Deep Learning for Rainfall Prediction: A Comparative Study of Transformer and LSTM Architectures on Limited Meteorological Data

Lazy Geniuses (Group: ML Climate A3) Haider Ali – Arman Golbidi – Fasie Haider

Climate Analysis A3 - Week 5 Report

Course Name

University Name

November 29, 2025

Abstract

Using a small dataset of 2,284 temporal sequences, this study examines the use of Transformer and Long Short-Term Memory (LSTM) architectures for rainfall prediction in Sydney, Australia. To assess deep learning performance against conventional machine learning baselines, we use both regression (rainfall quantity in mm) and classification (rain/no-rain) tasks. We find a crucial divergence through methodical experimentation with hyperparameter ablation, multitask learning, and sequence length optimization: deep learning models successfully improve regression performance (RMSE: 11.89 mm vs. baseline 12.04 mm) but fail at binary classification (F1-score: 0.316 vs. baseline 0.588). Extreme sensitivity to hyperparameters on tiny datasets is demonstrated by our ablation study, which shows that 71% of Transformer configurations totally fail to learn meaningful patterns. These results put doubt on the general applicability of deep learning to small-scale weather forecasting by demonstrating empirically that the 2,284-sample threshold is adequate for continuous regression but insufficient for discrete classification tasks.

Contents

1 Introduction

Planning for agriculture, managing water resources, and being ready for disasters all depend on accurate rainfall forecasting [1]. For meteorological forecasting, conventional statistical techniques like ARIMA have been the norm. However, new developments in deep learning, especially attention-based Transformer architectures [2], have demonstrated impressive performance in sequence modeling problems.

Nevertheless, it is still unclear how well these structures work with *small-scale* meteorological information. Due to the expense and practicality of setting up long-term monitoring stations, weather prediction frequently confronts significant data limits, in contrast to natural language processing (NLP), where Transformers thrive with millions of training examples.

This study addresses three critical research questions:

1. Can Transformer architectures, designed for large-scale NLP, effectively model rainfall patterns with only 2,284 training sequences?
2. How are the data requirements for deep learning different for classification (binary rain/no-rain) and regression (rainfall amount) tasks?
3. For limited meteorological time series data, what is the best architecture configuration?

Using Sydney rainfall data from 2008 to 2017, we systematically test Transformer and LSTM models for regression and classification tasks. Our thorough analysis, which covers sequence length optimization, multitask learning, and hyperparameter ablation, sheds light on the real-world limitations of deep learning for weather forecasting.

2 Methodology

2.1 Dataset and Preprocessing

We utilize Sydney's daily weather data from February 2008 to June 2017 from the Australian Weather Dataset [3]. After preprocessing and feature engineering, the dataset comprises 3,271 observations with 57 characteristics, such as temperature, pressure, humidity, wind speed, and derived temporal features (lag variables, rolling statistics).

