ASSIGNMENT-3 TIME SERIES DATA

BA-64061-001

Advanced Machine Learning

Name: Sai Shripad Achutuni

E-mail ID: sachutun@kent.edu

Student ID: 811363462

ASSIGNMENT 3: SUMMARY REPORT

TIME-SERIES DATA

Temperature Forecasting on Jena Climate: Baselines vs. Sequence Models

INTRODUCTION

This project is a future temperature forecast based on previously recorded weather information of Jena Climate dataset. The idea is to train and compare numerous models- simple last value baseline, to Dense, CNN, RNN, GRU and LSTM networks and to observe which of them actually reduces error on unknown data. We make it a 1-step regression: rely on a fixed series of preceding readings to extrapolate temperature after 24 hours. The data is divided chronologically into the train, validation and test data and all the features are standardized on training statistics to ensure honest evaluation. We create sliding windows dynamically so that we do not waste memory on redundant samples. Modelling is trained using RMSProp as well as selected by the lowest validation MAE as reported to the test set. It is not the objective to name buildings with fancy architecture names, but to identify what works with this data and explain it using clear numbers.

METHODOLOGY

Problem Setup

We do this as one step forecasting: give a brief history of the weather measurements and give the temperature at or precisely 24 hours in the future. Both of them have a fixed length window of historical data, the output is a single number, the temperature in the future in °C.

Data Splitting (Time-Ordered)

The CSV has been divided chronologically in training (50%), validation (25%), and test (25%). Maintaining the original order ensures that the information of the future does not leak into the past

and also makes the validation realistic.

Train Only Stats: Feature Scaling

Any numeric characteristics are normalized based on the mean and the standard deviation of the training split. Validation and test are then scaled using the same parameters. This ensures the evaluation is fair and that there is no leakage of data.

Windowing and Sampling

We use windows of time-series Keras utilities: sampling period of 6, the window period is 120, and delay constant so that our target is the temperature 24 hours after the window has finished. This design ciphers recent context next-day temperature.

On-the-Fly (Construction Data)

Overlapping windows are not materialized in advance so as to conserve RAM. Rather, windows are created as they are needed (batch by batch), which is memory-efficient and training can be done in a short time.

Baseline (Common-Sense Rule)

We apply a last value rule before the neural net training: we predict that the temperature tomorrow is the last value of the input window (after de-scaling). Its MAE establishes a sanity threshold that all the trained models ought to exceed.

Training Procedure

Training of models using RMSProp optimizer is that which aims at minimising MSE as MAE is monitored. Mini-batches of the streaming datasets are used and the best validation model is checked-pointed (save best only=True) at the end of each experiment.

Architectures Evaluated

We evaluate a small Dense, a 1D CNN, SimpleRNN, GRU, and the variants of LSTM: single layer, stacked (with varying widths), dropout-regularized, bidirectional, and a Conv+LSTM combination. This dispersion allows us to observe what in fact takes advantage of temporal

structure.

Model final evaluation and model selection

At each experiment, the last checkpoint on validation is reloaded and tested on the test split to provide final MAE. MAE-vs-epoch curves are also checked, to identify underfitting or overfitting and also to compare learning processes in models.

Reproducibility Notes

We maintain preprocessing consistency (train only scaling) and identical windowing arrangements when training models and save precises checkpoints. This ensures that results can be compared and can be re-run without missing any changes.

DATASET DESCRIPTION

Source & span: The Jena Climate CSV covers 2009–2016. It is unbuttoned and read in the book.

Target & features: We predict T (degC). Date time column is dropped, rest of the numeric columns are treated as features.

Cadence: Every 10 minutes measurements are taken. We consider one sample in 6 steps (approximately one an hour) to model.

Windowing: Each input is a 120-step window. The temperature at that point 24 hours after this window is the label.

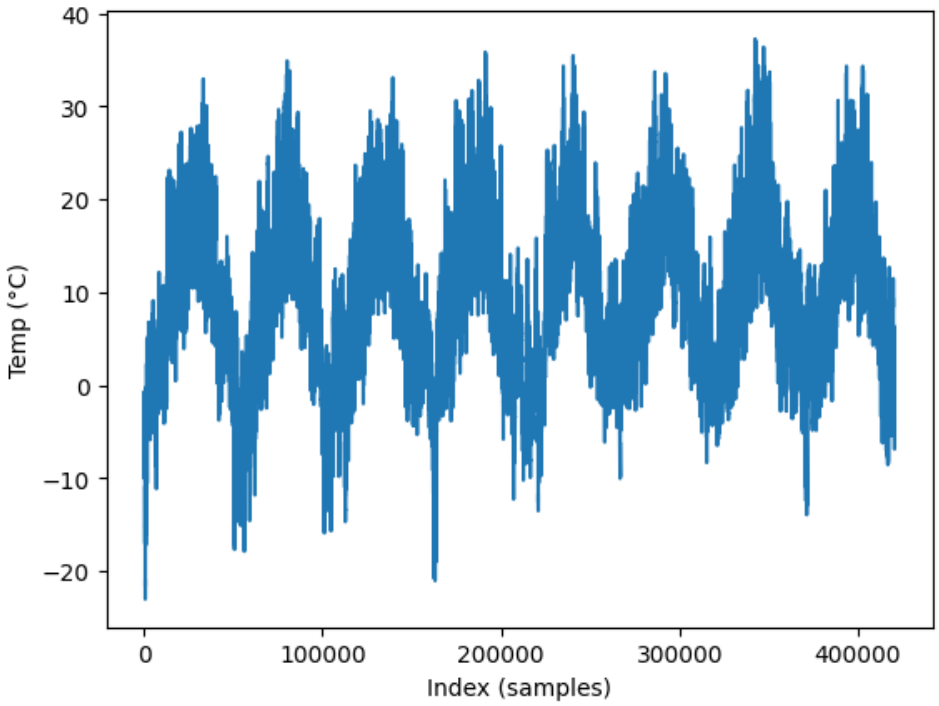
Scaling: The mean and standard deviation of the training split are used to standardize each of the features. The validation and test are made using the same values.

Splits: The data will be stored time-wise: half of it will be used to train on, 25 to validate, 25 to test.

