DOI - 10.68648/ijte-q-2026-49-03-009

Cross-City Transfer Learning for Daily Rainfall Forecasting at Pune, India

Sumit Suman, Vishal Kumar Chaudhary
Assistant Professors
Dept. of Civil Engineering
Government Engineering College Khagaria
Khagaria, Bihar
✉ suman.sumit@geckhagaria.org.in
✉ vishal@geckhagaria.org.in

Tanmay Kumar
Assistant Professor
Dept. of Civil Engineering
Government Engineering College Samastipur
Samastipur, Bihar
✉ tanmaykumar790@gmail.com

ABSTRACT

Data-driven rainfall models normally need long local records, which many stations lack. This study asks whether a transformer pre-trained on data-rich coastal Mumbai can cut both the local data and the computation needed to forecast daily rainfall at Pune, a rain-shadow city 120 km inland. RainFormer, a transformer encoder-decoder, first trained on Pune's full record, gave a reference test RMSE of 12.522 mm. It was then pre-trained on Mumbai, and only its last decoder block and output head, 52,481 of its 194,177 parameters, were fine-tuned on 10 to 100% of Pune's record, against models trained from scratch on identical data. Transfer beat the scratch model at every share. With 10% of the local record its RMSE was within 0.43% of the reference, it equalled the reference at 50% and bettered it at 60% (12.441 mm), so full-data accuracy was reached with 40% less local data while training 73% fewer parameters. Beyond that point the advantage narrowed and the transfer RMSE rose slightly, to 12.479 mm at 100%, indicating over-fitting of the fine-tuned layers to Pune's historical regime. Cross-city transfer can therefore replace much of a station's record at a fraction of the training cost.

KEYWORDS *: Transfer learning, Transformer, Rainfall forecasting, Fine-tuning, Computational efficiency.*

NOTATIONS

Symbol	Meaning
A(S)	Transfer advantage at share S (mm)
C	Reduction in trainable parameters
D	Saving in local training data (%)
n	Number of test samples (567)
S	Percentage share of Pune's record
S_{match}	Smallest share matching Experiment A
RMSE	Root mean square error (mm)
y_i, $\hat{y}_i$	Observed and forecast rainfall (mm)

INTRODUCTION

Daily rainfall forecasts support flood preparedness and water management in Indian cities, where the southwest monsoon delivers most of the annual rain in intense spells. Extreme rain events have become markedly more frequent over central India [1], and in the 2019 monsoon heavy rain and flooding affected several districts of Maharashtra, among them were Mumbai and Pune [2].

Deep learning has improved precipitation forecasting [3], but such models learn from data and their skill depends on long, complete local records. Rain-gauge records in India vary widely in length and completeness [4], so a newly instrumented station cannot support a model trained from scratch. Transfer learning addresses this by reusing knowledge from a data-rich source for a related target with scarce data [5]. Freezing the general lower layers of a network and retraining only the task-specific upper ones [6] also reduces the number of parameters trained. In hydrology, streamflow models pre-trained on a data-rich region and fine-tuned locally have outperformed locally trained models [7], and networks trained on many catchments have been shown to learn behaviour that generalises across basins [8].

Two questions remain open for station-scale rainfall: how much local data a transferred model needs to match a model trained on the full local record, and whether the benefit persists as local data accumulate, given evidence that the advantage of pre-training shrinks as target data grow [9, 10]. This study answers both for Mumbai, on the windward coast, and Pune, 120 km inland in the rain shadow of the Western Ghats. RainFormer, a transformer encoder-decoder, was trained on Pune's full record (Experiment A) and, separately, pre-trained on Mumbai and fine-tuned on increasing shares of Pune's record, each against a model trained from scratch on the same data (Experiment B).

LITERATURE REVIEW

Deep learning for precipitation has developed largely around nowcasting, in which convolutional recurrent networks extrapolate radar fields minutes to hours ahead [3]. A radar nowcasting network named Rainformer applies window-based attention to that task [11]; it shares its name with, but is unrelated to, the station-based model used here. In India, Khan and Maity [12] predicted daily rainfall across Maharashtra, Sunil et al. [13] forecast Mumbai rainfall 6 and 12 hours ahead with a ConvLSTM2D model, and Ranjan et al. [14] combined LSTM layers with transformer encoders for eight cities using the same public dataset as this study. Each model is trained on the location it forecasts, from a long local record.

