Stochastic Super-Resolution for Downscaling Time-Evolving Atmospheric Fields With a Generative Adversarial Network

Jussi Leinonen, Daniele Nerini, and Alexis Berne

***Abstract*— Generative adversarial networks (GANs) have been recently adopted for super-resolution, an application closely related to what is referred to as "downscaling" in the atmospheric sciences: improving the spatial resolution of low-resolution images. The ability of conditional GANs to generate an ensemble of solutions for a given input lends itself naturally to stochastic downscaling, but the stochastic nature of GANs is not usually considered in super-resolution applications. Here, we introduce a recurrent, stochastic super-resolution GAN that can generate ensembles of time-evolving high-resolution atmospheric fields for an input consisting of a low-resolution sequence of images of the same field. We test the GAN using two data sets: one consisting of radar-measured precipitation from Switzerland; the other of cloud optical thickness derived from the Geostationary Earth Observing Satellite 16 (GOES-16). We find that the GAN can generate realistic, temporally consistent super-resolution sequences for both data sets. The statistical properties of the generated ensemble are analyzed using rank statistics, a method adapted from ensemble weather forecasting; these analyses indicate that the GAN produces close to the correct amount of variability in its outputs. As the GAN generator is fully convolutional, it can be applied after training to input images larger than the images used to train it. It is also able to generate time series much longer than the training sequences, as demonstrated by applying the generator to a three-month data set of the precipitation radar data. The source code to our GAN is available at https://github.com/jleinonen/downscaling-rnn-gan.**

***Index Terms*— Atmosphere, clouds, image processing, meteorological radar, neural networks, remote sensing.**

I. Introduction

SUPER-RESOLUTION refers to enhancing the spatial resolution of an image beyond the original resolution. In digital image processing, the term describes various algorithms that take one or more low-resolution images and generate an estimate of a higher resolution image of the same target [1]. In climate science, downscaling[1] is a concept closely related to super-resolution [2]–[4]. It is used especially in connection with precipitation, which can vary sharply over spatial scales of 1 km or less, while global climate models typically have resolutions of tens or hundreds of kilometers. Downscaling bridges this gap by producing precipitation fields at a finer resolution for the purpose of assessing the impacts of phenomena such as extreme rainfall.

Like many other image processing applications, super-resolution has benefited from the introduction of the techniques of deep learning and particularly convolutional neural networks (CNNs). Early attempts at super-resolution using deep CNNs focused on finding image quality metrics that could serve as loss functions that produce sharp images [5]–[7]. More recently, generative adversarial networks (GANs) have been used to train super-resolution CNNs [8], [9]. GANs are a general technique for generating artificial samples [10] from the training distribution. When used to train CNNs, they can create visually realistic artificial images of, e.g., human faces [11] and landscapes [12]. In super-resolution applications, GANs create reconstructed high-resolution images by using one neural network (the discriminator) to evaluate the quality of the high-resolution outputs, while another network (the generator) is trained to output images that the discriminator considers to be of high quality. The two networks are trained simultaneously against each other (hence "adversarial"), and thus, the discriminator adaptively learns an appropriate reconstruction metric for the data set rather than relying on expert-provided metrics. The GAN generator may also have a noise input, which the generator learns to map to the variability of the output.

Producing a super-resolution image from only one source image (referred to as single-image super-resolution) is an underdetermined problem that generally does not have a unique solution. Super-resolution techniques, therefore, try to produce an image that is consistent with the input and that also takes advantage of prior knowledge about the structure of the high-resolution images. Despite the inherent uncertainty in the super-resolution reconstruction, often, these methods

Manuscript received May 20, 2020; revised July 22, 2020, August 26, 2020, and October 12, 2020; accepted October 18, 2020. This work was supported by the Swiss National Science Foundation under Grant #200020_175700. The GPU computing resources used in this study were provided through a grant from the Swiss National Supercomputing Centre (CSCS) under project ID sm35. *(Corresponding author: Jussi Leinonen.)*

Jussi Leinonen is with Environmental Remote Sensing Laboratory, École Polytechnique fédérale de Lausanne, Lausanne 1015, Switzerland, and also with the Federal Office of Meteorology and Climatology MeteoSwiss, 6605 Locarno-Monti, Switzerland (e-mail: jussi.leinonen@meteoswiss.ch).

Alexis Berne is with Environmental Remote Sensing Laboratory, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland.

Daniele Nerini is with the Federal Office of Meteorology and Climatology MeteoSwiss, 6605 Locarno-Monti, Switzerland.

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2020.3032790

[1]The terminology here is potentially confusing. Upsampling, meaning an operation that increases the number of pixels in an image and, thus, reduces the physical size of each pixel, is sometimes referred to as "upscaling" in image processing. In climate science, the term "downscaling" is used instead for an operation that reduces the physical size of pixels and, thus, improves the resolution. We attempt to avoid this unfortunate contradiction by using the terms "upsampling" and "downsampling" as they are defined in image processing and the term "downscaling" as it is used in climate science.

produce a single output for a given input and rarely estimate the uncertainty of the output. For instance, the state-of-the-art enhanced super-resolution GAN (ESRGAN) architecture does not include a noise input at all and is, therefore, completely deterministic for a given input [9]. This is often acceptable in applications such as enhancing the resolution of natural photographs, where a single plausible solution tends to be sufficient.

In contrast to photograph processing, in climate and weather applications, it is crucial to understand and quantify the uncertainty of predictions. Classical precipitation downscaling algorithms have used techniques such as randomized autoregressive models [13], [14] or multifractal cascades [15], to produce different random realizations of the high-resolution field for a given low-resolution input. GANs offer a natural way to model uncertainty using modern machine-learning methods, less dependent on particular statistical assumptions than the traditional methods. Regardless, the uncertainty aspect has been largely ignored in earlier attempts at improving the resolution of climate fields using deep learning even when employing GANs for this problem [16] or for other super-resolution applications related to climate or remote sensing [17]–[19] although a few studies have used GANs to represent uncertainty in other atmospheric data problems [20], [21]. Moreover, while GANs have been recently also used to model the time evolution of atmospheric fields [22], few studies using deep learning have investigated modeling the uncertainty of the generated high-resolution image in a manner consistent with the time evolution of atmospheric fields—a problem analogous to video super-resolution, which has also been studied using GANs [23], [24].

In this article, we introduce a stochastic super-resolution GAN that can produce an ensemble of plausible high-resolution outputs for a given input. The GAN architecture also includes a recurrent neural network (RNN) structure, which permits the generated outputs to evolve in time in a consistent manner. The architecture is fully convolutional, and thus, the networks can be trained with small images and later applied to larger ones. We use this GAN to stochastically downscale time series of images from two atmospheric remote-sensing data sets: precipitation measured by the MeteoSwiss ground-based weather radar network and cloud optical depth imaged by the Geostationary Operational Environmental Satellite 16 (GOES-16). The same architecture is used for both data sets, and thus, we expect that the method can be generalized to other atmospheric variables and further applications beyond the atmospheric field.

The rest of this article is structured as follows. Section II describes the network architecture and training and the validation of the results. Section III describes the data sets and their preprocessing, and Section IV presents and discusses the evaluation results. Finally, Section V concludes this article and presents objectives for future work.

II. Methods

A. Overview

A GAN consists of two neural networks: the generator (G) and the discriminator (D). The discriminator is trained to determine whether or not its input is an example from the training data set, while the generator is simultaneously trained to produce artificial samples that the discriminator classifies as real. Thus, the generator learns to produce realistic-looking artificial samples. In this study, we use a conditional GAN [25], in which both G and D are given an additional condition. In the case of super-resolution, the condition is a low-resolution image, and the discriminator is trained to distinguish between real high-resolution images from the training data set and artificial high-resolution images produced by the generator, conditionally to the corresponding low-resolution images.

For additional background on GANs, we refer the reader to [26], while a general overview of deep-learning methods can be found in [27].

B. Network Architecture

In our GAN, both G and D are deep CNNs which make extensive use of residual blocks [28]. The residual blocks process their input through two activation and convolution layers and, finally, add the input to the output at the end of processing. Consequently, an inactive residual block (one with near-zero weights in the convolutional layers) acts as an identity map. Thus, the number of residual blocks in a network is often flexible since the blocks that the network does not use simply pass their input through. As training progresses, residual networks may activate additional blocks as the network learns to take advantage of deeper features. The numbers of residual blocks in our networks were determined by an iterative design process, but, for the abovementioned reasons, their exact number is not critically important as having too many residual blocks need not be harmful to performance, although it does increase computational cost.

In contrast to most GANs, our networks also employ recurrent layers in the form of convolutional gated recurrent units (ConvGRUs), variants of the gated recurrent unit (GRU) [29]. These recurrent layers permit the network to learn the temporal evolution of the fields, while the convolutional and residual blocks learn the spatial structure. ConvGRUs replace learned affine transforms in the standard GRU with 2-D convolutions. ConvGRU layers learn the appropriate update rules from one time step to the next, enabling the GAN generator to model the evolution of the fields with time, and allowing the discriminator to evaluate the plausibility of image sequences rather than single images. These layers, along with the closely related convolutional long short-term memory (LSTM) layers, have been previously applied to modeling the time evolution of precipitation fields [30], [31].

