps ($20\ \mathrm{min}$) of precipitation as its input, and predicting precipitation up to 20 time steps ($100\ \mathrm{min}$) to the future. Quantitative comparisons to DGMR, a GAN-based precipitation nowcasting model, and to PySTEPS, a commonly used statistical nowcasting algorithm, reveal that LDCast outperforms them in accuracy (measured by CRPS). LDCast has a particularly distinct advantage over the benchmark models in characterizing the uncertainty of its predictions, generating diverse forecasts that result in rank distributions that are much closer to uniform. This diversity makes it easier for the model to reveal the possibility of less likely but higher impact events, such as extreme weather. The advantage of LDCast over the benchmark models increases when precipitation averaged over a larger scale is considered. Meanwhile, the results are more mixed with regard to the ability of the models to predict whether the precipitation exceeds predetermined thresholds, as measured by the FSS. A possible factor in this is that LDCast was trained on a logarithmic transformation of the precipitation rate, thus emphasizing the relative error, while DGMR was trained directly with the precipitation rate, which can be expected to emphasize the absolute error.

We evaluated the models using two different test datasets. One was from Switzerland and its surroundings, the same region where the model was trained. To assess how well the model generalizes to outside its training domain, we also performed the evaluation with rain rate data from northern Germany. The comparisons to the benchmark models indicate that LDCast loses some of its advantage over DGMR in CRPS and FSS when evaluated in the out-of-domain dataset. One reason for this may be that northern Germany and United Kingdom, from where the DGMR training data were obtained, are at similar latitudes and in proximity of the North Sea, and therefore have climates that resemble each other more closely than that of Switzerland, which experiences more convective precipitation, and where the evolution of precipitation patterns is expected to be different due to the orographic influence of the Alps. In contrast to FSS, LDCast retains its superiority in the rank histograms also with the German dataset. Thus, its ability to quantify its own uncertainty appears to be quite robust.

Another advantage of LDMs is the relative ease of training compared to GANs. On our system of eight Nvidia V100 GPUs, we initially trained our model for approximately $53\ \mathrm{h}$ with 128×128 pixel samples, then fine tuned it for approximately $5\ \mathrm{h}$ with 256×256 pixel samples. These computational costs, while significant, are considerably lower compared to GAN-based models. For comparison, we briefly experimented with implementing DGMR from

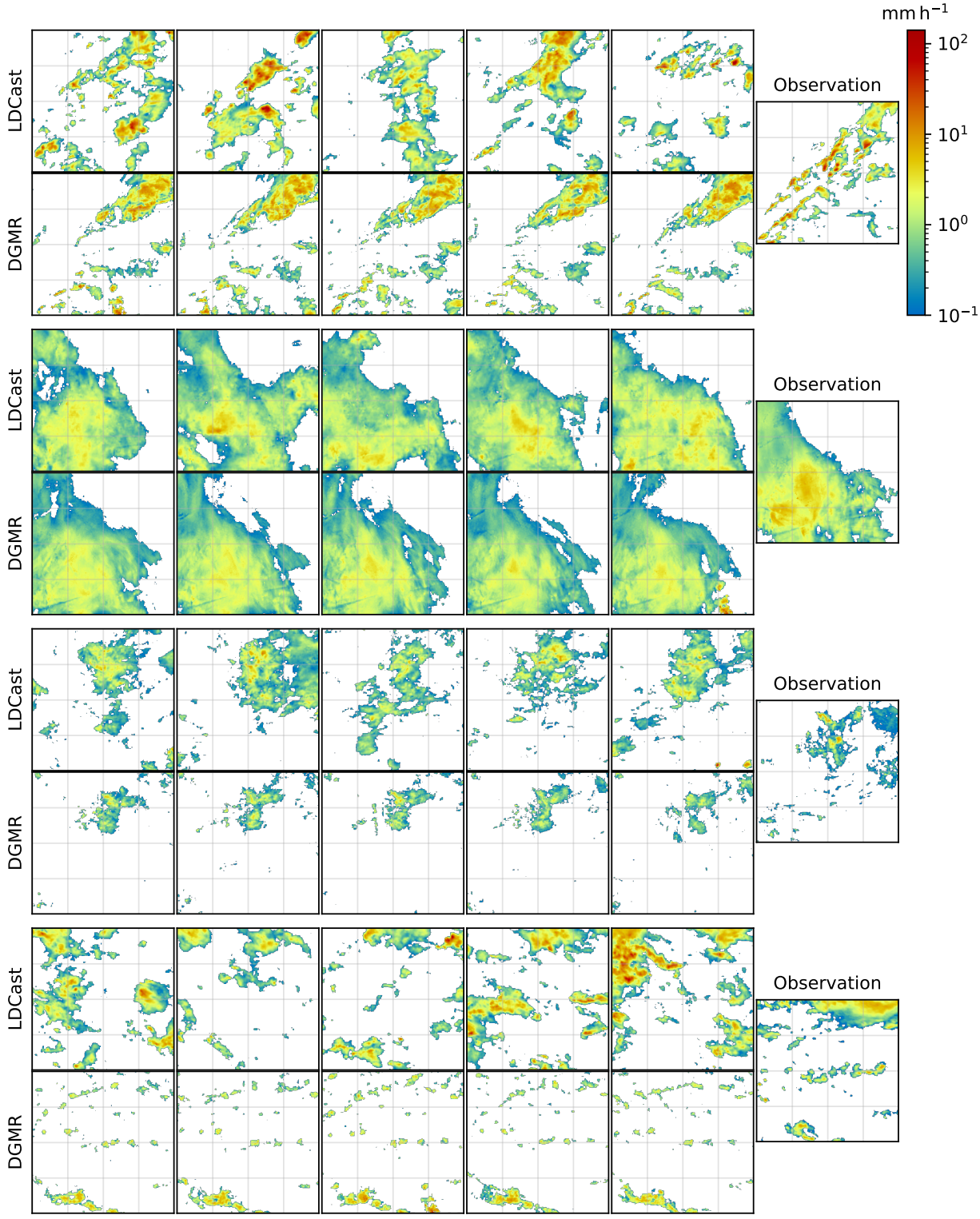

Figure 3: Ensemble members of predicted precipitation at 90 min lead time. In each of four cases, the results from LDCast are shown on the first row on the left and the results from DGMR on the second row. The actual observed precipitation is shown for comparison on the right.