The transformer [15] replaces recurrence with self-attention and has been widely adapted to time-series forecasting; its appetite for data makes it a natural candidate for transfer learning. In hydrology, Kratzert et al. [8] showed that networks trained on many catchments learn transferable behaviour, and Ma et al. [7] transferred streamflow models from data-rich to data-sparse continents by weight initialisation and freezing. These studies concern streamflow, in which rainfall is an input rather than the quantity forecast. Transfer can also fail: the benefit of pre-training diminishes as target data increase [9], can reverse when they are plentiful [10], and fine-tuning progressively overwrites what was learned from the source [16].

Research gap. No study appears to have examined how much local data a transformer pre-trained at one Indian city needs to forecast daily rainfall at another, what computation partial fine-tuning saves, or where the benefit of transfer ends. This paper addresses that gap for the Mumbai-Pune pair.

METHODOLOGY

Study area and data

Mumbai (19.08°N, 72.88°E) lies close to sea level on the windward Konkan coast and Pune (18.52°N, 73.86°E) about 560 m above sea level, 120 km east-southeast, in the lee of the Western Ghats (Fig. 1). Monsoon flow from the Arabian Sea is forced over the Ghats, giving heavy orographic rain on the windward slopes and much lower totals inland, where rainfall at Pune also depends on the dynamical response of the flow crossing the Ghats [17].

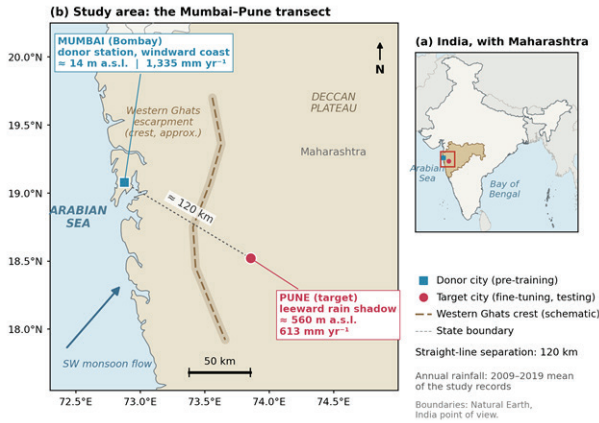


Fig. 1 Study area: (a) India, with Maharashtra highlighted (boundaries: Natural Earth, India point of view); (b) Mumbai, the donor station, and Pune, the target station, on either side of the Western Ghats.

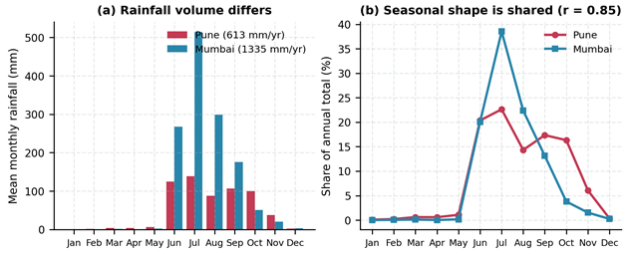


Fig. 2 Rainfall at Pune and Mumbai, 2009-2019: (a) mean monthly totals; (b) the same curves normalised by each city's annual total, showing the shared timing of the monsoon.

The data were taken from the open Kaggle collection

Historical Weather Data for Indian Cities [18], compiled from the World Weather Online historical service and used in recent Indian forecasting work [14]. The Pune and Mumbai files each hold 96,432 hourly rows of 25 fields covering 1 January 2009 to 1 January 2020; they are provider-processed city-level estimates rather than gauge observations. Mumbai receives 1,335 mm of rain a year against Pune's 613 mm, yet the monthly curves, normalised by each city's annual total, correlate at r = 0.85 (Fig. 2). In 2019 Pune received 1,787 mm, 3.6 times its 2009-2018 mean.