The architectures of our G and D networks are shown in Fig. 1. In the following, we give brief descriptions of the organization of the networks; the exact implementation using TensorFlow [32] and tf.keras [33], which is TensorFlow's high-level API for building and training deep learning models, can be found in the source code published at https://github.com/jleinonen/downscaling-rnn-gan.

The generator G starts with a time series of low-resolution fields (the conditioning variable), given as a 4-D tensor of dimension $N_\mathrm{t} \times h \times w \times N_\mathrm{v}$, where N_t is the number of time steps, h and w are the pixel height and width of the image,

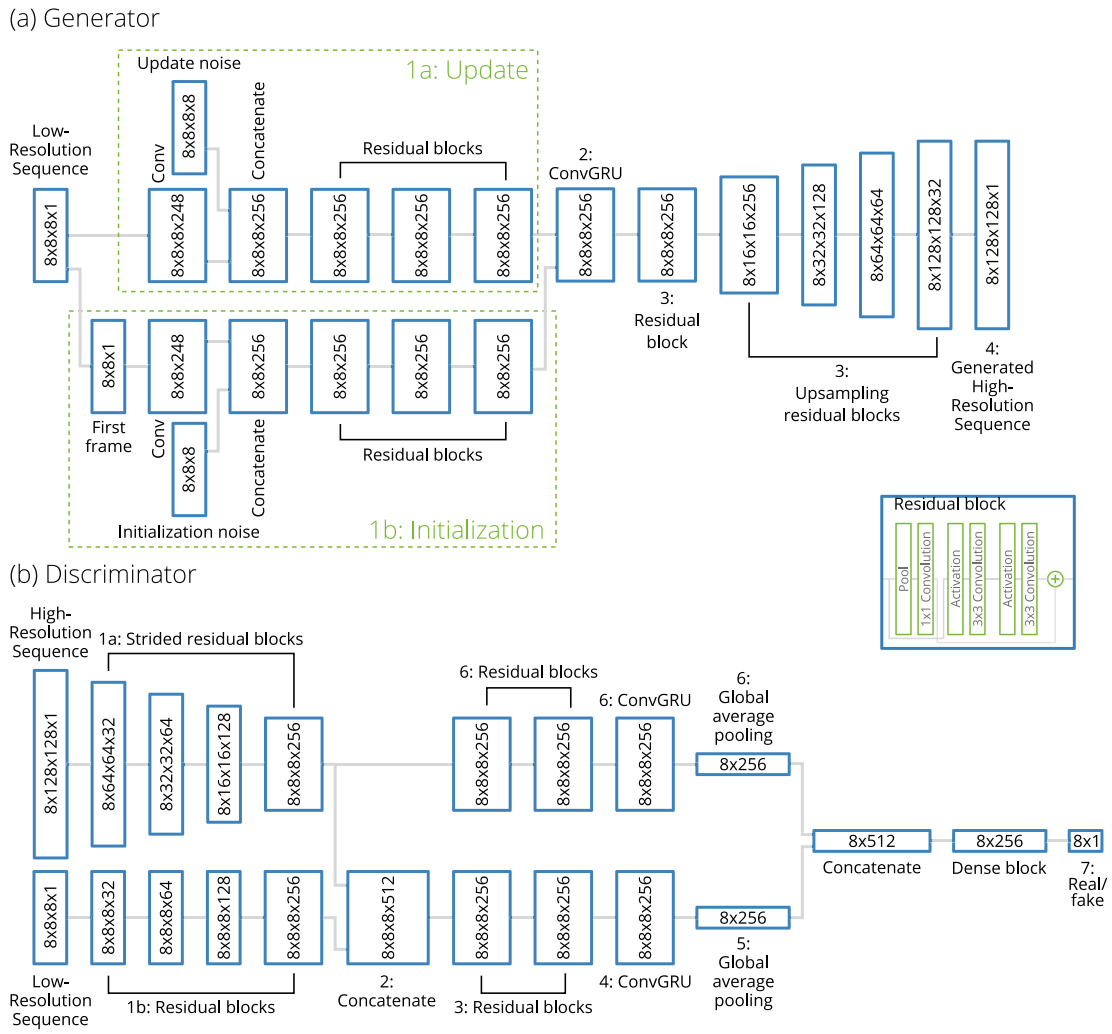

Fig. 1. Architectures of (a) generator and (b) discriminator. The numbered labels correspond to the descriptions in Section II-B. The dimensions shown here are for the training configuration where high-resolution image size $h \times w = 128 \times 128$, the number of frames per sequence $N_t = 8$, and the number of variables $N_v = 1$ for both data sets considered here. After training, the network can be evaluated using different values of these parameters.

respectively, and N_v is the number of variables. The time steps are assumed to be at constant intervals, and the size of one pixel is assumed to always correspond to a constant, well-defined physical size. The time series is processed through the following steps of the network.

1) *Encoding:*
   a) The low-resolution input tensor is mapped to a larger number of channels using a convolutional layer and concatenated with the noise input using a different noise instance for each time step. These data are then processed through a series of three residual blocks. The inputs are, thus, encoded into a deep representation.
   b) Using a similar series of layers as with step 1a above but with independent weights and only for the first time step, the initial state of the recurrent layer is derived.
2) *Recurrence:* The time evolution of the deep representation of the field is modeled with a ConvGRU layer. The input to the ConvGRU layer is the result of step 1a, while the initial state is derived from step 1b.
3) *Decoding/Upsampling:* The result of the ConvGRU layer is processed through a series of alternating residual blocks and upsampling layers. Each upsampling operation increases both spatial dimensions by a factor of two, using bilinear interpolation on the hidden representation. The residual blocks process the information to a less deep level of representation. We use four upsampling blocks, resulting in a resolution enhancement by a factor of $K = 16$. Different numbers of upsampling blocks could be used to obtain different factors of $K = 2^N$ with N a positive integer, but this would require retraining the GAN, requiring increased computation time for training, and hence, we concentrate on $K = 16$ in this work.
4) *Output:* The output of the last hidden layer is mapped using one final convolution to a high-resolution tensor of dimension $N_t \times Kh \times Kw \times N_v$. A sigmoid activation constrains the final output between 0 and 1.

L_2 weight regularization is used in the generator. All nonrecurrent layers use shared weights for each time step; this allows the generator to operate with any number of time steps. The generator has approximately 13.6 million trainable weights.

The discriminator D starts with a pair of high- and low-resolution sequences. The task of the discriminator is to determine whether or not these are a pair originating from the training data set. The processing steps below are used to achieve this.

1) *Encoding/Downsampling:*
   a) The high-resolution input is processed using a series of three residual blocks that use strided convolutions to downsample the input and encode it into a deep representation. As with the generator, the same weights are used for each time step.
   b) The low-resolution input is processed identically to step 1a, except the convolutions are not strided, and thus, no downsampling is performed. As a result, the output has the same dimensions as that of step 1a.
2) *Combination:* The outputs of steps 1a and 1b are concatenated.
3) *Further Encoding:* The joint output from step 2 is processed through two residual blocks for additional encoding.
4) *Recurrence:* The time consistency of the field is evaluated with a ConvGRU layer; unlike with the generator, we simply initialize the state to zeros.
5) *Global Average Pooling:* The average of each feature map is taken, pooling the activations at the different locations.
6) *High-Resolution Processing:* We also process the output of step 1a separately through steps 3–5 using independent weights. The motivation for this branch is to evaluate the quality of the high-resolution image separately from the consistency of the low-/high-resolution pair.
7) *Output:* The results of steps 5 and 6 are concatenated. The result is processed through one more fully connected layer and then mapped to N_t scalar values.

Spectral normalization [34] is used to constrain the discriminator. The number of trainable weights in the discriminator is approximately 15.1 million.

Leaky rectified linear unit (ReLU) activations [35] with negative slope 0.2 are used in both G and D except for the update and initialization networks in G (items 1a and 1b in the description of G), which uses regular ReLU activations. Using the regular ReLU in these parts of the network proved useful for improving stability when the generator is evaluated over long time series, we speculate that this is because the ReLU activation can become completely inactive, while the leaky ReLU cannot. Meanwhile, using leaky ReLU in the upsampling part of G (item 3 in the description) produced fewer artifacts than regular ReLUs.

We derived the architecture by examining the performance impact of the various design choices, such as the width and depth of the networks, the use of residual layers, different types of normalization, and the choice of activation functions. This can be analyzed rigorously using an approach called ablation study, in which various features and components of the network are removed individually, and the network is retrained and reevaluated with each modification; each such removal is adopted to the final architecture if it does not significantly degrade the results produced by the network. Although an ablation study is an effective method for optimizing neural network designs, the high computational cost of training the GAN would have made it impractical for us to perform a full ablation study with the available computational resources. Instead, we approached the design process using the same principles but somewhat more informally, examining the effect of the abovementioned design choices on training metrics and image quality in the initial stages of training and selecting those features that appeared to have a positive effect on performance. Similar considerations were applied to selecting the training strategies and hyperparameters.

