ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

MeteoSwiss

Alexandra Butoi, Matus Zilinec, Shubhangi Ghosh

Post-processing of Numerical Rainfall Forecasts Using Generative Adversarial Networks

Data Science Lab

Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Daniele Nerini, Jonas Bhend
Prof. Dr. Ce Zhang

January 2021

Contents

Abstract

Numerical Weather Prediction (NWP) models are the current standard of the advances in weather forecasting. Such physics-based models, however, are not free of errors due to the finite spatio-temporal resolution and imperfect model formulations. We treat the post-processing of weather forecasts as an image processing problem. We use conditional Generative Adversarial Networks (cGANs) to post-process NWP precipitation forecasts, in an effort to reduce some of their systematic biases and thus, increase their performance as probabilistic forecasts (with respect to resolution and realism). We evaluate the quality of the ensemble forecats produced by our model using Root Mean Squared error (RMSE), Log Spectral Distance and wetness ratio. We find that our model produces realistic forecasts and improves the quality of the generated images in terms of RMSE but it tends to overestimate the amount of rainfall.

Chapter 1

Introduction

Weather forecasting is the application of the principles of physics in order to predict the future state of the atmosphere for a given location and time. Due to the wide range of uses for weather forecasts (e.g. in agriculture, aviation, etc.), people have tried to predict the weather for millenia [3].
Modern weather forecasting is carried out using Numerical Weather Prediction (NWP), which takes the current weather conditions (initial conditions) and processes them using physics-based models (based on partial differential equations) of the atmosphere in order to determine future states [2]. The output of a NWP system is the forecast for a discrete grid of points. For this reason, the finite spatio-temporal resolution, as well as the model misspecification, make the NWP forecasts prone to systematic biases. For instance, the NWP forecats often look blockish when compared to the observation-based estimates combining radar and in-situ measurements, which exhibit smaller-scale structures which often differ in location and intensity. Fortunately, there has been some progress made towards correcting these biases using a variety of methods, ranging from simple bias corrections to sophisticated methods based on deep learning. This task is called post-processing and it aims to make any adjustments to the output of the NWP system in order to alleviate or eliminate these inconsistencies for better future predictions [8].
Current NWP systems produce probabilistic forecasts instead of a single deterministic forecast in order to obtain better predictions. A set of forecasts (an ensemble) is produced using the same model and taking as input slightly different initial conditions and stochastic perturbations. The ensemble forecats are produced in order to account for the errors introduced by the imperfect initial conditions (which might come from measurement errors), as well as the errors introduced by the imperfect model formulation. Ideally, the output of the post-processing method should verify well when evaluated as a probabilistic weather forecast.
Forecast verification is the process of evaluating the quality of an ensemble forecast [1]. In particular, the NWP post-processing should "maximize sharpness subject to calibration". This means that the model should make predictions with as high confidence as possible while still being accurate.
In this project, we perform post-processing of ensembles of precipitation forecasts for the area of Switzerland, using real data produced (NWP forecats) and collected (observations) by MeteoSwiss. We employ deep learning methods, namely generative adversarial networks (GANs) in order to post-process the NWP precipitation forecasts and produce ensemble forecasts that look physically realistic and perform well as probabilistic forecasts.
The rest of the paper is organized as follows: Chapter 2 introduces the datasets used for this project, Chapter 3 describes the evaluation measures, Chapter 4 details on the methods that were used, including the neural network architectures, hyperparameters and other training details, Chapters 5 presents and discusses the main findings and Chapter 6 concludes with a summary of the main achievements and results.

Chapter 2

Data

The post-processing of an ensemble forecast requires access to an ensemble of NWP forecasts, paired with observed values. We use 2 datasets provided by MeteoSwiss for the interval May, 2016 - September, 2020, namely COSMO-E (forecast) and CombiPrecip (observations).

