**Introduction**

This project investigates whether deep learning models can improve the accuracy and spatial interpretability of monthly weather forecasting. Specifically, I compare LSTM and ARIMA models to evaluate improvements in temporal prediction using historical data from weather stations in Maryland. I then extend the analysis by introducing a CNN-LSTM model that incorporates spatial structure across stations. The goal is twofold: first, to show that LSTM outperforms ARIMA in capturing nonlinear seasonal trends; second, to examine whether CNN-LSTM can better represent spatial dependencies in the learned hidden states, as measured by Moran’s I. This dual focus allows for a systematic evaluation of both temporal and spatial modeling advantages offered by deep learning methods.

**Data Source and Preprocessing**

I used daily weather data from the Global Historical Climatology Network – Daily (GHCN-D) via the NOAA Climate Data Online (CDO) API, focusing on stations in Maryland (FIPS: 24) from 2010 to 2024. Out of 752 total GHCND stations in Maryland, I retrieved data for 473 stations that had at least some records for temperature or precipitation. After filtering and aggregating the data to monthly resolution, only 47 stations had sufficient data to be retained.

I converted raw values by dividing by 10 to obtain standard units (°C, mm), computed daily average temperature as the mean of TMAX and TMIN, and excluded days with missing temperature values. Among the 47 stations, only 14 had full temporal coverage across all months. However, in order to utilize as much spatial information as possible, I applied a binary mask during model fitting to handle missing values. As a result, all 47 stations were included in the analysis.

**Comprare LSTM to ARIMA**

To compare deep learning and traditional approaches, I selected a spatially indexed weather station in Maryland that had continuous monthly records from 2010 to 2024. I focused on four target variables: average temperature (AVG), minimum temperature (TMIN), maximum temperature (TMAX), and precipitation (PRECIP).

For the traditional baseline, I implemented an ARIMA(1,1,0) model separately for each of the four variables, using a sliding window of 16 months as historical input. Forecasts were generated iteratively across the test set, and the model was retrained for each prediction point.

For the deep learning model, I implemented a Long Short-Term Memory (LSTM) network using PyTorch. The model takes a 16-month window of four normalized input features and predicts the values for the next month. The network architecture includes:

- An LSTM layer with 64 hidden units

- A fully connected output layer mapping to 4 variables

- A mean squared error loss function and Adam optimizer

The first 80% of the time series (2010 to mid-2022) was used for training, and the remaining 20% (mid-2022 to late 2024) served as the test set for evaluation and forecasting. All features were normalized using MinMaxScaler, and precipitation values were log-transformed during preprocessing and reversed after prediction because precipitation has extreme peaks.

I evaluated both models using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) on the test set. The results are summarized below:

| Variable | Model | MAE | RMSE |
| --- | --- | --- | --- |
| AVG | LSTM | 1.143 | 1.477 |
|  | ARIMA | 3.221 | 3.670 |
| PRECIP | LSTM | 38.575 | 47.951 |
|  | ARIMA | 48.822 | 63.262 |
| TMIN | LSTM | 1.156 | 1.518 |
|  | ARIMA | 3.373 | 3.826 |
| TMAX | LSTM | 1.314 | 1.698 |
|  | ARIMA | 3.246 | 3.769 |

The LSTM model consistently outperformed the ARIMA model across all variables, particularly for temperature metrics where ARIMA exhibited large prediction errors. The improvement in precipitation forecasting was also noticeable, although both models showed higher variance due to the inherent noise in precipitation patterns.图表, 折线图

AI 生成的内容可能不正确。

The plots compare observed weather variables (blue) with forecasts from an LSTM model (orange) and an ARIMA model (green) between mid-2022 and late 2024. For temperature-related variables (AVG, TMIN, TMAX), the LSTM model closely follows seasonal trends, while ARIMA shows noticeable lag and local overfitting. For precipitation, both models face challenges due to its high variability and irregular distribution. Interestingly, ARIMA attempts to capture fluctuations, whereas the LSTM outputs appear overly smoothed and tend toward the mean. Overall, LSTM provides more stable and accurate predictions for temperature, but struggles to represent the sharp dynamics of rainfall.

**Compare CNN-LSTM with LSTM**

To incorporate spatial information across weather stations, I compared two deep learning architectures: a basic LSTM and a convolutional LSTM (CNN-LSTM). Both models were trained using monthly weather data from 47 stations in Maryland, using four features: average temperature (AVG), maximum temperature (TMAX), minimum temperature (TMIN), and precipitation (PRECIP).

