Precipitation and Cloud Forecasting Using Radar Data and Encoder-Decoder Convolutional LSTM

Bianka Szepesiová[1], Richard Staňa[1,*]

[1]Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University in Košice, Jesenná 5, 040 01 Košice, Slovakia

Abstract

Precipitation and cloud forecasting is critical for weather forecasting, impacting sectors like agriculture, transportation, and renewable energy. Traditional methods, such as satellite imagery, radar systems, and numerical models, often struggle with short-term accuracy. This paper explores the application of neural networks for precipitation and cloud forecasting using sequences of radar images collected every 5 minutes. This approach enables temporal modeling of precipitation and cloud dynamics. We analyze neural network-based methods, propose a ConvLSTM model extended with an encoder-decoder architecture inspired by U-Net, and evaluate its performance on radar data from the Slovak Hydrometeorological Institute (SHMÚ). Results show that while the baseline ConvLSTM model replicates the last input frame, the encoder-decoder extension improves cloud movement prediction, though with reduced image quality. Metrics like Mean Absolute Error (MAE) and Structural Similarity Index Measure (SSIM) quantify performance, suggesting avenues for future optimization.

Keywords

Precipitation and cloud forecasting, Radar data, ConvLSTM, Encoder-Decoder

1. Introduction

Precipitation and cloud forecasting is crucial for accurate weather forecasting. It has an impact on everyday activities, such as trips and sporting events, and also significantly affects many other areas, including agriculture, transportation, and renewable energy. Traditional methods, including satellite imagery, radar systems, and numerical models, provide valuable insights but face limitations in short-term forecasting due to resolution or computational constraints [1, 2]. Recent advances in neural networks, particularly Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models, offer promising solutions by effectively analyzing temporal and spatial patterns in meteorological data [3]. This paper examines the application of neural networks in precipitation and cloud prediction using radar data. We review neural network-based methods, propose a Convolutional LSTM (ConvLSTM) model, and extend it with an encoder-decoder architecture inspired by U-Net [4]. The models are trained and tested on radar data from the Slovak Hydrometeorological Institute (SHMÚ). Unlike similar research papers described in the next section, we use images with a higher resolution (in most cases, more than twice as large, Table 1) than other projects, and we focus on RGB images. We evaluate performance using Mean Absolute Error (MAE) and Structural Similarity Index Measure (SSIM), comparing the baseline ConvLSTM and its encoder-decoder extension to assess improvements in predicting cloud movement.

The paper is structured into six sections. Following the introduction, Section 2 provides an overview of existing research in the field of precipitation and cloud prediction. Section 3 provides the methodology of this paper, including the used dataset and neural network models. The process of data preprocessing and training of models is described in Section 4. In Section 5, the results of our experiments are provided and discussed. Finally, Section 6 concludes our work and provides future possibilities.

ITAT'25: Information Technologies – Applications and Theory, September 26–30, 2025, Telgárt, Slovakia

*Corresponding author.

bianka.szepesiova@student.upjs.sk (B. Szepesiová); richard.stana@upjs.sk (R. Staňa)
0000-0001-7938-2117 (R. Staňa)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-4092/paper5.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2025-11-13

Table 1
Comparison of existing research.

Research	Resolution	Data	Methods
[6]	280×280×1	radar	RDCNN, COTREC
[7]	256×256×1	satelite binarized	U-Net, AROME
[8]	100×100×1	radar	PredRNN_v2
[9]	150×150×1	radar	RainPredRNN
[10]	96×96×1	radar	variants of U-Net
[11]	256×256×2	radar, satelite	Optical flow, ConvLSTM, U-Net, MSDM
[12]	344×315×1	radar	ConvLSTM
[13]	64×64×1	radar	ConvLSTM
our	512×288×3	radar	ConvLSTM, encoder-decoder ConvLSTM

2. Related Works

Traditional cloud prediction methods depend on satellite images, radar data, and numerical models. Electromagnetic radiation captured by meteorological satellites is used for monitoring cloud cover and atmospheric conditions. Although they offer a wide atmospheric view and fast updates in real-time, they are limited by low resolution [5]. Radar systems emit radio waves that reflect off precipitation particles, creating real-time images. However, they have limited range, difficulty detecting signals close to the radar, and may produce occasional false echoes from objects like planes or birds [1]. Numerical models simulate atmospheric dynamics based on physical equations. However, it requires a significant amount of computational power, and the initial conditions must be precise [2].

Neural network approaches have shown promise in overcoming these limitations. The RDCNN model utilizes a recurrent dynamic subnet (RDSN) comprising convolutional, sampling, and hidden layers, as well as a probability prediction layer, to forecast radar data. RDCNN achieved better results compared to traditional methods, such as COTREC [6]. The U-Net [4] architecture has also been proven more successful than traditional methods, such as the AROME model [7]. RainPredRNN combines U-Net [4] and PredRNN_v2 [8], which results in lower computational costs while still being able to produce good-quality predictions through spatiotemporal LSTM layers and memory decoupling [9]. Other variants of U-Net were used in research [10] for precipitation nowcasting. A combination of radar and satellite images was used in the paper [11] for precipitation nowcasting with methods such as Optical flow, ConvLSTM, U-Net, and MSDM. Variants of ConvLSTM were used for the same task on radar data in works [12, 13]. In Table 1 is the comparison of the described research in this field with the resolutions of images, the type of the dataset, and the methods they use.

3. Methodology

3.1. Dataset

The dataset was provided by the Slovak Hydrometeorological Institute (SHMÚ) [14] and comprises radar images captured at 5-minute intervals between January 2016 and September 2023 (814,499 images in total, each with a resolution of 2270 x 1560 pixels). The current radar network consists of four Meteor 735 CDP10 units located at Malý Javorník, Kojšovská hoľa, Kubínska hoľa, and Španí laz. All data are processed centrally at the SHMÚ Koliba facility, and actual data are publicly accessible via the SHMÚ website [14]. The example of the provided images is in Fig. 3.