2.1 Observed Precipitation Data - CombiPrecip

Combiprecip is a derived observation dataset, combining radar measurements with in-situ rainfall measurements from observation stations. While radar provides great spatial coverage and thus allows to measure precipitation even away from observation stations, inferring precipitation intensity from radar is somewhat unreliable in particular in complex topography. In particular, the radar beam is shadowed by nearby mountains and lower-lying areas are shaded from âviewâ. This shadowing can affect quantitative precipitation estimates, in particular when precipitation formation happens close to the ground. To mitigate some of the disadvantages of radar-based precipitation estimates, MeteoSwiss has developed CombiPrecip. Using all available rainfall measurements in the domain, the radar images are calibrated to match station-based rainfall intensities. As such, CombiPrecip represents the best observation-based estimate of the spatially comprehensive, short-term rainfall evolution available to MeteoSwiss.
CombiPrecip is archived on Swiss coordinates (CH1903 / LV03[1]). Precipitation is expressed in accumulated precipitation in mm over one hour, where the timestamp (always expressed as UTC) represents the end-of-accumulation time. The data is recorded hourly from May, 2016 until September, 2020.

2.2 Numerical Weather Prediction (NWP) Data - COSMO-E

The Numerical Weather Prediction (NWP) data is made available in the dataset COSMOE[2]. The COSMO-E NWP system is run on a rotated latitude-longitude lattice. The north pole is rotated to -170E 43N (in standard latitude/longitude coordinates) in order to minimize distortion over the region of interest (Switzerland). That is, the equator in the rotated coordinate system passes through the domain of interest. The subregion of the COSMO-E domain archived for analysis is shown in Figure 2.3.
COSMO-E provides NWP forecasts for a 2km x 2km grid over Switzerland, at a hourly temporal resolution. The model is run twice a day (at 00:00 and 12:00), with different initial conditions, offering forecasts for each hour for up to 5 days into the future. Each forecast is an ensemble made of 21 members: a 'control run', in which the NWP model is initialized with the actual observed

[1] https://epsg.io/21781
[2] https://www.meteoswiss.admin.ch/home/measurement-and-forecasting-systems/warning-and-forecasting-systems/cosmo-forecasting-system.html

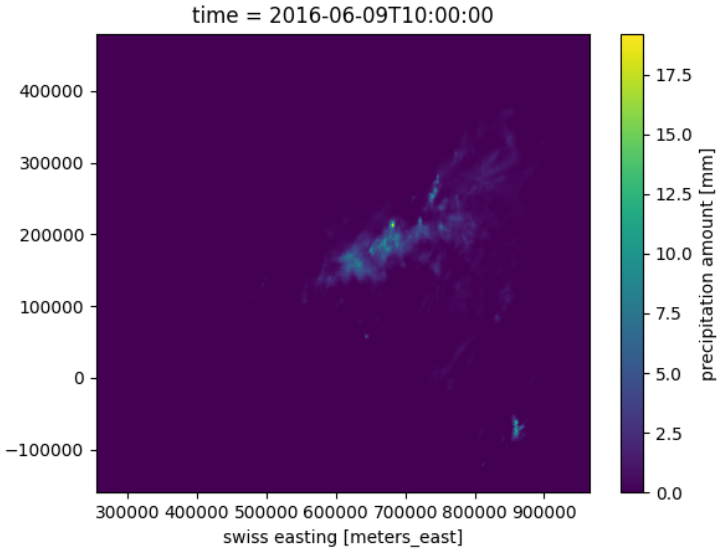

Figure 2.1: Example hourly precipitation estimate from CombiPrecip for 2016-06-09 at 10:00 UTC (accumulated precipitation over previous hour)

initial conditions, and 20 other members which were initialized using small perturbations in the initial conditions. As the datasets are very large, when training we use only the control run.

2.3 Grid Mismatch

The CombiPrecip and COSMO-E datasets use different coordinate reference systems. Therefore, we need to reproject the COSMO-E coordinates to the Swiss coordinate system. An example of a reprojection can be found in Figure 2.4. In addition, the observations are available for an area that spreads outside of Switzerland. In order to select the same region from both datasets, after reprojecting we use k-nearest neighbours in order to select the closest point from the CombiPrecip grid for each point on the COSMO-E grid.