Data preparation

Ten uninformative columns were removed, among them clock-time strings, a snowfall column that was zero throughout and three apparent-temperature indices that duplicate air temperature. The remaining 14 variables were aggregated to daily values and expanded into 27 predictors: calendar terms, current weather, physical interactions such as the dew-point depression, and antecedent-rainfall statistics including lags and rolling means. Every history-based predictor uses only days before the forecast day, leaving 3,988 usable days per city, split chronologically into training (70%), validation (15%) and test (15%, May 2018 to January 2020) periods and scaled with each city's own training statistics. A sliding window pairs 30 days of predictors with the next three days of rainfall, giving 2,759 training and 567 test samples per city; the model predicts scaled day-to-day changes, from which forecasts are rebuilt and clipped at zero. The test period averages 3.79 mm of rain per day against 1.39 mm in training.

RainFormer model

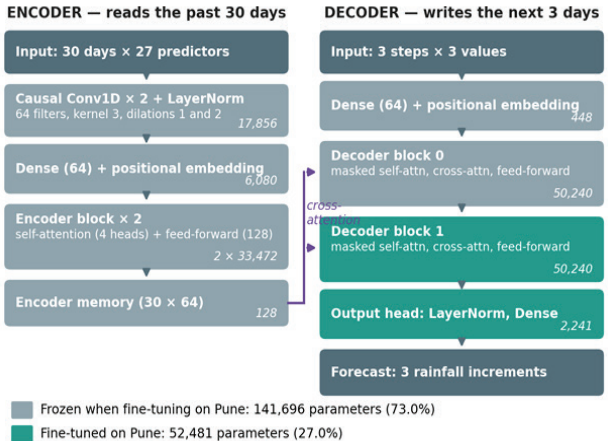


Fig. 3 RainFormer architecture, with the parameters of each block. Grey blocks are frozen and teal blocks fine-tuned in phase 2 of Experiment B; elsewhere every block is trained

RainFormer is a transformer encoder-decoder with 194,177 trainable parameters (Fig. 3). Two causal convolutions and two self-attention blocks encode the 30-day history into a memory, and two decoder blocks, each with masked self-attention, cross-attention to that memory and a feed-forward network, generate the three-day forecast through an output head. Attention follows the scaled dot-product form [15]. All models were trained with a horizon-weighted Huber loss [19] and the Adam optimiser [20], with early stopping on the validation error.

Experimental set-up

The experiments are summarised in Fig. 4 and Table 1. In Experiment A, a randomly initialised RainFormer was trained on all 2,759 Pune samples, about 7.6 years, with every parameter trainable; its test RMSE is the study's reference. Experiment B first pre-trained a new model on Mumbai, using no Pune data, and applied it to Pune's test period without fine-tuning to measure zero-shot transfer.

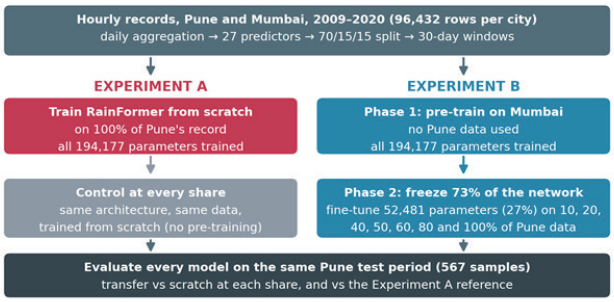


Fig. 4 Workflow: shared preprocessing, Experiment A on Pune only, and Experiment B pre-training on Mumbai and fine-tuning on shares of Pune data against a same-share scratch control.

In phase 2 the stem, both encoder blocks, the decoder projection and the first decoder block were frozen, and only the second decoder block and the output head, 52,481 parameters or 27.0% of the model, were fine-tuned on Pune data: the frozen layers hold what the cities share, the trainable layers map to Pune's smaller totals [6]. Fine-tuning was repeated on the most recent 10, 20, 40, 50, 60, 80 and 100% of Pune's samples, recent data

being used because a new station holds recent records. At each share a model was also trained from scratch on the same samples, so the difference between them isolates the contribution of pre-training. The work used TensorFlow 2.20 on one GPU with a fixed random seed, and all models were tested on the same Pune period.