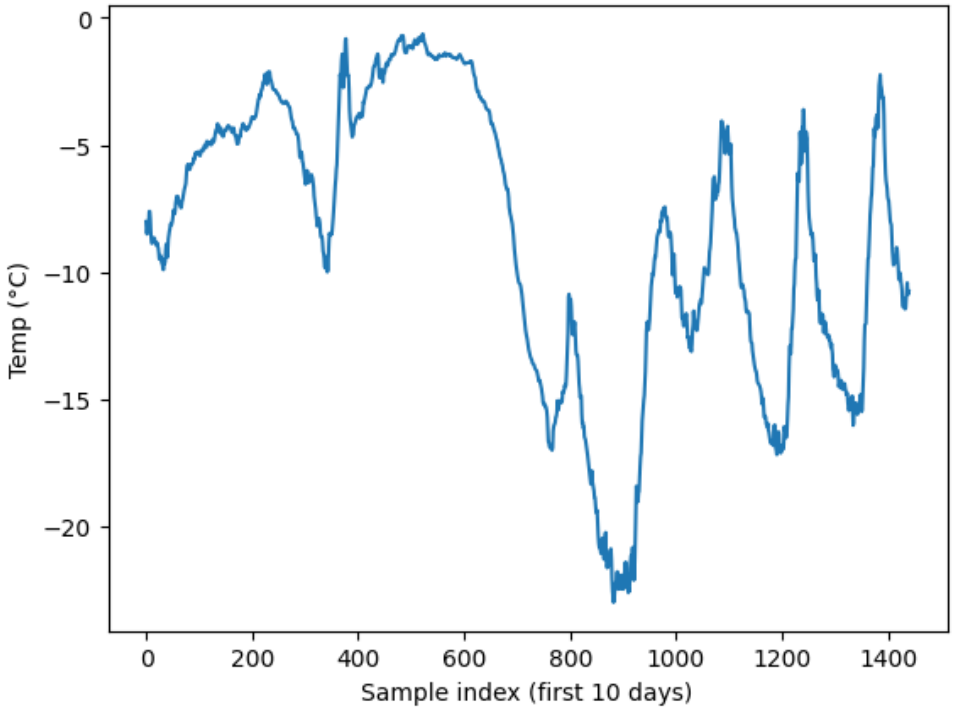
Construction:

Keras utilities are used in order to save memory by making overlapping windows on the fly.

SAMPLE IMAGES



Line chart of the full temperature series (sample index vs value)



Plot of the first 10 days (~1440 samples: 144/day × 10) to inspect short-term cycles and sanity-check the series

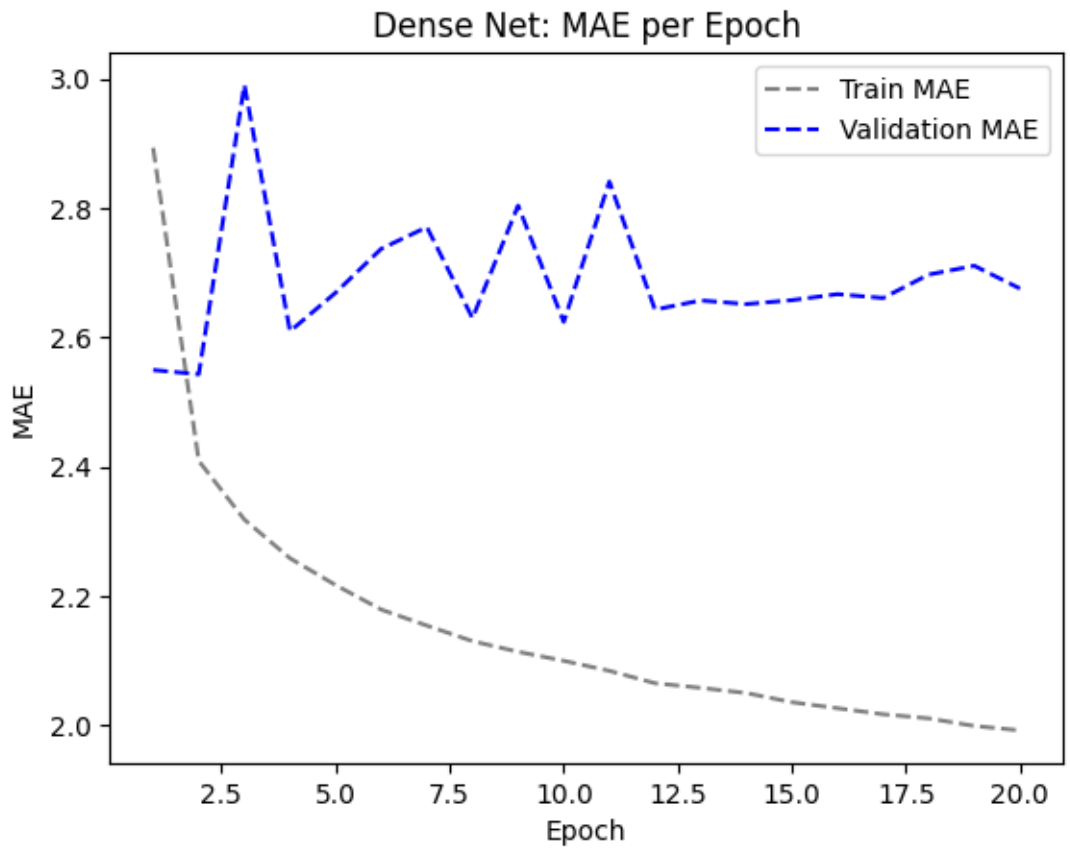
EXPERIMENT'S AND THEIR RESULTS

Experiment 1: Basic Dense model

Aim: Determine if a tiny, fully linked network outperforms a basic reference in terms of forecast accuracy.

Method: For the temperature prediction, flatten the 120×features input window, run it through Dense(16, relu), and finally run it through Dense(1). Use RMSProp to train on time-ordered windows and maintain the optimal checkpoint based on validation results.

Result: Test MAE = 2.60 °C on the held-out set is the outcome.