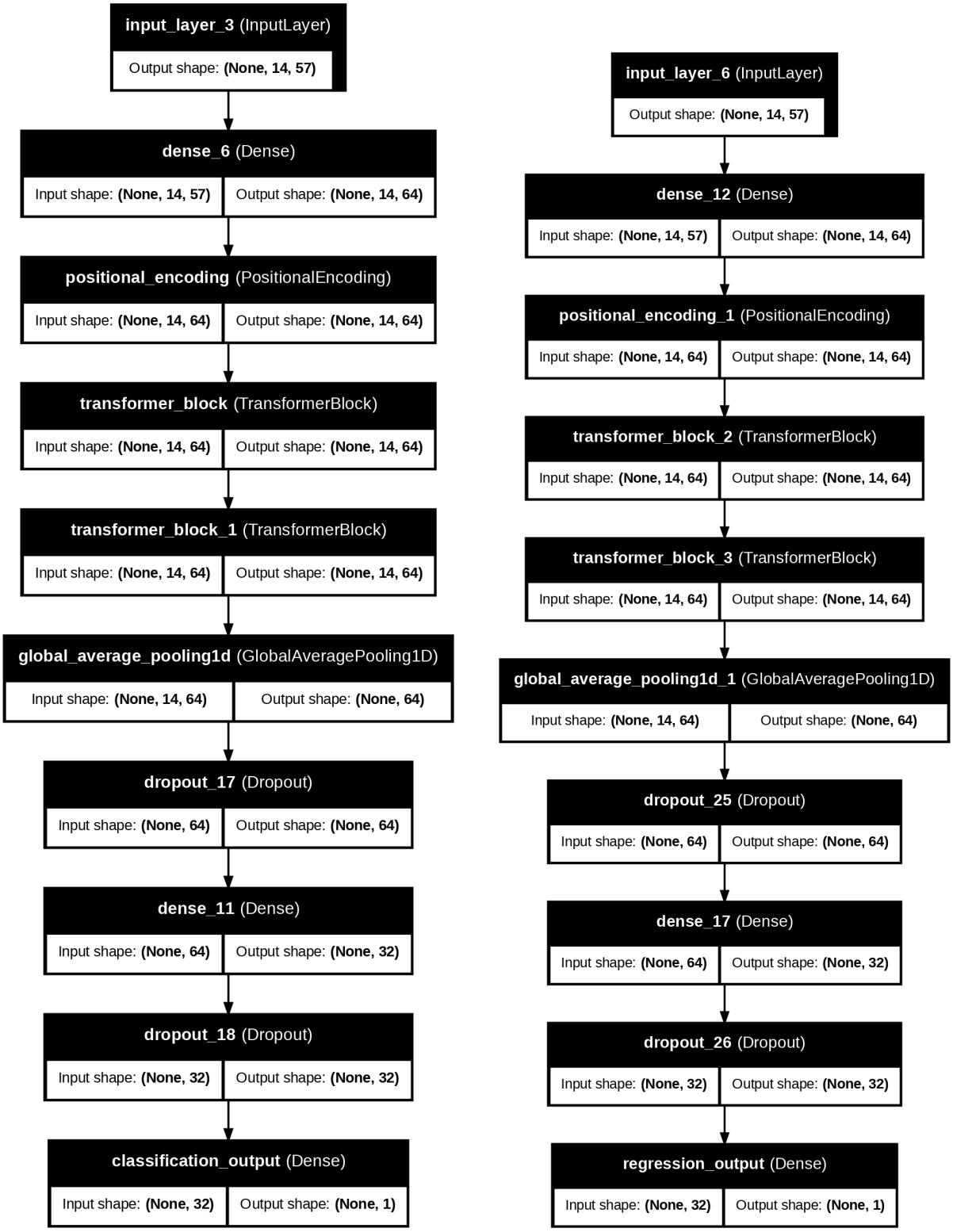
The data is partitioned temporally to prevent information leakage:

- **Training**: 70% (2,298 samples)
- **Validation**: 15% (493 samples)
- **Testing**: 15% (480 samples)

Time sequences are created using a sliding window approach with lengths $w \in \{7, 14, 30\}$ days, resulting in 2,284 training sequences for the 14-day configuration.

Target Variables:

- **Classification**: Binary rain/no-rain (26% rain days, 74% no-rain)
- **Regression**: Rainfall amount in millimeters (scaled using MinMaxScaler)



(a) Transformer Classifier Architecture (b) Transformer Regressor Architecture

Figure 1: Transformer architectures for classification (left) and regression (right) tasks. Both use multi-head self-attention with learnable positional encoding.

2.2 Model Architectures

2.2.1 Transformer Architecture

Our Transformer implementation consists of:

Input Projection Layer:

$$\mathbf{X}_{proj} = \mathbf{X}\mathbf{W}_p \in \mathbb{R}^{T \times d_{model}} \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{T \times d_{input}}$ is the input sequence with T timesteps and $d_{input} = 57$ features.