C. Training

Formally, the conditional GAN optimization objectives are

$$\min_{\boldsymbol{\theta}_D} \ \mathrm{E}_{\mathbf{x},\mathbf{y},\mathbf{z}}[L_D(\mathbf{x},\mathbf{y},\mathbf{z};\boldsymbol{\theta}_D)] \tag{1}$$

$$\min_{\boldsymbol{\theta}_G} \ \mathrm{E}_{\mathbf{y},\mathbf{z}}[L_G(\mathbf{y},\mathbf{z};\boldsymbol{\theta}_G)] \tag{2}$$

where $\mathbf{x}$ represents real samples (for us, high-resolution sequences), $\mathbf{y}$ represents the condition (low-resolution sequences), and $\mathbf{z}$ is the noise. We denote the discriminator loss as L_D, the generator loss as L_G, and the corresponding trainable weights as $\boldsymbol{\theta}_D$ and $\boldsymbol{\theta}_G$, respectively. We trained our GAN as a Wasserstein GAN with gradient penalty (WGAN-GP) [36], using a gradient penalty weight of $\gamma = 10$. The combined conditional WGAN-GP losses for D and G are

$$\begin{aligned} L_D(\mathbf{x},\mathbf{y},\mathbf{z};\boldsymbol{\theta}_D) = {} & D(\mathbf{x},\mathbf{y}) - D(G(\mathbf{y},\mathbf{z}),\mathbf{y}) \\ & + \gamma\left(||\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}},\mathbf{y})||_2 - 1\right)^2 \end{aligned} \tag{3}$$

$$L_G(\mathbf{y},\mathbf{z};\boldsymbol{\theta}_G) = D(G(\mathbf{y},\mathbf{z})) \tag{4}$$

where the samples $\hat{\mathbf{x}}$, used to compute the gradient penalty term, are randomly weighted averages between real and generated samples

$$\hat{\mathbf{x}} = \epsilon\mathbf{x} + (1-\epsilon)G(\mathbf{y},\mathbf{z}) \tag{5}$$

with ϵ sampled randomly from the uniform distribution between 0 and 1. Intuitively, the Wasserstein loss can be understood as the discriminator trying to make its output as large as possible for generated samples and as small as possible for real samples. The gradient penalty acts to constrain the discriminator output, which is otherwise unbounded.

As the optimization goals in (1) and (2) are contradictory, D and G must be trained adversarially. We alternated between training D with five batches and G with one, a strategy that was generally found to be beneficial in [37]. We used a batch size of 16, determined by the amount of memory available in the graphics processing unit (GPU). The Adam optimizer [38] was used for most of the optimization, with a learning rate of 10^{-4} for both G and D. We found that Adam converged quickly to reasonable image quality, but the solutions tend to oscillate, even with reduced learning rates. Therefore, near the end of the training after 350 000 training sequences, we switched to stochastic gradient descent (SGD) with a learning rate of 10^{-5}.

The generator was trained with 400 000 sequences, corresponding to 3.2 million individual images, and the discriminator with two million sequences (ten million images). This corresponded to roughly 48 h for each application using an Nvidia P100 GPU. Sample diversity was increased by using random rotation (by 0°, 90°, 180°, or 270°) and random mirroring on the image time series. This makes the GAN approximately invariant with respect to 90° rotations in addition to the translation and time invariance that are features of the network design.

D. Validation

While GANs are expected to converge towards the underlying data distribution of their input data set, frequently [39] they do not reproduce enough variability. There has been progress in quantifying the quality and variability of generated samples for unconditional GANs using metrics, such as the Frechet inception distance (FID) [40], but the FID is not directly applicable to the type of conditional GAN considered here because the training data set generally contains only one output for each input, and therefore, the underlying distribution cannot be reliably estimated.

As a simple metric of image quality, we use the root-mean-square error

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{\mathrm{real},i} - x_{\mathrm{gen},i})^2} \tag{6}$$

where x_i are the individual pixel values of the real image, $x_{\mathrm{gen},i} = G(\mathbf{y}, \mathbf{z})_i$ are the corresponding pixels of the generated image, and N is the number of pixels. To evaluate if the generated images properly reproduce the spatial structure of the true images, we also compute the multiscale structural similarity index (MS-SSIM), as defined in [41], and the log spectral distance (LSD) that gives the difference of the power spectra in decibels (dB)

$$\mathrm{LSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(10\log_{10}\frac{P_{\mathrm{real},i}}{P_{\mathrm{gen},i}}\right)^2} \tag{7}$$

where P_{real} and P_{gen} are the power spectra of the real and generated images, respectively.

For assessing whether the GAN generates the correct amount of variability, we propose to adapt a rank-statistics approach from ensemble weather forecasting [42], [43] to obtain a heuristic measure of the variability of the sequences produced by the conditional GAN. The underlying concept is as follows. For each sample, we have a single "ground truth" (the real high-resolution sequence) and an ensemble of N_{p} predictions (we can generate as many predictions as we wish by reevaluating the GAN with different instances of the noise). Then, for each pixel in the image, we can define the normalized rank of the actual value among all N_{p} predictions as $r = N_{\mathrm{s}}/N_{\mathrm{p}}$, where N_{s} is the number of predictions in the ensemble for which the value of that pixel is smaller than the corresponding ground-truth pixel (the rank is randomized for ties). It is clear that $0 \leq r \leq 1$, and if the sample is from the same distribution as the predictions, r should be uniformly distributed over this range when averaged over many pixels and many sequences. Consequently, we can use the uniformity of the distribution of r as an evaluation metric for the correct variability of the generated images.

The distribution of r can be evaluated visually by examining the histogram of r, as demonstrated in [44]. We can also quantify the uniformity with various distribution distance metrics between the rank distribution P_r and the uniform distribution over the possible values of r (since we take a finite sample of predictions, the possible values are discrete). Here, we investigate several such metrics. First, the Kolmogorov–Smirnov (KS) statistic [45] between two sets of probabilities P and Q is defined as

$$\mathrm{KS} = \sup|C - D| \tag{8}$$

where C and D are the cumulative distribution functions (CDFs) of P and Q, respectively. Second, the Kullback–Leibler divergence (D_{KL}) [46] of P with respect to Q is

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(r_i)\log\left(\frac{P(r_i)}{Q(r_i)}\right) \tag{9}$$

where r_i are the different values that the rank can attain. Unlike KS, D_{KL} is generally not symmetric between P and Q. Typically, P denotes the "ideal" distribution and Q an approximation, so, in this work, we use the uniform distribution for P and the observed rank distribution for Q. As the KS statistic measures the distance of the CDFs and D_{KL} relates to the information content difference of the probabilities, these two statistics capture different aspects of the differences between the rank distribution and the uniform distribution. We also compute the outlier fraction (OF), also called outlier percentage (OP) when given in percent units, which is defined as the fraction of ground-truth samples that lie outside the ensemble of predictions.

Using the complete ensemble, we can also evaluate the image quality with a metric that utilizes the entire ensemble of predictions and the continuous ranked probability score (CRPS) [47]. For a given pixel, CRPS is defined as the integral of the squared difference of the CDF of the ensemble members (denoted as F) and the CDF of the observations. For a single observation (the pixel $x_{\mathrm{real},i}$ from the real image), the observation CDF is a Heaviside step function H shifted to the point $x_{\mathrm{real},i}$, giving CRPS for the pixel i as

$$\mathrm{CRPS} = \int_{-\infty}^{\infty}(F(x') - H(x' - x_{\mathrm{real},i}))^2\mathrm{d}x'. \tag{10}$$

The CRPS for an entire image is obtained as the mean of the pixelwise CRPS scores. CRPS can be understood as a generalization of the mean absolute error (MAE), to which it is reduced if there is only one ensemble member.

In this article, all the abovementioned metrics are calculated for the data transformed to the [0, 1] range, as explained in Section III.

III. Data

To demonstrate that the network can learn the structures of different types of atmospheric fields, we trained it independently with two data sets. The first was a collection of samples

drawn from the MeteoSwiss weather radar composite [48] over the year 2018 (hereafter, referred to as the "MCH-RZC" data set). The samples were selected and processed as described in [49] and released in [50]. The data set contains 180 000 image sequences, each of which consists of eight images of 128×128 pixel size, each pixel corresponding to physical size of 1 km. The time interval between subsequent images in a sequence is 10 min. The image size and the number of images in each sequence were chosen as a compromise between the amount of training data and the available computational resources. The pixel values express the precipitation rate R in units of mm h^{-1}; this has been derived from the radar reflectivity, quality controlled, and corrected for various biases. We preprocessed the RZC data by taking the logarithm of R, which leads to regular distribution, since R is known to have a near-lognormal distribution for $R > 0$ [51], making learning easier. The $R = 0$ case will be discussed later in this section.

The other data set is derived from the cloud optical thickness τ observed by the GOES-16 satellite [52] (we refer to this data set as "GOES-COT" in the rest of this article). We used data from April to December 2019, the period after GOES-16 full-disk scans were switched to Mode 6 that provides data every 10 min (which is only coincidentally the same as with the MCH-RZC data set; any time interval would work). As the cloud optical thickness is only available at daytime and its accuracy can be affected by high solar zenith angles, we limited the data use to hours between 14 and 20 UTC, corresponding to approximately 09 to 15 local solar time at the subsatellite point. From these data, we randomly extracted 108 544 image time series of the same dimensions as the weather radar data. The geometric distortion caused by the Earth's curvature and the satellite point of view was corrected by projecting the data to orthographic projection [53] with a spatial resolution of 2 km per pixel. In order to minimize distortion, the sampling was constrained to a box bounded by 30°S and 30°N latitude and 105°W and 45°W longitude (the center of the longitude range being the subsatellite point at 75°W). As with the precipitation data set, we took the logarithm of τ to make the distribution more even following [54].