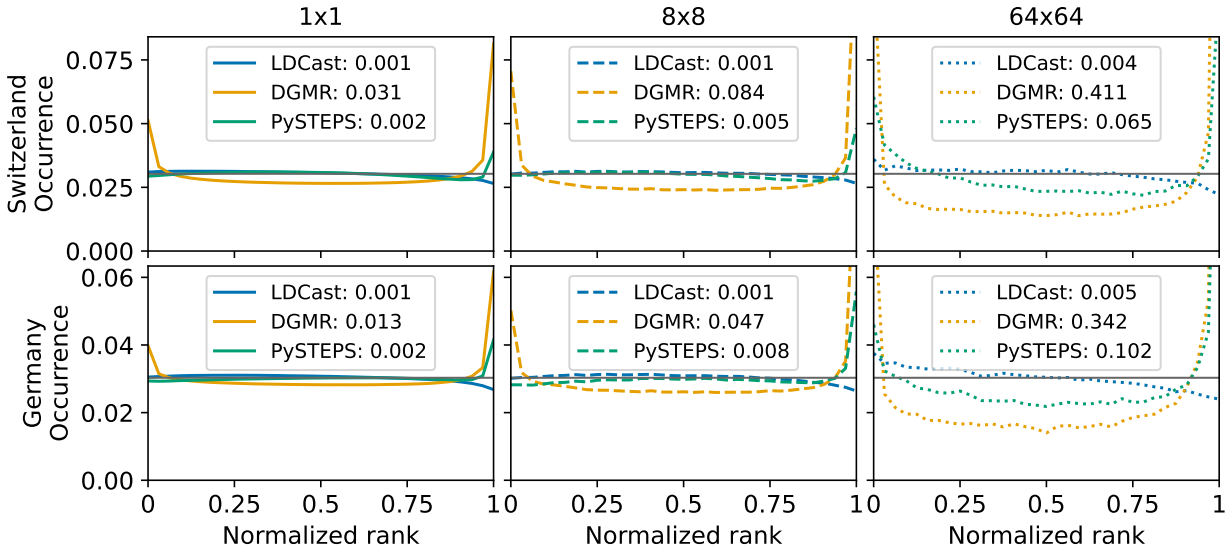

Figure 4: Rank distributions for the LDCast, DGMR and PySTEPS models. The columns correspond to different averaging scales as with Fig. 2. The numbers in the legend indicate the Kullback–Leibler divergence from the uniform distribution. The gray line in each plot indicates the ideal uniform distribution.

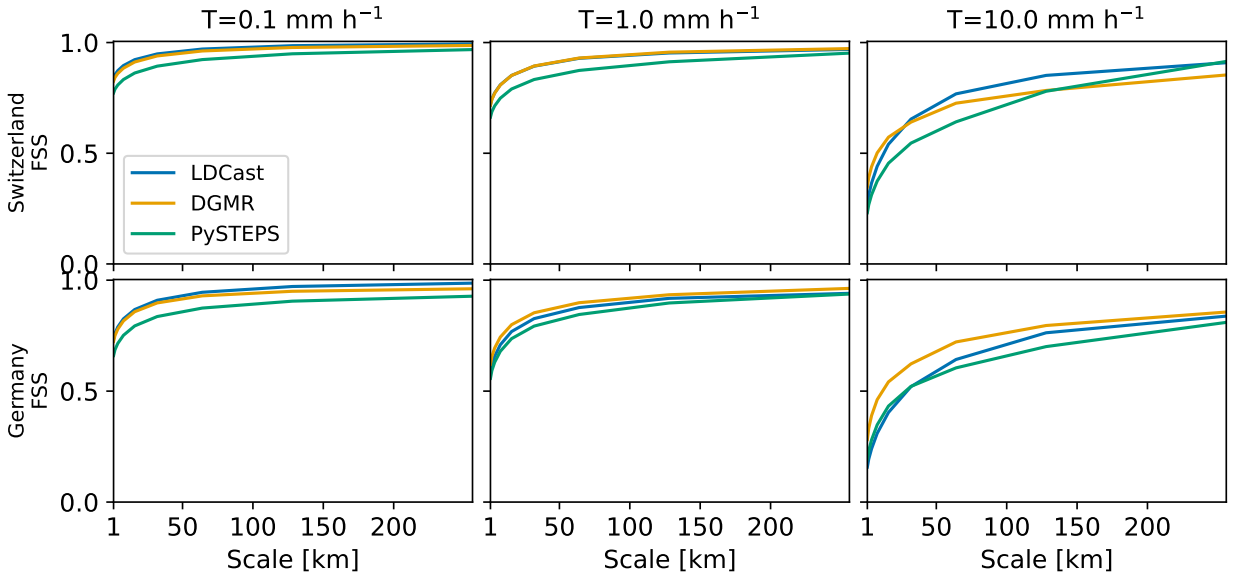

Figure 5: FSS as a function of scale for LDCast, DGMR and PySTEPS. The three columns show the FSS for thresholds of $0.1\ \mathrm{mm\,h^{-1}}$, $1.0\ \mathrm{mm\,h^{-1}}$ and $10.0\ \mathrm{mm\,h^{-1}}$, respectively.

the pseudocode available in [22] (the full source code for DGMR is not available; for the DGMR benchmark, we used the saved generator released by the developers). The training speed that we achieved indicated that training for the full 5×10^5 generator steps described in [22] would have required approximately 1100 h, i.e. 46 days, on the abovementioned hardware. As this was not critical to our investigation, we decided to forgo training the model to completion. Optimized implementations might improve the training times for both models; nevertheless, it seems clear that LDMs make generative modeling in weather and climate sciences more approachable to researchers with limited computational resources. Beyond training speed, our experience with training the model was also that the stability of training diffusion models makes the development process easier compared to GANs. Furthermore, compared to our initial attempts to generate samples in the pixel space, we found that the latent-space encoding in LDMs not only reduces computational costs but also improves training stability by regularizing the input and output variable space.