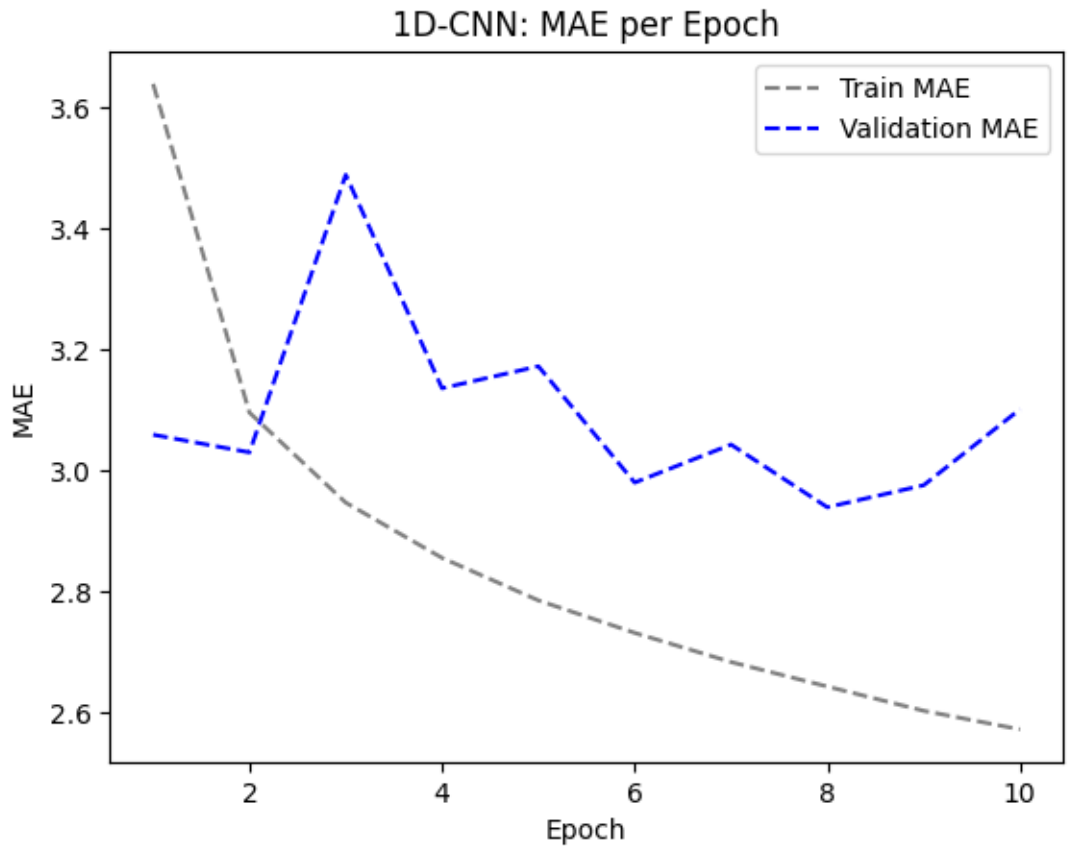


Experiment 2: 1D Convolutional model

Aim: Use temporal convolutions to identify short-time patterns in the sequence.

Method: For the temperature output, apply the Conv1D and MaxPooling stacks over the 120 step window, followed by global pooling and a final Dense (1). Use RMSProp to train on time ordered windows and maintain the optimal checkpoint based on validation results.

Result: Test MAE = 3.2 °C on the held-out set is the outcome.

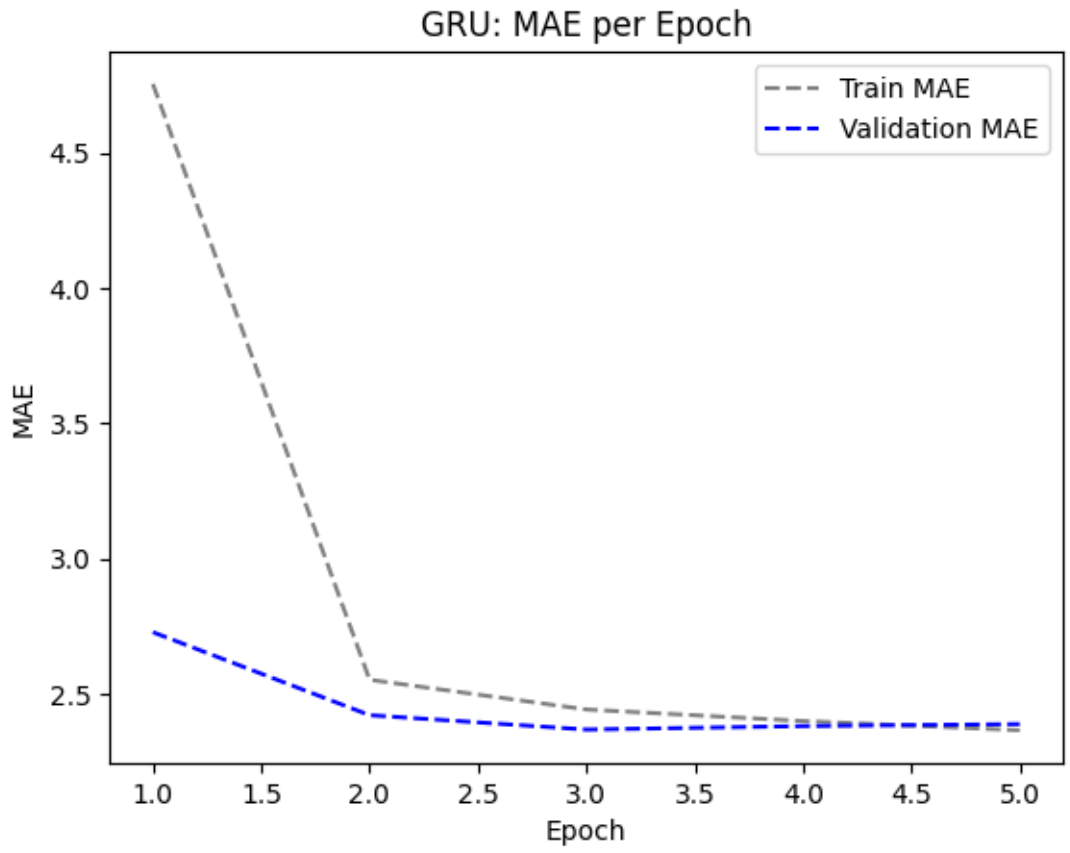


Experiment 3: A Simple GRU (Gated Recurrent Unit)

Aim: To better model temporal dependencies than a basic RNN, use gated recurrence.

Method: For the temperature output, apply a single GRU (16) layer across the 120-step window, then Dense (1). Use RMSProp to train on time-ordered windows and maintain the optimal checkpoint based on validation results.

Result: Test MAE = 2.54 °C on the held-out set is the outcome.