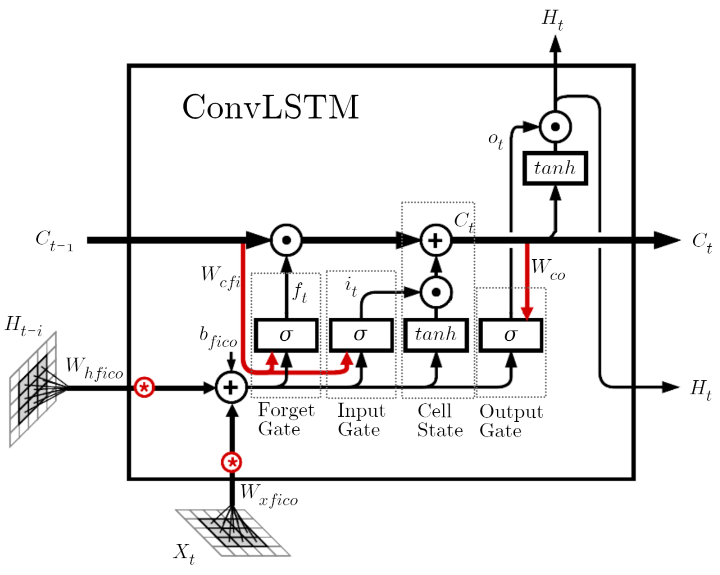

Figure 1: Structure of a ConvLSTM cell. Source: [16].

3.2. Architecture Models

3.2.1. Baseline ConvLSTM

The baseline architecture consists of three components: ConvLSTMCell, ConvLSTM, and Seq2Seq [15].

ConvLSTMCell is the core unit of the model. This cell adopts principles of gates from a standard LSTM cell. The simplified structure of the ConvLSTMCell is shown in Figure 1. The Forget Gate decides how much information from the previous cell state (C_{t-1} in image 1) should be kept or discarded. The input Gate controls how much of the new incoming information should be added to the cell state. The New Cell State updates the new cell state (C_t in image 1), combining retained old information and selected new information. The Output Gate determines how much of the cell state should be exposed as output and passed on to the output (next hidden state H_t in image 1). In the ConvLSTMCell, the convolution is applied to the concatenation of the current input tensor (which can be the input frame or the feature map from the previous ConvLSTM layer) and the previous hidden state along the channel dimension. The number of output channels from this convolution is four times the number of the cell's designated output channels, allowing the result to be split into four separate tensors corresponding to the input gate, forget gate, new cell state, and output gate. The input and forget gates are computed by applying the sigmoid activation function to the element-wise addition of their respective convolutional outputs and the Hadamard product of the previous cell state with learned weights. The input gate controls the amount of new information added to the cell state, while the forget gate determines how much past information is retained. The new cell state is then derived as a combination of the previous cell state (modulated by the forget gate) and the candidate cell state (modulated by the input gate), where the candidate is the corresponding convolutional output passed through an activation function (ReLU or tanh). Subsequently, the output gate is computed by applying a sigmoid activation to the element-wise addition of its corresponding convolutional output and the Hadamard product of the newly computed cell state with dedicated learned weights. Finally, the new hidden state is obtained through the Hadamard product of the output gate and the activated new cell state, where the activation function is also ReLU or tanh. The final outputs of the ConvLSTMCell are the new hidden state and new cell state. The internal operations of the ConvLSTM cell follow the equations introduced in [3]:

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f) \\
C_t &= f_t \odot C_{t-1} + \\
&\quad i_t \odot \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \odot C_t + b_o) \\
H_t &= o_t \odot \tanh(C_t)
\end{aligned}
$$

where:

- X_t is the input tensor at time step t,
- H_{t-1} and C_{t-1} are the hidden state and cell state tensors from the previous time step,
- σ denotes the sigmoid activation function,
- $*$ denotes the convolution operation,
- $\odot$ denotes the element-wise (Hadamard) product,
- $W_{xi}, W_{xf}, W_{xc}, W_{xo}$ are the convolutional weight matrices applied to the input X_t for the input, forget, cell candidate, and output gates respectively,
- $W_{hi}, W_{hf}, W_{hc}, W_{ho}$ are the convolutional weight matrices applied to the previous hidden state H_{t-1} for the respective gates,
- W_{ci}, W_{cf}, W_{co} are learned weight parameters for the element-wise multiplication with the previous or current cell state,
- b_i, b_f, b_c, b_o are the bias terms for the respective gates.

ConvLSTM processes image sequences by unrolling a single ConvLSTMCell over time. At each time step, it receives the current image along with the previous hidden and cell states, and produces updated states that serve as input to the next ConvLSTMCell. The hidden states across all time steps form the output sequence.

Seq2Seq stacks one or more ConvLSTM layers, each followed by batch normalization. The first layer processes raw input frames, while subsequent layers operate on the hidden states generated by earlier layers. The last frame of the output from the final ConvLSTM layer is passed through a convolutional layer, with a Sigmoid activation function applied to predict the next frame.

3.2.2. Encoder-Decoder ConvLSTM

Inspired by U-Net [4], we extended the ConvLSTM with an encoder-decoder architecture, as shown in Figure 2. The encoder consists of five blocks, each containing two convolutional layers followed by max-pooling, except for the last block, which excludes pooling. This progressively reduces the spatial dimensions by a factor of 16 while increasing the feature channels to 256. The compressed feature maps from the encoder serve as input to the ConvLSTM layers, which process temporal dependencies throughout the sequence. The output from the ConvLSTM is then passed to the decoder, which consists of four blocks. Each block performs upsampling followed by two convolutional layers to restore the spatial dimensions to their original size. A final 1×1 convolutional layer generates the predicted output frame. For simplicity, skip connections were omitted in this implementation.