2.4 Training and test set split

In order to be able to measure the quality of the models, it is important to devise a proper validation strategy. It is well-known that precipitation formation shows a strong seasonal and diurnal cycle. Therefore, we cannot use a randomized train-test split in our data as this would result in information leak. Instead, we split the data by year, using the last year for testing for and the 3 years prior to it for training. When using this strategy, we make the assumption that precipitation patterns do not change significantly from one year to another.

2.5 Caching data

We observe that converting the data to PyTorch tensors from large xarray matrices is a time bottleneck. Hence, we cache the observation and forecast data relevant to us in the form of numpy matrices beforehand. In the process of caching, we also filter out the samples which have missing values.

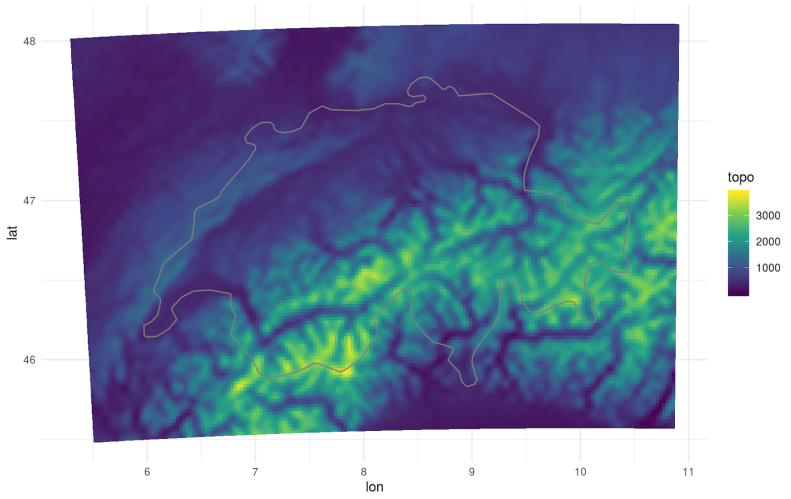

Figure 2.2: COSMO-E topography on rotated latitude-longitude

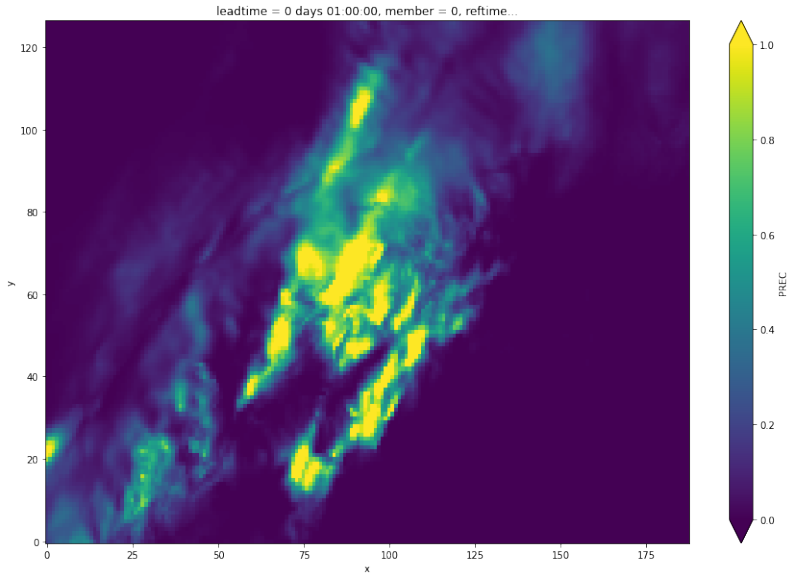

Figure 2.3: Example hourly precipitation forecast from COSMOE for 2018-05-01 at 01:00 UTC, forecast made at 00:00 UTC (accumulated precipitation over previous hour)