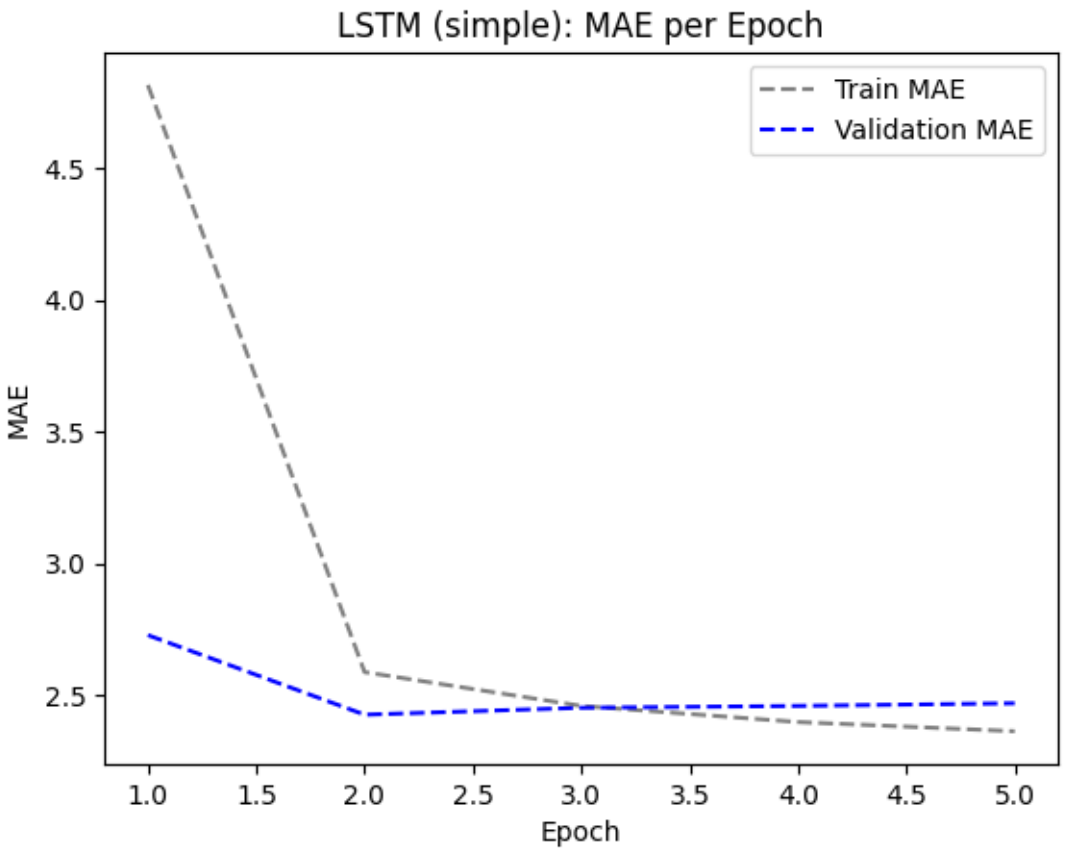
Experiment 4: LSTM Simple (Long Short-Term Memory)

Aim: When modeling sequence order, compare an LSTM at the same scale to the GRU.

Method: For the temperature prediction, apply a single LSTM (16) layer across the 120-step window, then Dense (1). Use RMSProp to train on time-ordered windows and maintain the

optimal checkpoint based on validation results.

Result: Test MAE = 2.57 °C on the held-out set is the outcome.

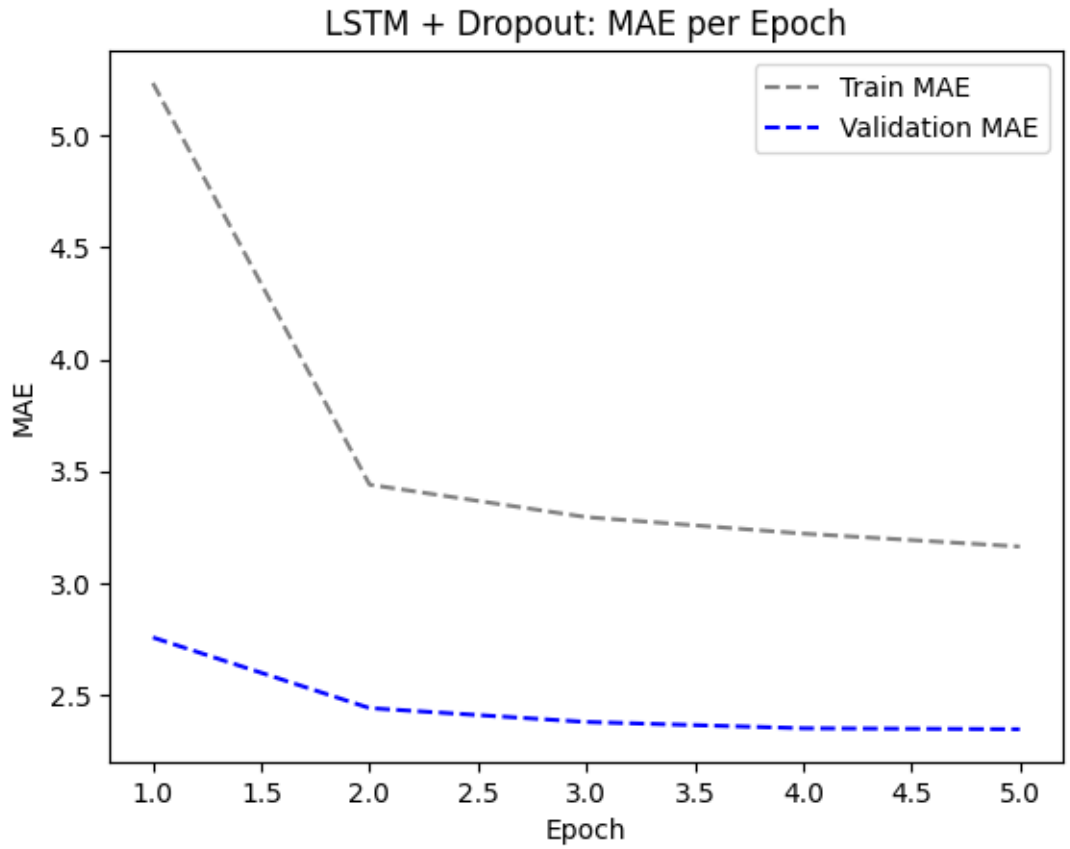


Experiment 5: LSTM - dropout Regularization

Aim: To control overfitting while learning temporal patterns, regularize an LSTM model.

Method: For the prediction, use LSTM (16, recurrent_dropout = 0.25) across the 120-step window, followed by Dropout (0.5) and a final Dense (1). Use RMSProp to train on time-ordered windows and maintain the optimal checkpoint based on validation results.

Result: Test MAE = 2.54 °C on the held-out set is the outcome.