The image given to the GAN during training and evaluation is a transformed variant of the variable x [where x can be either $\log(R)$ or $\log(\tau)$]. While the distribution of both variables becomes smoother with the logarithmic transformation, it necessitates special processing in empty (nonprecipitating or noncloudy) regions where the logarithm is not defined. We solve this with the following transformation: empty pixels are mapped to 0, and the detectable range $[x_{\min}, x_{\max}]$ is shifted and scaled to $[\theta, 1]$, thus transforming the entire data set to $[0, 1]$. The threshold θ is a small positive value that separates the nonprecipitating values from the precipitating ones. The transformation is reversible, and consequently, when postprocessing the GAN-generated fields, we consider every pixel with a value below θ as empty, while the values larger than θ are mapped back to x. We used $\theta \approx 0.17$ for both data sets and did not find the results particularly sensitive to the choice of this parameter. To suppress artifacts that would sometimes appear at the sharp edges caused by the thresholding, we smoothen the images with a Gaussian filter before feeding them to the network. This filter also has the effect of inhibiting certain artifacts in the MCH-RZC data set that occasionally result from processing the data from multiple radars into a single composite on a regular Cartesian grid.

Each low-resolution image is obtained from its high-resolution counterpart by taking the average of the linear (not logarithmic) values of R or τ for each nonoverlapping 16×16 pixel tile in the image and then applying the logarithmic transformation and the mapping to $[0, 1]$, as described earlier. Due to the averaging process, some of the averaged pixels may initially have values between 0 and θ; these are truncated to 0 in order to prevent the GAN from taking advantage of data that are invisible in the visualizations.

To ensure that we avoid the scenario where the GAN simply memorizes the training set, we set aside 10% of samples, randomly selected, from each data set to be used as the validation set. The samples from the validation set were not used for training but were used to monitor the progress of the training. Furthermore, to examine how well the GAN generalizes to data that are not sampled from exactly the same data as the training set, we constructed test data sets of 1024 samples for both data sources using data from a different time period. For MCH-RZC, the test data were selected from the year 2017, while, for GOES-COT, they were sampled from the April 1–20, 2020, time period. Except where mentioned otherwise, all visualizations shown in this article were generated using the test sets, ensuring that the examples are from data that the GAN has not been exposed to during training.

IV. Results

A. Examples of Generated Sequences

We show three examples of GAN-reconstructed time series from the MCH-RZC test data set in Fig. 2. These were generated using the generator saved after 361 600 training sequences, selected based on the metrics shown in Sections IV-B and IV-C, as well as a subjective check of the quality and stability of the generated sequences. For each example, Fig. 2 shows the true high-resolution sequence, the 16×16 downsampled sequence, and three different reconstructions. The first example [see Fig. 2(a)] shows a region with different rainfall structures in different parts of the image, with a relatively uniform structure at the top center and a highly spatially variable structure at the bottom. At the top, all three reconstructions produce a similar, uniform structure that strongly resembles the texture of the original. Meanwhile, we see a significant difference in the structure of the cells developing at the bottom where reconstructions #1 and #2 produce much more granular structures than reconstruction #3, which creates a much more uniform structure at the bottom. This example demonstrates how the difference in granularity remains consistent over time: #1 and #2 are more spatially variable than #3 for all time steps. In Fig. 2(b), the precipitation is organized over very short scales everywhere in the image. The structure and orientation of the generated cells vary between the reconstructions. None of the three generated examples captures exactly the orientation of the

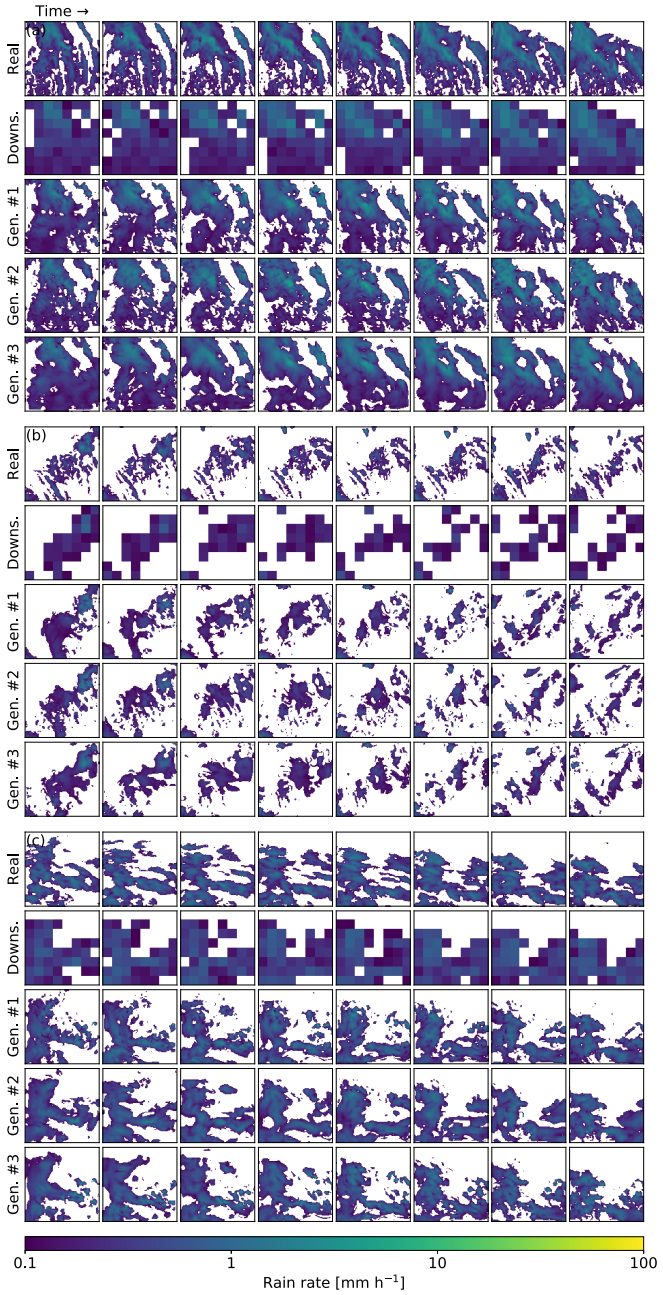

Fig. 2. Examples of reconstructed image sequences from the MCH-RZC test data set. (a)–(c) Real high-resolution image on the first row, the downsampled version on the second row, and three examples of reconstructions created by the GAN on the last three rows.

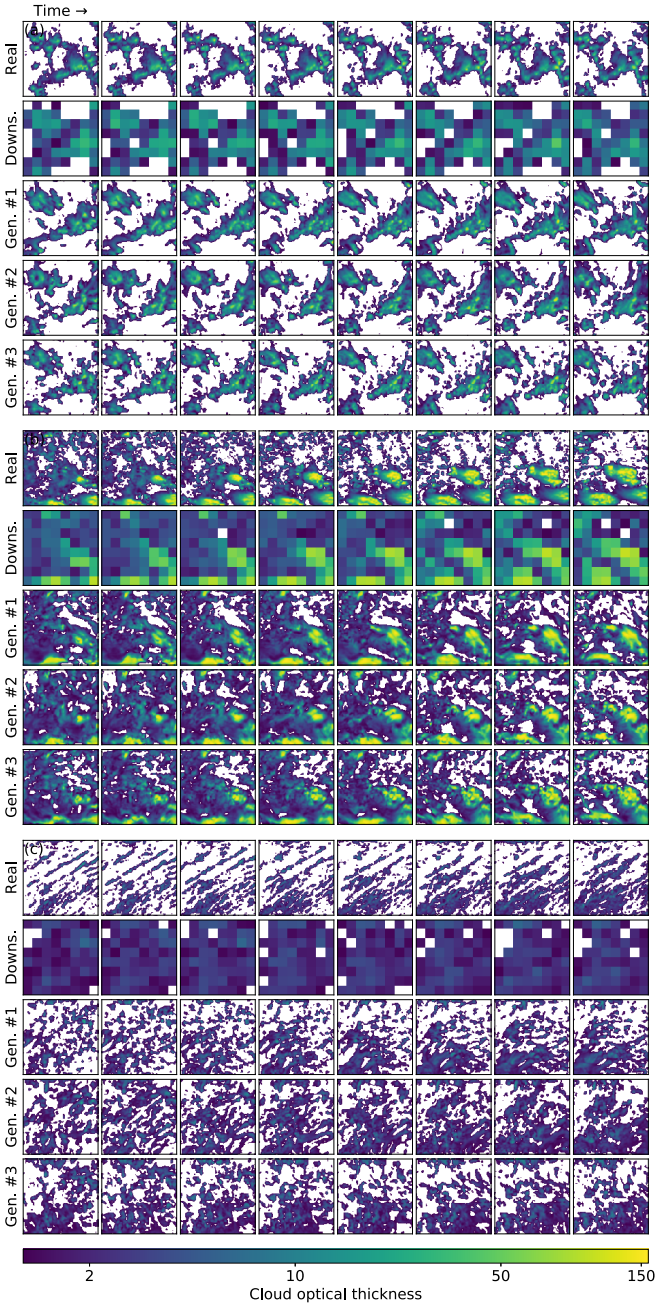

Fig. 3. Examples of reconstructed image sequences from the GOES-COT test data set. (a)–(c) Real high-resolution image on the first row, the downsampled version on the second row, and three examples of reconstructions created by the GAN on the last three rows.

original cells, the information from which is lost in the downsampling process. Regardless, the GAN can clearly infer the type and scale of precipitation cells fairly accurately from the low-resolution image and produce different guesses about the underlying structure. The last example, in Fig. 2(c), shows another complex scene that contains different structures in different parts of the image. Here, too, it can be seen that the GAN can generate different solutions for a given scene: The overall structure is the same in all reconstructions, but the details are quite different.

