# Deep Temperature Prediction

Miles Young, Blaine Mensa, Matthew Ng, Alex Hoff

**Abstract**

The prediction of weather has long been an important task. Important applications in this field range from producing temperature forecasts for the next day, precipitation probability distributions over several hours, or identifying and tracking extreme weather events like tornadoes. In this report, we concentrate on the task of temperature prediction using deep learning. We show that the prediction of temperature is best done through the use of quality data and responds to various factors of importance in atmospheric measurement. The code can be found at https://colab.research.google.com/drive/1DbkbOFQzW5SGGuFkC_FFTfMGVqqnBb-W?usp=sharing. The repository containing the same jupyter notebook file (titled Team4 code) and the data used to train (Oakland weather station data in the saved data folder) can be accessed at: https://github.com/backwaterstation/671-Deep-Learning-Term-Project.

## 1 Introduction

This project explores the development of deep learning models in order to predict daily average temperature. The importance of the study of weather modeling cannot be understated. Many of us check weather forecasts daily, in order to prepare for what to wear, how many layers to pack, and whether we need to be prepared for precipitation. More important predictions of weather include forecasting for extreme weather events such as tornadoes. The latter can actually contribute to planning that may save lives.

In this experiment, we use data from the National Oceanic and Atmospheric Administration (NOAA) in order to train our model. This data is very detailed, high quality weather sensor data. In particular, we focus on an arbitrarily chose station-the airport in Oakland California.

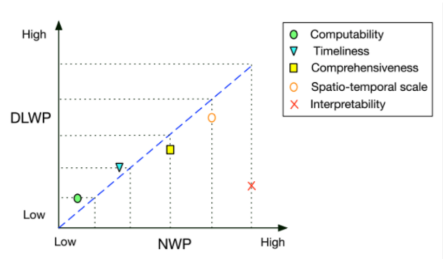
Multiple results from multiple models are reported in the paper, however we discuss the full breadth of the experiment with involved extensive iteration with hyperparameters and model architectures, however this was not the main focus.

It is our hope that we may provide some insight into the challenges of developing deep learning models for weather prediction.

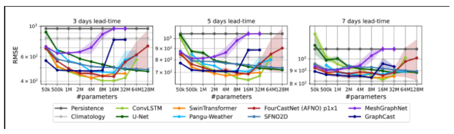
## 2 Related Work

Traditional weather modeling utilizes domain expertise regarding the physical and atmospheric processes contribute to the events perceived as weather. Traditionally, many models are of the non-deep learning approach including general circulation models (GCM) or numerical weather prediction (NWP) which are essentially a set of nonlinear equations, known as the primitive equations[1]. While these models incorporate powerful and accurate domain expertise, they can be very costly and require the use of supercomputers to run the level of prediction required by many potential users[2].

Recent work and computational advances have led to a shift towards deep learning for weather modeling [3]. Figure 1 a and b show some details on the computational cost and performance of deep learning models for weather prediction.



(a) Computational cost of models



(b) Comparison of model types

**Fig. 1**: Figures 1 and 2 show some details on the computational cost and performance of deep learning models for weather prediction.

## 3 Method

The experiment discussed in the paper took a very simple approach. The data for one weather station was downloaded from the NOAA weather cite https://www.ncei.noaa.gov/access/search/data-search/global-summary-of-the-day. The data is the post-processed daily averages for various sensor values including temperature among others. In our case we chose the Oakland weather station at the airport in Oakland (station id: 72493023230) due to the quality and consistency of measurements available at the site.

Once the data was available, it was split into variable length sequences for analysis, ultimately we ended up deciding on a sequence length of about 30 days through experimentation. In short, each input value was a sequence of sequence-length days measurements and the correspond label was the value observed at sequence-length+1 day. The data was also normalized before use in training. Additionally, we made sure that when the data was converted back to original scale to view predictions, the mean and std for rescaling was used from the original training data to avoid leakage into model at test time.

As far as models, we used PyTorch to create several models. In the course of the experiment, we tried changing the hyperparameters, subsets of data trained on, and model architecture. We present the results from 4 models including basic LSTMs, CNN-LSTM hybrids, Bi-LSTMs and a model trained on larger subsets of the data. Our findings show fairly low variability in the models which was echoed in our attempts to improve performance via hyperparameter and architecture tuning. The models were also compared with a baseline "naive" model which simply predicts the same temperature as the previous day.