A downside of DMs is that the network needs to be evaluated several times during sample generation. Our sampling process required approximately 19 s to generate one $20 \times 256 \times 256$ ensemble member on one of the GPUs used for training. This can potentially be reduced in operational use by using fewer sampler iterations (possibly at the cost of lower sample quality and diversity), with lower precision floating point arithmetic, and with architectural and implementation optimizations to minimize redundant calculations between iterations. Ensemble members can also be generated in parallel with multiple GPUs. Nevertheless, the computational requirements make DMs less likely to be adopted in performance-critical applications, such as using neural networks to emulate computationally expensive components of weather and climate models. For such applications, latent-space autoencoders can also be used in combination with a GAN [39], which may provide performance benefits. Another limitation of the iterative nature of DMs is that as implicit models, it is not straightforward to include physics-based or statistical constraints in them. Further research is needed to determine how such constraints could be implemented in DMs. Nevertheless, LDCast performs well compared to DGMR, which does include statistical constraints on the generated precipitation, implying that such constraints are not necessarily needed in practice.

Precipitation nowcasting has for several years drawn considerable attention as an application of deep learning. However, nowcasting turned out to be a challenging application for deep generative models, and appeared relatively late with the recent introduction of models like DGMR. The success of LDMs at this task, combined with the computational advantages, suggests that they will find applications in nowcasting different atmospheric variables, as well as in other weather and climate applications in which accurate uncertainty quantification is important. We also expect that the LDCast methodology can be extended to exploit multiple predictor variables, potentially including satellite observations and forecasts from numerical weather prediction models similar to [40]. Our forecaster stack based on AFNO and temporal attention with positional encoding is naturally suited for this as it can flexibly handle inputs at different time coordinates.

4 Methods

4.1 Datasets

We trained the model on a dataset of precipitation rate estimates from the MeteoSwiss operational radar network [41, 42]. The network consists of five scanning C-band Doppler radars, whose overlapping ranges, optimized scanning strategy and processing algorithms (vertical profile, visibility and clutter correction) mitigate the issue of topographic blocking in the complex Swiss terrain. The radar composite is produced every 5 min at 1 km resolution in a rectangular area 710 km in the east–west direction and 640 km north–south, covering all of Switzerland and some surrounding regions. The data were gathered from the years 2018–2021, using the period from April to September for each year to focus the training more on the convective season, when the variability of rain rates is largest.

In order to test models outside the region in which they were trained, we also obtained precipitation rate data from the radar composite of the German Weather Service (DWD) [43] from April–September 2022. This network covers all of Germany, but the southern part partially overlaps with the Swiss radar network, so we only use the northern half for testing. The 5 min / 1 km temporal and spatial resolutions of the DWD and MeteoSwiss composites are identical to each other, and also to those of the UK MetOffice radar network, which was used to train DGMR. Thus, both LDCast and DGMR can be evaluated without retraining in both the Swiss and German domains.

We split the Swiss dataset to training, validation and testing sets such that each UTC day is assigned entirely to only one of the splits; this is done to reduce the temporal proximity, and hence correlation, of the training and validation/testing data. Approximately 10% of the data is assigned to the validation set and another approximately 10% to the testing set. The German dataset is used only for testing. The final evaluation is performed with 1024 samples from each testing dataset, with 32 ensemble members generated for each sample with each model.

When generating training, validation and testing samples, rather than sampling the datasets uniformly we sample them such that the model sees similar numbers of cases from different precipitation intensities R. This is achieved by oversampling cases containing higher R. We divide the dataset into 32×32 pixel tiles, and compute R_{m}, the 99th percentile of precipitation rate in each tile (representing a soft maximum less sensitive to outliers). Each tile is then assigned to one of 11 bins, where the first bin is for $R_{\mathrm{m}} < 0.2\ \mathrm{mm\,h^{-1}}$, the last bin for $R_{\mathrm{m}} \geq 50\ \mathrm{mm\,h^{-1}}$, and the rest are logarithmically spaced between 0.2–$50\ \mathrm{mm\,h^{-1}}$. Training samples are then generated such that each bin is sampled with equal probability.

For preprocessing we follow the strategy of [16]. Before feeding samples to the model, they are preprocessed with a logarithmic transformation

$$f(R) = \begin{cases} \log_{10} R & R \geq 0.1\ \mathrm{mm\,h^{-1}} \\ \log_{10} 0.02 & R < 0.1\ \mathrm{mm\,h^{-1}} \end{cases} \tag{1}$$

The discontinuity at $0.1\ \mathrm{mm\,h^{-1}}$ is useful for giving the model a clearer distinction between the raining and non-raining points, but we found it could create artifacts in generative models. To mitigate this and other artifacts in the input data, we further apply antialiasing to the samples with a Gaussian filter of 0.5 pixel standard deviation.

4.2 Latent diffusion model

LDCast is a conditional LDM that consists of three main network components: a forecaster stack, a denoiser stack and a variational autoencoder. An overview of the network structure is shown in Fig. 6. Below, we describe the components of the network and the training process. Implementation details such as hyperparameters can be found in Supplementary Information Table S1. The exact information can be found in the published code as indicated under Code Availability.

4.2.1 Forecaster

The forecaster stack is based on the AFNO. In the FourCastNet architecture [38], a series of 2D AFNO blocks is used to process the atmospheric state at time step t to predict the state at $t + 1$. Each block consists of an AFNO and a pixelwise multilayer perceptron (MLP) network. The model is initially trained to predict one time step, then fine tuned to predict two time steps, and can then be evaluated iteratively to predict further time steps. We modify this procedure for the nowcasting application, where we want to train the model to predict $D_{\mathrm{out}} = 5$ encoded output time steps simultaneously from $D_{\mathrm{in}} = 1$ encoded input time steps. The modified architecture consists of three stages:

1. **Analysis**: The input of dimension $C \times D_{\mathrm{in}} \times W \times H$, where C, W and H are the number of latent-space channels, width and height of the encoded input respectively, is processed with a series of AFNO+MLP blocks.
2. **Temporal transformer**: The input is projected to the $C \times D_{\mathrm{out}} \times W \times H$ output space using a cross-attention transformer [45] block that is only evaluated along the temporal dimension. The query of the cross attention is computed from sinusoidal positional encoding (as in [45]) of the time coordinates of the outputs.
3. **Forecast**: The forecast stage is identical in architecture to the analysis stage, but operates in the output space.