Fig. 3 displays three examples for the GOES-COT test data set, using the generator obtained after 371 200 training sequences. These data generally have more intricate texture than the MCH-RZC data set, with patterns occurring over shorter scales. This is partially a result of the different spatial resolutions of the data sets, 1 km for MCH-RZC and 2 km for GOES-COT. The case of Fig. 3(a) has very strong contrasts in the cloud optical thickness, sometimes occurring over distances of only a few pixels. These contrasts are lost in the downsampling; regardless, the GAN is able to generate a pattern at an approximately correct scale and spatial structure. The reconstructions differ in terms of the exact location of the generated clouds, reflecting the uncertainty of the GAN about the correct solution. Fig. 3(b) shows another case of highly complex cloud organization

with high COT maxima and strong contrasts over short distances. This example demonstrates the time consistency of the solutions particularly well; for example, the empty regions are in different locations in the different images, but their location remains consistent from one time step to the next. Furthermore, there are differences in the texture of the clouds between the generated images: for example, the high-COT region in the center right of the last few frames contains a cell structure in reconstruction #3, while it is more uniform in reconstructions #1 and #2. Finally, Fig. 3(c) shows a highly anisotropic case where the clouds have a strongly preferred orientation in the original high-resolution image. The GAN has some difficulty inferring the correct orientation, which is lost in the downsampling and generates fairly different solutions to reflect its uncertainty of the correct answer. Some solutions in the corresponding figure in the Supplementary Material (`examples-goescod-random-02.pdf`) include even more strongly oriented clouds although none match the correct solution exactly. The generated clouds in reconstruction #2 exhibit some preferred orientation.

We selected the examples in Figs. 2 and 3 manually in order to illustrate the behavior of the network in different cases. As such, they are a limited and nonrepresentative sample of the data sets. Moreover, it is impossible to convey the full variability of the generated solutions using only the three ensemble members that we are limited to because of space constraints. To address this issue, we have included more examples, randomly selected from the test data sets and with more ensemble members generated with the GAN, in the Supplementary Material available online alongside this article.

B. Reconstruction Quality

To assess the development of image quality as the GAN is trained, we computed the RMSE, MS-SSIM, LSD, and CRPS metrics, as described in Section II-D, at intervals of 3200 generator training sequences. All of these metrics were calculated for the data transformed to the $[0, 1]$ range, as explained in Section III. The evolution of the average of these metrics over a sample drawn from the validation data set is shown in Fig. 4. The numbers for the fully trained GAN are shown in Table I for both the test and validation data sets.

The RMSE and MS-SSIM metrics improve rapidly in the first 15 000 generator training sequences, converging quickly to a near-equilibrium. After this, there is little improvement in these scores. LSD keeps improving considerably longer especially for the MCH-RZC data set, showing signs of improvement until approximately 70 000 sequences. The CRPS metric, which utilizes all ensemble members, keeps improving longer than the single-image metrics but with much more noise. After the switch to the SGD optimizer, the noise in the single-image metrics (but not the CRPS) is reduced, but the switch seems to have almost no effect on the metrics except for a slight degradation in the MS-SSIM metric for the GOES-COT data set just after the switch.

Our subjective assessment of the generated image quality indicated that the quality keeps increasing for longer than the single-image metrics indicate, until at least 100 000 sequences.

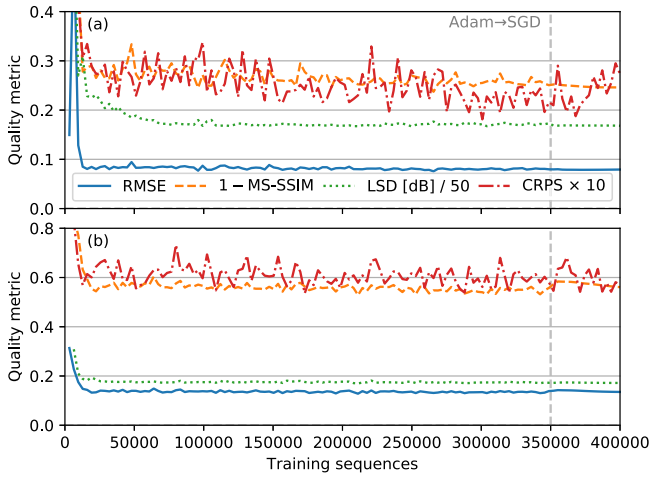

Fig. 4. Metrics of the image quality from the GAN-generated ensemble. The blue solid line shows the RMSE, the orange dashed line shows 1−MS-SSIM, the green dotted line shows the LSD (divided by 50 to bring it to a similar scale as the other metrics), and the red dashed-dotted line shows the CRPS multiplied by 10. (a) Results for the MCH-RZC validation data set. (b) Results for the GOES-COT validation data set.

We believe that the poor performance of the metrics is caused by them not capturing the desired qualities of the super-resolution reconstruction particularly well. The RMSE, in particular, is minimized at the mean of possible solutions and, therefore, is of limited use in assessing the performance of GANs. The MS-SSIM is affected by similar issues because the objective of the GAN is to generate an ensemble of plausible solutions, and only a small fraction of those can be expected to be a close match to the original. For instance, when precipitation consists of small convective cells, the GAN might create cells of the correct size and intensity but in slightly wrong locations, leading to poor metrics in spite of perceptual similarity. The LSD, which compares the power spectra, does capture some of the structure but taking the power spectrum loses information about the location of the signals. The CRPS appears promising for evaluating conditional GANs as it detects improvement for much longer than the other metrics.

C. Variability

In Fig. 5, we show the evolution of the variability metrics of the GAN over time during the training, evaluated using the validation data set using 100 ensemble members for each validation sample. We consider the KS statistic, D_{KL} and OF as defined in Section II-D, and also plot the bias of the mean rank from the optimal value of $1/2$. During the training using the Adam optimizer, the metrics improve rapidly at first, and slow improvement continues for much longer than with the single-image quality metrics discussed in the previous section. Improvement continues until at least 300 000 sequences. After the switch to the SGD optimizer at approximately 350 000 training sequences, the oscillation in the metrics is reduced.

The variability metrics for the fully trained GAN are shown in Table I alongside the quality metrics. The metrics near the end of training indicate that the rank distribution is close to uniform. At the time steps used in Section IV-A, the KS statistic indicates that the CDF of the rank distribution differs

TABLE I

IMAGE QUALITY AND VARIABILITY METRICS COMPUTED FOR THE TEST AND VALIDATION SETS FOR THE TRAINED GAN. (TOP) METRICS FOR THE VALIDATION AND TESTING SETS OF BOTH THE MCH-RZC AND THE GOES-COT DATA SETS. (BOTTOM) METRICS FOR DIFFERENT METHODS USING THE MCH-RZC TEST SET, BOLD NUMBERS DENOTING THE BEST METHOD FOR EACH METRIC

	RMSE	MS-SSIM	LSD (dB)	CRPS	KS	D_{KL}	OF	Mean rank
GAN, MCH-RZC valid.	0.079	0.750	8.445	0.020	0.029	0.014	0.046	0.502
GAN, MCH-RZC test	0.097	0.680	8.365	0.029	0.040	0.024	0.056	0.501
GAN, GOES-COT valid.	0.140	0.422	8.652	0.054	0.059	0.046	0.073	0.494
GAN, GOES-COT test	0.133	0.456	8.817	0.061	0.052	0.044	0.073	0.506
GAN, MCH-RZC test	0.097	0.680	**8.365**	**0.029**	**0.040**	**0.024**	**0.056**	**0.501**
Lanczos, MCH-RZC test	0.092	0.617	18.700	—	—	—	—	—
RCNN, MCH-RZC test	**0.076**	**0.683**	23.268	—	—	—	—	—
RainFARM, MCH-RZC test	0.243	0.134	16.484	0.131	0.202	0.318	0.294	0.516

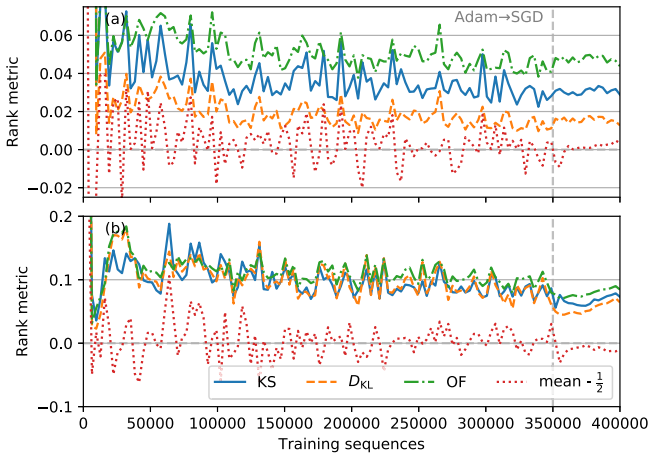

Fig. 5. Metrics of the rank distribution (as defined in Section II-D) of ground-truth images in the GAN-generated ensemble, shown as a function of training samples given to the generator. The blue solid line shows the KS statistic, the orange dashed line shows the Kullback–Leibler divergence, the green dashed-dotted line shows the OF, and the red dotted line shows the difference of the mean rank and 1/2. (a) Results for the MCH-RZC validation data set. (b) Results for the GOES-COT validation data set.