This approach reduces computational complexity by decreasing the dimensionality of the input to the ConvLSTM, which enables the use of deeper recurrent layers.

4. Experimental Evaluation

4.1. Data Preprocessing

Radar images were processed using OpenCV [17] to reduce noise and decrease resolution. Morphological opening, a combination of erosion followed by dilation, was applied to disconnect thin connections

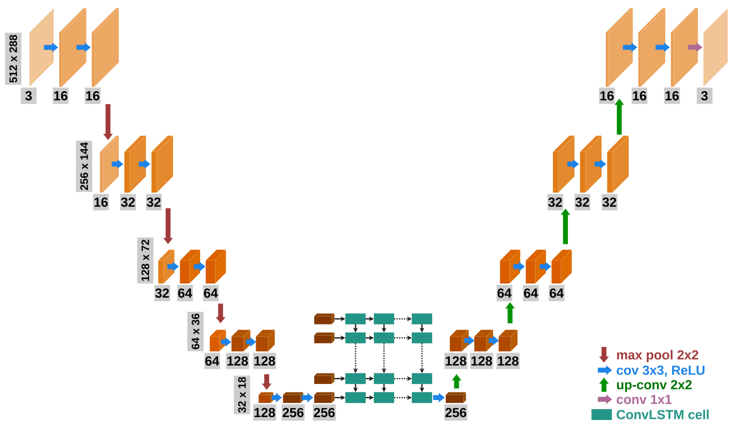

Figure 2: Architecture of the encoder-decoder ConvLSTM model. Orange boxes represent multi-channel feature maps, with the number of channels specified below each box. Spatial dimensions are shown on the left and correspond to each box at that level. Teal boxes denote ConvLSTM cells. Arrows indicate various operations. For simplicity, only the encoding of one input image is illustrated.

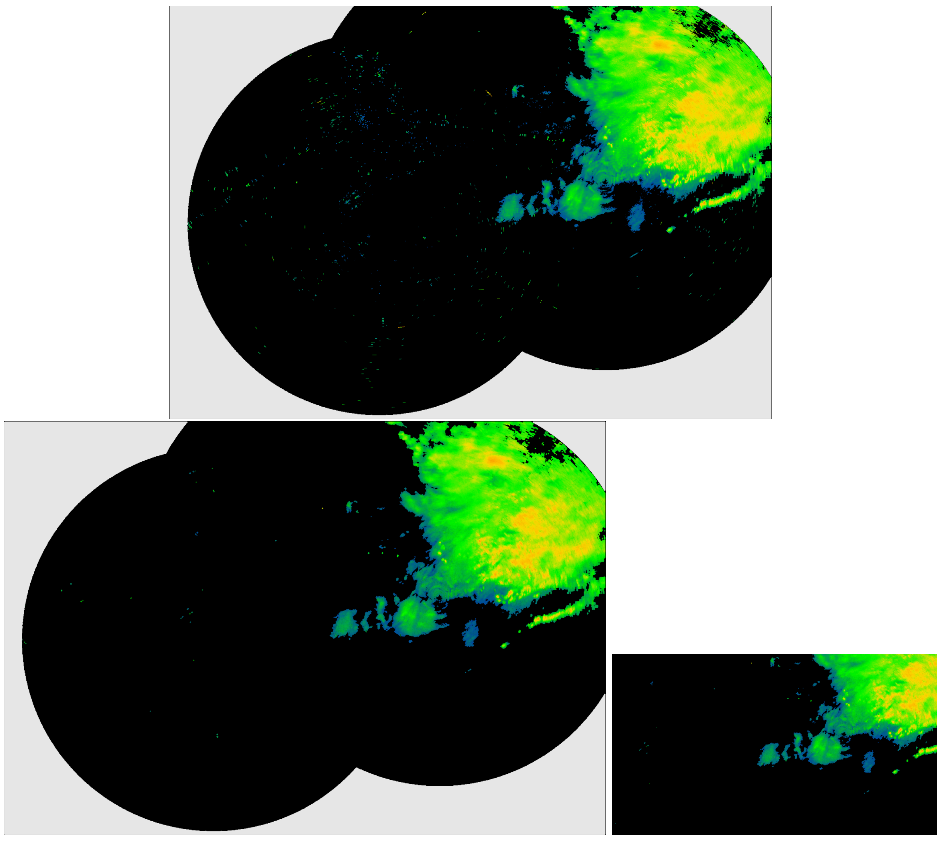

Figure 3: Up: original radar image. Middle: image after morphological processing. Down: cropped and resized image.

between objects, suppress noise, and smooth the images by eliminating irrelevant details. Subsequently, the images were cropped to a region primarily covering Slovakia and resized to 517×288 pixels (or 512×288 in the encoder-decoder model to maintain compatibility with max-pooling operations). The transformation process is illustrated in Fig. 3.