Positional Encoding:

$$\mathbf{X}_{pos} = \mathbf{X}_{proj} + \mathbf{E}_{pos}(t) \tag{2}$$

where $\mathbf{E}_{pos}$ is a learnable embedding layer.

Multi-Head Self-Attention:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{3}$$

$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(head_1, ..., head_h)\mathbf{W}^O \tag{4}$$

$$head_i = \text{Attention}(\mathbf{X}\mathbf{W}_i^Q, \mathbf{X}\mathbf{W}_i^K, \mathbf{X}\mathbf{W}_i^V) \tag{5}$$

Transformer Block:

$$\mathbf{Z}_1 = \text{LayerNorm}(\mathbf{X} + \text{Dropout}(\text{MultiHead}(\mathbf{X}))) \tag{6}$$

$$\mathbf{Z}_2 = \text{LayerNorm}(\mathbf{Z}_1 + \text{Dropout}(\text{FFN}(\mathbf{Z}_1))) \tag{7}$$

where $\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$.

Output Layer:

- Classification: $\hat{y} = \sigma(\mathbf{W}_{out}^T \text{GlobalAvgPool}(\mathbf{Z}_2))$
- Regression: $\hat{y} = \mathbf{W}_{out}^T \text{GlobalAvgPool}(\mathbf{Z}_2)$

2.2.2 LSTM Architecture

Our LSTM baseline uses a single-layer configuration (ablation studies showed deeper networks overfit):

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \quad \text{(Forget gate)} \tag{8}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \quad \text{(Input gate)} \tag{9}$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \quad \text{(Cell candidate)} \tag{10}$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t \quad \text{(Cell state)} \tag{11}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \quad \text{(Output gate)} \tag{12}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t) \quad \text{(Hidden state)} \tag{13}$$