We note that this architecture can be used on its own for non-generative prediction. We also expect (although we do not utilize this capability in the current study) that the cross-attention mapping can be used naturally with inputs that have a variable time difference to the outputs and/or a different time resolution compared to the outputs. This adds to the flexibility of the architecture compared to the convolutional and recurrent-convolutional networks that have been frequently used for precipitation nowcasting (e.g. [14, 46]).

4.2.2 Denoiser

Our denoising stack is a modification of the U-Net-type network used by the original LDM implementation [36]. To model spatiotemporal relationships, we replaced the 2D convolutions with 3D convolutions. We removed the spatial attention layers of the original network since they add considerable computational cost and removing them did not seem to degrade performance; this is likely due to the spatiotemporally equivariant nature of our data. Furthermore, we noticed that when using the layer normalization employed in the original LDM network, LDCast often produced outputs with realistic spatial patterns but a shifted magnitude of the precipitation intensity. A simple solution was to remove the normalization layers; this allowed the model to reproduce the intensity of the rainfall better, and did not seem to impede convergence significantly.

To condition the denoising network with the forecasting network, we use blocks that concatenate the U-Net state to the conditioning variable, then apply an AFNO operation similar to that used in the forecaster to the concatenated input

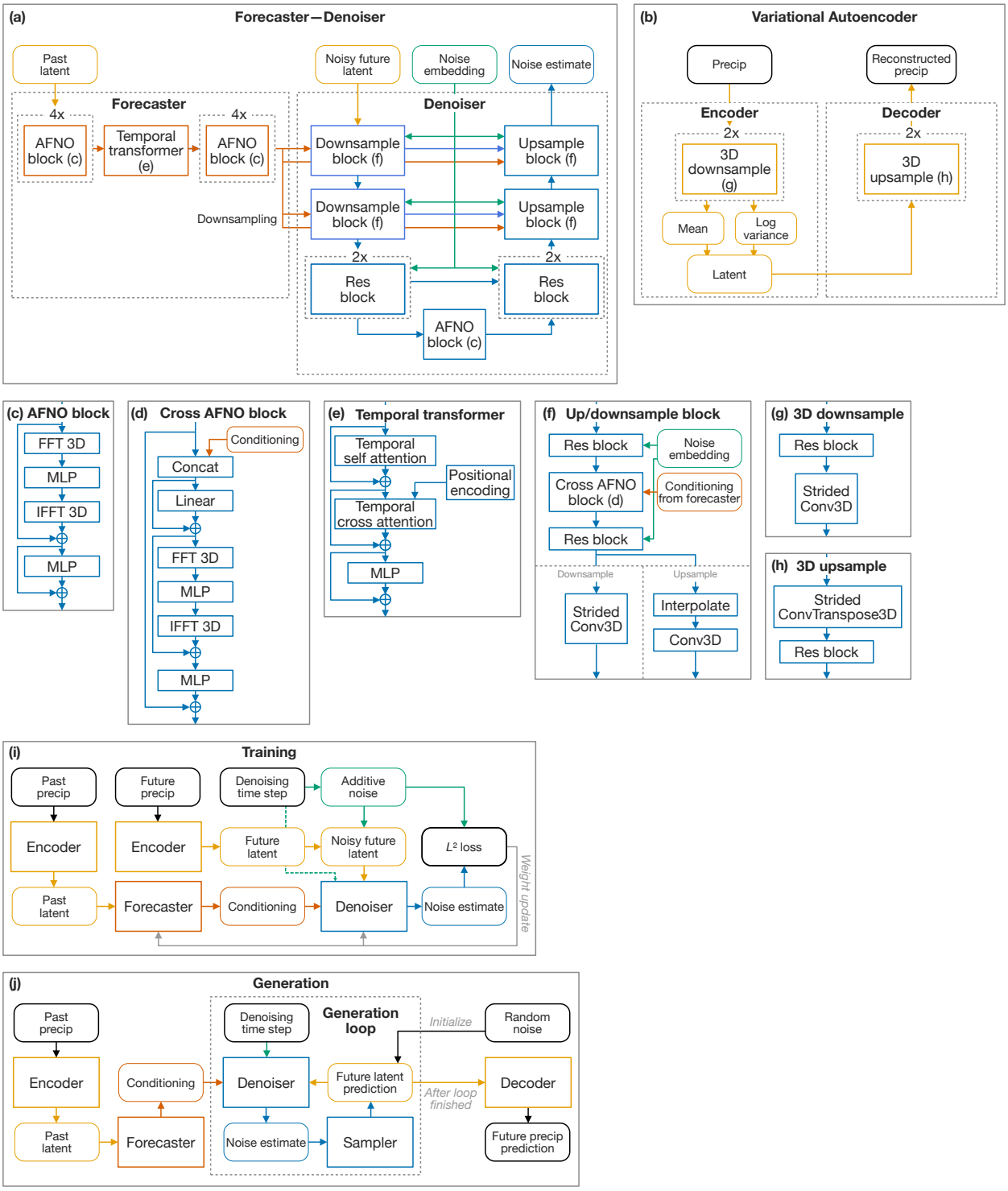

Figure 6: An overview of the LDCast neural networks. (a) The forecaster and denoiser stacks. (b) The VAE used to transform precipitation sequences to the latent space. (c)–(h) The layer blocks used in the network diagrams. (i) The training procedure. (j) The forecast generation procedure. "Conv" denotes convolution. "MLP" (multilayer perceptron) is a block consisting of a linear layer, activation function and another linear layer. "Res block" denotes a ResNet-type residual block [44]; the noise embedding is added to the input of the block.

(Fig. 6d). This is based on the reasoning of [37] that AFNO is used in a manner analogous to self-attention; we thus aim at a cross attention-like operation with this block.

4.2.3 Variational autoencoder

The VAE is used to encode samples from the pixel space to a continuous latent space and then decode them back to the pixel space. We construct the encoder and decoder parts of the VAE as simple 3D convolutional networks, where each level consists of a ResNet-type residual block and a downsampling (encoder) or upsampling (decoder) convolutional layer. Each level reduces each spatial and temporal dimension by a factor of 2; we use two levels to reduce the number of points by a factor of $4 \times 4 \times 4 = 64$. The encoder output is bottlenecked to 32 channels. Between the encoder and decoder stages, the VAE latent space is regularized with a loss based on Kullback–Leibler divergence (KL) between the latent variable and a multivariate standard normal variable.

