Spatiotemporal Weather Data Predictions with Shortcut Recurrent-Convolutional Networks: A Solution for the Weather4cast Challenge

Jussi Leinonen[1]

[1]*Federal Office of Meteorology and Climatology MeteoSwiss, Locarno-Monti, Switzerland*

Abstract

This paper presents the neural network model that was used by the author in the Weather4cast 2021 Challenge Stage 1, where the objective was to predict the time evolution of satellite-based weather data images. The network is based on an encoder-forecaster architecture making use of gated recurrent units (GRU), residual blocks and a contracting/expanding architecture with shortcuts similar to U-Net. A GRU variant utilizing residual blocks in place of convolutions is also introduced. Example predictions and evaluation metrics for the model are presented. These demonstrate that the model can retain sharp features of the input for the first predictions, while the later predictions become more blurred to reflect the increasing uncertainty.

Keywords

weather, satellite data, neural networks, gated recurrent units

1. Introduction

Weather forecasting requires the prediction of spatiotemporal changes in atmospheric fields. Most commonly, this prediction is carried out using numerical weather prediction (NWP, e.g. [1]), which models the evolution of the atmosphere by computationally evaluating approximate physical laws, such as forward-integrating the partial differential equations of fluid dynamics.

Although the NWP approach to forecasting is firmly grounded in first principles, NWP models are computationally expensive: state-of-the-art models are run on powerful supercomputers and yet take a significant amount of time to complete a model run, which limits their usefulness in short-term predictions. These issues have motivated considerable research on using artificial intelligence (AI) and machine learning (ML) methods for weather prediction, particularly after the prominent progress in convolutional neural networks (ConvNets) in the last decade. ConvNets can learn to recognize complex spatial structures, which make them well-suited for modeling atmospheric fields. They have been used as components of weather models, for example, to speed up computationally expensive modules of NWP codes. Recurrent neural networks have also been used to model the time evolution of atmospheric fields. Numerous applications and potential uses of ML/AI in weather in climate sciences have been discussed by, e.g., [2, 3, 4, 5]. Thus, there is considerable interest in further developing ML model architectures for these applications.

To support the development of ML/AI for weather prediction, the Weather4cast 2021 Challenge had the objective to predict the time evolution of atmospheric satellite data products using machine-learning methods. This paper presents the ML solution submitted by the author.

2. Challenge Data and Objectives

The Weather4cast 2021 Challenge Stage 1 presented the competitors with a data problem of predicting the time evolution of satellite-based measurements of four different atmospheric fields: *temperature*, representing either the surface temperature (where no cloud is present) or the cloud top temperature (in cloudy areas); convective rainfall rate (*crr_intensity*); probability of tropopause folding (*asii_turb_trop_prob*); and cloud mask (*cma*). These are data products created by the Nowcasting Satellite Application Facility (NWC SAF) processing from satellite data from the EUMETSAT-operated MeteoSat satellites. The products were given for six regions (numbered R1–R6) in Europe, Northern Africa and the Middle East, each 256×256 pixels of approximately $4 \, \mathrm{km} \times 4 \, \mathrm{km}$ each. A few other variables also were available such as cloud type, as well as static data with the elevation of the terrain, the latitude and the longitude.

The goal of the challenge was to predict, in each area, the next 32 frames of a sequence given the previous 4 frames, with frames separated by 15 min. Training and validation datasets, with complete $4 + 32$-frame samples, as well as a test dataset with only the 4-frame inputs,
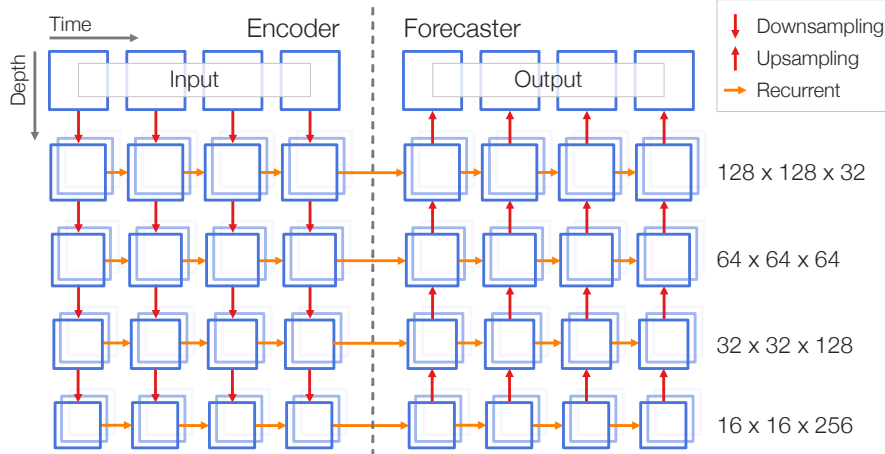
Figure 1: Illustration of the network architecture.