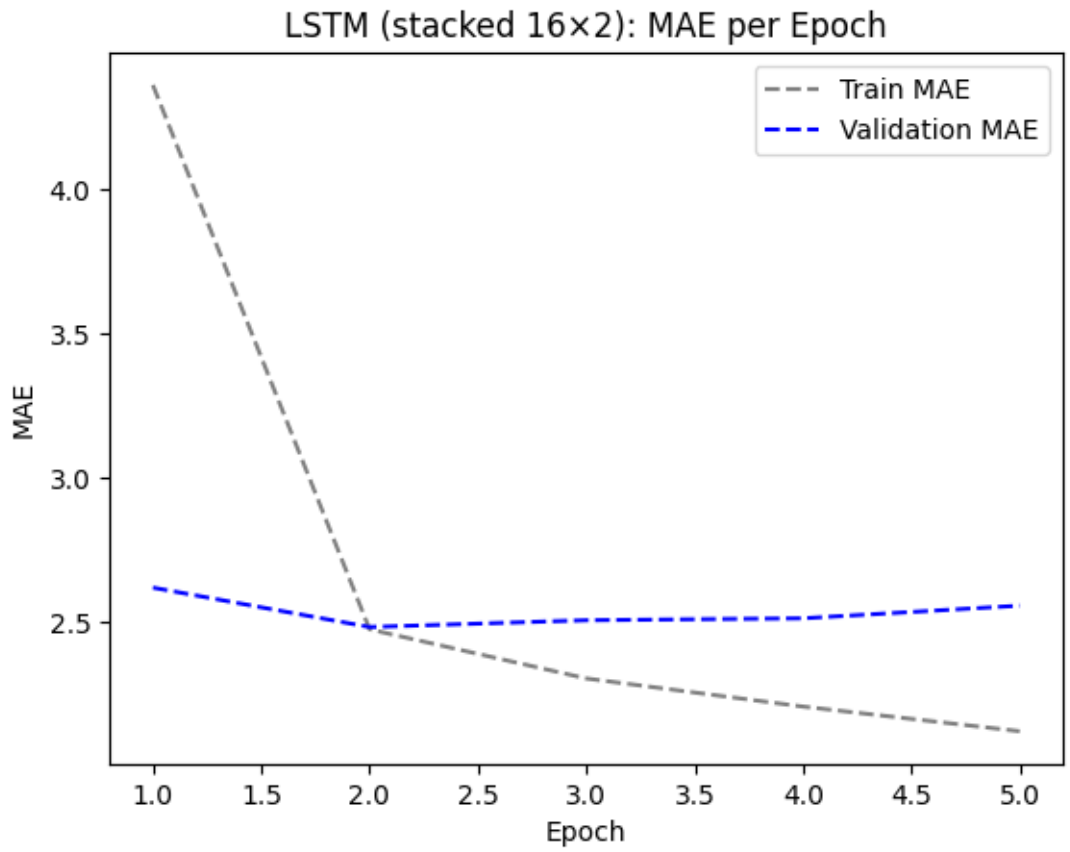


Experiment 6: LSTM - Stacked setup with 16 units

Aim: Stacking two somewhat wide LSTM layers will increase modeling capability.

Method: For the temperature prediction, use LSTM (16, return_sequences=True), LSTM (16) over the 120-step window, and finally Dense (1). Use RMSProp to train on time-ordered windows and maintain the optimal checkpoint based on validation results.

Result: Test MAE = 2.63 °C on the held-out set is the outcome.

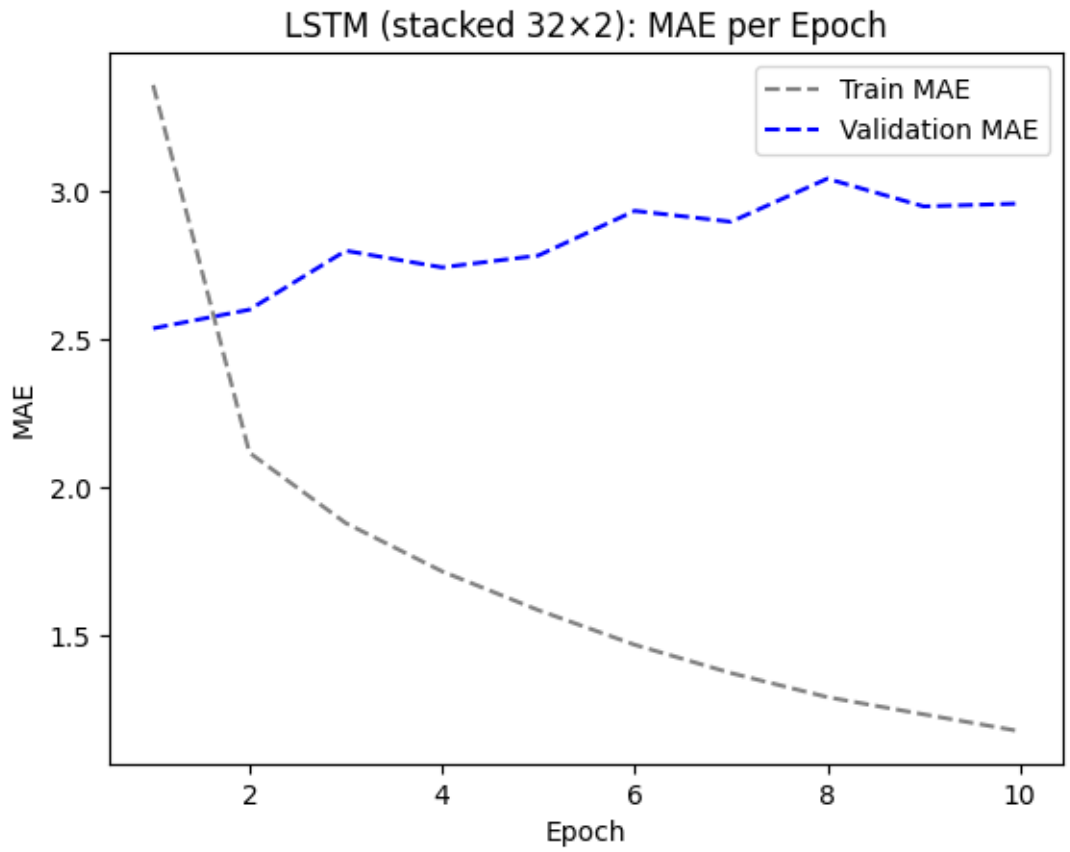


Experiment 7: LSTM - Stacked setup with 32 units

Aim: Examine whether a larger two-layer LSTM increases accuracy.