While the number of spatiotemporal grid points is reduced by a number of 64 in the encoding process, the number of channels is also increased from 1 to 32. Thus, the total amount of data is decreased only by a factor of 2. However, we found that the reduction in spatial resolution is more important for reducing the computational cost of the forecaster and denoiser stacks. Therefore, the performance gain obtained by operating in the latent space is considerably larger than the data reduction factor.

4.2.4 Training

The VAE was trained before the rest of the network, using L^1 loss and the KL regularization term. Once trained, the VAE weights were held fixed while the forecaster and denoiser stacks were trained simultaneously. The model was trained to predict 5 time steps from 1 time step in the latent space, corresponding to predicting 20 time steps (100 min) from 4 time steps (20 min) in the pixel space.

The conditional LDM training loss can be parameterized as an L^2 loss [36]

$$L_{\mathrm{LDM}} = \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,\mathbf{I}),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right] \tag{2}$$

where x is the condition (past precipitation), $\mathcal{E}$ is the encoder, y is the real sample (observed future precipitation), ϵ is random noise, t is the step of the denoising process, z_t is the noisy latent-space sample at step t, τ_θ is the conditioning (forecaster) stack, ϵ_θ is the denoiser and θ represents the trainable parameters of the networks.

We used the AdamW optimizer [47] to train both networks. The hyperparameters are found in Supplementary Information Table S1. The learning rate schedule was based on monitoring the loss in the validation set after every checkpoint, which were performed every 1000 training batches; if the validation loss did not decrease for a 3 consecutive checkpoints, the learning rate was reduced. Early stopping was also used, terminating the training after 6 checkpoints had passed without improvement in the validation loss. Exponential moving averaging (EMA) was applied to the network weights, following [36].

The model was initially trained to convergence with 128×128 pixel samples to reduce training time. It was then fine-tuned with another training run, in which the model was initialized with the weights obtained in the pre-training, using 256×256 pixel samples. This saves considerable training time compared to training the model from random initialization with 256×256 pixel samples.

4.2.5 Evaluation

We produced samples using the standard LDM approach (Fig. 6j):

1. The input precipitation is encoded to the latent space using the VAE encoder.
2. A prediction is computed from the latent-space inputs using the forecaster stack.
3. Starting from $\mathcal{N}(0, \mathbf{I})$ distributed random noise, we perform 50 iterations of the denoiser with the PLMS sampler, using the prediction obtained from the previous step for conditioning.
4. The denoised latent variables thus obtained are decoded to precipitation using the VAE decoder.

The AFNO layers operate similarly to fully convolutional layers, so the model can be trained with samples of one size and then applied to another. We performed the evaluation with 256×256 pixel samples that were also used in the fine-tuning phase of the training. We also experimented with evaluating the model trained only with 128×128 pixel samples; the results were quite similar to those for the final model, suggesting that the fine tuning may be omitted if desirable from a computational perspective.

4.2.6 Postprocessing

When using the LDCast output, we set all precipitation rate predictions below $0.1\ \mathrm{mm\,h^{-1}}$ to zero and cap the precipitation rate to the maximum in the Swiss dataset, approximately $118\ \mathrm{mm\,h^{-1}}$. When computing quantitative scores (CRPS, FSS and the rank histograms) for LDCast, we reduce bias by using probability matching (PM) based on results on the validation set of the Swiss dataset. That is, we compute the cumulative distribution functions (CDFs) of the predicted values and observed values on the Swiss validation set, and then apply adjustments to the predictions such that the CDFs match. The PM based on the Swiss validation set is used to adjust both the results for the Swiss test set and those for the German dataset. In order to compare the models fairly, we use the same postprocessing procedure also for the benchmarks.

4.3 Verification scores

4.3.1 Continuous ranked probability score

The CRPS [48] measures the accuracy of a probabilistic forecast, taking into account both the bias and the spread. Using i to denote a single point in a multidimensional dataset, let y_i be the observation at that point and $\hat{F}_i$ be the CDF of corresponding probabilistic forecast $\hat{y}_i$. The CRPS at i is defined as the integral of the squared difference of $\hat{F}_i$ and the CDF of y_i, a unit step function H:

$$\mathrm{CRPS}(\hat{F}_i, y_i) = \int_{-\infty}^{\infty} \left(\hat{F}_i(x) - H(x - y_i)\right)^2 \mathrm{d}x \tag{3}$$

where

$$H(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{4}$$

When an ensemble is used to represent the probability of the forecast, there are N_e discrete forecasts at i: $\hat{y}_{i,1}, \ldots, \hat{y}_{i,N_\mathrm{e}}$. The forecast CDF is then a function consisting of multiple steps:

$$\hat{F}_i(x) = \frac{1}{N_\mathrm{e}} \sum_{k=1}^{N_\mathrm{e}} H(x - \hat{y}_{i,k}). \tag{5}$$

The CRPS for an entire dataset (or a subset of it) of N_s samples is computed as the average $N_\mathrm{s}^{-1} \sum_{i=1}^{N_\mathrm{s}} \mathrm{CRPS}(\hat{F}_i, y_i)$. In the special case of $N_\mathrm{e} = 1$, the CRPS over a dataset is simply the mean absolute error (MAE) between the forecast and the observation. Thus, CRPS can be viewed as a generalization of the MAE for probabilistic forecasts.

4.3.2 Probability integral transform / rank distribution

The probability integral transform (PIT) tests whether a probabilistic prediction has the same probability distribution as the observations, that is, whether the uncertainty of the predictions is modeled correctly.

Using the notation adopted in Sect. 4.3.1, we first define at each point i

$$r_i = \hat{F}_i(y_i), \tag{6}$$

that is, $r_i \in [0, 1]$ is the value of the forecast CDF at the observation. PIT is based on the fact that if y and $\hat{y}$ come from the same distribution, the distribution of r_i over the dataset approaches the standard uniform distribution $U_{[0,1]}$ as $N_\mathrm{s} \to \infty$.