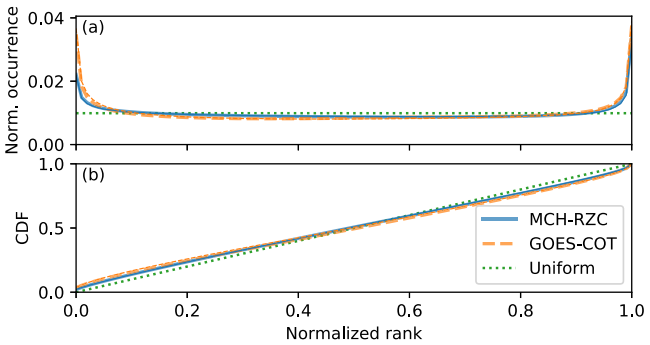

Fig. 6. (a) Occurrence of normalized ranks for the trained GAN (using the same generator weights as in Section IV-A). The blue solid lines correspond to the MCH-RZC data set and the orange dashed lines to the GOES-COT data set. The thick, lighter-colored lines show the results for the test data set, while the thin, darker lines show the results for the validation data set. The green dotted line shows the uniform distribution for comparison. (b) As (a), but showing the CDFs of the distributions.

from the CDF of the uniform distribution by at most 0.029 for the MCH-RZC data set and at most 0.059 for the GOES-COT data set. This suggests that, at least in this respect, the GAN generates close to the appropriate amount of variability in its outputs although there is clearly some difference in the distributions, and therefore, the actual KS test for equal distributions would fail. The similarity to and differences from the uniform distribution can also be seen visually in Fig. 6 where we show the rank distribution graphically. The visualization shows that while there are considerably more samples in the outlier ranks (r of either 0 or 1) than in the ranks near the middle of the distribution, these outliers represent only a minor fraction of all ranks (as also demonstrated by the OF in Table I). In a clear majority of cases, the real sample falls within the ensemble of predictions.

We also experimented with tuning the noise amplitude, which was noted in [20] to increase the variability of the generated fields. We tried different multiplication factors for the noise, ranging from 0.5 to 3.0. We found that, for inadequately trained generators, noise adjustment could significantly improve the variability metrics. On the other hand, for the models trained to near-convergence, the optimal adjustment factors were rather close to 1, ranging between 0.9 and 1.1 depending on the data set and the metric. Given that the difference is minor and that there is no clear theoretical justification for this *ad hoc* adjustment, we do not apply any adjustment to the noise amplitude in the final results.

D. Comparison to Alternative Methods

In Fig. 7, we show a comparison of our GAN-based method to alternative techniques: Lanczos interpolation, a recurrent CNN (RCNN) trained to optimize RMSE, and the rainfall filtered autoregressive model (RainFARM) algorithm of [13]. These represent conceptually different approaches to the downscaling problem. Lanczos interpolation is a traditional, widely used image scaling method and is used here as a baseline case. The RCNN trained with the RMSE loss is an example of a more straightforward deep-learning approach; in order to provide a fair comparison to the GAN, the RCNN uses the same architecture as our GAN generator, except with the noise input disabled. Finally, RainFARM is a downscaling method developed specifically for rainfall using more traditional statistical techniques based on the Gaussian random fields generated using power-law spectral scaling. RainFARM, such as the GAN, can be used stochastically to generate multiple realizations of the random field, while the RCNN and Lanczos methods are deterministic.

The examples illustrate that GAN produces more detail and more visually accurate reconstruction of the original image than the alternative methods. The Lanczos interpolation and the RMSE-trained RCNN both produce a smooth output but with little detail at smaller scales. We also tried training the RCNN using the MAE loss, but the results (not shown) were

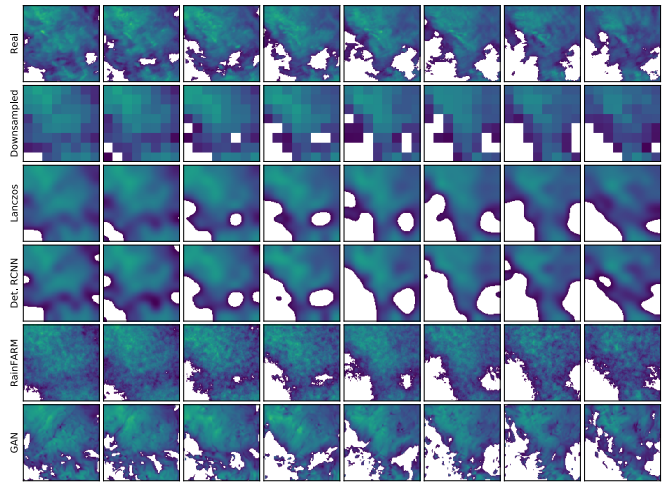

Fig. 7. Comparison of our GAN-based method to alternative methods. The first row shows the real high-resolution sequence, and the second row shows the downsampled version. The subsequent rows show the different reconstruction methods: Lanczos interpolation (third row), RCNN trained to optimize RMSE (fourth row), the RainFARM algorithm (fifth row), and our GAN (sixth row).

very similar to RMSE. The RainFARM algorithm can produce more small-scale detail than the previous two methods, but it is limited to producing the same texture everywhere in the image and does not reproduce the structure of the high-resolution field as accurately as the GAN. Moreover, the example shown in Fig. 7 is one where RainFARM performs relatively well. As Rebora *et al.* [13] note, RainFARM is quite sensitive to the choice of the scaling exponent, and we found that, in some cases, the textures produced were considerably less realistic than in this example as a result of a poorly estimated exponent. The GAN, on the other hand, works quite robustly and very rarely generates any implausible artifacts.

The performance metrics for the various methods are shown in the bottom half of Table I. These are consistent with what is shown in Fig. 7. The GAN, Lanczos, and RCNN methods give similar results for the RMSE and MS-SSIM metrics, which further demonstrates that these are not particularly good metrics for evaluating GAN performance as they penalize solutions with higher variance. The RCNN achieves the best RMSE metric, which is unsurprising as it was specifically trained to optimize this metric, and it also gives the best MS-SSIM score. With the LSD, the GAN achieves the best score by far, while RainFARM, which produces a detailed texture, performs better than the Lanczos and the RCNN that produce unrealistically smooth outputs. In the ensemble metrics, the GAN clearly outperforms RainFARM, while these scores cannot be evaluated for the deterministic methods.

In terms of computational resources, evaluating the GAN generator (and, by extension, the RMSE-trained RCNN, which uses the same architecture) for one sequence of eight 128 $\times$ 128 pixel images took approximately 660 ms on a modern quad-core Intel i7 central processing unit (CPU) and 20 ms on an Nvidia P100 GPU. These times were obtained with a batch size of 16; using a batch size of 1 instead increased the evaluation time per sequence by approximately a factor of 2 on both the CPU and the GPU (TensorFlow parallelization becomes less efficient with smaller batches). By comparison, the Lanczos interpolation took 11 ms seconds/sequence, and the RainFARM algorithm, using a fairly unoptimized implementation, took 240 ms/sequence. The latter two methods were evaluated only on the CPU. This performance comparison demonstrates that the GAN method is relatively resource-intensive, but evaluating the GAN for modest amounts of input data is possible in a reasonable amount of time also on a CPU, while a GPU is desirable for bulk processing large amounts of data.

E. Generalization to Larger Images and Longer Sequences

Since the GAN architecture is fully convolutional, we can apply the generator trained with relatively small (in our case, 128 $\times$ 128 pixel) inputs to fields of different sizes without any modifications. The only restriction is that the pixels should correspond to the same physical size as the pixels of the training sequences, and that pixel dimensions of the input must be divisible by the resolution enhancement factor of 16. Similarly, the recurrent structure allows us to apply the generator to longer or shorter sequences than the training sequences of length 8 as long as the time interval between the frames of the sequence is the same as that used for training.

We demonstrated this capability by applying the generator (using the same version as in Section IV-A) to the data from the June–August 2017 archive of full frames of the MCH-RZC data at 10-min time intervals. These data are from a different year than the training set and, thus, are completely independent. The frames in the data are 710 pixels wide and 640 pixels high; the width was cropped to 704 pixels to satisfy the requirement that the dimensions be divisible by 16. The generator was applied sequentially to each frame; the hidden state of the ConvGRU layer was propagated to the following frame at each step. For the first step, and wherever there is a longer than 10-min time gap between frames (which occasionally happens due to missing data), we used the initialization network to reinitialize the ConvGRU state, thus interrupting the time consistency in these situations.

We show one frame of the generated sequence in Fig. 8. This example shows a situation with different modes of precipitation in different regions. It demonstrates that the GAN can create realistic reconstructions even for much larger images than those from the training set. The time evolution of the generated fields can obviously not be properly illustrated with a single image, so we provide an animation that shows the June–August 2017 sequence as a video accompanying this article online in the Supplementary Material.