We used several specialized model architectures based on our reading of the existing literature. In particular, there were two specialized models: CNN-LSTM and Bi-LSTM with decomposition. The CNN-LSTM appears in several works, but the general architecture was based on the work by Li et. al [4]. The concept behind the model is to allow a 1d convolution layer to extract some feature map from the sequences and subsequently process the convolution output with LSTM layers. The Bi-LSTM model was inspired by the work of Zhang et. al where not only was the architecture unique, but they did additional data preprocessing. We tried to recreate their model. Firstly, STL decomposition is performed to decompose the original time series into trend, seasonal, and residual components. These new time series along with the original series are then processed just as for the other models, but the unique model architecture uses 4 separate Bi-LSTMs to process the components which are then subsequently fused through a linear layer. More details of this approach can be found in the article cited [5]. The following table shows the model by parameter count.

| **Model Name** | **Parameter Count** |
|---|---:|
| LSTM v1 | 207,489 |
| LSTM v13 | 213,633 |
| CNN-LSTM | 2,072,297 |
| Bi-LSTM | 535,047 |

**Table 1**: Parameter counts of different model architectures

## 4 Results

In short, the best architecture for this task turned out to be the simple LSTM. The best performance was on the data including 13 variables, while the 1 variable LSTM (of the same architecture) closely followed. The next was the CNN-LSTM with the Bi-LSTM architecture coming in last. Table 1 summarizes the findings.

The training curves along with the predictions generated on our test set for the various models are shown in figure 2.

| Model Name | MSE | MAE |
|---|---|---|
| Naive Model | 7.47 | 2.08 |
| LSTM 1 var | 6.68 | 1.98 |
| LSTM 13 var | 6.09 | 1.98 |
| Bi-LSTM 1 var | 25.4 | 3.99 |
| CNN-LSTM | 7.26 | 2.04 |

**Table 2**: Model performance comparison

## 5 Discussion

The project was a learning experience and an excellent first step towards understanding weather modeling. While we were unable to achieve extensive progression in model iteration, some of the models that we produced were able to exceed at least the naive forecasting benchmark.

### 5.1 Model performance and limitations

What really surprised us in this experiment was the inability to make improvements to the models. We started with the simple LSTM architecture and attempted steadily increasing model size, tuning hyperparameters and slowly increased numbers of layers. The most complex model (CNN-LSTM) was a very large model based on the work on previous works that had found great success with such models [4].

Given that we had such resistance when attempting to produce quality models, we felt that increasing the number of variables was the last direction we could turn towards. While we did see some improvement in the model by adding more features, the improvement was minimal. Ultimately, we found that more exhaustive measures must be taken to achieve good performance.

### 5.2 Future work

The main goal of future work should be to understand why so little improvement was observed through experimentation with changing the model. Furthermore, it was quite unexpected that we would see worse performance on models with significantly more parameters (in the case of the CNN-LSTM).

Additionally, we believe that different types of data could be used to improve the model performance. It is common to use spatio-temporal data for these tasks and more investigation of the effects of data on the performance of models would be another interesting avenue for exploration.

## References

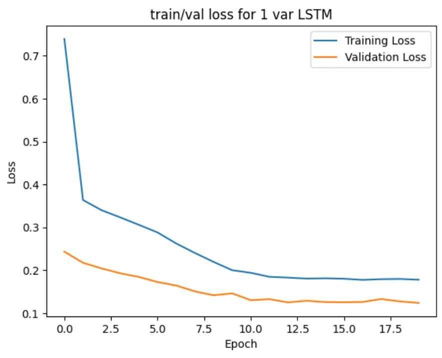
[1] Chen, S., Long, G., Jiang, J., Liu, D., Zhang, C.: Foundation Models for Weather and Climate Data Understanding: A Comprehensive Survey (2023). https://arxiv.org/abs/2312.03014

[2] Shi, J., Shirali, A., Jin, B., Zhou, S., Hu, W., Rangaraj, R., Wang, S., Han, J., Wang, Z., Lall, U., Wu, Y., Bobadilla, L., Narasimhan, G.: Deep Learning and Foundation Models for Weather Prediction: A Survey (2025). https://arxiv.org/abs/2501.06907