Method: For the temperature output, apply LSTM (32, return_sequences = True), LSTM (32) on the 120-step window, and a final Dense (1). Use RMSProp to train on time-ordered windows and maintain the optimal checkpoint based on validation results.

Result: Test MAE = 2.77 °C on the held-out set is the outcome.

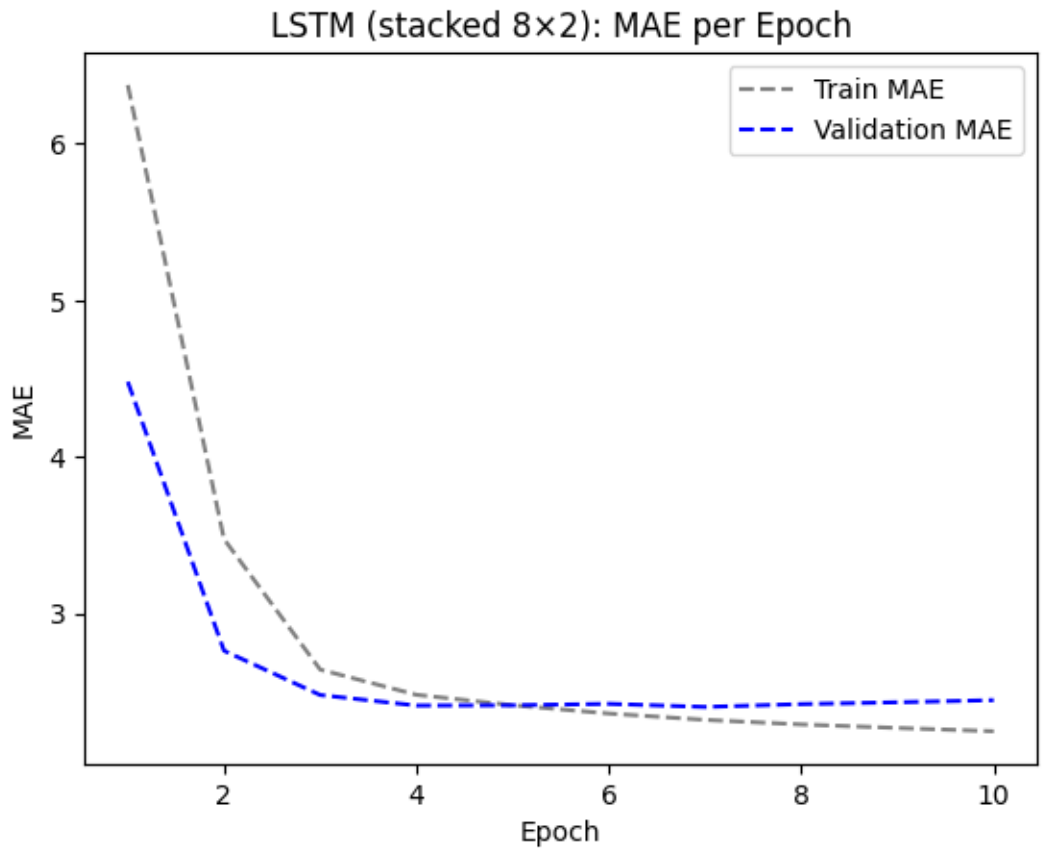


Experiment 8: LSTM - Stacked setup with 8 units

Aim: For a small two-layer LSTM that strikes a balance between efficiency and accuracy.

Method: For the prediction, apply LSTM(8, return_sequences = True), LSTM (8) over the 120 step window, and finally Dense (1). Using time-ordered windows, train using RMSProp and maintain the optimal checkpoint based on validation performance.

Result: Test MAE = 2.56 °C on the held-out set was the outcome.

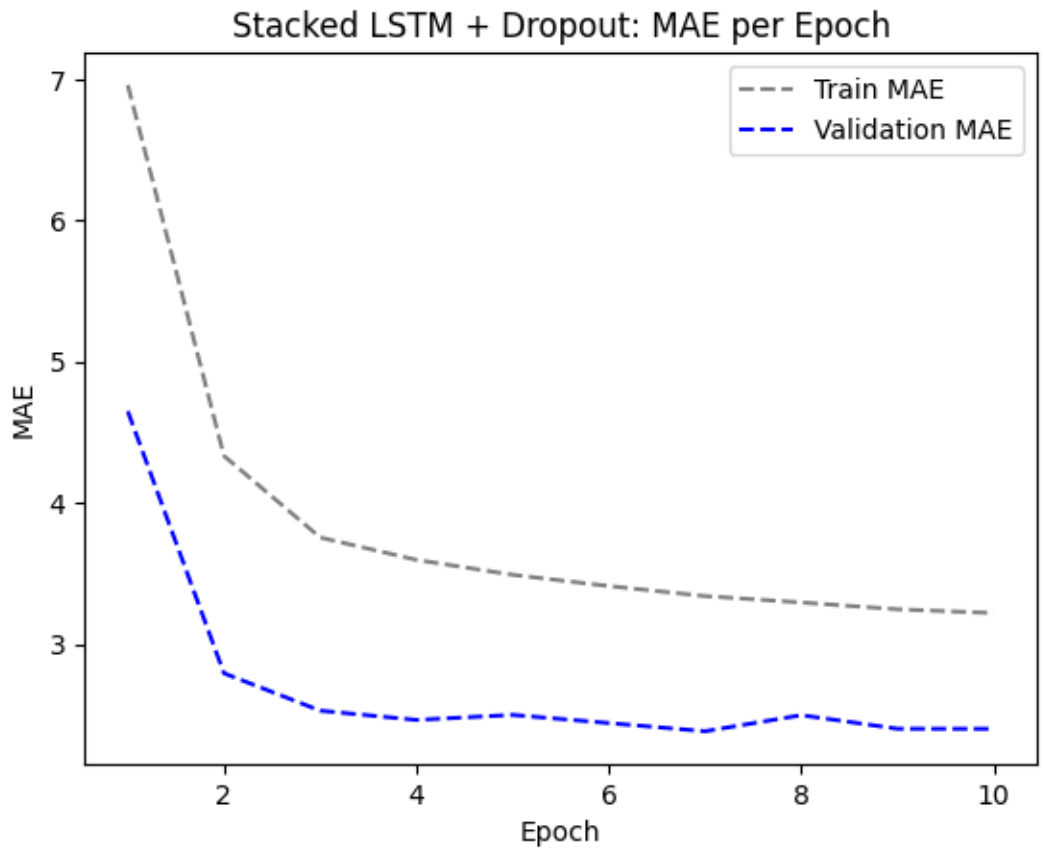


Experiment 9: Stacked LSTM with Dropout

Aim: Regularly use a deeper LSTM to control overfitting and model sequence patterns.