were provided for regions R1–R3, constituting the "Core" competition. Meanwhile, R4–R6 only had test data available, meaning that they had to be evaluated using models trained on R1–R3; this was called the "Transfer Learning" competition. Furthermore, all regions had a set of "held-out" data which were made available only during the final week of the competition; the final results were based on the performance with these data.

The performance of the models was evaluated using the mean-square error (MSE) for each variable. However, some adjustments were made to the MSE to account for the particular needs of each variable, except for *crr_intensity*. First, the loss for *temperature* was modified to account for varying amounts of missing data in each region. Second, *asii_turb_trop_prob* is a probabilistic variable and the output of the model was passed through a truncated and normalized logit transform before the evaluation of the MSE. Third, although *cma* is technically evaluated using the MSE, the variable in the output data file is required to be quantized such that the value is either 0 or 1; therefore, model output values < 0.5 are rounded to 0 and outputs ≥ 0.5 are rounded to 1 before evaluation. The details of the metrics can be found in [6].

3. Solution

3.1. Models

The model presented here is a neural network combining recurrent-convolutional layers and shortcut connections in an encoder-forecaster architecture. The architecture is presented in Fig. 1. It is based on that developed in [7] for precipitation nowcasting and adopted by [8], as well as similar to that of [9], with some differences that are described below.

The encoder section consists of four recurrent downsampling stages. Each stage first passes the sequence through a residual block [10], with each frame processed using the same convolutional filters. A strided convolution in the residual block is used to downsample the input by a factor of 2. Then, the sequence is processed by a gated recurrent unit (GRU) layer [11]; a tensor of zeros is passed as the initial state of the GRU. The number of channels in the convolutions is increased with increasing depth in the encoder.

The forecaster section is approximately a mirror image of the encoder section. Each stage consists of a GRU layer which is followed by bilinear upsampling and a residual block. A shortcut similar to U-Net [12] is utilized: The final state of each GRU in the encoder is passed through a convolution and then used as the initial state of the GRU of corresponding depth in the forecaster. This allows the high-resolution features of the recent frames to be passed through, preventing the first predictions from being blurry. A final projection and a sigmoid activation produce the output as a single variable constrained between 0 and 1.

The main difference of the architecture presented here to that of [7] is that the use of Trajectory GRU (TrajGRU) is rejected as TrajGRU was found to cause training instability. Two variants are considered instead. The first utilizes the Convolutional GRU (ConvGRU) layer adopted by e.g. [9, 13, 14]. In the second variant, the convolution in the ConvGRU is replaced by a residual block modified to be used for this purpose. The use of the residual block increases the depth of the operations in the GRU and is thus expected to allow it to better process nonlinear transformations and also to increase the

distance at which pixels can influence each other at each step of the ConvGRU. The latter effect may recover some of the advantages of TrajGRU over ConvGRU that [7] found. The author is unaware of previously published instances of a residual layer being used in place of the convolution in GRU. In this paper, this variant is called "ResGRU", although the same abbreviation was used for a different combination of GRUs and residual connections in [15].

The models were implemented using TensorFlow/Keras [16] version 2.4. The source code and the pre-trained models can be found through the links in Appendix A.

3.2. Training

Since the scores for the target variables were evaluated independently from each other, a separate instance of the model was trained for each target variable, but using all variables as inputs for each model. The models were trained on the training dataset of R1–R3 such that every available gapless sequence of 36 frames was used for training, resulting in 72192 different sequences (albeit with considerable overlap). The training was performed with combined data from all regions R1–R3 in order to increase the training dataset size and improve the ability of the model to generalize; specializing the model to single regions was not attempted. The static data (latitude, longitude and elevation) were also used for training. Data augmentation by random rotation in 90° increments as well as random top-down and left-right mirroring was used to further increase the effective number of training samples. The model for *asii_turb_trop_prob* was trained using a custom logit loss corresponding to the metric specified in [6], while the other variables were trained using the standard MSE loss. The Adam optimizer [17] was used to train the models with a batch size of 32.