By computing r over a dataset, one can examine the uniformity of the resulting distribution p_r. This can be done either visually by plotting the distribution, or quantitatively by computing a distribution distance metric between p_r and the standard uniform distribution $U_{[0,1]}$. One possible metric is the Kullback–Leibler divergence (KL) frequently used in machine learning.

In the case of ensemble forecasts, the PIT is equivalent to the *rank distribution* (or rank histogram) [49] frequently used in ensemble forecast verification. In this case, r_i is equivalent to the rank of the observation among the forecasts (i.e. the number of forecasts that are smaller than the observation; ties are randomized) divided by the number of ensemble members N_e:

$$r_i = \frac{1}{N_\mathrm{e}} \sum_{k=1}^{N_\mathrm{e}} H(y_i - \hat{y}_{i,k}) \tag{7}$$

Consequently, the distribution p_r is discrete, with possible values $r_j = j/N_\mathrm{e}$, $j \in 0 \dots N_\mathrm{e}$. One should thus use the discrete version of KL. The discrete uniform distribution with $N_\mathrm{e}+1$ possible values is $U(r_j) = (N_\mathrm{e}+1)^{-1}$ at each r_j. We then get the KL as

$$\mathrm{KL}(U, p_r) = \sum_{j=0}^{N_\mathrm{e}} U(r_j) \ln\left(\frac{U(r_j)}{p_r(r_j)}\right) = -\frac{1}{N_\mathrm{e}+1} \sum_{j=0}^{N_\mathrm{e}} \ln\left((N_\mathrm{e}+1)\, p_r(r_j)\right). \tag{8}$$

4.3.3 Fractions skill score

In precipitation forecasts, one often wants to predict whether the precipitation exceeds a certain threshold level T. Using the notation of the previous sections, we define the occurrence of such events as binary variables:

$$S_i = H(y_i - T) \tag{9}$$

$$\hat{S}_{i,k} = H(\hat{y}_{i,k} - T) \tag{10}$$

where S_i is the observed occurrence of the threshold-exceeding event at point i and $\hat{S}_{i,k}$ is the occurrence in the forecast at i in the ensemble member k.

The FSS [50] is based on the notion that predicting the location of an event wrong by a short distance should be penalized less than mispredicting it by a long distance. Most scores such as root-mean-square error (RMSE), MAE or the critical success index (CSI; also known as the threat score or the intersection-over-union score) do not have this property; they penalize incorrect predictions equally regardless of whether or not there is a correct prediction nearby.

In the calculation of FSS, one first defines the *fraction* of events in a neighborhood V of points as:

$$M_V = \frac{1}{|V|} \sum_{i \in V} S_i \tag{11}$$

$$\hat{M}_V = \frac{1}{N_\mathrm{e}|V|} \sum_{i \in V} \sum_{k=1}^{N_\mathrm{e}} \hat{S}_{i,k} \tag{12}$$

where $|V|$ denotes the number of points in V. In the definition of $\hat{M}_V$, we use the generalization of [51] to ensemble forecasts. To calculate FSS at a given spatial scale n, we define $W_{(n)}$ as the set of all square neighborhoods of $n \times n$ size. This follows common practice and simplifies calculation; alternatively one can use, for instance, circular neighborhoods. The fractional Brier score $\mathrm{FBS}_{(\mathrm{n})}$ and the reference FBS (i.e. the FBS of a skilless forecast) $\mathrm{FBS}_{(\mathrm{n}),\mathrm{ref}}$ for the scale n are given by

$$\mathrm{FBS}_{(\mathrm{n})} = \frac{1}{|W_{(n)}|} \sum_{V \in W_{(n)}} \left(\hat{M}_V - M_V\right)^2 \tag{13}$$

$$\mathrm{FBS}_{(\mathrm{n}),\mathrm{ref}} = \frac{1}{|W_{(n)}|} \sum_{V \in W_{(n)}} \hat{M}_V^2 + M_V^2. \tag{14}$$

Finally, the FSS for scale n is

$$\mathrm{FSS}_{(\mathrm{n})} = 1 - \frac{\mathrm{FBS}_{(\mathrm{n})}}{\mathrm{FBS}_{(\mathrm{n}),\mathrm{ref}}}. \tag{15}$$

FSS is 1 for an ideal forecast and 0 for a skilless forecast.

4.4 Benchmarks

4.4.1 Deep Generative Models of Radar

DGMR [22] represents the current state of the art in generative nowcasting. It is a GAN generator that was trained with a GAN hinge loss combined with a regularization loss that encourages the ensemble mean of the generated precipitation fields to match the true precipitation amount. The generator is built using convolutional gated recurrent unit (ConvGRU) layers organized in a U-Net-like structure, while the discriminator is split into separate spatial and temporal discriminators that both use convolutional layers. The GAN was trained with a dataset of radar-measured precipitation from the UK Met Office RadarNet4 network of C-band polarimetric radars.

The DGMR authors have made a saved model available, and we use it as our main point of comparison to LDCast. The inputs are compatible with our model, as the available model is trained for 256×256 pixel inputs at $1\ \mathrm{km}$ spatial

and 5 min temporal resolution. DGMR produces an output up to 90 min to the future. Because of the spatiotemporal latent-space encoding our model must produce forecasts of a multiple of 4 time steps (20 min), so we trained it to predict up to 100 min into the future and truncated the results at 90 min when computing scores that are compared directly to DGMR.

4.4.2 PySTEPS

PySTEPS [10] is a nowcasting library that implements the STEPS algorithm for stochastic ensemble nowcasting. We include PySTEPS in the comparisons presented in this paper as a state-of-the-art non-ML-based method. Extensive comparisons between PySTEPS and DGMR can also be found in [22]. We used PySTEPS following the STEPS example on the PySTEPS website [1].

We produced an output of zero rainfall for PySTEPS whenever the input was all zeros. We also found that occasional samples in our datasets caused the PySTEPS processing to fail. Examination of these cases showed that the problems occurred with very low rain rates, so we produced an output of all zero precipitation whenever this happened.

Data availability

The pretrained models and the training and evaluation datasets can be found at [52].

Code availability

The code for replicating the results can be found at `https://github.com/MeteoSwiss/ldcast`.

The saved DGMR generator model can be found `https://github.com/deepmind/deepmind-research/tree/master/nowcasting`. The PySTEPS library website is `https://pysteps.github.io/`; PySTEPS can also be installed through many Python package managers.