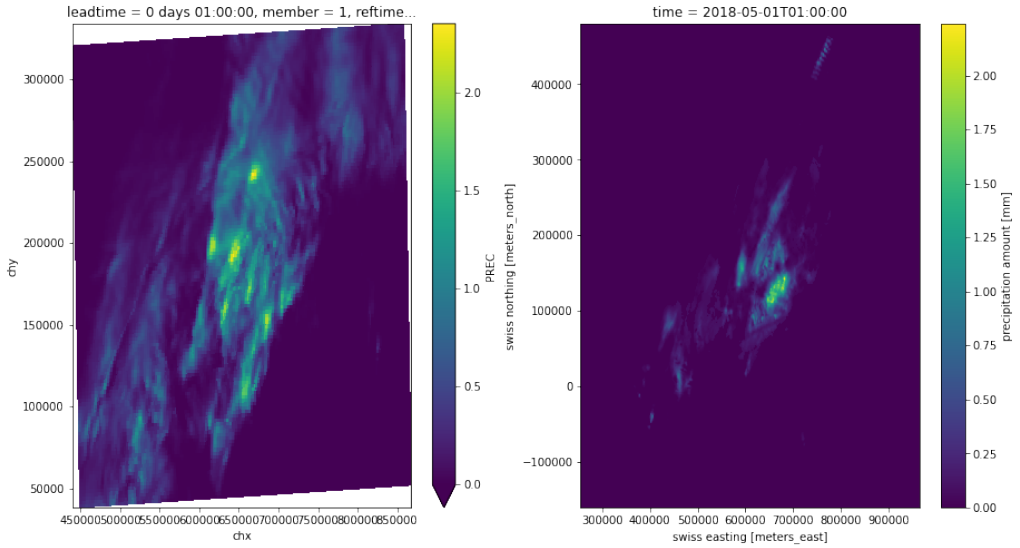

Figure 2.4: The figure on the left is the reporojected COSMOE forecast. The figure on the right is the CombiPrecip observation for the same time.

Chapter 3

Evaluation Measures

The objective of the project was to post-process numerical weather forecasts in order to correct some of the systematic biases and increase their resolution, potentially by making the model predict smaller-scale structures that are often present in the observed data. We evaluate quantitatively the performance of our model by comparing our post-processed forecasts with the observed data through the following verification measures: Root Mean Squared Error (RMSE), Log Spectral Distance and Wetness ratio. The RMSE quantifies the accuracy of the generated forecasts while the Log Spectral Distance and the wetness ratio determine whether the forecast is realistic.

3.1 Root Mean Squared Error

The Root Mean Squared Error between 2 grids of points is defined as:

$$RMSE = \sqrt{\frac{1}{MN}\Sigma_{i=1}^{M}\Sigma_{j=1}^{N}(g_{ij} - o_{ij})^2} \tag{3.1}$$

where
M = extent of the coordinate system in the chx direction in Swiss coordinates
N = extent of the coordinate system in the chy direction in Swiss coordinates
g_{ij} is the generated post-processed forecast at coordinate (i, j)
o_{ij} is the observed precipitation at coordinate (i, j)

RMSE is a measure of accuracy of the generated forecasts. We report the average RMSE over all the datapoints in the testing dataset.

3.2 Log Spectral Distance

The realism of a forecast field can be assessed qualitatively using visual inspection. However, it is useful to be able to evaluate the realism quantitatively as well, especially when comparing 2 precipitation fields or tracking the model's progress over time. One important measure that helps achieve this is the Log-Spectral Distance (LSD), which compares the power spectrum with of the simulated fields with the power spectrum of the observations. The metric is defined as follows:

$$LSD = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}((10log_{10}\frac{P_{obs,i}}{P_{gen,i}})^2)} \tag{3.2}$$

where N = the number of points in the dataset
$P_{obs,i}$ is the power spectrum of the observed precipitation
$P_{gen,i}$ is the power spectrum of the generated post-processed forecast

3.3 Wetness Ratio

When training the Generative Adversarial Networks, one of the challenges we faced was the fact that the generated images systematically underestimated the amount of precipitation. This is due to the fact that a large proportion of the images from the original dataset contained no precipitation. For this reason, we attempted to balance the estimation of precipitation by removing a fraction of samples from the dataset with low precipitation. However, it proves to be difficult to determine a right threshold, which results in generated images with a precipitation ratio similar to the one present in the observation dataset. We use the following measure to compare the amount of precipitation from a generated forecast with the amount o precipitation from the observed precipitation field:

$$\text{Wetness ratio} = \frac{\text{No. of non-zero precipitation coordinates in the forecast}}{\text{No. of non-zero precipitation coordinates in the observation}} \tag{3.3}$$

A wetness ratio close to 1 is ideal.