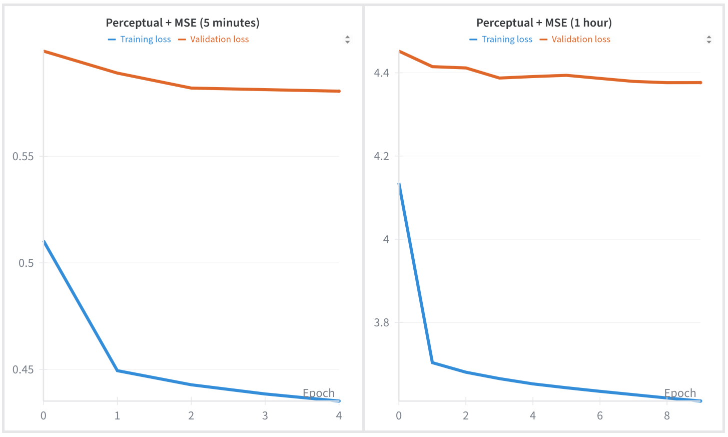

Figure 4: Training and validation loss curves for 5-minute and 1-hour interval models using combined MSE and Perceptual Loss.

4.2. Training

We experimented with different loss functions, including Mean Squared Error (MSE) and Perceptual Loss [18], both alone and in combination. Additionally, models were trained on sequences with time intervals of 5 minutes and 1 hour, and we tested their ability to predict one and four future frames. For training, we utilized a server with two NVIDIA A100 GPUs, both with 40GB of memory.

For the **baseline ConvLSTM** models, we used 50,000 images for 5-minute interval sequences, which we split into 40,000 for training, and 5,000 each for validation and testing. These images covered the period from September 24, 2021, 00:00 to March 16, 2022, 14:40. (We also tested a larger dataset of 100,000 images without significant performance improvement.) For 1-hour interval sequences, we applied the same split using images from January 1, 2016, 00:00 to September 14, 2021, 08:00.

All models contained a single ConvLSTM layer with 20 cells, matching the input sequence length. The input had 3 channels, corresponding to the color channels of the images. We used 32 convolutional kernels of size 3×3 with ReLU activation. For models predicting a single output frame, a batch size of 8 was used; for models predicting four frames, the batch size was set to 1 due to memory constraints.

Models trained on 5-minute interval sequences were trained for 5 epochs, whereas models trained on 1-hour interval sequences were trained for 10 epochs. During training, we monitored both training and validation loss, and to avoid overfitting, the training was stopped when the validation loss no longer decreased while the training loss continued to decrease. Figure 4 shows an example of the loss evolution for 5-minute and 1-hour models using the combined MSE and Perceptual Loss. Each epoch lasted approximately 1.5 hours for 5-minute models and 45 minutes for 1-hour models, resulting in a total training time of around 7.5 hours per model. The batch size was set as large as allowed by the available GPU memory. A similar approach was applied to the encoder-decoder model, where batch size and number of epochs were chosen based on memory constraints and validation loss convergence.

For the **encoder-decoder model**, we used 80,000 images with a 5-minute interval, spanning from September 24, 2021, 00:00 to June 18, 2022, 18:40. We split the dataset into 64,000 training, 8,000 validation, and 8,000 testing images.

The encoder and decoder were first trained jointly as an autoencoder for 80 epochs, divided into four phases of 20 epochs each. This training took approximately 16 hours in total. The pretrained encoder and decoder were then integrated into a ConvLSTM network, where the input to the ConvLSTM part was of size $20 \times 256 \times 32 \times 18$, corresponding to the encoder's output. Compared to the baseline model's input size of $20 \times 3 \times 512 \times 288$, this reduces the input to the ConvLSTM network by a factor of three. It is possible to reduce the input size even more, but for the purpose of this paper, we find it sufficient, since it enabled us to utilize more ConvLSTM layers without encountering memory issues. The total number of trainable parameters was approximately 14.3 million for the baseline model and 13.8 million for the encoder-decoder ConvLSTM model.

Due to the complexity of the problem and the input of the network, we initially decided to use a more complex approach with 8 ConvLSTM layers, each containing 20 cells corresponding to the input sequence length. The number of input channels for the ConvLSTM was set to 256, matching the encoder's output channels. We used 128 convolutional kernels of size 3×3 with ReLU activation, and the batch size was set to 24.

Training was divided into two phases, with the first phase consisting of 10 epochs and the second phase consisting of 5 epochs. During the first phase, encoder and decoder weights were frozen. Once loss convergence plateaued, the weights were unfrozen for fine-tuning in the second phase. This approach was inspired by the training process commonly used in transfer learning [19], where a pretrained network (in our case, the encoder-decoder model) with frozen weights is expanded by new layers (in our case, ConvLSTM layers) for a specific task and by training only weights in the new layer changes. When training is done, all weights are unfrozen and fine-tuned.

Total training time was approximately 18 hours, averaging 1 hour and 12 minutes per epoch.

As shown in Section 5, the results of 8 layers show significant problems with color accuracy. After consulting this problem, inspired by research [20], we also tried to reduce the number of ConvLSTM layers to 5 and trained a new model following the same approach. This model had a total of approximately 9.6 million trainable parameters.

The code used for our experiments can be found on the GitHub repository: https://github.com/bszepesiova/Cloud-and-Precipitation-Forcasting-Using-Convolutional-LSTM.

4.3. Evaluation of Models

Models were evaluated using MAE and SSIM [21] on the original test set prepared for the encoder-decoder model, which contained 8,000 images. Since the images were normalized, the MAE values range from 0 to 1, with lower values indicating more accurate predictions. SSIM measures image similarity based on luminance, contrast, and structure and ranges from -1 to 1, where higher values reflect better structural and visual similarity to the target.

Evaluation considered two aspects. First, the quality of the predicted outputs was assessed in terms of color fidelity, structure, and overall visual similarity to the target images. This evaluation was performed on the raw model outputs and their corresponding target images. Second, the shape and movement of precipitation were analyzed independently of visual quality by binarizing the images to indicate cloud presence or absence. Binarized comparisons using true/false positive and negative metrics, inspired by [7], were used solely to evaluate the precipitation shape and motion, unaffected by the color or visual details of the predictions. We decided on this binarized evaluation because accurately capturing precipitation patterns is often more important for practical use than producing visually appealing images.