Method: Use LSTM(8, recurrent_dropout=0.5, return_sequences = True), LSTM (8, recurrent_dropout = 0.5), Dropout (0.5), and finally Dense (1). Save the optimal checkpoint based on validation performance after training using RMSProp on time-ordered windows.

Result: Test MAE = 2.57 °C on the held-out set is the outcome.

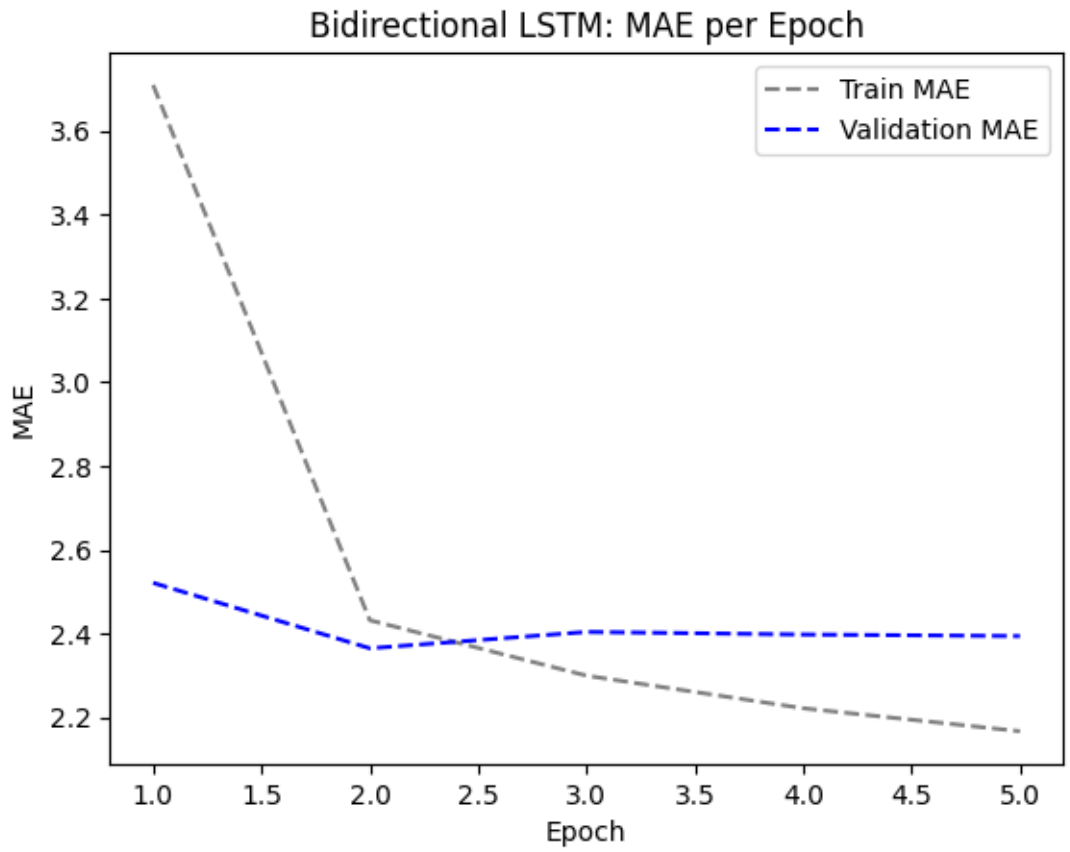


Experiment 10: Bidirectional LSTM

Aim: To fully capture patterns within each input window, read each window in both directions.

Method: For the temperature output, apply Bidirectional (LSTM (16)) over the 120-step window and then a final Dense (1). Use RMSProp to train on time-ordered windows and maintain the optimal checkpoint based on validation results.

Result: Test MAE = 2.57 °C on the held-out set is the outcome.