Chapter 4

Methods

In this chapter, we provide a description of the generative models that we use in approximating CombiPrecip observations using COSMO-E forecasts. All models that we use are based on neural networks. Specifically, these are generative adversarial networks in two variants. First, we use generative adversarial networks (GANs) to generate realistic-looking forecasts without requiring that the output is similar to a specific observation. Then, we extend this by using conditional generative adversarial networks (cGANs) to generate forecasts for a specific date and time, using the COSMO-E forecast as the input.

4.1 Generative Adversarial Network

In general, a generative adversarial network [4] is an adversarially trained neural network architecture used for generative modeling, generating new examples based on real examples from a given probability distribution. It consists of two functions, a generator G and a discriminator D. In our model, G and D are parametrized as convolutional neural networks. The generator is trained to output plausible examples as a function of the input, which is a uniformly distributed random vector. This approximates drawing from a probability distribution, where the input noise ensures a non-zero variance of the generated examples. Conversely, the discriminator is trained to distinguish between real examples and examples generated by the generator, outputting a confidence score. In theory, by training the model in this way, the distribution of images generated by G becomes indistinguishable from the real probability distribution as D and G improve. We will now describe the details of our implementation.

In the next sections, we will use the following abbreviations to denote layers of a neural network:

- $Conv2d(c, k, s, p)$ denotes a 2D convolution layer with c channels, a kernel size of k, a stride of s and p zero-padding pixels on the image boundaries.
- $Deconv(c, k, s, p)$ denotes a 2D transposed convolution operator. The arguments are identical to $Conv2d(c, k, s, p)$.
- $Tanh$ denotes the hyperbolic tangent function.
- $\sigma(.)$ denotes the sigmoid function.
- $ReLU$ denotes a Rectified Linear Unit [10]. $\text{ReLU}(x) = \max(0, x)$.
- $BatchNorm$ denotes a batch normalization layer [5].

4.1.1 GAN architecture

In the unconditional case, the input to the generator is a random vector $x \in \mathbb{R}^{100}$. The generator outputs an image matrix $y \in \mathbb{R}^{256x384}$. The input to the discriminator is an image matrix $x' \in \mathbb{R}^{256x384}$, with either $x' = y$, or $x' \in \mathrm{D}_{\mathrm{obs}}$, the dataset of observations. The discriminator outputs a single scalar, $y' \in [0, 1]$, indicating the confidence that x' came from the distribution of $\mathrm{D}_{\mathrm{obs}}$.

The generator is of the following form:

$$
\begin{aligned}
x \rightarrow\ & \mathrm{Deconv}(1024, (4, 6), 1, 0) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Deconv}(512, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Deconv}(256, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Deconv}(128, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Deconv}(64, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Deconv}(1, 4, 2, 1) \rightarrow \mathrm{Tanh} \rightarrow y \in \mathbb{R}^{256x384}
\end{aligned}
$$

The discriminator takes the following form:

$$
\begin{aligned}
x' \rightarrow\ & \mathrm{Conv2d}(64, 4, 2, 1) \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Conv2d}(64, 4, 2, 1) \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Conv2d}(128, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Conv2d}(256, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Conv2d}(512, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Conv2d}(1024, 4, 2, 1) \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU} \\
\rightarrow\ & \mathrm{Conv2d}(1, (4, 6), 1, 0) \rightarrow \sigma(.) \rightarrow y' \in [0, 1]
\end{aligned}
$$

4.1.2 cGAN architecture

Using GAN from the previous section, we are able to generate realistic-looking observation images. However, we are not able to generate an observation based on a specific input image from the forecast COSMO-E model. Thus, we have to modify the network to condition on this input image. For this, we use a conditional generative adversarial network [9] which takes the forecast as its input. Our implementation is based on the pix2pix paper [6], which proposed various optimizations to the default cGAN architecture in order to perform image-to-image translation with high resolution images.