Table 1. Training configuration and computational load

Setting	Experiment A (scratch)	Experiment B, phase 1 (pre-training)	Experiment B, phase 2 (fine-tuning)
Training data	Pune, 100% (2,759 samples)	Mumbai, 100% (2,759 samples)	Pune, 10-100% (276-2,759 samples)
Initial weights	Random	Random	Mumbai model of phase 1
Trainable parameters	194,177 (100%)	194,177 (100%)	52,481 (27.0%)
Frozen parameters	0	0	141,696 (73.0%)
Adam optimiser state (values)	388,354	388,354	104,962
Back-propagation through	Whole network	Whole network	Last decoder block and head
Learning rate; maximum epochs	1×10^{-3}; 90	1×10^{-3}; 70	2×10^{-4}; 50
Epochs run; training time	28; 85 s	39; 77 s	Varies with share

Performance metrics

Forecasts for the first day were evaluated over the 567 test samples by the root mean square error,

$$RMSE = \sqrt{[\ (1/n)\ \Sigma_i\ (\hat{y}_i - y_i)^2\ ]} \quad (1)$$

where ŷi and yi are forecast and observed rainfall and n = 567. Persistence (tomorrow equals today) and an always-dry forecast serve as naive baselines. Writing S for the percentage share of Pune's record used, the transfer advantage is

$$A(S) = RMSE_{scratch}(S) - RMSE_{transfer}(S) \quad (2)$$

the saving in local data is

$$D = 100 - S_{match} \quad (\%) \quad (3)$$

where S_{match} is the smallest share whose error does not exceed the reference, and the computational saving is the reduction in parameters trained,

$$C = 1 - N_{trainable} / N_{total} \quad (4)$$

RESULTS AND DISCUSSION

Reference performance

Trained on the full Pune record, RainFormer converged after 28 epochs at a test RMSE of 12.522 mm, 4.9% better than persistence (13.169 mm) and better than the always-dry forecast (12.759 mm). The margin is modest because daily point rainfall is hard to forecast and the test period, dominated by the 2019 monsoon, is 2.7 times wetter than training.

Local data needed after transfer

Applied to Pune without fine-tuning, the model pre-trained on Mumbai reached 12.728 mm, only 1.6% above the reference and better than both naive baselines: a model that has never seen Pune captures much of its rainfall behaviour, reflecting the monsoon rhythm the two cities share (Fig. 2).

Table 2 and Fig. 5 give the effect of fine-tuning. Transfer was more accurate than scratch at every share. With a 10% share, 276 samples or about nine months, transfer reached 12.576 mm, only 0.43% above the reference, whereas scratch on the same data reached 12.775 mm, 2.0% above it and worse than forecasting no rain at all. The transfer error was almost unchanged up to 40%, equalled the reference at 50% (12.525 mm) and fell below it at 60% (12.441 mm) and 80%. Full-data accuracy was therefore reached with a 60% share, a saving of D = 40%, about three years of data.

Table 2. Test RMSE of the transferred and same-share scratch models at each share of Pune data

Percentage share of Pune data	Training samples (≈ years)	Transfer RMSE (mm)	Scratch RMSE (mm)	Advantage (mm)
0% (zero-shot)	0	12.728	—	—
10%	276 (0.76)	12.576	12.775	+0.199
20%	552 (1.51)	12.574	12.721	+0.147
40%	1,104 (3.02)	12.574	12.683	+0.108

50%	1,380 (3.78)	12.525	12.635	+0.110
60%	1,655 (4.53)	12.441	12.585	+0.145
80%	2,207 (6.04)	12.442	12.514	+0.072
100%	2,759 (7.55)	12.479	12.522	+0.043

Experiment A reference 12.522 mm; persistence 13.169 mm; always-dry 12.759 mm. The scratch model at 100% is Experiment A itself.

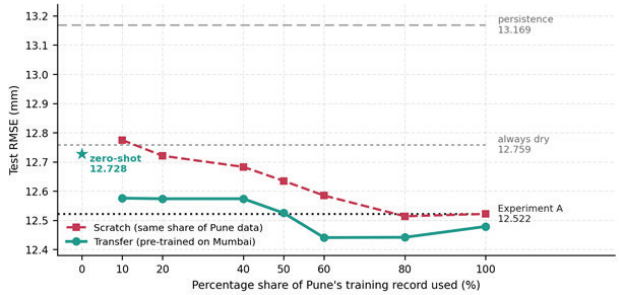


Fig. 5 Test RMSE of the transferred model and of a scratch model trained on the same share of Pune data; the dotted line is the Experiment A reference and the star the zero-shot model.