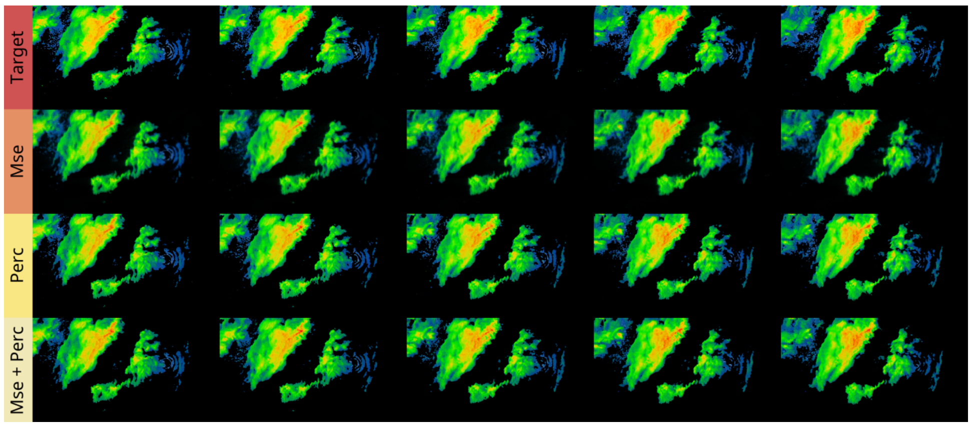

Figure 5: Comparison between target images and baseline ConvLSTM model predictions using different loss functions (MSE, perceptual loss, and a combination of both) at a 5-minute interval.

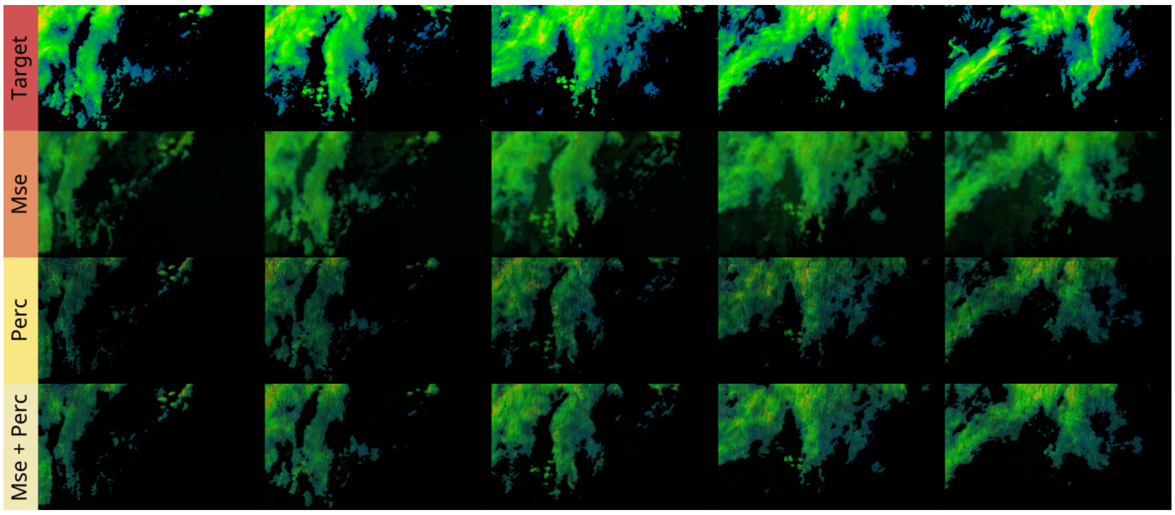

Figure 6: Comparison between target images and baseline ConvLSTM model predictions using different loss functions (MSE, perceptual loss, and a combination of both) at a 1-hour interval.

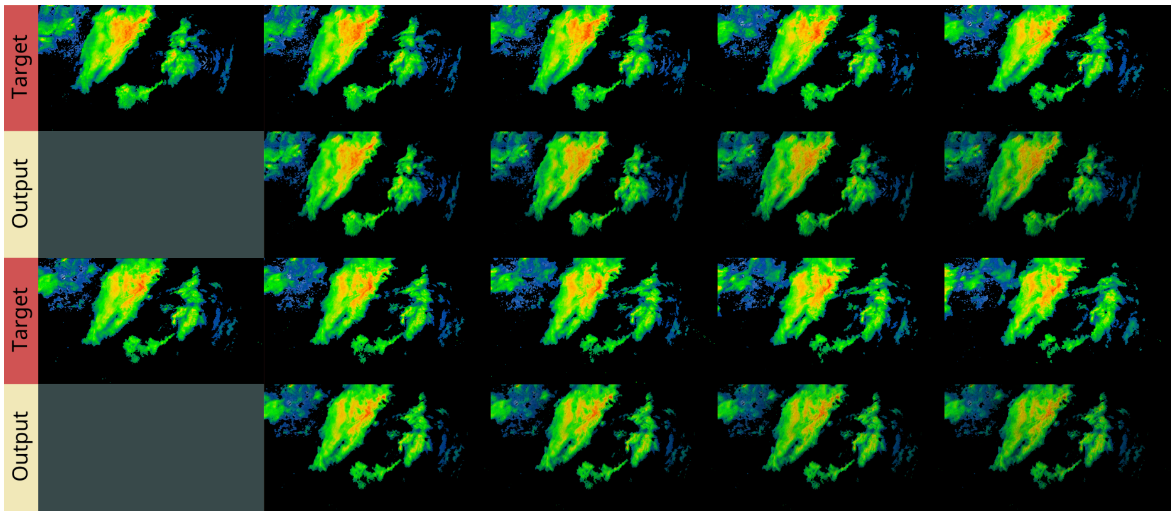

Figure 7: Two examples of the comparison between target images and predictions of the model using a combination of MSE and perceptual loss at a 5-minute interval, predicting four consecutive frames. On the target row, the first image is the last image of the network input sequence. All predicted images are very much alike this last image.