In the conditional case, the input to the generator is a zero-padded forecast image from the COSMO-E model $x \in \mathbb{R}^{128x192}$. The generator outputs an image matrix $y \in \mathbb{R}^{256x384}$. The discriminator is identical as in the unconditional case. It takes as input the image matrix $x' \in \mathbb{R}^{256x384}$, with either $x' = y$, or $x' \in \mathrm{D}_{\mathrm{obs}}$, the dataset of observations, and outputs a single scalar, $y' \in [0, 1]$, indicating the confidence that x' came from the distribution of $\mathrm{D}_{\mathrm{obs}}$.

The generator is composed of a convolutional encoder, which compresses the forecast image x to a latent vector representation $z \in \mathbb{R}^{96}$, followed by a deconvolutional decoder, which attempts to recover the observation image conditioned on z.

Additionally, we use skip connections between encoder and decoder. A skip connection is used to improve the flow of gradient by connecting layer $\mathbf{d}^i$ in the decoder with a previous layer in the encoder $\mathbf{e}^i$, thus skipping the intermediate layers. Specifically, the output channels of $\mathbf{e}^i$ are appended to the input of $\mathbf{e}^i$. Skip connections are denoted by superscripts in the following description of layers.

Encoder

$$
\begin{aligned}
x' \rightarrow\ & \text{Conv2d}^5(64, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Conv2d}^4(64, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Conv2d}^3(128, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Conv2d}^2(128, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Conv2d}^1(1, 1, 1, 0) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
& \rightarrow z \in \mathbb{R}^{96}
\end{aligned}
$$

Decoder

$$
\begin{aligned}
z \rightarrow\ & \text{Deconv}(2048, (4, 6), 1, 0) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Deconv}(1024, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Deconv}^1(512, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Deconv}^2(256, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Deconv}^3(128, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Deconv}^4(64, 4, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \\
\rightarrow\ & \text{Deconv}^5(1, 4, 2, 1) \rightarrow \text{Tanh} \rightarrow y \in \mathbb{R}^{256x384}
\end{aligned}
$$

4.1.3 Hyperparameters

In training, we use the Adam [7] optimizer with learning rate $\gamma_D = 0.0002$ for the discriminator, and $\gamma_G = 0.0001$ for the generator. We set $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The model is trained for 100 epochs on the MeteoSwiss dataset with a batch size of 64.
As the loss function, we use Binary Cross Entropy Loss for both GAN and cGAN. To enforce the similarity of generated images with the ground truth in cGAN, this loss is summed with the L1 loss with $\lambda = 100$ (see [6] for details).

Chapter 5

Results and Discussion

We will now evaluate the models proposed in Chapter 4. We will first use the previously discussed evaluation measures to perform quantitative evaluation. Finally, we will perform visual (human) evaluation of the quality of generated images.

5.1 Generating fake observations

We performed some experiments using unconditional GANs in order to generate realistic-looking observations, similar to those present in the CombiPrecip dataset. As GANs are generally difficult to evaluate, we relied extensively on human evaluation. The fake images generated by the GAN look physically realistic but sometimes overestimate the amount of precipitation. This is due to the fact that we had to remove a large proportion of the images from the original dataset (the observations that contained no or very little precipitation) in order to obtain a stable GAN. Figure 5.1 and Figure 5.2 show samples of generated images and images from the original dataset. In addition to human evaluation, we use the log-spectral distance to check whether the generated images resemble the spatial structure of the original images. Despite looking physically realistic, the generated images have a higher log-spectral distance than the original images, indicating that there is some power loss at some frequencies. More hyperparameter tuning (and perhaps a more powerful neural network architecture) is required in order to generate fields that look both realistic for the human eye and match the power spectrum of the original images.

5.2 Post-processing of NWP forecasts

We compute the same metrics for both the COSMO-E forecasts and the post-processed forecasts using the conditional GAN. A comparison between the 2 forecasts can be seen in Table 5.1.