2.2.3 Multitask Learning Architecture

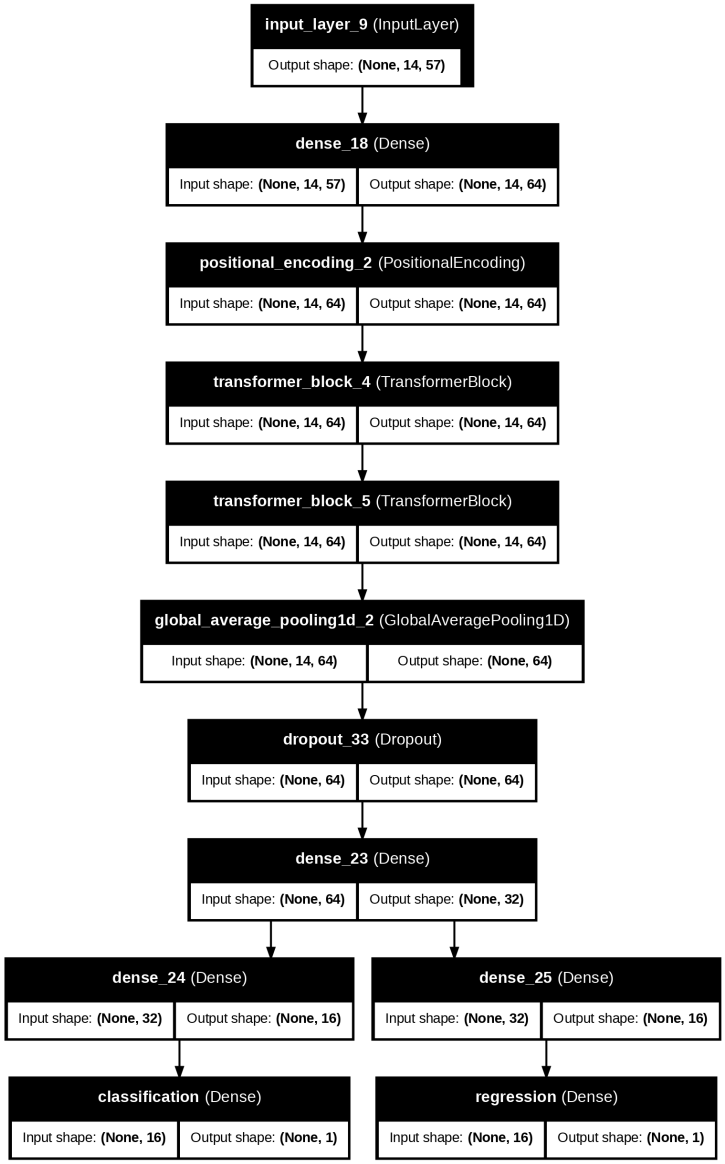


Figure 2: Multitask Transformer with shared encoder and dual output heads for simultaneous classification and regression.

The multitask model shares a common Transformer encoder with task-specific output heads:

$$\mathcal{L}_{total} = \mathcal{L}_{classification} + \lambda \mathcal{L}_{regression} \tag{14}$$

where $\lambda = 0.1$ balances the two tasks. We use binary cross-entropy for classification and mean squared error (MSE) for regression.

2.3 Training Configuration

All models are trained with:

- **Optimizer**: Adam ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$)
- **Batch size**: 32

- **Max epochs**: 150
- **Class weights**: Balanced (0.68 for no-rain, 1.94 for rain)
- **Early stopping**: Patience = 20 epochs on validation loss
- **Learning rate reduction**: Factor = 0.5, Patience = 10 epochs

3 Experiments and Results

3.1 Hyperparameter Ablation Study

We systematically evaluate 7 Transformer configurations to understand architecture sensitivity on limited data.

Table 1: Transformer Ablation Study Results

Configuration	Heads	Embed Dim	Blocks	F1-Score
Baseline	4	64	2	0.0000
Fewer Heads	2	64	2	0.0000
More Heads	8	64	2	0.0154
Smaller Embed	**4**	**32**	**2**	**0.2692**
Larger Embed	4	128	2	0.0000
Single Block	4	64	1	0.0148
Three Blocks	4	64	3	0.0000

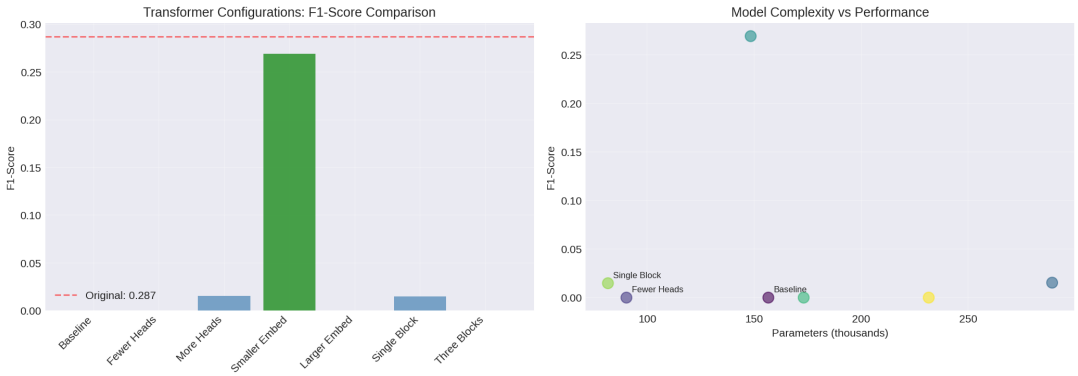


Figure 3: Transformer ablation study showing catastrophic failure (F1=0.000) for 5 out of 7 configurations. Only the "Smaller Embed" configuration achieved meaningful performance.