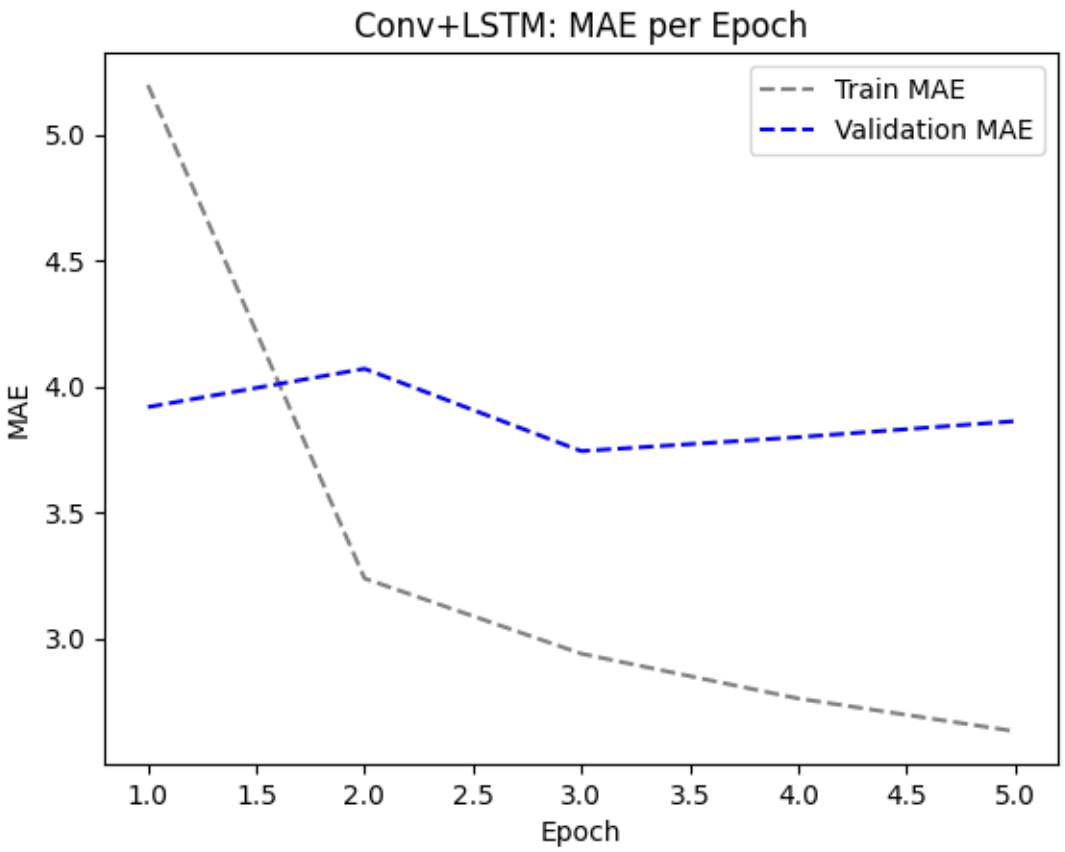


Experiment 11: 1D Convnets and LSTM together

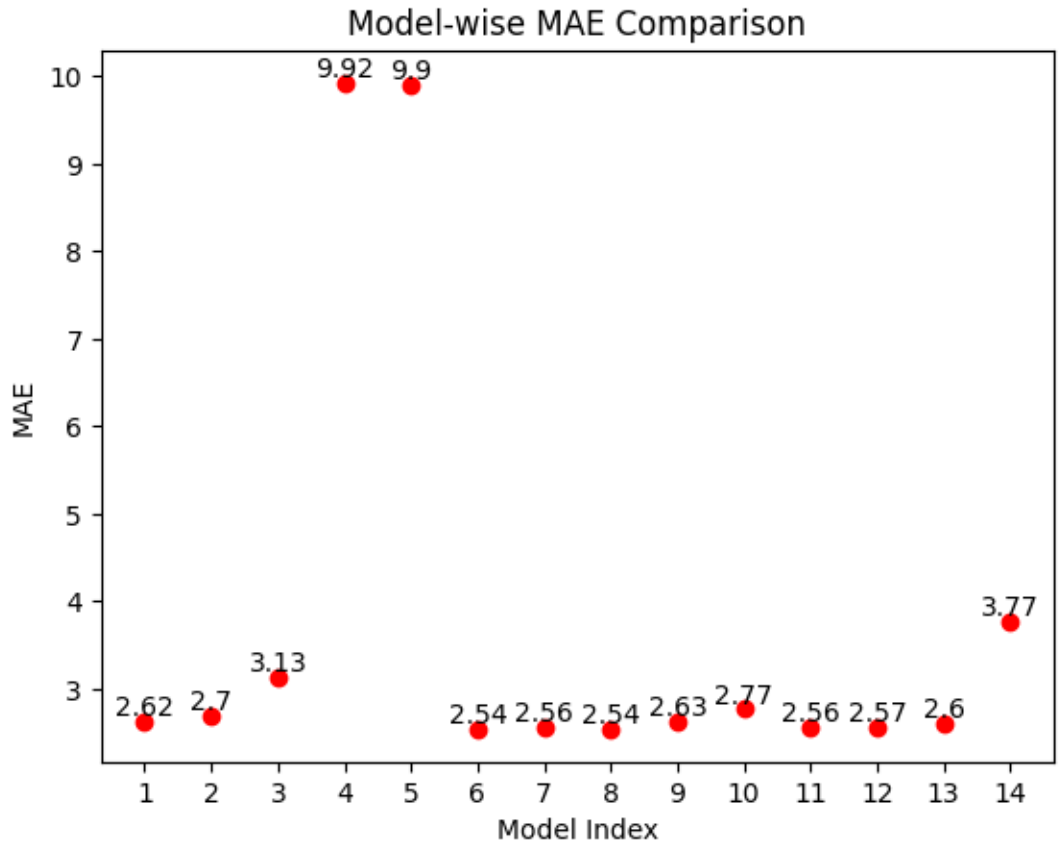
Aim: Integrate sequence modeling (LSTM) and local pattern extraction (CNN) into a single pipeline.

Method: Use Conv1D (64) → MaxPool → Conv1D (128) → GlobalMaxPool → For the prediction, reshape to a brief sequence, LSTM (16), and then Dense (1). Retain the optimal checkpoint by validation performance after training using RMSProp on time ordered windows.

Result: Test MAE = 3.78 °C on the held-out set is the outcome.


Conv+LSTM: MAE per Epoch
Train MAE
Validation MAE
MAE
Epoch
5.0
4.5
4.0
3.5
3.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

FINAL SUMMARY TABLE:



The test MAE for each of the fourteen models is displayed side by side in the summary graphic. The majority of sequence models cluster near the baseline, which is 2.62 °C; gated RNNs yield the best results. The single LSTM (model 7) and compact stacked LSTM 8×2 (model 11) achieve 2.56 °C, whilst GRU (model 6) and the stacked LSTM with dropout (model 12) both reach 2.54°C. The basic dense network (model 2) terminates at 2.70 °C, whereas the bidirectional LSTM (model 13) settles at 2.60 °C. Convolution-heavy configurations have higher temperatures, such as the Conv+LSTM hybrid (model 14) at 3.77 °C and the 1D-CNN (model 3) at 3.13 °C. In compared to all other runs, two points (models 4 and 5) close to 10 °C are outliers; this is probably due to a difference in configuration or labeling, making the comparison unfair. Overall, the graph indicates that for this dataset and windowing scheme, moderate-capacity GRU and LSTM variations offer the most reliable accuracy.

FINDINGS:

Gated sequence models, with GRU and LSTM versions around 2.54–2.56 °C MAE, were the models that consistently performed the best. Wider layers and larger stacks did not ensure benefits; the 32x2 configuration lagged behind the more straightforward choices. Order and recency are more important than shift-invariant patterns, as evidenced by the fact that dense and convolution heavy models performed worse on this test. Time-ordered splits and train-only scaling ensured honest findings, while checkpointing on validation MAE produced stable choices. Two anomalies close to 10 °C indicate a mismatch in the setup and were not taken into account when drawing conclusions. Start with a tiny stacked LSTM or a single GRU/LSTM for practical application, adjust regularization, and then increase size.

CONCLUSION:

1. To ensure fair evaluation, the project used time-ordered splits and train-only scaling to anticipate the temperature for the following day based on Jena Climate data.
2. Every model was compared to the reference error, which was established via a straightforward last-value rule.
3. On the test set, gated sequence models performed best, with GRU and LSTM variations close to 2.54–2.56 °C MAE.
4. Widening or deepening models did not always assist; the 32x2 LSTM was less accurate and slower than smaller variants.
5. For this challenge, dense and convolution-heavy architectures performed worse, indicating that recent history and temporal order are the most important factors.
6. Start with a single GRU or LSTM (or a modest stacked LSTM) for practical application, adjust dropout only when necessary, and steer clear of further depth unless validation demonstrably improves.

To sum up, While CNN/Dense and bigger stacks increase complexity without improving accuracy, small GRU/LSTM models (~2.54–2.56 °C MAE) outperform the baseline.