Verification measures		
Verification Measure	Original	Generated
Log-Spectral Distance	6.97	12.42
Root Mean Squared Error	0.117	0.113
Wetness Ratio	1.07	2.07

Table 5.1: Comparison of verification measures between the NWP forecasts and the post-processed forecasts

Similar to the images produced by the unconditional GAN, the post-processed forecasts overestimate the amount of precipitation (for the same reasons previously mentioned). A sample of NWP forecasts, together with the observations and the post-processed forecasts can be seen in Figure 5.3. The slight improvement in the RMSE indicates that, even though the generated images overestimate the amount precipitation (the amount of non-zero pixels), the intensity of these is not

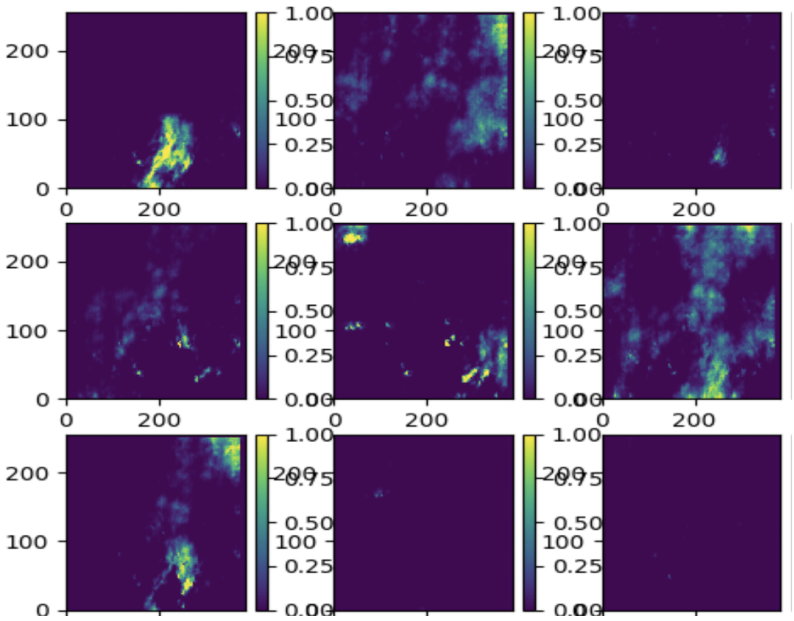

Figure 5.1: Sample of generated images using the unconditional GAN

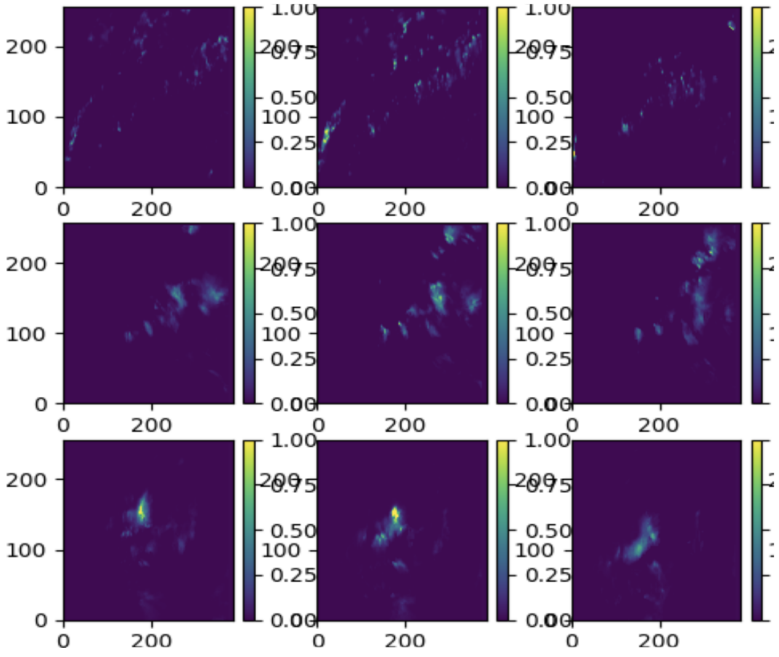

Figure 5.2: Sample of observations from the training dataset

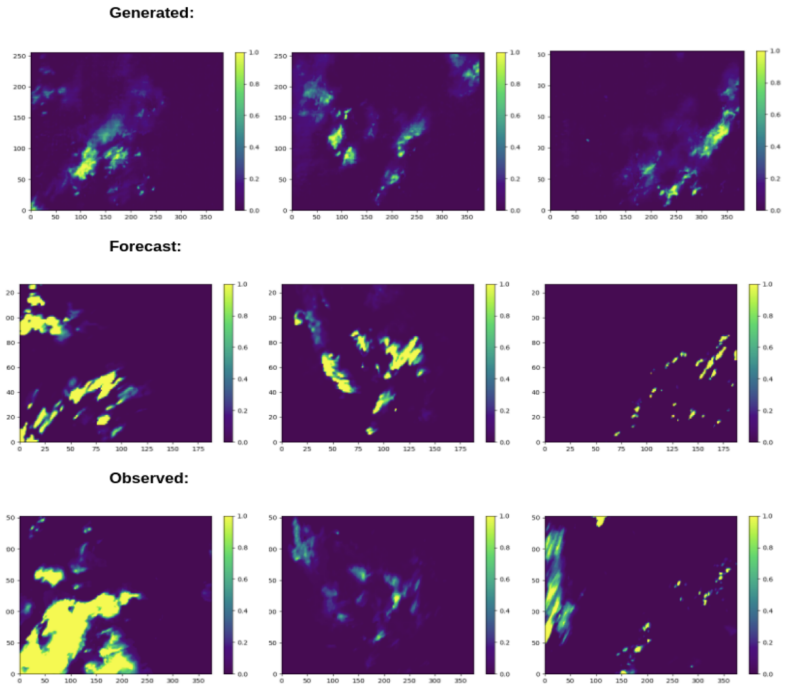

Figure 5.3: Sample of NWP forecast, together with the ground-truth (observation) and the post-processed forecast generated by the GAN

significant and the post-processed images match better the intensity and location of the rainfall present in the observations. Even though the generated fields look physically realistic, the increase in log-spectral distance indicates that the generated images miss some power at certain frequencies.

Chapter 6

Conclusion

Generative models, and conditional Generative Adversarial Networks (GANs) in particular, prove to be an effective method for post-processing of precipitation forecasts. The GANs are able to generate realistic-looking precipitation forecasts, which look very similar to the true observations and perform reasonably well when evaluated using verification measures such as log-spectral distance, root mean squared error and wetness ratio. Even though some of these metrics are worse for the post-processed forecasts, the conditional GAN manages successfully to increase the resolution of the original forecasts and match better the location and intensity of the rainfall present in the observations.