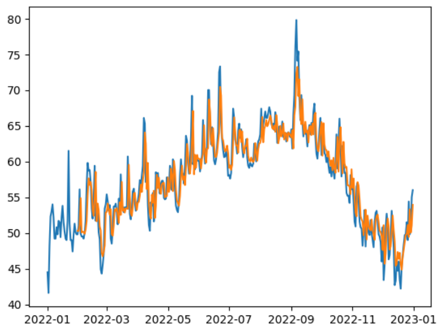
[3] Ren, X., Li, X., Ren, K., Song, J., Xu, Z., Deng, K., Wang, X.: Deep learning-based weather prediction: A survey. Big Data Research **23**, 100178 (2021) https://doi.org/10.1016/j.bdr.2020.100178

[4] Li, B., Qian, Y.: Weather Prediction Using CNN-LSTM for Time Series Analysis: A Case Study on Delhi Temperature Data (2024). https://arxiv.org/abs/2409.09414

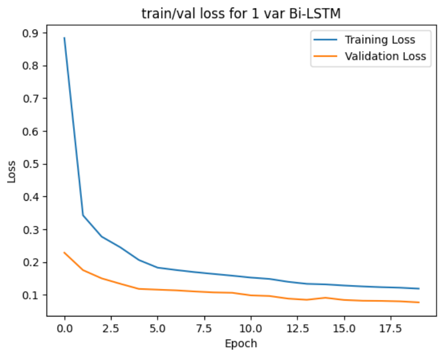
[5] Zhang, K., Huo, X., Shao, K.: Temperature time series prediction model based on time series decomposition and bi-lstm network. Mathematics **11**(9) (2023)



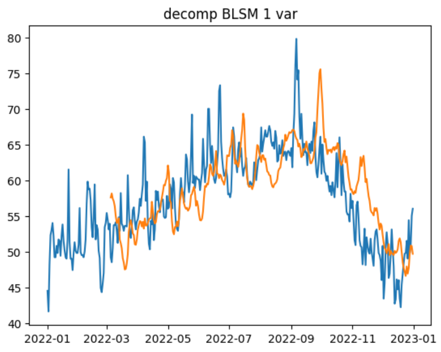
(a) LSTM 1 training curves



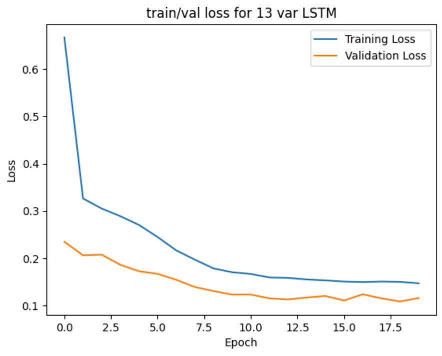
(b) LSTM 1 predictions



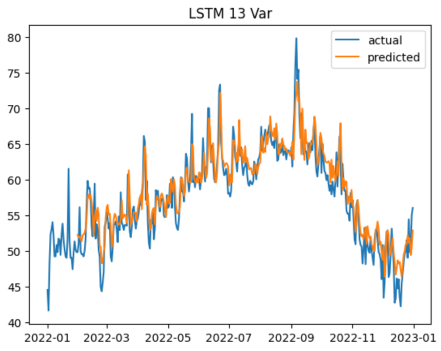
(c) Bi-LSTM training curves



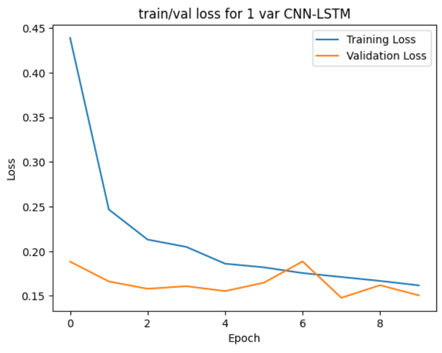
(d) Bi-LSTM predictions



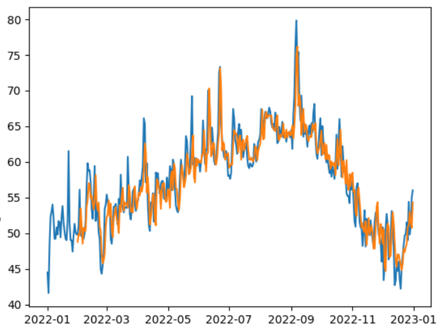
(e) LSTM 13 training curves



(f) LSTM 13 predictions



(g) CNN-LSTM training curves



(h) CNN-LSTM predictions

**Fig. 2**: Training/Validation Curves for All Models