The progress of the training was evaluated using the provided validation dataset for R1–R3. After each training epoch, the evaluation metric was computed on the validation set and then:

1. If the metric improved upon the best evaluation result, the model weights were saved.
2. If the metric had not improved in 3 epochs, the learning rate was reduced by a factor of 5.
3. If the metric had not improved in 10 epochs, the training was stopped early.

In practice, condition 3 was never activated as the model continued to achieve marginal gains on the validation data at least every few epochs until the maximum training time of 12 h or 24 h (depending on the training run) was reached. This suggests that the model did not suffer significantly from overfitting, which typically causes the validation loss to start increasing even as the training loss keeps decreasing. This is perhaps due to the relatively modest number of weights in the models by standards of modern ConvNets, approximately 12.1 million weights in the ConvGRU variant and 18.6 million in the ResGRU variant. The loss over the validation set was used as the metric for each variable except *cma* for which a rounded MSE that takes the 0, 1 quantization into account was used.

A parallel setup of eight Nvidia Tesla V100 GPUs was used to train the models. Training for one epoch took approximately 20 minutes with this hardware. The eight parallel GPUs only provided a speedup of a factor of approximately 3 compared to training on a single GPU, suggesting that single-GPU training of the models should be feasible, although the batch size would likely have to be reduced as the models require rather large amounts of GPU memory.

4. Results

Both the ConvGRU and ResGRU variants of the model were trained for each target variable. The evaluation results for the validation dataset are shown in Table 1. Comparisons to TrajGRU were found impractical as the models using TrajGRU would not converge properly due to the training instability mentioned in Sect. 3.1.

Based on the evaluation results, three submissions were made to the final leaderboards of Weather4cast Stage 1: one using the ConvGRU variant for all variables (codenamed V4c), another using ResGRU (V4rc), and a third using the best model for each variable based on the validation metrics (V4pc). It was indeed this last model that produced the best results also on the leaderboards for both the Core and Transfer Learning competitions, as shown in Table 2.

Figures 2–5 show examples of the predictions using the validation dataset. These are all shown for the same scene except for Fig. 3, where a different scene was chosen because the one used for the others did not contain precipitation. It is clear that the predictions start relatively sharp and get blurrier over time, reflecting the increasing uncertainty. The blurriness is likely exacerbated by the use of the MSE metric, specified in the data challenge, which is prone to regression to the mean. Especially in Fig. 4, one can also see that the model can predict the motion of features in the images.

5. Conclusions

The model presented here reached the top of the final leaderboards in both the Core and the Transfer Learning categories of the Weather4cast 2021 Challenge Stage 1. It is a versatile solution to the problem of predicting the evolution of atmospheric fields, producing sharp predictions for the near term and increasing the uncertainty for

Table 1
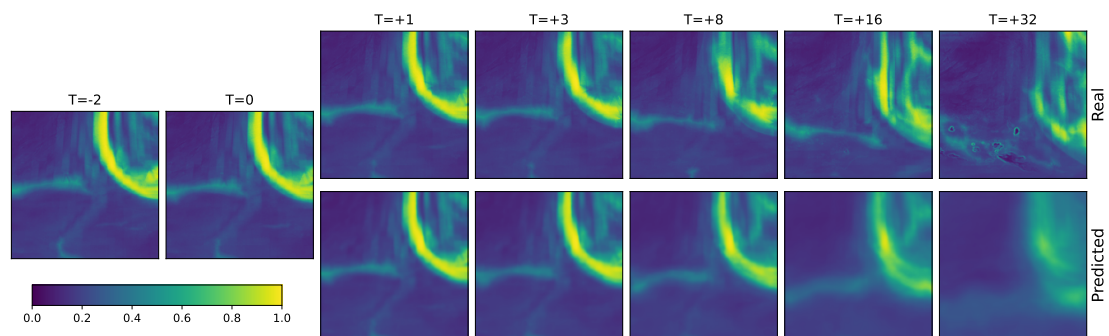
Evaluation metrics for the validation dataset.