This work could be extended by modifying the original pix2pix architecture so that the output of the generator is randomized, for example using dropout[1]. This way, the generator could be used to output any number of ensemble members by using a different random state. Alternatively, the generator could be used to post-process each of the original ensemble members, resulting in an ensemble of the same size. An interesting avenue for research would be comparing the performance of the two methods. Additionally, further experiments can be conducted using other neural network architectures and more hyperparameter tuning in the attempt to improve the performance of our method.

[1] It is harder to randomize cGAN compared to GAN, since the network learns to ignore random noise in input.

Bibliography

[1] https://www.cawcr.gov.au/projects/verification/. `https://www.cawcr.gov.au/projects/verification/`. (Accessed on 01/22/2021).

[2] Numerical weather prediction | national centers for environmental information (ncei) formerly known as national climatic data center (ncdc). `https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/numerical-weather-prediction`. (Accessed on 01/22/2021).

[3] weather forecasting | methods, importance, & history | britannica. `https://www.britannica.com/science/weather-forecasting`. (Accessed on 01/22/2021).

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448â456. JMLR.org, 2015.

[6] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[8] H. Medina and D. Tian. Comparison of probabilistic post-processing approaches for improving numerical weather prediction-based daily and weekly reference evapotranspiration forecasts. *Hydrology and Earth System Sciences*, 24(2):1011–1030, 2020.

[9] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014. cite arxiv:1411.1784.

[10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes FÃ¼rnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.