Radar Nowcasting with Persistence and Deep Learning (U-Net) Models

Paul Lacotte

July 2, 2025

1 Introduction

This project focuses on short-term precipitation forecasting (nowcasting) using radar reflectivity data. Two models are compared:

- A simple **persistence model**, which assumes that the radar image at time $t+1$ is the same as at time t.
- A **deep learning model** based on a simplified **U-Net** architecture, which predicts the next radar frame using several previous frames.

2 Dataset Description

The radar dataset is stored in `NetCDF` format and contains reflectivity values over a spatial grid and across time.

Using the function `load_radar_dataset()`, we load the main variable and convert it to a NumPy array. A quick summary is printed to understand the data distribution (min, max, mean, standard deviation, and NaNs). The first few radar frames are also plotted for visualization.

Key preprocessing steps include:

- Visualizing first radar maps
- Checking the distribution of values
- Building input/output sequences for training using the function `create_unet_sequences()`

This yields a dataset shaped as (N, H, W, n_{input}) for inputs and $(N, H, W, 1)$ for targets.

3 Baseline Model: Persistence

The **persistence model** serves as a naive baseline. It simply copies the last input frame as the prediction for the next timestep. Despite its simplicity, this model can yield surprisingly reasonable predictions for short-term nowcasting, particularly when the radar pattern evolves slowly.

4 Deep Learning Model: U-Net

The deep learning model is a compact version of the **U-Net** architecture. It features:

- 2 downsampling blocks
- A bottleneck
- 2 upsampling blocks with skip connections

The model takes a stack of radar images (e.g., 3 time steps) and outputs the predicted frame at $t+1$. It is trained with the **Mean Squared Error (MSE)** loss and evaluated using **Root Mean Squared Error (RMSE)** and **Mean Absolute Error (MAE)**.

Training settings:

- Optimizer: Adam
- Batch size: 5
- Epochs: configurable via CLI (default = 5)

5 Evaluation

Model evaluation is based on the RMSE, calculated on the test set using:

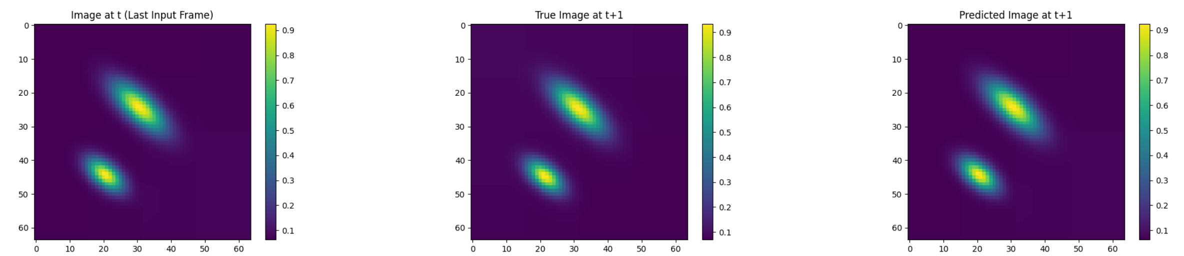
```
np.sqrt(mean_squared_error(y_true.flatten(), y_pred.flatten()))
```

Visualizations include:

- Last input frame
- Ground truth for $t+1$
- Predicted frame
- Error heatmap

Baseline (Persistence) Model

RMSE: for the RMSE we have 0.0138 for the Unet and 0.0220 for the baseline and in terms of error map:



(a) Last Input Frame (b) Ground Truth (c) Prediction

Figure 1: Persistence Model Prediction

Error map:

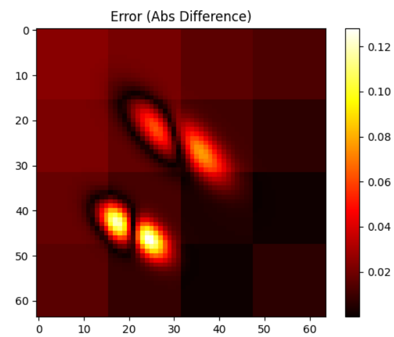
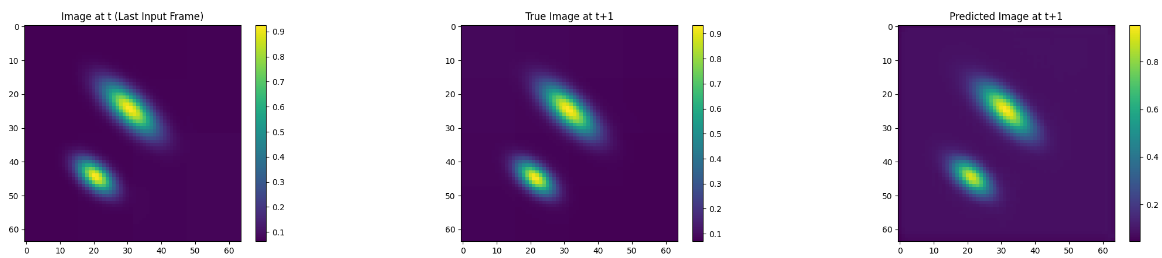


Figure 2: Error Map - Persistence Model

Deep Learning Model (U-Net)

RMSE: *(e.g., 2.187)*



(a) Last Input Frame (b) Ground Truth (c) Prediction

Figure 3: U-Net Model Prediction

Error map:

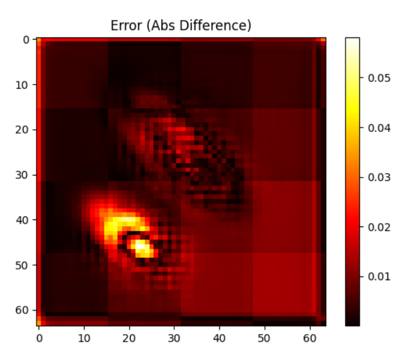


Figure 4: Error Map - U-Net

6 Observations

- The **U-Net model clearly outperforms** the baseline in terms of prediction quality and error distribution.
- The **error maps** show that the U-Net makes finer adjustments in areas of rapid change (e.g., storm edges), where the persistence model fails.
- Training time is relatively short (¡2s) given the small size of the U-Net, making it suitable for fast experimentation.

7 Future Improvements

- Use **ConvLSTM** instead of U-Net to better capture spatiotemporal dependencies.
- Include **more input frames** (e.g., 5 or 6 past images).

8 Code Availability and Execution

All functions are wrapped in a CLI interface using `argparse`, allowing execution like:

```
python nowcast.py --data_path radar_data.nc --model unet --epochs 10
```

The models and plots are automatically saved to disk for later analysis.