MODEL DEVELOPMENT

`

3 Chapter 3

3.1 Model Development

3.1.1 Network Architecture

In this study, multi-step (eight-step) ahead prediction was achieved using a deep learning framework. The model design consists of an encoder and decoder, forming a hybrid model for Seq2Seq prediction. This model's primary constituents were 3DCNN, ConvLSTM, and BLSTM. Figure 3-1 illustrates the basic architecture of our model.

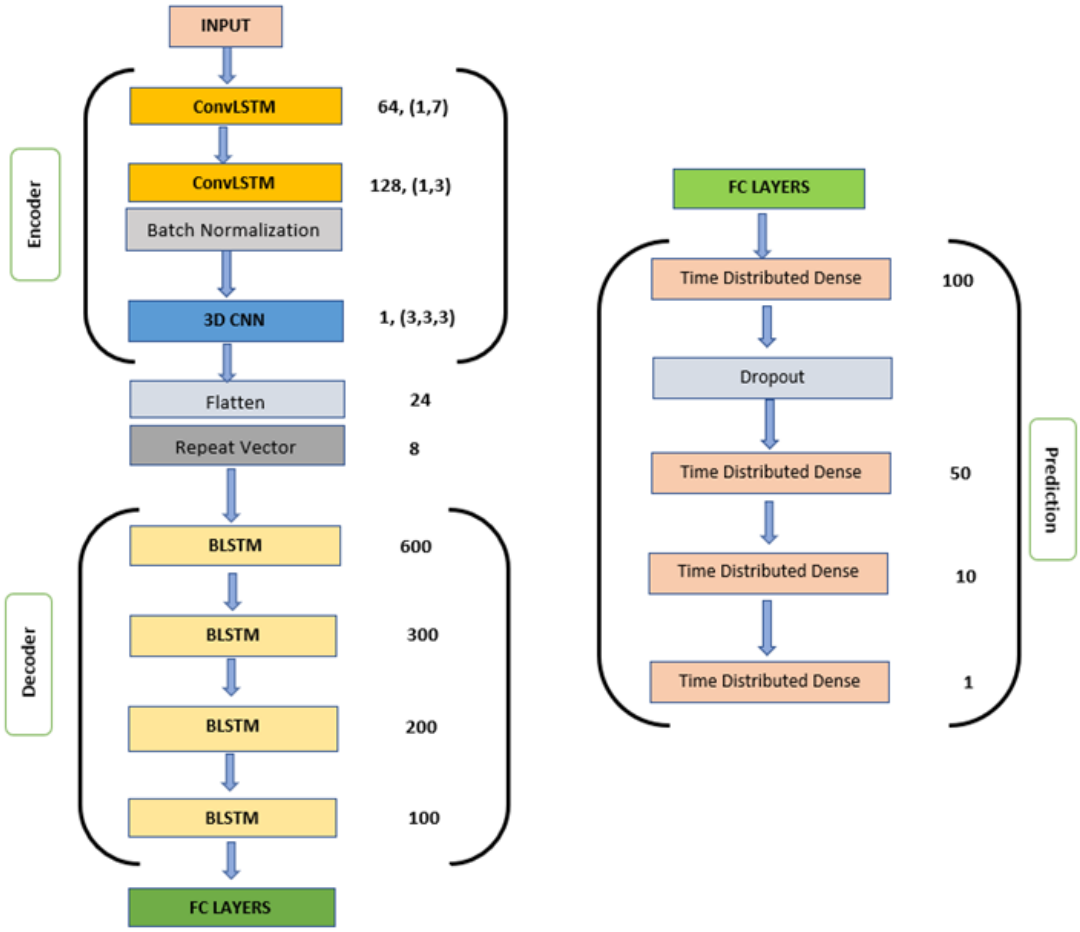

Figure 3-1: Hybrid Model

This architecture is basically built with LSTM and 3D-CNN networks. Because the LSTM model can store data in its inherent periodic cells and that could be retrieved at

`

any time steps, it has been used extensively for time series prediction. Although non-Gaussian noise inherent in the data set cannot be filtered out, LSTM networks perform incredibly well in minimizing Gaussian noise of the data [126]. In order to mitigate these drawbacks, a BLSTM network was utilized to lessen overfitting of data [127]. ConvLSTM also performs better on datasets having a lot of temporal information and long-term sequential features. Additionally, 3DCNN models were an improvement over 2DCNN models because they were better at processing huge amounts of contextual data that were useful for extracting spatiotemporal features. 3DCNN can extract features from data by segmenting data into different frequency domains over time. This property is also useful in minimizing noise present in the data.