|	temperature	crr_intensity	asii_turb_trop_prob	cma
ConvGRU	0.004564	**0.0001259**	0.002250	0.1393
ResGRU	**0.004356**	0.0001278	**0.002161**	**0.1376**
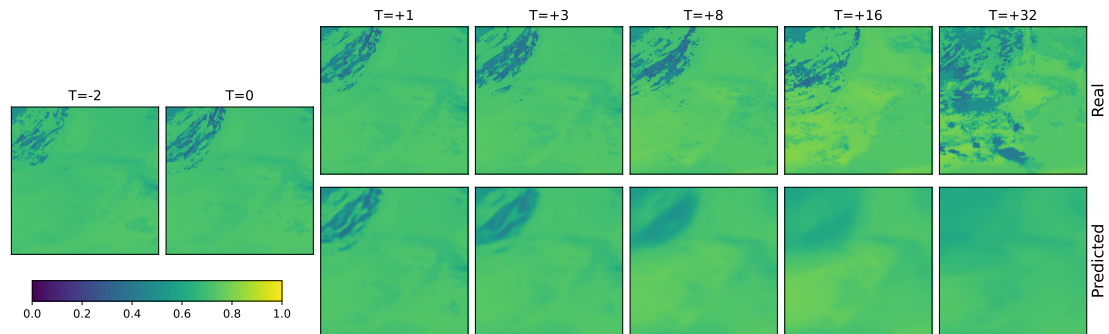
Figure 2: An example of predictions for the *temperature* variable. The frames on the left correspond to past temperature, while the frames on the right show the real future temperature (top row) and the predicted temperature (bottom row). The T coordinate refers to the index of the frame in the sequence, with $T = 0$ represents the last input data point and $T = 1$ the first prediction. The model output normalized to the range $(0, 1)$ is shown.
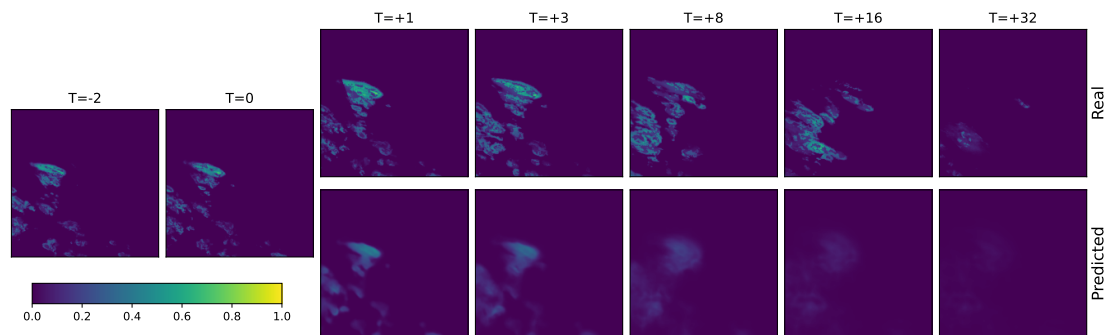
Figure 3: As Fig. 2, but for *crr_intensity*. A different case is shown as the case of Fig. 2 does not contain precipitation.

Figure 4: As Fig. 2, but for *asii_turb_trop_prob*.
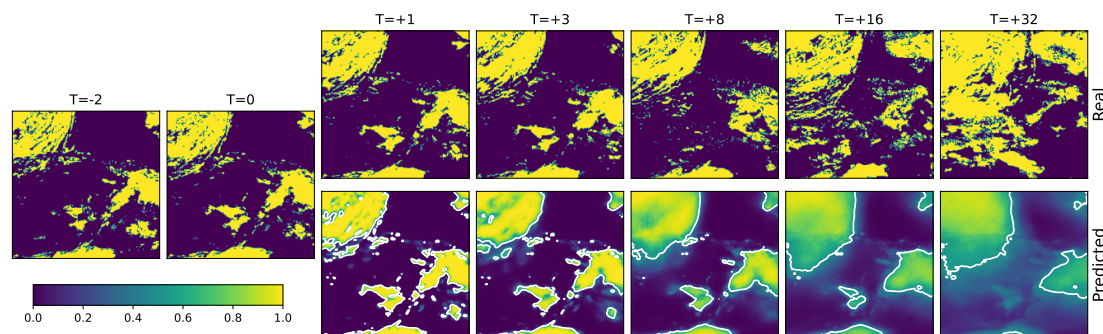
Figure 5: As Fig. 2, but for *cma*. The white contours in the predictions indicate 0.5, the threshold of the cloud mask in the output.

Table 2

Evaluation metrics for the held-out test dataset, as computed by the Weather4cast website (https://www.iarai.ac.at/weather4cast/).