References

[1] Surcel, M., Zawadzki, I. & Yau, M. K. A study on the scale dependence of the predictability of precipitation patterns. *J. Atmos. Sci.* **72**, 216–235 (2015). URL `https://doi.org/10.1175/JAS-D-14-0071.1`.

[2] Sun, J. *et al.* Use of NWP for nowcasting convective precipitation: Recent progress and challenges. *Bull. Amer. Meteor. Soc.* **95**, 409–426 (2014).

[3] Bellon, A. & Austin, G. L. The evaluation of two years of real-time operation of a short-term precipitation forecasting procedure (SHARP). *J. Appl. Meteor.* **17**, 1778–1787 (1978). URL `http://www.jstor.org/stable/26178613`.

[4] Germann, U. & Zawadzki, I. Scale-dependence of the predictability of precipitation from continental radar images. Part I: Description of the methodology. *Mon. Wea. Rev.* **130**, 2859–2873 (2002). URL `https://doi.org/10.1175/1520-0493(2002)130<2859:SDOTPO>2.0.CO;2`.

[5] Bowler, N. E., Pierce, C. E. & Seed, A. W. STEPS: A probabilistic precipitation forecasting scheme which merges an extrapolation nowcast with downscaled NWP. *Quart. J. Roy. Meteor. Soc.* **132**, 2127–2155 (2006). URL `https://doi.org/10.1256/qj.04.100`.

[6] Sideris, I. V., Foresti, L., Nerini, D. & Germann, U. NowPrecip: localized precipitation nowcasting in the complex terrain of Switzerland. *Quart. J. Roy. Meteor. Soc.* **146**, 1768–1800 (2020). URL `https://doi.org/10.1002/qj.3766`.

[7] Panziera, L., Germann, U., Gabella, M. & Mandapaka, P. V. NORA–nowcasting of orographic rainfall by means of analogues. *Quart. J. Roy. Meteor. Soc.* **137**, 2106–2123 (2011). URL `https://doi.org/10.1002/qj.878`.

[8] Foresti, L., Sideris, I. V., Panziera, L., Nerini, D. & Germann, U. A 10-year radar-based analysis of orographic precipitation growth and decay patterns over the Swiss Alpine region. *Quart. J. Roy. Meteor. Soc.* **144**, 2277–2301 (2018). URL `https://doi.org/10.1002/qj.3364`.

[9] Seed, A. W., Pierce, C. E. & Norman, K. Formulation and evaluation of a scale decomposition-based stochastic precipitation nowcast scheme. *Water Resour. Res.* **49**, 6624–6641 (2013). URL `https://doi.org/10.1002/wrcr.20536`.

[1] `https://pysteps.readthedocs.io/en/stable/auto_examples/plot_steps_nowcast.html`

[10] Pulkkinen, S. *et al.* Pysteps: an open-source Python library for probabilistic precipitation nowcasting (v1.0). *Geosci. Model Dev.* **12**, 4185–4219 (2019). URL https://doi.org/10.5194/gmd-12-4185-2019/.

[11] Shi, X. *et al.* Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M. & Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 28 (Curran Associates, Inc., 2015). URL https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

[12] Agrawal, S. *et al.* Machine learning for precipitation nowcasting from radar images (2019). URL https://arxiv.org/abs/1912.12132.

[13] Ayzel, G., Scheffer, T. & Heistermann, M. RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. *Geosci. Model Dev.* **13**, 2631–2644 (2020). URL https://doi.org/10.5194/gmd-13-2631-2020/.

[14] Franch, G. *et al.* Precipitation nowcasting with orographic enhanced stacked generalization: Improving deep learning predictions on extreme events. *Atmosphere* **11** (2020). URL https://doi.org/10.3390/atmos11030267.

[15] Goodfellow, I. *et al.* Generative adversarial networks. *Commun. ACM* **63**, 139–144 (2020). URL https://doi.org/10.1145/3422622.

[16] Leinonen, J., Nerini, D. & Berne, A. Stochastic super-resolution for downscaling time-evolving atmospheric fields with a generative adversarial network. *IEEE Trans. Geosci. Remote Sens.* **59**, 7211–7223 (2021). URL https://doi.org/10.1109/TGRS.2020.3032790.

[17] Price, I. & Rasp, S. Increasing the accuracy and resolution of precipitation forecasts using deep generative models. In Camps-Valls, G., Ruiz, F. J. R. & Valera, I. (eds.) *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, vol. 151 of *Proceedings of Machine Learning Research*, 10555–10571 (PMLR, 2022). URL https://proceedings.mlr.press/v151/price22a.html.

[18] Harris, L., McRae, A. T. T., Chantry, M., Dueben, P. D. & Palmer, T. N. A generative deep learning approach to stochastic downscaling of precipitation forecasts. *J. Adv. Model. Earth Sys.* **14**, e2022MS003120 (2022). URL https://doi.org/10.1029/2022MS003120.

[19] Hayatbini, N. *et al.* Conditional generative adversarial networks (cGANs) for near real-time precipitation estimation from multispectral GOES-16 satellite imageries — PERSIANN-cGAN. *Remote Sens.* **11** (2019). URL https://doi.org/10.3390/rs11192193.

[20] Wang, C., Tang, G. & Gentine, P. PrecipGAN: Merging microwave and infrared data for satellite precipitation estimation using generative adversarial network. *Geophys. Res. Lett.* **48**, e2020GL092032 (2021). URL https://doi.org/10.1029/2020GL092032.

[21] Scher, S. & Peßenteiner, S. Technical note: Temporal disaggregation of spatial rainfall fields with generative adversarial networks. *Hydrol. Earth Syst. Sci.* **25**, 3207–3225 (2021). URL https://doi.org/10.5194/hess-25-3207-2021.

[22] Ravuri, S. *et al.* Skilful precipitation nowcasting using deep generative models of radar. *Nature* **597**, 672–677 (2021). URL https://doi.org/10.1038/s41586-021-03854-z/.

[23] Mescheder, L., Geiger, A. & Nowozin, S. Which training methods for GANs do actually converge? In Dy, J. & Krause, A. (eds.) *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, 3481–3490 (PMLR, 2018). URL https://proceedings.mlr.press/v80/mescheder18a.html.

