Weather Forecasting from Time Series

Mircea-Ionuț Meche

***Abstract*—This project was undertaken with the goal of understanding time series in the context of forecasting, in particular, weather forecasts, and how machine learning systems can be used for this purpose. To that end, three notable architectures were considered for training and testing, and three ablation studies have been performed with the goal of finding an optimal solution.**

I. State of the Art

Neural networks have seen a great rise in usage in the last few years, and one of the most prevalent use cases has been that of forecasting based on recorded data, arranged as time series. To that end, technological progress, as well as advancements in the field, have shifted the dynamic, thus various solutions to forecasting had arrived and left in the last 10 to 15 years.

The authors of [1] have thoroughly gathered several architectures and compared the their advantages and disadvantages, while also highlighting their evolution throughout the years. Support Vector Machines (SVM) were among the first to be used in such tasks, and remained the dominant architecture until the arrival of neural networks, albeit in a rudimentary state compared to today's deep architectures with multiple hidden layers. Examples of such shallow neural networks include the Multi-Layer Perceptron (MLP) and error Back Propagation (BP) systems.

One particular breed of classical networks that became relevant in recent years are Recurrent NNs (RNNs), which are noted in [1] to have the ability to memorize data and thus improve forecasting. RNNs became the base of much more capable deep architectures, represented by the Long-Short Term Memory recurrent network, which used a "forget gate" to filter out unnecessary information. An ulterior improvement of the LSTM system is the Gated Recurrent Unit (GRU), which simplifies the architecture by combining the forget gate with the input gate, allowing for much smaller architectures that manage to keep an acceptable level of performance.

Finally, on the topic of deep neural networks, the Convolutional Neural Network (CNN) has become the de facto solution for a great deal of problems, ranging from classification to regression problems, and that includes forecasting, in no small part thanks to their flexibility in dimensionality and computation methods.

Going back to the RNN derivatives, the authors of [2] used these architectures to forecast traffic flow, and successfully showcased the increased performance compared to a classical Auto-Regressive Integrated Moving Average (ARIMA) architecture. Meanwhile. works such as [3] and [4] showcase a hybrid architecture that combines the LSTM and CNN architectures in order to gain greater performance in short term forecasting and to reduce the necessary training time. In addition, it allows the architecture to retain both size and time dependent information, thus proving to be more flexible.

II. Methods and Dataset

The project was created using the given guidelines, as well as several tutorials detailing the methodology, those being [7], [8] and [9]. Thus, the basic structure is to initialize and analyze the dataset, following that with processing techniques before the training stage.

For the purpose of weather forecasting, a Kaggle dataset, detailed in [5], was used. The reason why I have chosen this dataset over the more established Jena climate dataset used in [7] is because this dataset seems more refined, as no specific processing is required, as would be the case with the "wind speed" parameter in the case of Jena Climate. Otherwise, it is very similar and allows the same logic to be used. Weather forecasting specific datasets are not as prevalent as those meant for other tasks, such as stock price time series sets. If I had the time, I would also like to try testing on the WEATHER-5K dataset [11], as it is a relatively new and comprehensive dataset for weather forecasts. Still, the chosen set is quite robust, with several parameters available. such as temperature, atmospheric pressure, cloud coverage, precipitation value and so on, so it will do for the scope of this project.

The dataset is first inspected for periodicity. Looking at monthly recorded statistics, seasonality does seem to affect most of the recorded parameters. Precipitation and temperature maps show that temperature and humidity tends to be higher around the middle of the year, in particular precipitations are most present around August, while the highest temperatures are recorded in April. Meanwhile, an analysis on a larger scale shows that temperatures are increasing yearly starting around the beginning of the current millennium, thus showcasing the consequences of global warming.

After visualization, the dataset is normalized and split for training and validation with an approximate 70-30 split respectively. This gives us the opportunity to record the previously stated characteristics of the weather as recorded (high temperatures in the first third of the year, yearly increasing temperatures and so on). The evaluation metric used for measuring the performance of the architectures is the mean square error.

III. Experiments and Results

The architectures that were chosen for this work are LSTM, GRU and CNN, due to their good performance and prevalence in research papers. In addition, they are easy to implement using libraries such as PyTorch [**?**, torch] After initialization, the models are trained and tested using only a portion of the parameters present in the weather data set. After that, a

prediction is made for the next 72 hours past the end of the test dataset and compared against the real data. This allows us to get a good baseline prior to ablation studies. The training is done in 10 epochs with the Adam optimizer, and the mean squared error is recorded. The prediction is made for the "temperature" parameter.

For more granular architecture designs, the LSTM and GRU both have a single layer and they are both unidirectional. In the case of the CNN it has a single convolutional layer, a Rectified Linear activation layer and a pooling layer, followed by a flattening layer for reducing dimensions and a final linear layer for the output.