In our model, a ConvLSTM encoder layer receives input and creates a feature map. The feature map is then transferred to a second ConvLSTM layer. The second ConvLSTM layer filters and further refines the feature map generated by previous layer. At this stage Batch Normalisation layer was used to improve training of the model. To extract patterns of spatio-temporal nature, the output is passed into a 3DCNN. After the 3DCNN layer, the output is sent to the decoder layer consisting of four BLSTM networks. A sequence of values of 8 hours, will be produced by BLSTM. The three fully connected BLSTM layers function as interpretation layers for each time step in the output sequence. The model's output could be obtained by fully connected (FC) layer, which gives the eight steps ahead prediction. In order to reduce overfitting, a dropout layer was added after the initial BLSTM. In a CNN model, each filter layer abstracts a feature. Because the first layers of the network receive noisy input, less numbers of filters were applied to capture only the essential features of the layers. In order to capture a deeper level of feature abstraction, further filters were added in the following levels. More features can be captured by a filter or kernel with a smaller size than one with a greater size. In the initial ConvLSTM layer, 64 filters of size (1,7) were applied. In the second ConvLSTM layer, the kernel size was reduced to (1,3) and the number of filters was raised to 128. Odd numbers of kernel sizes were selected in order to ensure symmetry around the center.

The model uses BLSTM layer, which functions as a decoder and produces a sequence of values as output. Out of sample testing and cross validation methods were applied to judge the model's performance. Earlier, [128] used a similar architecture for time

`

series data to learn smart manufacturing challenges without use of 3DCNN. In the current work, the 3DCNN layer received input from the Stacked ConvLSTM layer. The time series data for air pollution are the result of a complicated interaction between a variety of stochastic and dynamic processes with various characteristic frequencies [129]. By using 3DCNN to account for the unique features, the network's capacity to make better predictions was subsequently enhanced.

We used three consecutive sequences of 8-hour length as input (24 hour input) and the next sequence (8-hour data) as target to forecast 8 hour ahead prediction of $PM_{2.5}$. concentration. Deep learning model demands a large size data set for better performance. Therefore in order to create a large data set for training the model, overlapping moving window technique was applied [130,131].

Let's have a look at the time series $u(t) = \{u_1, u_2, u_3, \ldots\ldots, u_t\}$. Based on the last value and a constant moving window of size w, the subsequent k values in the series $\hat{s} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_k)$ comparable to $(u_{t+1}, u_{t+2}, \ldots\ldots, u_{t+k})$ could be described as follows:

$$\hat{s} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_k) = f(u_{t-w}, u_{t-w+1}, u_{t-w+2}, \ldots, u_t)$$

An input set $U \in R^{n \times w}$ and an output set $S \in R^{n \times k}$ is produced with help of applying above formula in an univariate time series. The training data size (n) is indicated by the formula: $n = (N - w - k + 1)$.

Model parameters were generalized in order to produce an optimal value for the majority of the stations because it was employed in numerous stations distributed across various Indian regions.

In order to find best fit model following encoder decoder based architectures (Figure 3-2 to Figure 3-8) were designed and explored :