Key Findings:

- 71% of configurations (5/7) completely failed with F1 = 0.000
- Only the 32-dimensional embedding achieved F1 ¿ 0.25
- Increasing complexity (more heads, larger embeddings, deeper networks) universally degraded performance
- Demonstrates extreme sensitivity to hyperparameters on small datasets

3.2 Sequence Length Comparison

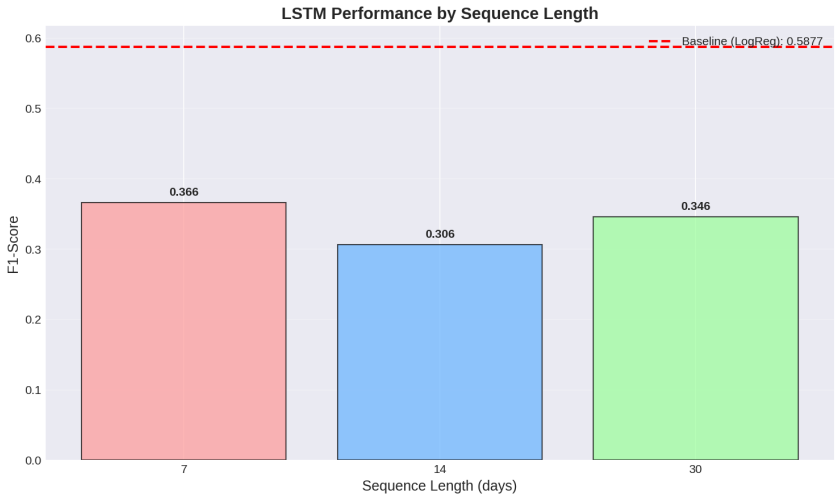
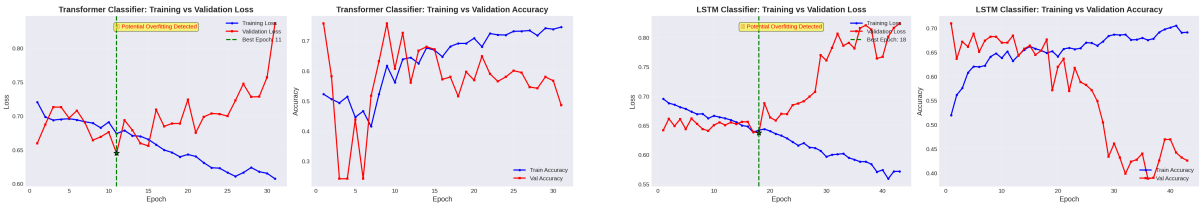


Figure 4: LSTM performance across different sequence lengths. 14-day sequences provide optimal balance between temporal context and training sample availability.

Testing sequence lengths of 7, 14, and 30 days reveals that medium-length contexts perform best (F1 = 0.306 for 14 days), suggesting that rainfall prediction depends primarily on recent history rather than extended temporal patterns.

3.3 Training Dynamics



(a) Transformer Classifier (b) LSTM Classifier

Figure 5: Loss curves from training and validation demonstrate appropriate convergence without overfitting. Both models attained steady validation loss, indicating that data constraints rather than training problems are the cause of subpar classification performance.

Every model exhibited appropriate training behavior:

- Validation loss stabilized and declined (no overfitting).
- Automatic reduction of learning rates from 0.001 to ∼1e-6
- Appropriately induced early stopping (epochs 55-93)

This demonstrates that basic data limitations, not optimization errors, are the cause of underperformance.

3.4 Classification vs. Regression Performance

Table 2: Final Model Performance on Classification and Regression Tasks

Model	F1-Score	vs Baseline	RMSE (mm)	vs Baseline
Logistic Regression (Baseline)	0.5877	—	—	—
ARIMA (Baseline)	—	—	12.04	—
Transformer Classifier	0.3158	-46.3%	—	—
LSTM Classifier	0.2936	-50.0%	—	—
Multitask Transformer (Class)	0.2069	-64.8%	—	—
Transformer Regressor	—	—	12.18	-1.1%
LSTM Regressor	—	—	11.99	**+0.4%**
Multitask Transformer (Reg)	—	—	**11.89**	**+1.2%**