While generating these long time series, we found that some versions of the generator could produce artifacts when left running for a long time. For the purposes of generating Fig. 8 and the corresponding video, we were able to suppress these artifacts sufficiently by adjusting the generator architecture and choosing a version of the generator that was less prone to them. However, for those cases where the artifacts cannot be avoided, we found a simple stabilization method, which we describe in the Appendix.

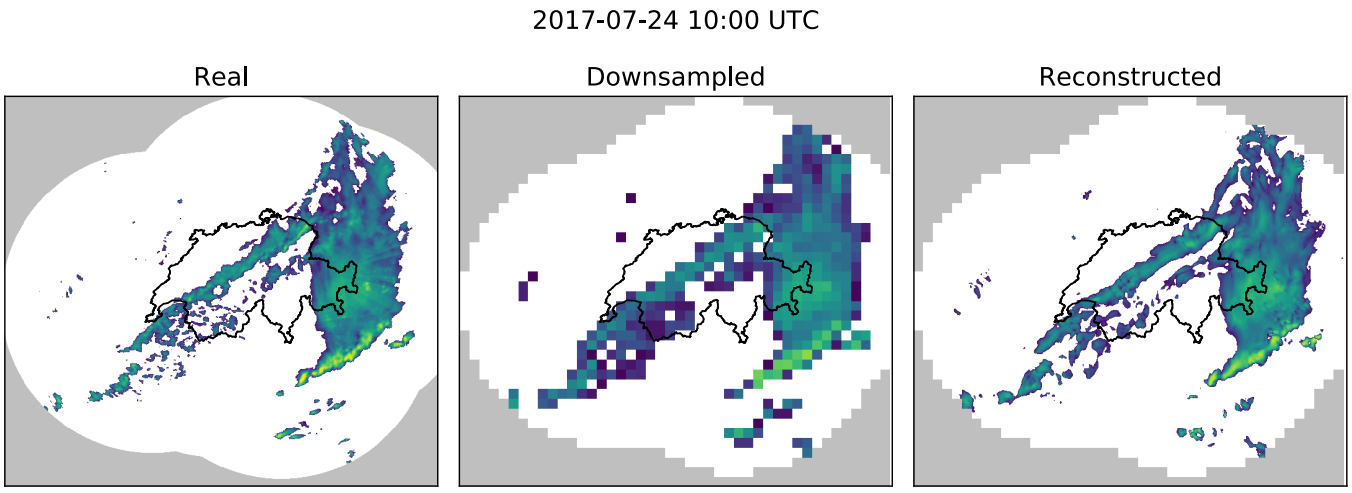

Fig. 8. Example of the results of the GAN applied to full frames of the June–August 2017 data from the MCH-RZC data set, showing the situation of July 24 at 10:00 UTC. The gray areas mask the points that are unavailable due to a lack of radar coverage. The borders of Switzerland are shown in the middle in order to provide spatial context. (Left) Original frame. (Middle) Downsampled version fed to the generator. (Right) High-resolution frame reconstructed by the GAN.

V. Conclusion and Summary

Deep learning has enabled significant advances in image and video super-resolution, with GANs being among the most prominent methods. Resolution enhancement also has many applications in the processing of observational and model data in the weather and climate sciences. However, in weather and climate applications, uncertainty quantification is essential. This work addresses this need with a conditional super-resolution GAN that operates on sequences of 2-D images and creates an ensemble of predictions for each input. The spread between the ensemble members represents the uncertainty of the super-resolution reconstruction.

Rather than processing each image in a sequence independently, our generator architecture uses a recurrent layer to update the state of the high-resolution reconstruction in a manner that is consistent with both the previous state and the newly received data. The recurrent layer can, thus, be understood as performing a Bayesian update on the ensemble member, resembling an ensemble Kalman filter. Besides being recurrent, the generator is fully convolutional, meaning that it can operate on variable-sized inputs and produce consistent time evolution for arbitrarily long sequences.

The representativeness of the ensemble was quantitatively evaluated using ensemble statistics. We found that rank metrics take longer to converge than image quality metrics, such as MS-SSIM and RMSE, and therefore, the rank metrics can be used to monitor the progress of the training even after image quality metrics saturate. The CPRS quality metric, which uses the entire ensemble, also appears to provide a better estimate of image quality than the single-image metrics. The ensemble metrics, therefore, seem promising for evaluating the quality and variability produced by conditional GANs in general and may be useful in applications beyond the geoscience domain.

The evaluation of the GAN indicates that it produces realistic high-resolution fields with appropriate amounts of variability. Moreover, the GAN was trained separately for two distinct applications, proving that it can generalize to different types of input data. We expect that it can be applied to other similar applications as well. The GAN generator also generalizes well to larger input images and longer sequences than those in the training set, reducing the computational cost of training as the GAN can be trained with relatively short sequences of small images and then evaluated with sequences of different length and image size.

Besides increasing the range of applications, potential future improvements include the following.

1) Optimization of the network architecture for performance and memory footprint using rigorous ablation analysis.
2) Generalization of different scaling factors or possibly producing high-resolution images for multiple scaling factors at once (the current version is specific to the factor of 16).
3) Resolution enhancement in the temporal in addition to the spatial dimensions to allow time interpolation.
4) Extrapolation of the time series to allow short-term predictions into the future, possibly in conjunction with additional methods, such as optical flow.
5) The inclusion of auxiliary variables to help the generator produce the right kind of fields; for instance, orography affects precipitation formation and could be included as an additional variable, as was previously done in a deep-learning context in [55].
6) Further development of the rank-based methods for evaluating conditional GANs. In particular, the ensemble metrics in this article were evaluated pixelwise, but it may be possible to develop a more feature-based method similar to the FID.

Appendix
Optional Stabilization for Long Time Series

We found that some versions of the generator were prone to generating artifacts when left running recurrently for many time steps. In these cases, the generator was stable over the

eight frames used in the training, but this was apparently not always sufficient to guarantee stability over longer periods of time. While we were able to avoid this in our reported experiments, as described in Section IV-E, we found a relatively simple technique to suppress the artifacts when they appear. We report it here as it may be useful for further experiments with such recurrent GANs.

As the initialization network did not produce any artifacts, we were able to use the following procedure to stabilize the evaluation of the generator. On each time step k, after evaluating the update network, the ConvGRU state h_k is adjusted as follows:

$$h_k := h_{\text{null}} + (1 - \lambda_r)(h_k - h_{\text{null}}) \tag{11}$$

where h_{null} is the ConvGRU state produced by the initialization network for an all-zeros input, and λ_r is a relaxation constant (we experimented with $0.01 \leq \lambda_r \leq 0.2$ for the MCH-RZC data set). This process nudges the ConvGRU state toward the null state. This effectively suppresses artifacts while still allowing the update network to operate on the state from the previous step. This procedure seems to reduce (but not completely eliminate) the variability present in the generated images. Therefore, while it serves to stabilize the evaluation over long periods of time, it should only be used when the artifacts cannot be removed using improvements to the generator network.

ACKNOWLEDGMENT

The authors would like to thank MeteoSwiss for providing the MCH-RZC data and Dr. D. Wolfensberger for assisting us with using it.

REFERENCES

[1] P. Milanfar, Ed., *Super-Resolution Imaging*. Boca Raton, FL, USA: CRC Press, 2011.

[2] D. Maraun and M. Widmann, *Statistical Downscaling and Bias Correction for Climate Research*. Cambridge, U.K.: Cambridge Univ. Press, 2018.

[3] A. W. Wood, L. R. Leung, V. Sridhar, and D. P. Lettenmaier, "Hydrologic implications of dynamical and statistical approaches to downscaling climate model outputs," *Climatic Change*, vol. 62, nos. 1–3, pp. 189–216, Jan. 2004, doi: 10.1023/B:CLIM.0000013685.99609.9e.

[4] H. J. Fowler, S. Blenkinsop, and C. Tebaldi, "Linking climate change modelling to impacts studies: Recent advances in downscaling techniques for hydrological modelling," *Int. J. Climatol.*, vol. 27, no. 12, pp. 1547–1578, Oct. 2007, doi: 10.1002/joc.1556.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016, doi: 10.1109/TPAMI.2015.2439281.

[6] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1646–1654, doi: 10.1109/CVPR.2016.182.

[7] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision—ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 694–711, doi: 10.1007/978-3-319-46475-6_43.

[8] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690, doi: 10.1109/CVPR.2017.19.

[9] X. Wang *et al.*, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proc. Comput. Vis. Workshops*, L. Leal-Taixe and S. Roth, Eds. Cham, Switzerland: Springer, 2019, pp. 63–79, doi: 10.1007/978-3-030-11021-5_5.

[10] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014, pp. 2672–2680. [Online]. Available: https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[11] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410, doi: 10.1109/CVPR.2019.00453.

[12] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2337–2346, doi: 10.1109/CVPR.2019.00244.

[13] N. Rebora, L. Ferraris, J. von Hardenberg, and A. Provenzale, "RainFARM: Rainfall downscaling by a filtered autoregressive model," *J. Hydrometeorol.*, vol. 7, no. 4, pp. 724–738, Aug. 2006, doi: 10.1175/JHM517.1.