TABLE I
BASELINE PERFORMANCE

Baseline	Train Loss	Test Loss
LSTM	0.0002	0.0004
GRU	0.0001	0.0005
CNN	0.0002	0.0005

Following the training and testing procedure, predictions for the next 72 hours were made and compared with the given data. Of the 3 models, the LSTM makes the most accurate prediction, followed by the GRU and CNN systems.

The first ablation study was concerned with data augmentation. To that end, the tests were performed again, but first on a dataset with more parameters, and afterwards on a dataset with only one parameter. Otherwise, the same testing procedure was used.

TABLE II
ENRICHED DATASET PERFORMANCE

1st Study	Train Loss	Test Loss
LSTM	0.0001	0.0006
GRU	0.0001	0.0005
CNN	0.0002	0.0006

From the testing data, it would appear that performance is slightly worse when combining more elements for the prediction, and the same is true for the predictions. A repeat of this situation occurs when data is only trained on one parameter, as now all models share the same performance metrics.

TABLE III
LIMITED DATASET PERFORMANCE

1st Study	Train Loss	Test Loss
LSTM	0.0002	0.0005
GRU	0.0002	0.0005
CNN	0.0002	0.0005

For the second ablation study, the models themselves were enriched. As shown in [10], Bidirectional LSTM gain greater accuracy because the training procedure is done both from left to right and vice-versa. As such, the LSTM and GRU architectures were replaced with bidirectional variants with 2 layers each in the hopes of improving performance. Meanwhile, in the case of the CNN, the group of [Convolutional layer, ReLU layer] was doubled and augmented with a dropout layer for better learning sessions. However, despite it being a feasible idea in theory, in practice my results show little to no improvement.

TABLE IV
LIMITED DATASET PERFORMANCE

2nd Study	Train Loss	Test Loss
LSTM	0.0002	0.0004
GRU	0.0001	0.0005
CNN	0.0002	0.0004

Finally, for the third and final ablation study, a new architecture was tested, that being the hybrid consisting of a LSTM chained with a CNN, each identical to the original architectures tested in the baseline. In theory, as discussed in the state of the art section, the performance should be improving, but once again, the results are stagnant in this instance.

TABLE V
LIMITED DATASET PERFORMANCE

3rd Study	Train Loss	Test Loss
Baseline LSTM	0.0002	0.0005
LSTM-CNN	0.0001	0.0005

IV. CONCLUSION

This study was quite limited in scope and applicability, as I wasn't able to properly give the project the time it deserved. It could also be that the complexity of the data and applied methods in question simply don't allow for vast of significant improvements. Even so, this has been an interesting project and I would like to see how it can be improved upon in the future. Until then, the conclussion is that a simple LSTM implementation has allowed for an acceptable level of performance while maintaining a simple architecture. The Jupyter Notebook used in this project, as well as the data, can be found in the corresponding Github repository.

REFERENCES

[1] Z. Han, J. Zhao, H. Leung, K. F. Ma and W. Wang, "A Review of Deep Learning Models for Time Series Prediction," in IEEE Sensors Journal, vol. 21, no. 6, pp. 7833-7848, 15 March 2021, doi: 10.1109/JSEN.2019.2923982.

[2] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, China, 2016, pp. 324-328, doi: 10.1109/YAC.2016.7804912.

[3] S. H. Rafi, Nahid-Al-Masood, S. R. Deeba and E. Hossain, "A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network," in IEEE Access, vol. 9, pp. 32436-32448, 2021, doi: 10.1109/ACCESS.2021.3060654.

[4] T. Li, M. Hua and X. Wu, "A Hybrid CNN-LSTM Model for Forecasting Particulate Matter (PM2.5)," in IEEE Access, vol. 8, pp. 26933-26940, 2020, doi: 10.1109/ACCESS.2020.2971348.

[5] Kaggle Weather Analysis dataset, https://www.kaggle.com/datasets/parthdande/timeseries-weather-dataset. Accessed: 26.05.2025.

[6] Torch Neural Network library, https://docs.pytorch.org/docs/stable/nn.html. Accessed: 26.05.2025.

[7] Keras time-series forecasting tutorial, https://keras.io/examples/timeseries/timeseries_weather_forecasting/. Accessed: 25.05.2025.

[8] Tensorflow time-series forecasting tutorial, https://www.tensorflow.org/tutorials/structured_data/time_series. Accessed: 25.05.2025.

[9] GfG time-series forecasting tutorial, https://www.geeksforgeeks.org/time-series-forecasting-using-pytorch/. Accessed: 26.05.2025

[10] S. Siami-Namini, N. Tavakoli and A. S. Namin, "The Performance of LSTM and BiLSTM in Forecasting Time Series," 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 3285-3292, doi: 10.1109/BigData47090.2019.9005997.
[11] Weather-5K dataset, https://github.com/taohan10200/WEATHER-5K?tab=readme-ov-file. Accessed: 27.05.2025