5. Results and Discussion

The **baseline ConvLSTM** predicted the last input frame rather than the target, which is likely due to the network's limited depth. This can be seen in Fig. 5, where each image in the second, third, and fourth row closely resembles the target image from the preceding column, corresponding to the final frame of the input sequence.

At 5-minute intervals, the images were of higher quality and appeared sharper because the differences between the last input frame and the target frame were small, as illustrated in Fig. 5. At 1-hour intervals, the model also tended to predict frames similar to the final input frame; however, these images were noticeably less sharp and of lower quality, as shown in Fig. 6.

This behavior can also be observed in the model predicting four consecutive frames, as shown in Fig. 7. All four predicted images exhibit the same shape, closely resembling the final target image three rows above, which corresponds to the last input frame. The shapes in the predicted frames remained static and gradually decreased in intensity over time.

MSE caused blurring, Perceptual Loss reduced color intensity, and their combination balanced

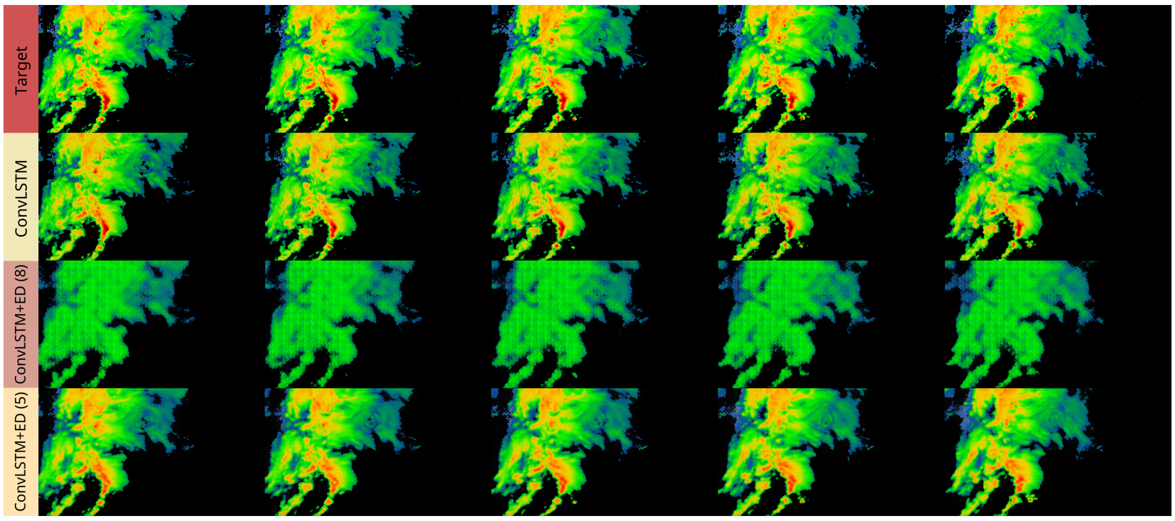

Figure 8: Comparison between target images and predictions of the baseline model and the encoder-decoder models using 8 and 5 ConvLSTM layers.

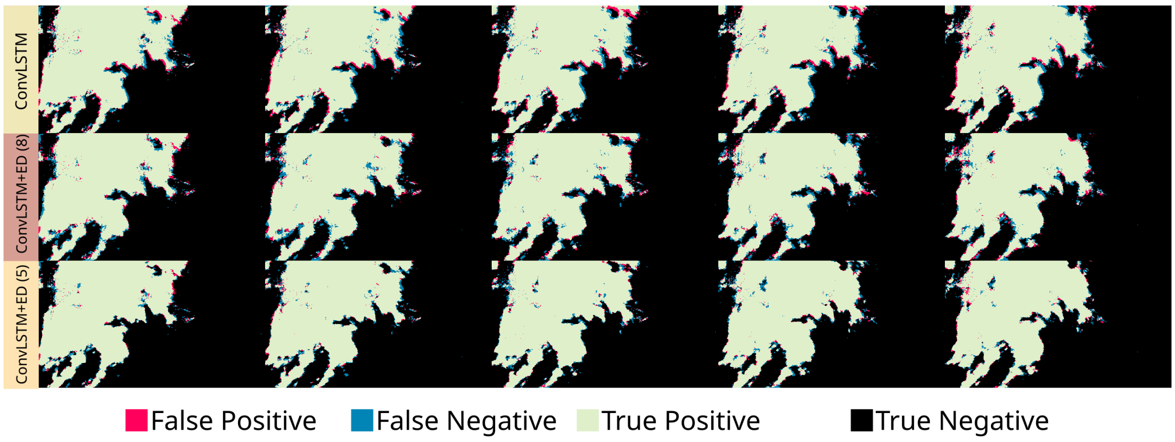

Figure 9: Differences between target and predicted binary images of the baseline model and the encoder-decoder model using 8 and 5 ConvLSTM layers. Red indicates areas where the model incorrectly predicted cloud presence, blue indicates areas where it incorrectly predicted cloud absence, black indicates correctly predicted cloud absence, and light yellow indicates correctly predicted cloud presence.

sharpness and color. Therefore, we decided to employ this combined loss in the second model.

The **encoder-decoder** model with 8 layers predicted images that lacked yellow and red tones and were generally of lower quality. After reducing the number of ConvLSTM layers to five, the image quality improved significantly, as shown in Fig. 8.