Computational benefit

Fine-tuning also reduced the computation required. Only 52,481 of the 194,177 parameters were updated, so from equation (4) the saving is C = 73.0%: gradients and weight updates covered about a quarter of the network, and back-propagation was confined to the last decoder block and the output head. The Adam optimiser keeps two moment estimates per trainable parameter [20], so it held 104,962 values instead of 388,354 (Table 1). The schedule was lighter too, at most 50 epochs against 90, and at the 60% share each epoch processed 1,655 samples rather than 2,759. The forward pass still runs through the whole network, so the saving lies in back-propagation and the weight update, not inference. Pre-training on Mumbai is a one-time cost, 77 s here, reusable for other stations.

Diminishing advantage and over-fitting

The advantage of transfer was largest where local data were scarcest, 0.199 mm at a 10% share, and fell to 0.072 mm at 80% and 0.043 mm at 100% (Table 2); averaged over the two smallest and the two largest shares it declined by two-thirds, from 0.173 to 0.058 mm. Beyond 60 to 80% the transfer error itself rose, reaching 12.479 mm with the full record, although it stayed below both the reference and the scratch model trained on the same data.

This is attributed to over-fitting of the fine-tuned layers. The model had been pre-trained on the whole Mumbai record, whose rainfall is about twice as variable as Pune's (training standard deviations 7.63 mm and 3.65 mm), so its response to heavy rain comes largely from Mumbai. Fitting the unfrozen layers to more of Pune's drier training period, 1.39 mm per day against 3.79 mm in testing, specialises them to that regime and erodes the inherited response on the heavy-rain days that dominate the RMSE, while the frozen 73% cannot improve with more data. This agrees with evidence that the benefit of pre-training diminishes as target data grow [9], can reverse when they are plentiful [10], and that fine-tuning overwrites source knowledge [16].

Earlier hydrological transfer moved streamflow models between regions [7], and Indian rainfall studies trained their models on the stations they forecast [12, 14]. This work instead quantifies, against a control trained on identical data, the local record and the computation that transfer replaces, and locates where the benefit ends.

CONCLUSION

The knowledge needed to forecast Pune's daily rainfall proves to be largely contained in the Mumbai record: what the monsoon does on the windward coast carries across the Western Ghats, and only the mapping to Pune's smaller rainfall has to be relearned locally. The contribution of this work is to measure, against a like-for-like control, how much of a station's record and how much training computation such a transfer replaces, and where it ceases to pay.

The implication is general rather than prescriptive. Where a station's record is short, a model transferred from a data-rich neighbour is the more economical option, since it approaches full-data accuracy while only about a quarter of the network is trained. As the record lengthens the benefit narrows and the two approaches converge, so further data may be better used to adapt more of the network, or to train locally; the share at which this happens will depend on the stations, the model and the period studied. The study can be

extended by repeating the experiments over several seeds, pre-training on more than one donor station, and verifying heavy-rain events against gauge observations.