[14] S. Terzago, E. Palazzi, and J. von Hardenberg, "Stochastic downscaling of precipitation in complex orography: A simple method to reproduce a realistic fine-scale climatology," *Natural Hazards Earth Syst. Sci.*, vol. 18, no. 11, pp. 2825–2840, Nov. 2018, doi: 10.5194/nhess-18-2825-2018.

[15] S. Lovejoy and D. Schertzer, "Multifractals, cloud radiances and rain," *J. Hydrol.*, vol. 322, nos. 1–4, pp. 59–88, May 2006, doi: 10.1016/j.jhydrol.2005.02.042.

[16] H. Chen, X. Zhang, Y. Liu, and Q. Zeng, "Generative adversarial networks capabilities for super-resolution reconstruction of weather radar echo images," *Atmosphere*, vol. 10, no. 9, p. 555, Sep. 2019, doi: 10.3390/atmos10090555.

[17] W. Ma, Z. Pan, J. Guo, and B. Lei, "Super-resolution of remote sensing images based on transferred generative adversarial network," in *Proc. IGARSS-IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2018, pp. 1148–1151, doi: 10.1109/IGARSS.2018.8517442.

[18] K. Jiang, Z. Wang, P. Yi, G. Wang, T. Lu, and J. Jiang, "Edge-enhanced GAN for remote sensing image superresolution," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5799–5812, Aug. 2019.

[19] K. Stengel, A. Glaws, D. Hettinger, and R. N. King, "Adversarial super-resolution of climatological wind and solar data," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 29, pp. 16805–16815, Jul. 2020, doi: 10.1073/pnas.1918964117.

[20] J. Leinonen, A. Guillaume, and T. Yuan, "Reconstruction of cloud vertical structure with a generative adversarial network," *Geophys. Res. Lett.*, vol. 46, no. 12, pp. 7035–7044, Jun. 2019, doi: 10.1029/2019GL082532.

[21] D. J. Gagne, H. M. Christensen, A. C. Subramanian, and A. H. Monahan, "Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz'96 model," *J. Adv. Model. Earth Syst.*, vol. 12, no. 3, pp. 1–20, Mar. 2020, doi: 10.1029/2019MS001896.

[22] S. Scher and S. Peßenteiner, "Temporal disaggregation of spatial rainfall fields with generative adversarial networks," *Hydrol. Earth Syst. Sci. Discuss.*, to be published, doi: 10.5194/hess-2020-464.

[23] A. Lucas, S. Lopez-Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3312–3327, Jul. 2019, doi: 10.1109/TIP.2019.2895768.

[24] X. Wang, A. Lucas, S. Lopez-Tapia, X. Wu, R. Molina, and A. K. Katsaggelos, "Spatially adaptive losses for video super-resolution with GANs," in *Proc. ICASSP-IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 1697–1701.

[25] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: https://arxiv.org/abs/1411.1784

[26] D. Foster, *Generative Deep Learning*. Newton, MA, USA: O'Reilly Media, 2019.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[29] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proc. SSST-8, 8th Workshop Syntax, Semantics Struct. Stat. Transl.*, 2014, pp. 103–111.

[30] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015, pp. 802–810. [Online]. Available: http://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf

[31] L. Tian, X. Li, Y. Ye, P. Xie, and Y. Li, "A generative adversarial gated recurrent unit model for precipitation nowcasting," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 4, pp. 601–605, Apr. 2020, doi: 10.1109/LGRS.2019.2926776.

[32] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement.*, 2016, pp. 265–283.

[33] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://keras.io

[34] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–27. [Online]. Available: https://arxiv.org/abs/1802.05957

[35] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Workshop Deep Learn. Audio, Speech Lang. Process.*, 2013, p. 3.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 5767–5777. [Online]. Available: https://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf

[37] K. Kurach, M. Lučić, X. Zhai, M. Michalski, and S. Gelly, "A large-scale study on regularization and normalization in GANs," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds. Long Beach, CA, USA: Proceedings of Machine Learning Research, Jun. 2019, pp. 3581–3590.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2014, pp. 1–15. [Online]. Available: https://arxiv.org/abs/1412.6980

[39] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. 6th Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, 2018, pp. 1–25. [Online]. Available: https://openreview.net/forum?id=Hk99zCeAb

[40] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 6626–6637. [Online]. Available: http://papers.nips.cc/paper/7240-gans-trained-by-a-two-time-scale-update-rule-converge-to-a-local-nash-equilibrium.pdf

[41] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. 37th Asilomar Conf. Signals, Syst. Comput.*, 2003, pp. 1398–1402, doi: 10.1109/ACSSC.2003.1292216.

[42] O. Talagrand, R. Vautard, and B. Strauss, "Evaluation of probabilistic prediction systems," in *Proc. ECMWF Workshop Predictability*, 1997, pp. 1–25. [Online]. Available: https://www.ecmwf.int/node/12555

[43] G. Candille and O. Talagrand, "Evaluation of probabilistic prediction systems for a scalar variable," *Quart. J. Roy. Meteorol. Soc.*, vol. 131, no. 609, pp. 2131–2150, Jul. 2005, doi: 10.1256/qj.04.71.

[44] T. M. Hamill, "Interpretation of rank histograms for verifying ensemble forecasts," *Monthly Weather Rev.*, vol. 129, no. 3, pp. 550–560, 2001, doi: 10.1175/1520-0493(2001)129<0550:IORHFV>2.0.CO;2.

[45] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, *A Modern Introduction to Probability and Statistics: Understanding Why and How*. London, U.K.: Springer, 2005.

[46] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, Mar. 1951, doi: 10.1214/aoms/1177729694.

[47] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *J. Amer. Stat. Assoc.*, vol. 102, no. 477, pp. 359–378, Mar. 2007, doi: 10.1198/016214506000001437.

[48] U. Germann, M. Boscacci, M. Gabella, and M. Sartori, "Peak performance: Radar design for prediction in the Swiss Alps," *Meteorol. Technol. Int.*, pp. 42–45, Apr. 2015. [Online]. Available: https://www.researchgate.net/publication/310740877_Peak_Performance_Radar_design_for_prediction_in_the_Swiss_Alps

[49] J. Leinonen, T. Yuan, and A. Berne, "Generative adversarial network for climate data field generation," in *Proc. 9th Int. Workshop Climate Inform.*, 2019, pp. 37–42.

[50] J. Leinonen, "Weather radar observation dataset for machine learning," Harvard Dataverse, V1, 2019, doi: 10.7910/DVN/ZDWWMG.

[51] B. Kedem and L. S. Chiu, "On the lognormality of rain rate," *Proc. Nat. Acad. Sci. USA*, vol. 84, no. 4, pp. 901–905, Feb. 1987, doi: 10.1073/pnas.84.4.901.

[52] A. K. Heidinger *et al.*, "ABI cloud products from the GOES-R series," in *The GOES-R Series: A New Generation of Geostationary Environmental Satellites*, S. J. Goodman, T. J. Schmit, J. Daniels, and R. J. Redmon, Eds. Amsterdam, The Netherlands: Elsevier, 2020, ch. 6, pp. 43–62.

[53] J. P. Snyder, "Map projections—A working manual," United States Government Printing Office, Washington, DC, USA, U.S. Geol. Survey Prof. Paper 1395, 1987.

[54] J. Leinonen, M. D. Lebsock, G. L. Stephens, and K. Suzuki, "Improved retrieval of cloud liquid water from CloudSat and MODIS," *J. Appl. Meteorol. Climatol.*, vol. 55, no. 8, pp. 1831–1844, Aug. 2016, doi: 10.1175/JAMC-D-16-0077.1.

[55] G. Franch, D. Nerini, M. Pendesini, L. Coviello, G. Jurman, and C. Furlanello, "Precipitation nowcasting with orographic enhanced stacked generalization: Improving deep learning predictions on extreme events," *Atmosphere*, vol. 11, no. 3, p. 267, Mar. 2020, doi: 10.3390/atmos11030267.

Jussi Leinonen received the Dr. Tech. Sci. degree from Aalto University, Espoo, Finland, in 2013. He performed his doctoral research at the Finnish Meteorological Institute, Helsinki, Finland. Subsequently, he was a Post-Doctoral Scholar and later a Data Scientist at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, from 2014 to 2019.

In April 2019, he joined Environmental Remote Sensing Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. Since October 2020, he has been with the Radar, Satellite, and Nowcasting Division, MeteoSwiss, Locarno-Monti, Switzerland. His research interests include machine learning in the atmospheric sciences, precipitation radars, electromagnetic scattering, cloud and precipitation microphysics, and probabilistic data analysis.

Daniele Nerini received the M.S. degree from Imperial College London, London, U.K., in 2013, and the Ph.D. degree from ETH Zurich, Zurich, Switzerland, in 2019. He performed his doctoral research in the Radar, Satellite, and Nowcasting Division, MeteoSwiss, Locarno-Monti, Switzerland.

He is a Research Associate with Forecast Development Division, Swiss Federal Office for Meteorology and Climatology MeteoSwiss. His research focuses on radar hydrology, precipitation nowcasting, and the verification and postprocessing of numerical weather predictions.

Alexis Berne received the Ph.D. degree from Université Joseph Fourier, Grenoble, France, in 2002.

From 2003 to 2006, he was a Marie Curie Fellow with Wageningen University, Wageningen, The Netherlands. Since 2006, he has been leading Environmental Remote Sensing Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. His research interests include radar meteorology in mountainous and polar regions.