[24] Bau, D. *et al.* Seeing what a GAN cannot generate. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). URL https://doi.org/10.1109/ICCV.2019.00460.

[25] Song, Y. & Ermon, S. Generative modeling by estimating gradients of the data distribution. In Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 32 (Curran Associates, Inc., 2019). URL https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf.

[26] Song, Y. & Ermon, S. Improved techniques for training score-based generative models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, 12438–12448 (Curran Associates, Inc., 2020). URL https://proceedings.neurips.cc/paper/2020/file/92c3b916311a5517d9290576e3ea37ad-Paper.pdf.

[27] Dhariwal, P. & Nichol, A. Diffusion models beat GANs on image synthesis. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. & Vaughan, J. W. (eds.) *Advances in Neural Information Processing Systems*, vol. 34, 8780–8794 (Curran Associates, Inc., 2021). URL https://proceedings.neurips.cc/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

[28] Saharia, C. *et al.* Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22 (Association for Computing Machinery, New York, NY, USA, 2022). URL https://doi.org/10.1145/3528233.3530757.

[29] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. & Chen, M. Hierarchical text-conditional image generation with CLIP latents (2022). URL https://arxiv.org/abs/2204.06125. 2204.06125.

[30] Saharia, C. *et al.* Photorealistic text-to-image diffusion models with deep language understanding. In Oh, A. H., Agarwal, A., Belgrave, D. & Cho, K. (eds.) *Advances in Neural Information Processing Systems* (2022). URL https://openreview.net/forum?id=08Yk-n5l2Al.

[31] Li, H. *et al.* SRDiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing* **479**, 47–59 (2022). URL https://doi.org/10.1016/j.neucom.2022.01.029.

[32] Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, 6840–6851 (Curran Associates, Inc., 2020). URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[33] Song, J., Meng, C. & Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations* (2021). URL https://openreview.net/forum?id=St1giarCHLP.

[34] Liu, L., Ren, Y., Lin, Z. & Zhao, Z. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations* (2022). URL https://openreview.net/forum?id=PlKWVd2yBkY.

[35] Addison, H., Kendon, E., Ravuri, S., Aitchison, L. & Watson, P. A. Machine learning emulation of a local-scale UK climate model (2022). URL https://arxiv.org/abs/2211.16116.

[36] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10684–10695 (2022). URL https://arxiv.org/abs/2112.10752.

[37] Guibas, J. *et al.* Efficient token mixing for transformers via Adaptive Fourier Neural Operators. In *International Conference on Learning Representations* (2022). URL https://arxiv.org/abs/2111.13587.

[38] Pathak, J. *et al.* FourCastNet: A global data-driven high-resolution weather model using Adaptive Fourier Neural Operators (2022). URL https://arxiv.org/abs/2202.11214. 2202.11214.

[39] Esser, P., Rombach, R. & Ommer, B. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12873–12883 (2021). URL https://doi.org/10.1109/CVPR46437.2021.01268.

[40] Leinonen, J., Hamann, U. & Germann, U. Seamless lightning nowcasting with recurrent-convolutional deep learning. *Artif. Intell. Earth Syst.* **1**, e220043 (2022). URL https://doi.org/10.1175/AIES-D-22-0043.1.

[41] Germann, U., Galli, G., Boscacci, M. & Bolliger, M. Radar precipitation measurement in a mountainous region. *Quart. J. Roy. Meteor. Soc.* **132**, 1669–1692 (2006). URL https://doi.org/10.1256/qj.05.190.

[42] Germann, U., Boscacci, M., Gabella, M. & Schneebeli, M. Weather radar in Switzerland. In Willemse, S. & Furger, M. (eds.) *From weather observations to atmospheric and climate science in Switzerland: Celebrating 100 years of the Swiss Society for Meteorology*, chap. 9 (Vdf Hochschulverlag AG an der ETH Zürich, Zürich, Switzerland, 2016).

[43] Stephan, K., Klink, S. & Schraff, C. Assimilation of radar-derived rain rates into the convective-scale model COSMO-DE at DWD. *Quart. J. Roy. Meteor. Soc.* **134**, 1315–1326 (2008). URL https://doi.org/10.1002/qj.269.

[44] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). URL https://doi.org/10.1109/CVPR.2016.90.

[45] Vaswani, A. *et al.* Attention is all you need. In Guyon, I. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 30 (Curran Associates, Inc., 2017). URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[46] Leinonen, J., Hamann, U., Sideris, I. V. & Germann, U. Thunderstorm nowcasting with deep learning: a multi-hazard data fusion model. *Geophys. Res. Lett.* (2023). URL https://doi.org/10.1029/2022GL101626. Accepted for publication.

[47] Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations* (2019). URL https://openreview.net/forum?id=Bkg6RiCqY7.

[48] Gneiting, T. & Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *J. Am. Stat. Assoc.* **102**, 359–378 (2007). URL https://doi.org/10.1198/016214506000001437.

[49] Candille, G. & Talagrand, O. Evaluation of probabilistic prediction systems for a scalar variable. *Quart. J. Roy. Meteor. Soc.* **131**, 2131–2150 (2005). URL https://doi.org/10.1256/qj.04.71.

[50] Roberts, N. M. & Lean, H. W. Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Mon. Wea. Rev.* **136**, 78–97 (2008). URL https://doi.org/10.1175/2007MWR2123.1.

[51] Duc, L., Saito, K. & Seko, H. Spatial-temporal fractions verification for high-resolution ensemble forecasts. *Tellus A: Dyn. Meteorol. Oceanogr.* **65**, 18171 (2013). URL https://doi.org/10.3402/tellusa.v65i0.18171.

[52] Leinonen, J., Hamann, U., Nerini, D., Germann, U. & Franch, G. Pretrained models and results for "Latent diffusion models for generative precipitation nowcasting with accurate uncertainty quantification" (2023). URL https://doi.org/10.5281/zenodo.7780914.

Acknowledgments

We thank Markus Schultze and Ulrich Blahak of DWD for providing the German radar dataset, and Nathalie Rombeek for feedback on the article.

Funding

JL was supported by the fellowship "Seamless Artificially Intelligent Thunderstorm Nowcasts" from the European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT). The hosting institution of this fellowship is MeteoSwiss in Switzerland.