(i) Stacked LSTM, (ii)CNN-LSTM, (iii) ConvLSTM-LSTM,

(iv) ConvLSTM-BLSTM, (v) BConvLSTM-LSTM, (vi) BConvLSTM-BLSTM

and (vii) BLSTM-BLSTM

`

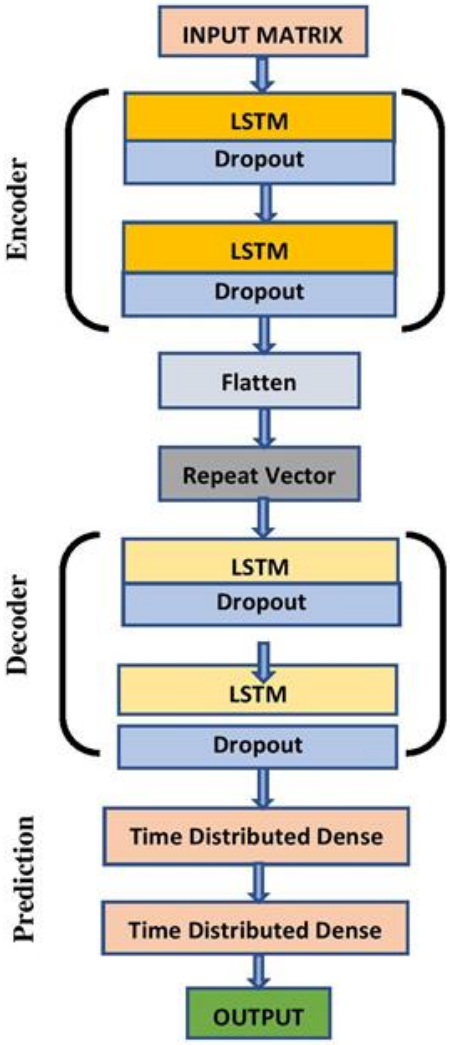

Figure 3-2: Stacked LSTM

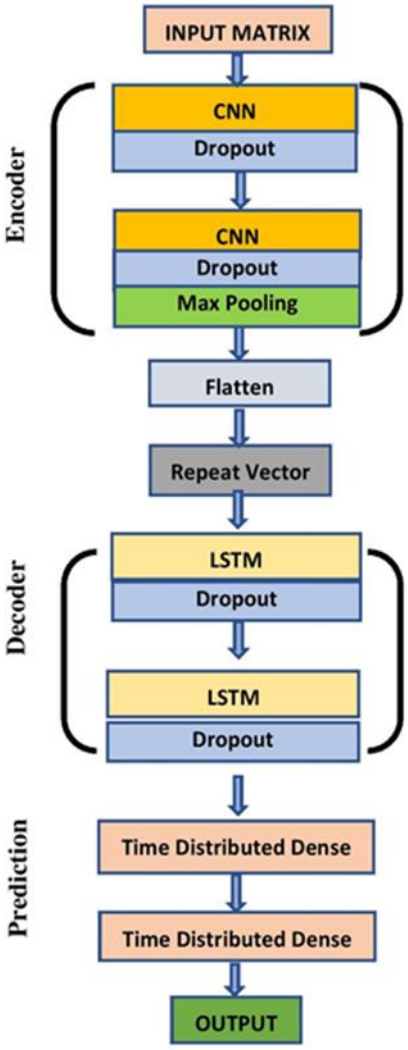

Figure 3-3: CNN -LSTM

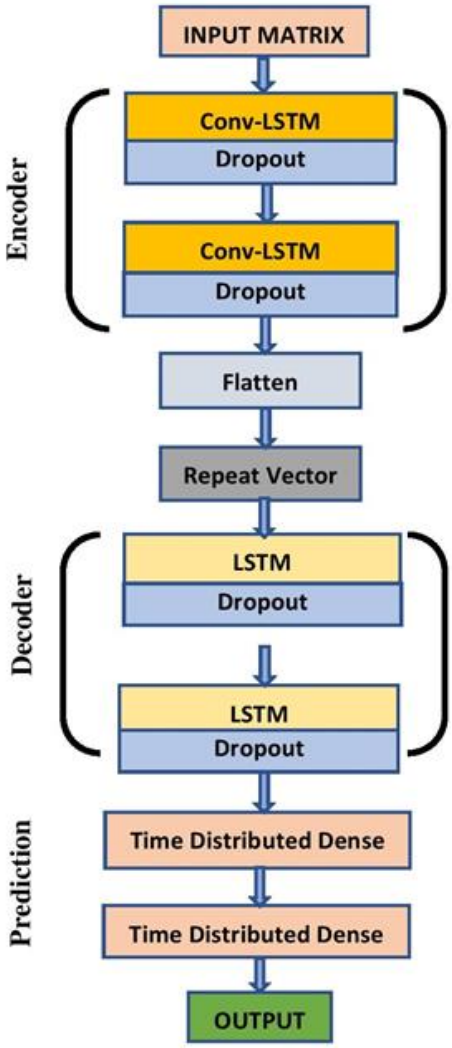

Figure 3-4: ConvLSTM-LSTM

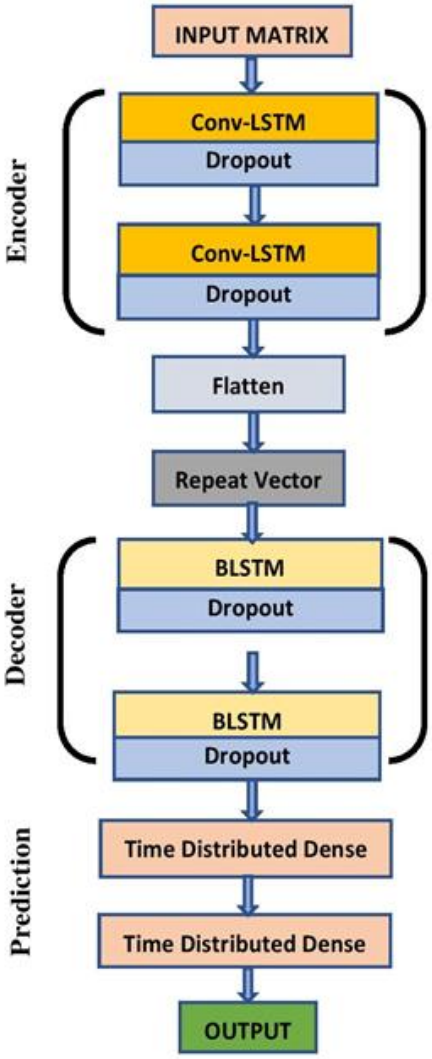

Figure 3-5: ConvLSTM-BLSTM

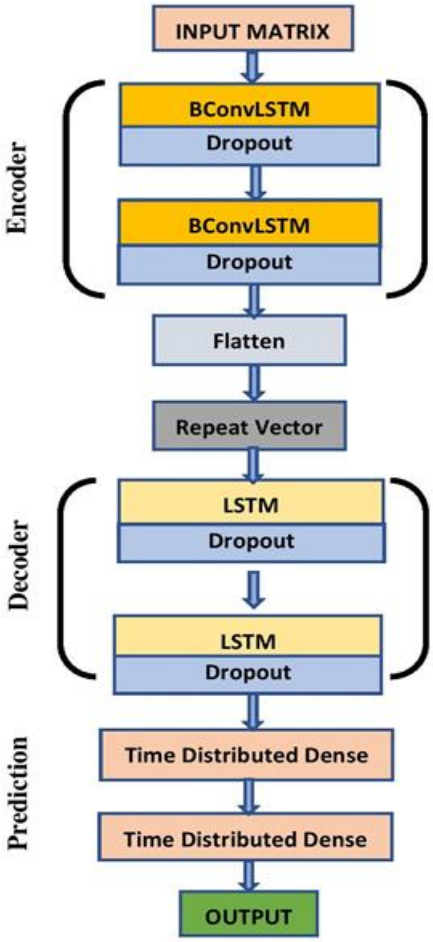

Figure 3-6: BConvLSTM-LSTM

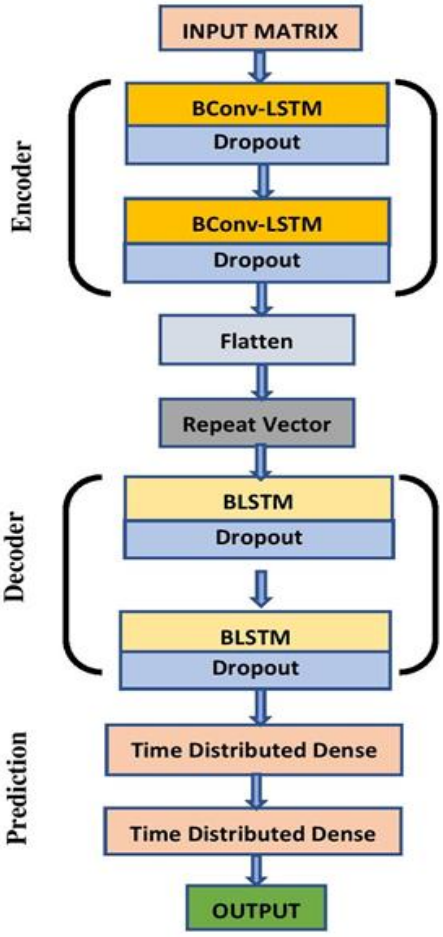

Figure 3-7: BConvLSTM-BLSTM