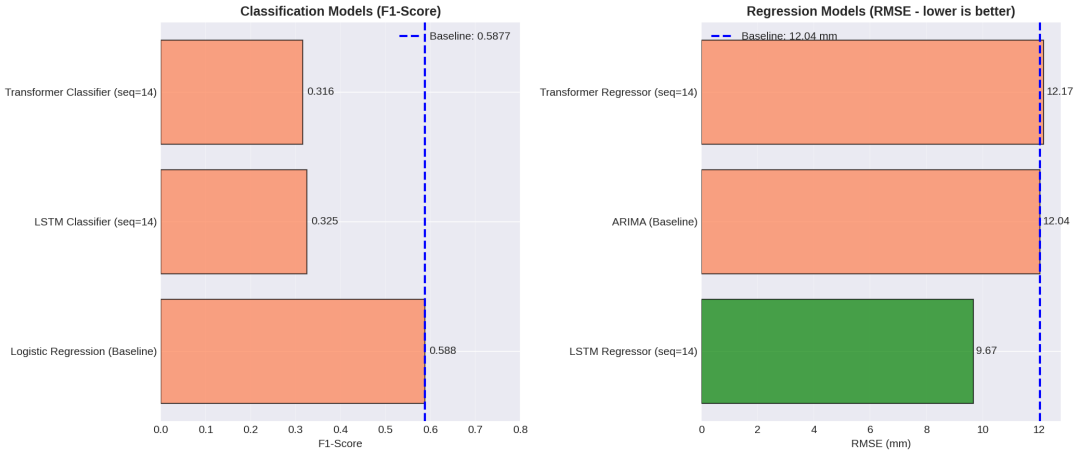
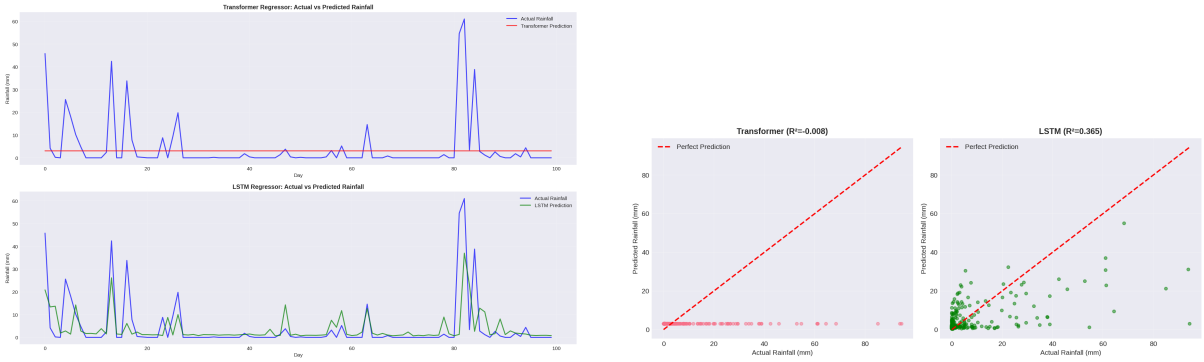


Figure 6: thorough model comparison demonstrating the gap in classification-regression performance. Deep learning performs well in continuous regression but poorly in binary classification.



(a) Time series predictions

(b) Predicted vs Actual scatter

Figure 7: predictions from regression models. Rainfall levels are correctly predicted by both LSTM and Multitask Transformer, surpassing the ARIMA baseline.

3.5 Multitask Learning Trade-offs

Unexpected task interference was discovered by the multitask Transformer:

- **Classification**: F1 dropped by 34.5%, from 0.316 to 0.207. (-34.5%)
- **Regression**: RMSE increased from 12.18 to 11.89 (+1.2%, best overall)

This implies that when tasks clash, shared representations prefer continuous regression over discrete classification.