	Core	Transfer learning
ConvGRU	0.5051	0.4658
ResGRU	0.5014	0.4626
Best combination	**0.4987**	**0.4607**

longer lead times. The architecture can be easily adapted to other tasks such as probabilistic predictions or outputs that are different from the inputs. Further research is needed to handle, for instance, different spatial and temporal resolutions of inputs and data available for future time steps.

Acknowledgments

References

[1] P. Bauer, A. Thorpe, G. Brunet, The quiet revolution of numerical weather prediction, Nature 525 (2015) 47–55. doi:doi.org/10.1038/nature14956.

[2] A. McGovern, K. L. Elmore, D. J. Gagne, II, S. E. Haupt, C. D. Karstens, R. Lagerquist, T. Smith, J. K. Williams, Using artificial intelligence to improve real-time decision-making for high-impact weather, Bull. Amer. Meteor. Soc. 98 (2017) 2073–2090. doi:10.1175/BAMS-D-16-0123.1.

[3] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, Prabhat, Deep learning and process understanding for data-driven earth system science, Nature 566 (2019) 195–204. doi:10.1038/s41586-019-0912-1.

[4] C. Huntingford, E. S. Jeffers, M. B. Bonsall, H. M. Christensen, T. Lees, H. Yang, Machine learning and artificial intelligence to aid climate change research and preparedness, Environmental Research Letters 14 (2019) 124007. doi:10.1088/1748-9326/ab4e55.

[5] S. E. Haupt, W. Chapman, S. V. Adams, C. Kirkwood, J. S. Hosking, N. H. Robinson, S. Lerch, A. C. Subramanian, Towards implementing artificial intelligence post-processing in weather and climate: proposed actions from the oxford 2019 workshop, Philos. Trans. R. Soc. London, Ser. A 379 (2021) 20200091. doi:10.1098/rsta.2020.0091.

[6] IARAI, Weather4cast 2021: Competition metrics, 2021. URL: https://www.iarai.ac.at/weather4cast/wp-content/uploads/sites/3/2021/04/w4c.pdf.

[7] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, W.-c. WOO, Deep learning for precipitation nowcasting: A benchmark and a new model, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/a6db4ed04f1621a119799fd3d7545d3d-Paper.pdf.

[8] G. Franch, D. Nerini, M. Pendesini, L. Coviello, G. Jurman, C. Furlanello, Precipitation nowcasting with orographic enhanced stacked generalization: Improving deep learning predictions on extreme events, Atmosphere 11 (2020). doi:10.3390/atmos11030267.

[9] S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge, R. Prudden, A. Mandhane, A. Clark, A. Brock, K. Simonyan, R. Hadsell, N. Robinson, E. Clancy, A. Arribas, S. Mohamed, Skillful precipitation nowcasting using deep generative models of radar, 2021. `arXiv:2104.00954`.

[10] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. doi:`10.1109/CVPR.2016.90`.

[11] K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, in: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, 2014, pp. 103–111.

[12] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: N. Navab, J. Hornegger, W. M. Wells, A. F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, 2015, pp. 234–241. doi:`10.1007/978-3-319-24574-4_28`.

[13] L. Tian, X. Li, Y. Ye, P. Xie, Y. Li, A generative adversarial gated recurrent unit model for precipitation nowcasting 17 (2020) 601–605. doi:`10.1109/LGRS.2019.2926776`.

[14] J. Leinonen, D. Nerini, A. Berne, Stochastic super-resolution for downscaling time-evolving atmospheric fields with a generative adversarial network, IEEE Trans. Geosci. Remote Sens. (2020). doi:`10.1109/TGRS.2020.3032790`.

[15] W. Gao, R.-J. Wai, A novel fault identification method for photovoltaic array via convolutional neural network and residual gated recurrent unit, IEEE Access 8 (2020) 159493–159510. doi:`10.1109/ACCESS.2020.3020296`.

[16] F. Chollet, et al., Keras, https://keras.io, 2015.

[17] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference for Learning Representations, San Diego, California, USA, 2014. URL: https://arxiv.org/abs/1412.6980.

[18] J. Leinonen, Model weights for a Weather4cast 2021 Challenge Stage 1 solution, 2021. doi:`10.5281/zenodo.5101213`.

A. Online Resources

The source code with instructions to replicate the results presented in this paper can be found at https://github.com/jleinonen/weather4cast-stage1. The model weights used in the challenge submissions can be downloaded at [18].