REFERENCES

1. M. K. Roxy, S. Ghosh, A. Pathak et al. (2017), "A threefold rise in widespread extreme rain events over central India", Nature Communications, Vol. 8, Article 708. https://doi.org/10.1038/s41467-017-00744-9
2. A. K. Mishra, M. S. Meer and V. Nagaraju (2024), "Exploring extreme flood events of a western state of India during monsoon season of 2019 from space", MAUSAM, Vol. 75, No. 1, Pp. 245-248. https://doi.org/10.54302/mausam.v75i1.3568
3. X. Shi, Z. Chen, H. Wang et al. (2015), "Convolutional LSTM network: A machine learning approach for precipitation nowcasting", Advances in Neural Information Processing Systems, Vol. 28, Pp. 802-810.
4. D. S. Pai, L. Sridhar, M. Rajeevan et al. (2014), "Development of a new high spatial resolution (0.25° × 0.25°) long period (1901-2010) daily gridded rainfall data set over India and its comparison with existing data sets over the region", MAUSAM, Vol. 65, No. 1, Pp. 1-18.
5. S. J. Pan and Q. Yang (2010), "A survey on transfer learning", IEEE Transactions on Knowledge and Data Engineering, Vol. 22, No. 10, Pp. 1345-1359. https://doi.org/10.1109/TKDE.2009.191
6. J. Yosinski, J. Clune, Y. Bengio and H. Lipson (2014), "How transferable are features in deep neural networks?", Advances in Neural Information Processing Systems, Vol. 27, Pp. 3320-3328.
7. K. Ma, D. Feng, K. Lawson et al. (2021), "Transferring hydrologic data across continents: Leveraging data-rich regions to improve hydrologic prediction in data-sparse regions", Water Resources Research, Vol. 57, No. 5, e2020WR028600. https://doi.org/10.1029/2020WR028600
8. F. Kratzert, D. Klotz, G. Shalev et al. (2019), "Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets", Hydrology and Earth System Sciences, Vol. 23, No. 12, Pp. 5089-5110. https://doi.org/10.5194/hess-23-5089-2019
9. K. He, R. Girshick and P. Dollár (2019), "Rethinking ImageNet pre-training", Proceedings of the IEEE/CVF International Conference on Computer Vision, Pp. 4918-4927.
10. B. Zoph, G. Ghiasi, T.-Y. Lin et al. (2020), "Rethinking pre-training and self-training", Advances in Neural Information Processing Systems, Vol. 33, Pp. 3833-3845.
11. C. Bai, F. Sun, J. Zhang et al. (2022), "Rainformer: Features extraction balanced network for radar-based precipitation nowcasting", IEEE Geoscience and Remote Sensing Letters, Vol. 19, Art. no. 4023305, Pp. 1-5. https://doi.org/10.1109/LGRS.2022.3162882
12. M. I. Khan and R. Maity (2020), "Hybrid deep learning approach for multi-step-ahead daily rainfall prediction using GCM simulations", IEEE Access, Vol. 8, Pp. 52774-52784. https://doi.org/10.1109/ACCESS.2020.2980977
13. A. Sunil, A. Devda, R. Murthy and B. Deepthi (2025), "Enhanced rainfall forecasting for Mumbai using data-driven ConvLSTM2D models at fine spatial and temporal scales", Natural Hazards, Vol. 121, Pp. 11931-11956. https://doi.org/10.1007/s11069-025-07267-3
14. P. Ranjan, V. Akshay, P. R. Girijesh Kumar and M. Jagannath (2025), "Improving weather predictions accuracy with a hybrid LSTM-transformer model: A study on Indian climate", IEEE Access, Vol. 13, Pp. 169014-169025. https://doi.org/10.1109/ACCESS.2025.3614473
15. A. Vaswani, N. Shazeer, N. Parmar et al. (2017), "Attention is all you need", Advances in Neural Information Processing Systems, Vol. 30, Pp. 5998-6008.
16. J. Kirkpatrick, R. Pascanu, N. Rabinowitz et al. (2017), "Overcoming catastrophic forgetting in neural networks", Proceedings of the National Academy of Sciences, Vol. 114, No. 13, Pp. 3521-3526. https://doi.org/10.1073/pnas.1611835114
17. S. Dutta, S. I. Laskar and M. Maiti (2006), "Does the Western-Ghats play any dynamical role in the distance effect of vortex over the Bay of Bengal on the enhancement of monsoon rainfall over Pune?", MAUSAM, Vol. 57, No. 4, Pp. 567-578. https://doi.org/10.54302/mausam.v57i4.496
18. H. Soneji (n.d.), "Historical Weather Data for Indian Cities", Kaggle, https://www.kaggle.com/datasets/hiteshsoneji/historical-weather-data-for-indian-cities
19. P. J. Huber (1964), "Robust estimation of a location parameter", The Annals of Mathematical Statistics, Vol. 35, No. 1, Pp. 73-101. https://doi.org/10.1214/aoms/1177703732
20. D. P. Kingma and J. Ba (2015), "Adam: A method for stochastic optimization", Proceedings of the 3rd International Conference on Learning Representations, arXiv:1412.6980.