For further analysis, we binarized the predicted and target images using a threshold of 0.4 to indicate cloud presence (1) or absence (0). Pixels were then classified as true positive (light yellow), true negative (black), false positive (red), or false negative (blue), following the scheme in Fig. 9. The choice of 0.4 was made empirically, as illustrated in Fig. 10, which shows target images together with contours of their binarized versions at thresholds 0.1, 0.4, 0.5, and 0.6. Thresholds between 0.1 and 0.4 produced very similar results and accurately captured the precipitation patterns, while thresholds of 0.5 and above failed to cover the full extent of the precipitation areas, leading to a visible loss of information.

The baseline model yielded the best MAE and SSIM scores on the non-binarized images as shown in Table 2. The encoder-decoder models showed lower performance on these images, possibly due to inaccuracies caused by the decoder, as well as the increased depth of the network. The baseline model tends to predict frames very similar to the last input frame, resulting in predictions with more accurate structures compared to those of the encoder-decoder models. On the other hand, the encoder-decoder

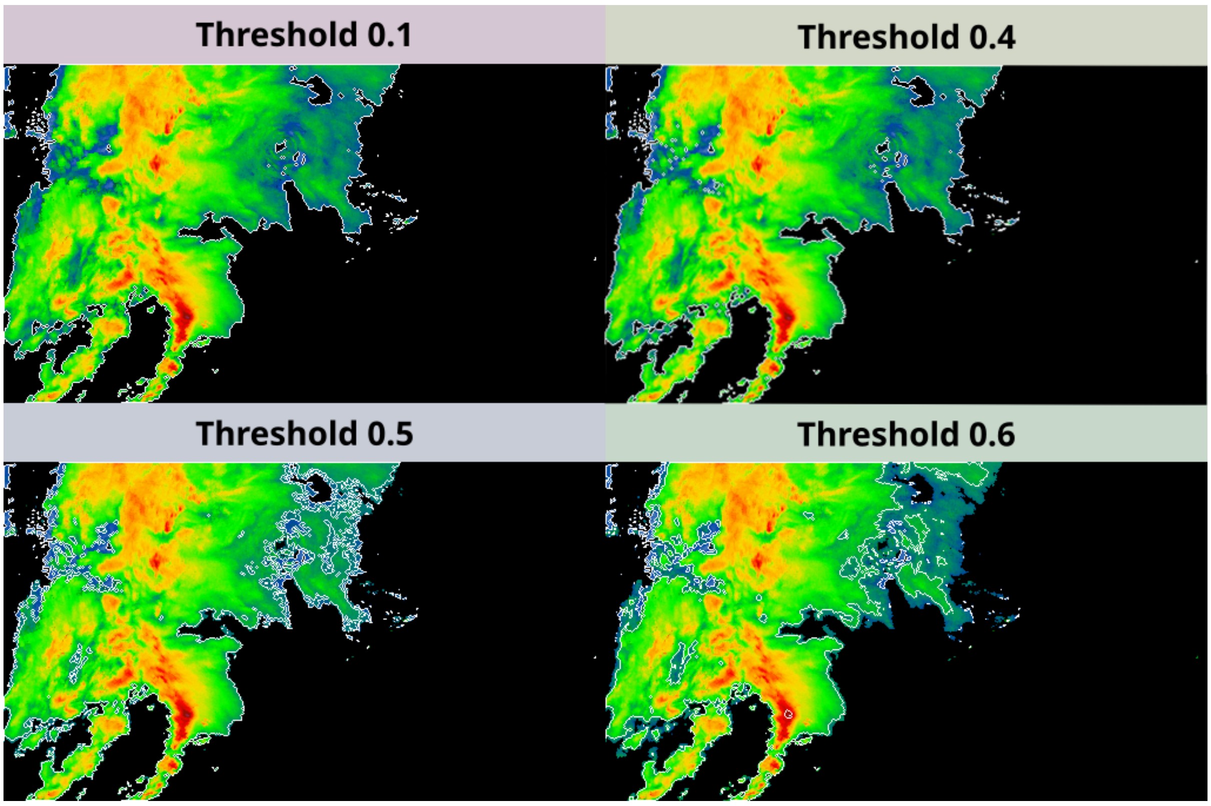

Figure 10: Target images with contours of their binarized versions at thresholds 0.1, 0.4, 0.5, and 0.6.

Table 2
Comparison of ConvLSTM models using MAE and SSIM metrics.

Model	MAE	SSIM	MAE (Binary)	SSIM (Binary)
ConvLSTM	**0.0183**	**0.8526**	0.0314	0.8433
ConvLSTM + ED (8)	0.0282	0.8040	0.0342	0.8382
ConvLSTM + ED (5)	0.0190	0.7755	**0.0268**	**0.8575**

Table 3
Comparison of binarized model outputs and targets using MAE and SSIM with different binarization thresholds ranging from 0.2 to 0.7.

Model	0.2		0.3		0.5		0.6		0.7	
	MAE	SSIM	MAE	SSIM	MAE	SSIM	MAE	SSIM	MAE	SSIM
ConvLSTM	0.0314	0.8468	0.0311	0.8475	0.0431	0.8279	0.0355	0.8481	0.0160	0.9076
ConvLSTM + ED (8)	0.0349	0.8407	0.0338	0.8415	0.0440	0.8249	0.0383	0.8453	0.0215	0.9064
ConvLSTM + ED (5)	**0.0260**	**0.8627**	**0.0259**	**0.8622**	**0.0361**	**0.8424**	**0.0281**	**0.8684**	**0.0132**	**0.9293**

model with 5 ConvLSTM layers obtained the best results on the binarized images, outperforming both the baseline and the encoder-decoder model with 8 layers. Additional results for thresholds ranging from 0.2 to 0.7 are provided in Table 3, where the 5-layer encoder-decoder model consistently outperformed other models on the binarized images across all thresholds.