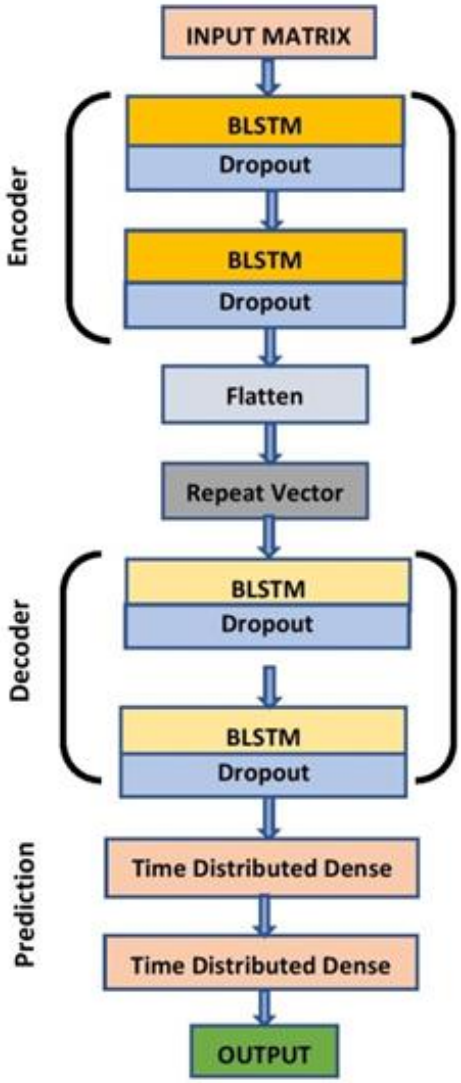

Figure 3-8: BLSTM-BLSTM

3.1.2 Overlapping Moving Window

This study attempted to predict $PM_{2.5}$ levels for the next 8 hours based on previous data from the last 24 hours. In order to achieve this, entire time series data was divided into 8 equal divisions. For training purpose, we have used 3 sequences or 24 hours data as input and one sequence or 8 hours data as output. However, this amount of instances was not enough to train a deep learning model. To create more training data, time series data were trained using an overlapping moving window method.