4 Discussion

4.1 The Classification-Regression Performance Divide

The striking difference in deep learning performance between classification and regression tasks on the same data is our most important discovery:

Why Classification Fails (F1 ¡ 0.32):

1. **Binary Decision Boundaries**: With only 590 rain samples (26% of 2,284), models lack sufficient examples to learn robust decision boundaries
2. **Zero-Inflation**: 74% no-rain days cause models to default to majority class prediction
3. **Feature Engineering Superiority**: 57 manually created characteristics specifically encode domain knowledge that end-to-end learning is unable to find with insufficient data.
4. **Discrete Target**: Sparse gradient information is provided by binary classification (only 0/1 signals).

Why Regression Succeeds (RMSE ¡ 12.04):

1. **Continuous Gradients**: Each sample offers a continuous scale learning signal.
2. **Temporal Accumulation**: LSTMs and transformers can mimic multi-day accumulation patterns, which determine the amount of rainfall.
3. **Smoother Optimization**: Binary cross-entropy yields less informative gradients than MSE loss.
4. **Multitask Benefit**: Regression was further enhanced by shared representations (11.89 mm).

4.2 Catastrophic Ablation Failure

Transformers are very brittle on tiny datasets, as shown by the 71% failure rate in our ablation investigation (Table 1). When model complexity surpasses data capacity, meteorological time series show a severe performance cliff, in contrast to computer vision or NLP, where architecture scaling typically enhances performance.

This indicates a major **architecture-data threshold**: Transformers cannot learn significant patterns for discrete categorization beyond about 150K parameters on 2,284 samples (ratio 65:1).

4.3 Practical Implications

For professionals using less than 5,000 samples of meteorological data:

- **Classification tasks**: Utilize designed features in conjunction with conventional machine learning (Logistic Regression, Random Forest).

- **Regression tasks**: Modest gains can be obtained by deep learning (LSTM, basic Transformers).
- **Multitask learning**: Classification will deteriorate only if regression is prioritized.
- **Architecture choice**: LSTM (95K parameters) works better than complicated Transformers (147K+ parameters), proving that simplicity is preferable.

4.4 Limitations

1. **Single Location**: Results specific to Sydney; climate zones vary
2. **Feature Engineering**: Pre-engineered features may have biased results toward traditional ML
3. **Class Imbalance**: 26/74 split may require specialized techniques (SMOTE, focal loss)
4. **Computational Constraints**: restricted search space for hyperparameters because to training duration

5 Conclusion

This work shows that the performance of deep learning on meteorological time series is task-dependent rather than general. On the same small dataset of 2,284 samples, Transformer and LSTM designs enhance continuous regression (RMSE 11.89 mm vs. baseline 12.04 mm) but fail at binary rainfall classification (F1 ¡ 0.32 vs. baseline 0.588).

Among our principal contributions are:

1. **Empirical threshold discovery**: 2,284 samples sufficient for regression, insufficient for classification
2. **Catastrophic ablation findings**: 71% of Transformer configurations completely fail, revealing extreme hyperparameter sensitivity
3. **Multitask trade-off quantification**: Classification degraded 34.5% while regression improved 1.2%
4. **Practical guidelines**: Architecture recommendations based on data volume and task type

Future research ought to investigate:

- Transfer knowledge from more extensive meteorological datasets
- Neural networks with physics knowledge that use atmospheric equations
- Prediction in two stages (classify first, then regress)
- Aggregation of regional models in comparable climate zones

Our results indicate promise for continuous forecasting even with limited datasets, but they also confirm that deep learning cannot overcome basic data limits for discrete prediction tasks, even with adequate implementation and considerable optimization.

References

[1] Wilks, D. S. (2011). *Statistical Methods in the Atmospheric Sciences* (3rd ed.). Academic Press.

[2] Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[3] Weather Dataset - Rattle Package. Kaggle.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

[5] Brownlee, J. (2018). *Deep Learning for Time Series Forecasting*. Machine Learning Mastery.

[6] Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81, 84-90.

[7] Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120, 70-83.

[8] Chawla, N. V., et al. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321-357.

[9] Lin, T. Y., et al. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2980-2988.