Although the baseline model shows better metric values in Table 2, predicting the last input frame limits its practical value, resulting in worse overall performance than the encoder-decoder models.

6. Conclusion and Future Works

This study provides an evaluation of ConvLSTM architectures for meteorological prediction tasks, demonstrating both the potential and limitations of these approaches when applied to radar data from the SHMÚ. Two variants of the ConvLSTM network were used, trained with different loss functions (MSE, Perceptual Loss, and their combination). Most of the tested combinations of used network and loss functions failed to produce sharp images, preserve colors, or capture movement patterns. Only the encoder-decoder variant with 5 ConvLSTM layers shows superior performance in capturing cloud movement patterns (with a slight reduction in image quality), particularly evident in binarized image analysis.

Future work should focus on optimizing multi-step predictions and conducting more experiments with more hyperparameters, incorporating additional meteorological data (e.g., wind direction, temperature), or exploring other architectures, such as DYffusion [22]. It might also be interesting to explore the use of a Variational Autoencoder [23] instead of the original autoencoder, as it models a well-structured and continuous latent space where even randomly selected or interpolated latent representations between training samples produce coherent and realistic images.

Acknowledgments

We would like to express our sincere gratitude to the Slovak Hydrometeorological Institute (shmu.sk) for generously providing the data that were essential for this research.

Declaration on Generative AI

During the preparation of this work, the authors utilized ChatGPT, Grok, and Grammarly to translate, paraphrase, refine the writing style, and verify the grammar and spelling of the entire paper, as well as to draft sections of the paper, including the Abstract, Introduction, and Conclusion. After using these tools/services, the authors reviewed and edited the content as needed, taking full responsibility for the publication's content.

References

[1] A. B. of Meterorology, How does a weather radar work?, Available on the Internet: https://media.bom.gov.au/social/blog/1459/how-does-a-weather-radar-work/, 2017. [cit. 2. 7. 2025].

[2] W. contributors, Numerical weather prediction — Wikipedia, the free encyclopedia, Available on the Internet: https://en.wikipedia.org/w/index.php?title=Numerical_weather_prediction&oldid=1185267885, 2023. [cit. 2. 7. 2025].

[3] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional lstm network: A machine learning approach for precipitation nowcasting, Advances in neural information processing systems 28 (2015).

[4] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, Springer, 2015, pp. 234–241.

[5] K. Singh, Types of satellite imagery, Available on the Internet: https://pangeography.com/types-of-satellite-imagery/, 2023. [cit. 2. 7. 2025].

[6] E. Shi, Q. Li, D. Gu, Z. Zhao, Convolutional neural networks applied on weather radar echo extrapolation, DEStech Trans. Comput. Sci. Eng (2017).

[7] L. Berthomier, B. Pradel, L. Perez, Cloud cover nowcasting with deep learning, in: 2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA), IEEE, 2020, pp. 1–6.

[8] Y. Wang, M. Long, J. Wang, Z. Gao, P. S. Yu, Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms, Advances in neural information processing systems 30 (2017).

[9] D. N. Tuyen, T. M. Tuan, X.-H. Le, N. T. Tung, T. K. Chau, P. Van Hai, V. C. Gerogiannis, L. H. Son, Rainpredrnn: A new approach for precipitation nowcasting with weather radar echo images based on deep learning, Axioms 11 (2022) 107.

[10] C. Kaparakis, S. Mehrkanoon, Wf-unet: Weather fusion unet for precipitation nowcasting, arXiv preprint arXiv:2302.04102 (2023).

[11] D. Li, Y. Liu, C. Chen, Msdm v1. 0: A machine learning model for precipitation nowcasting over eastern china using multisource data, Geoscientific Model Development 14 (2021) 4019–4034.

[12] P. Demetrakopoulos, Short-term precipitation forecasting in the netherlands: An application of convolutional lstm neural networks to weather radar data, arXiv preprint arXiv:2312.01197 (2023).

[13] S. Imran, T. Anuradha, R. Bharat, Radar based precipitation nowcasting prediction by using deep learning techniques, in: E3S Web of Conferences, volume 405, EDP Sciences, 2023, p. 04003.

[14] SHMÚ, Slovenská rádiolokačná sieť, Available on the Internet: https://www.shmu.sk/sk/?page=1566, 2025. [cit. 2. 7. 2025].

[15] R. Panda, Video frame prediction using convlstm network in pytorch, Available on the Internet: https://sladewinter.medium.com/video-frame-prediction-using-convlstm-network-in-pytorch-b5210a6ce582/, 2021. [cit. 2. 7. 2025].

[16] J. Kadupitiya, G. Fox, V. Jadhao, Survey on deep learning models for time series data, 2020. doi:10.13140/RG.2.2.26413.92649.

[17] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools (2000).

[18] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: CVPR, 2018.

[19] F. Chollet, Transfer learning & fine-tuning, Available on the Internet: https://keras.io/guides/transfer_learning/, 2023. [cit. 2. 7. 2025].

[20] M. Bock, A. Hölzemann, M. Moeller, K. Van Laerhoven, Improving deep learning for har with shallow lstms, in: Proceedings of the 2021 ACM International Symposium on Wearable Computers, 2021, pp. 7–12.

[21] P. Datta, All about structural similarity index (ssim): Theory + code in pytorch, Available on the Internet: https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e, 2020. [cit. 2. 7. 2025].

[22] S. Rühling Cachay, B. Zhao, H. Joren, R. Yu, Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting, Advances in neural information processing systems 36 (2023) 45259–45287.

[23] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).