`

Here, moving the entire sequence one step forward, have created more training datasets. (Figure 3-9).

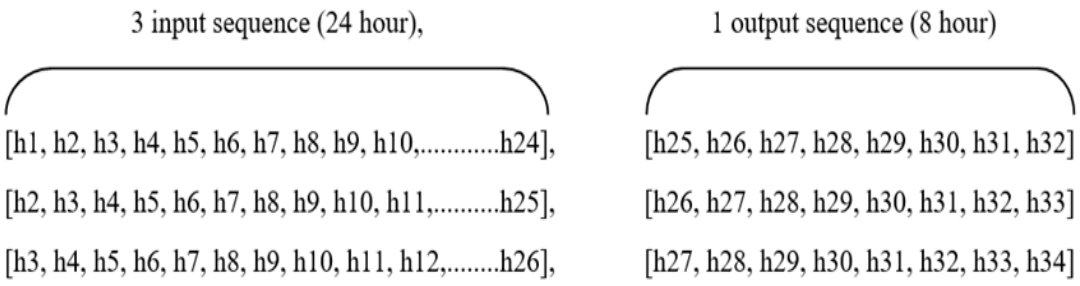

Figure 3-9: Overlapping Moving Window

3.1.3 Hyperparametrs

The performance of a machine learning model is highly governed by its hyperparameters. Hyperparametrs are special types of parameters in machine learning. By tuning the hyperparametrs we can obtain different results. The hyperparameters used in the hybrid model for all India implementation are listed in Table 3-1.