**LSTM Model**: I implemented a baseline LSTM using PyTorch. For each station, the input was a 12-month sequence of normalized features. The model used a shared LSTM layer followed by a fully connected output layer predicting all four variables per station. Dropout was applied to reduce overfitting.

**CNN-LSTM Model**: I implemented a three-layer ConvLSTM that explicitly modeled spatial structure. To do this:

* I projected station coordinates onto a regular 2D grid using nearest-neighbor interpolation.
* At each time step, spatial weather data were mapped into a pseudo-image grid.
* Each ConvLSTM cell updated spatial-temporal hidden states, followed by convolutional fusion layers producing the final prediction.

A binary mask was applied during training and evaluation to handle missing stations dynamically, and both models were trained using a masked MSE loss, optimizing only over stations with available observations. Inputs and targets were normalized using z-scores and scaled back for evaluation.

I evaluated both models on a held-out test set using RMSE, MAE, and R². The results are summarized below:

| CNN-LSTM | RMSE | MAE | R² |
| --- | --- | --- | --- |
| AVG | 3.18 | 2.53 | 0.835 |
| TMAX | 3.41 | 2.74 | 0.813 |
| TMIN | 3.18 | 2.55 | 0.837 |
| PRECIP | 53.51 | 42.19 | -0.286 |

| LSTM | RMSE | MAE | R² |
| --- | --- | --- | --- |
| AVG | 2.31 | 1.80 | 0.913 |
| TMAX | 2.41 | 1.89 | 0.907 |
| TMIN | 2.41 | 1.89 | 0.906 |
| PRECIP | 48.35 | 37.89 | -0.050 |

The LSTM model slightly outperformed CNN-LSTM in both temperature and precipitation forecasting across most metrics. Although CNN-LSTM tried to model spatial correlations, its performance gain was not evident in this study, potentially due to noise or the relatively small number of grid-aligned stations.

The figure below shows model predictions from both LSTM and CNN-LSTM compared to ground truth for one station among the 14 stations with complete data: Both models capture the strong seasonal patterns of temperature variables.

* LSTM predictions (green dashed) are slightly closer to the true values (solid blue) than CNN-LSTM (orange dashed) for most variables.
* Precipitation remains noisy and challenging for both models, with CNN-LSTM occasionally overshooting peaks.

图表, 折线图

AI 生成的内容可能不正确。

**Spatial Structure Comparison**

However, since the focus of this project is geospatial analysis, I further evaluated whether the CNN-LSTM model was better at capturing meaningful spatial patterns in the learned representations compared to a standard LSTM.

To do this, I extracted the hidden states from both models across time and projected them into a common representation space using Principal Component Analysis (PCA). I then computed Moran’s I, a classic spatial autocorrelation statistic, on the first principal component at each time step. Higher values of Moran’s I indicate stronger spatial clustering.

The analysis was conducted across 47 stations using 8-nearest neighbors to define the spatial weight matrix. The results were: LSTM Moran’s I: mean = –0.0209, std = 0.0403, CNN-LSTM Moran’s I: mean = –0.0163, std = 0.0539.

To formally test whether the CNN-LSTM captured spatial dependencies better than LSTM, I performed a permutation test on the difference in mean Moran’s I values across time. The result was: Observed difference = 0.0046, p-value = 0.1595.

The difference was not statistically significant at the 0.05 level, suggesting that although CNN-LSTM is structurally designed to model spatial relationships, this advantage did not capture stronger spatial autocorrelation in the hidden representations in this case.

**Conclusion**

This project demonstrates that deep learning models, particularly LSTM and CNN-LSTM, provide more accurate monthly weather forecasts than traditional statistical approaches like ARIMA. Across all key variables—average temperature, maximum and minimum temperature, and precipitation—LSTM consistently outperformed ARIMA in capturing temporal trends and reducing prediction error.

Although CNN-LSTM introduces spatial structure through convolutional operations over the station grid, it did not lead to clear improvements over LSTM in this case. Neither the predictive accuracy nor the spatial coherence of hidden states, as measured by Moran’s I, showed significant gains from the added spatial modeling.

Several factors likely contributed to this outcome. The geospatial data contained substantial missingness and uneven coverage, which limited the effective use of spatial information. Only 14 out of 47 stations had complete data across the entire period, and the rest required binary masking during training. Additionally, the relatively small number of spatial locations may have been insufficient for the CNN-LSTM to learn robust spatial correlations. In summary, while deep learning models offer clear advantages in accuracy, their potential to enhance spatial interpretability has not been proven in this case.

**Data Source**

[**https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by\_year**](https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year)