The most often used activation function for deep learning was the rectified linear unit (ReLU), which has the formula f(x)=max (0, x). The "swish" activation function, f(x)=x sigmoid(x), was developed by the Google Brain team [132] and works better in a deeper network. The 'tanh' activation function was employed for the BLSTM component of the study, and the 'swish' activation function for rest of the model component. Several kinds of optimizing algorithms are available to lower the loss function during model training. Some of the mostly used optimizers are Adaptive Moment Estimation (Adam), Gradient Descent (GD), Momentum Based Gradient Descent (MGD), Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSProp) and Nesterov Accelerated Gradient (NAG). We used the Adaptive Moment Estimation (Adam) optimizer in this work because of its efficiency and adaptability.

`

Table 3-1: Hyperparamters Value

Layer	Hyperparameters	Values
ConvLSTM -1	Filter	64
	Kernel Size	(1,7)
	Activation Function	Swish
ConvLSTM -2	Filter	128
	Kernel Size	(1,3)
	Activation Function	NA
3D CNN	Filter	1
	Kernel Size	(3,3,3)
	Activation Function	Swish
BLSTM -1	Units, Activation Function	600, tanh
BLSTM -2	Units, Activation Function	300, tanh
BLSTM -3	Units, Activation Function	200, tanh
BLSTM -4	Units, Activation Function	100, tanh
Dense -1	Units, Activation Function	100, swish
Dense -2	Units, Activation Function	50, swish
Dense -3	Units, Activation Function	10, swish
Dense -4	Units, Activation Function	1, linear
NA	Dropout	0.3
NA	Learning Rate	10e-4
NA	Optimizer	Adam
NA	Batch Size	64
NA	epoch	100

`

We have used Hyperband optimization while finding best fit model among encoder-decoder based deep learning model. The range of the hyperparameter values are listed in Table: 3-2.

Table 3-2: Hyperband Parameters

LAYER	**HYPERPARAMETER(S)**	**VALUE**
ConvLSTM -1, BiConvLSTM -1	Filter	8, 16, 32, 64, 128, 256, 512, 1024
	Kernel Size	3, 5, 7, 9
ConvLSTM -2, BiConvLSTM -1	Filter	16, 32, 64, 128, 256, 512
	Kernel Size	3, 5, 7, 9
LSTM1, BLSTM1	Units	100, 200, 300, 400, 500, 600, 700
LSTM -2, BLSTM -2	Units	100, 200, 300, 400, 500, 600
Dense -1	Units	10, 100
Dense -2	Units	1, linear activation function
NA	Dropout	0 to 0.5
NA	Learning Rate	1e-4, 1e-5, 1e-6
NA	Optimizer	Adam, SGD, Adadelta, Nadam,RMSprop
NA	Batch Size	16, 32, 64, 128, 256, 512
NA	Activation Function	relu, tanh, sigmoid, swish

`

3.1.4 Model Evaluation

The model's efficacy was evaluated using a Walkforward validation technique. Performance evaluation was performed using statistical measures namely root mean square error (RMSE), mean absolute error (MAE), and mean absolute percent error (MAPE).

The error metrics equations are as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_i - P_i)^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|A_i - P_i|$$

$$MAPE = \left(\frac{1}{n}\sum_{i=1}^{n}\frac{|A_i - P_i|}{|A_i|}\right) * 100$$

Further model performance was also tested through Factor of two of observations (FAC2) and Geometric variance (VG) [133].The following are the equations used in the error metrics:

$$FAC2 = \frac{1}{n}\sum_{i=1}^{n}\frac{P_i}{A_i}, \qquad Data\ fraction\ satisfying\ 0.5 \leq \frac{P_i}{A_i} \leq 2.0$$

$$VG = exp^{\frac{1}{n}\sum_{i=1}^{n}(lnA_i - ln\,P_i)^2}$$

Ai= Observed value, Pi= Model predicted value, n= Total number of samples

`