# Convolutional LSTM applied to dynamic systems

Priyesh Kakka

**Abstract** Deep learning methods such as the convolutional neural network(CNN) and recurrent neural network(RNN) have been widely used in various applications, predominantly for computer vision and natural language processing. The combination of these mentioned applications can be used to predict frames of data varying in time (i.e., video prediction). This idea of combining CNN and RNN has been implemented in the form of a neural network architecture called Convolutional LSTM. This article proposes using Convolutional LSTM with autoencoder architecture in predicting dynamical systems such as fluid flows. Once trained using data, the proposed model can accurately predict the flow sequence for simple problems such as lid-driven cavity and flow past cylinder at varying Reynolds number in the laminar region.

Dynamic system modeling, especially modeling of fluid flow, has been researched extensively in the past. Most dynamic systems are governed by non-linear partial differential equations(PDE's). For most practical problems, these PDEs are usually difficult to solve analytically, and thus, numerical modeling of these simulations is an ad-hoc approach to follow. These numerical simulations, especially turbulent flow, require significant computational resources and usually have a big turnaround time, which is a bottleneck for industrial applications.

With the advent of deep learning, a new data-driven approach for predicting dynamic systems has attracted the attention of researchers in recent years. The philosophy underlying the data-driven model using deep learning is based on the assumption that, given a large dataset, we can obtain a non-linear function $\mathcal{F}$, which maps inputs to its corresponding output. This function which is a neural network can then be used to predict outcomes given corresponding inputs. Based on this assumption, a data-driven model can be constructed for a dynamic system. For the specific case of fluid flow, the grid on which numerical solution is obtained can be considered a matrix of data that evolves in time. This matrix $\in \mathbb{R}^d$ of dimension d can be then trained using recurrent neural networks(RNN) [5].

Recent advances in RNN's to preserve long-term dependencies of data has made it a very effective tool for predicting long sequences. One such model is called the long-short term memory (LSTM) model [2]. LSTM model, was then used along convolutional neural network architecture(ConvLSTM) for precipitation nowcasting[6]. Here, the authors tried to forecast the precipitation with the images of clouds obtained

from the satellite. This approach can be translated to a fluid problem that has variable spatio-temporal dimensions. Similarly, we try to apply the ConvLSTM model to fluid flow problems to predict the sequence of flow in the future based on past data. We further tweak the model and incorporate autoencoders to reduce the parameters in the neural network to be trained and reduce the noise if present in the image.

The model is then tested on moving MNIST dataset and later applied to two-fluid flow problems. The first problem considered is a lid-driven cavity, and the second problem is a flow around the cylinder. It is observed that the trained neural network architecture does a good job in predicting the sequence to sequence input-output, where the network could accurately predict the sequence of five frames simultaneously, given five frames before it as input.

## Model

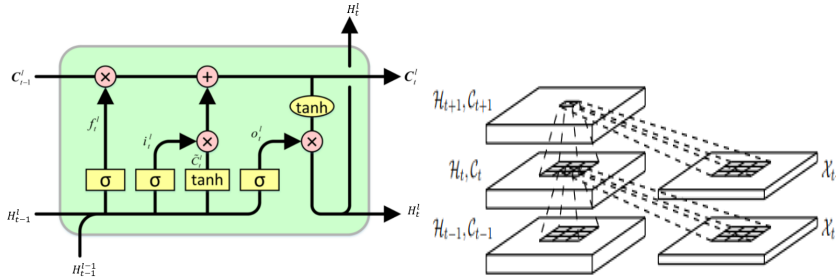
The inputs to the neural network are a series of 2-dimensional snaps of data $\Phi = \{\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \ldots \boldsymbol{\phi}_T\}; \boldsymbol{\phi}_i \in \mathbb{R}^{n \times d}$ for which $\boldsymbol{\phi}_i$ has been discretized by $d$ points in the domain $\Omega$.

By providing series of inputs and targets for optimizing its parameters, we train the network, which is basically finding the mentioned non-linear function $\mathcal{F}$ which maps $\boldsymbol{\phi}_i$ to a an output after $l$ time steps $\boldsymbol{\phi}_{i+l}$. Once the network is trained, it should be able to predict the future sequence in time, given input.

A convolutional neural network is required to handle these large parameters in the two-dimensional input and encode the spatial information. Traditionally, convolutional neural networks are used for image processing and recognition. To provide a temporal dimension to the network, a recurrent neural network is merged with CNN. Here, we use LSTM embedded in Convolutional nets called ConvLSTM. As described by [6], all the inputs $\mathcal{X}_1, \ldots, \mathcal{X}_t$, cell outputs $C_1, \ldots, C_t$, hidden states $\mathcal{H}_1, \ldots, \mathcal{H}_t$, and gates $i_t, f_t, o_t$ of the ConvLSTM are 3D tensors whose last two dimensions are spatial dimensions (rows and columns). To get a better picture of the inputs and states, we may imagine the block show in 1a as vectors standing on a spatial grid. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. This can easily be achieved by using a convolution operator in the state-to-state and input-to-state transitions 1b. The key equations of ConvLSTM are shown in 1 below, where '*' denotes the convolution operator and 'o', as before, denotes the Hadamard product:

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ C_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ C_{t-1} + b_f\right) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tanh\left(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ C_t + b_o\right) \\
\mathcal{H}_t &= o_t \circ \tanh\left(C_t\right)
\end{aligned}
\tag{1}
$$

Further, along with ConvLSTM blocks, usual CNN blocks are used, and an




**(a)** *Vanilla LSTM* **(b)** *ConvLSTM*

**FIGURE 1.** *Representation of LSTM blocks*

encoder-decoder architecture of the neural network is formed as shown 2. Here, we are feeding the input matrix data with the dimensions $B \times S \times H \times W$, where B is the batch size, S is the sequence length of the input (dimension in time), and H, W is the dimensions of the input data. For the architecture mentioned in this article, we have used three stages of encoder blocks, with each stage comprising a CNN block and a ConvLSTM block. There is a reduction in each block's input data matrix dimensions with an increase in its channel size. In every stage, the input data is first passed through the CNN block, whose output is fed to the ConvLSTM block. The ConvLSTM block concatenates the input received with the activation state $\mathcal{H}_t$ and outputs the data, along with the two hidden states $C_t$ and $\mathcal{H}_t$. These hidden states are later fed into the respective stages of the decoder. The decoder architecture is a mirror image of the encoder described and is similar to a U-Net structure [4].

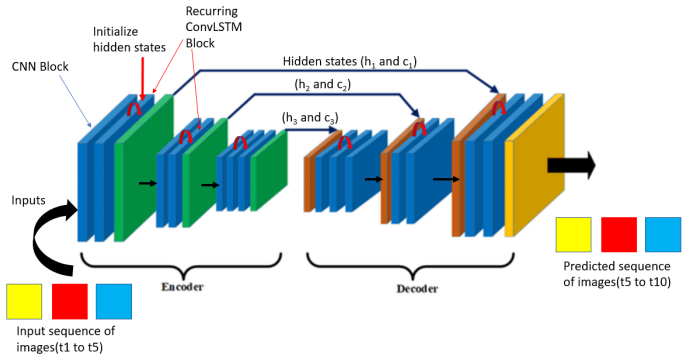
To code the described Autoencoder ConvLSTM model, we use an open-source tool Pytorch 1.6.0 [3]. The loss function used here is the mean square error. The optimizer used is Adams with the learning rate being kept as dynamic using an inbuilt scheduler "ReduceLROnPlateau" with a factor of 0.5. Leaky Relu activation function is used in all the blocks, and group normalization is applied over all the mini-batches.

## Results

### Validation: Moving MNIST

The third model configuration shown in table 1 was tested with a 10000 moving MNIST dataset. The dataset consists of twenty frames (ten input frames and ten target frames). These frames have three digits that move and bounce off from the boundaries of the frame, making a video sequence. Each image is of 64x64 pixels, which is reduced or encoded to 8 pixels at the end of 3rd stage of the encoder.

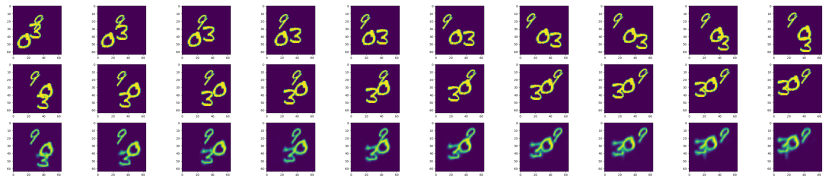
In figure 3, We can observe that the third row matches the second row (which is the actual output), based on the first row as input. Thus, we can say that the model works




*Notes:* Each of the three stages of encoder consist of first block as a CNN block with second block as ConvLSTM block,

**FIGURE 2.** *Representation of Encode-Decoder architecture used*

reasonably well in terms of predicting location and numbers. It can be noted that the accuracy of prediction decreases at the end of the third row but is still readable.



*Notes:* Row one is inputs, row two is targets and the last row is output predicted based on input by neural network

**FIGURE 3.** *Moving MNIST test case*

## Dynamic Problems

Now that we have established that the model works, we can apply it to dynamic problems. In the current work, we focus on fluid problems where the data from numerical simulations are fed as an input matrix, and its future time steps are marked as targets to train the model. Here, we use five-time steps as input and the next five time steps as output for both cases. Once the network is trained, it can be used to predict future time steps given an input.

### Lid-driven Cavity

A lid-driven cavity is an unsteady state problem, which is solved by using stream-vorticity formulation. The time stepping in simulation is explicit, with the time step $\Delta t$ being limited by the CFL of 1. The change in the contours of the streamlines is small for each time step; hence, we extracted the data after every $4\delta t$ to create our training and test datasets. The model has been trained with over 200 sequences of 10 frames. For the generation of 200 sequences, eight simulations of the lid-driven cavity with varying Reynolds numbers (Re), from 500 to 10000 are used, with each generating 25 sequences.

From figure 4, it is observed that the prediction contours are well defined with exact magnitudes. Thus, we can conclude here that the trained neural network was able to predict the outcome accurately.

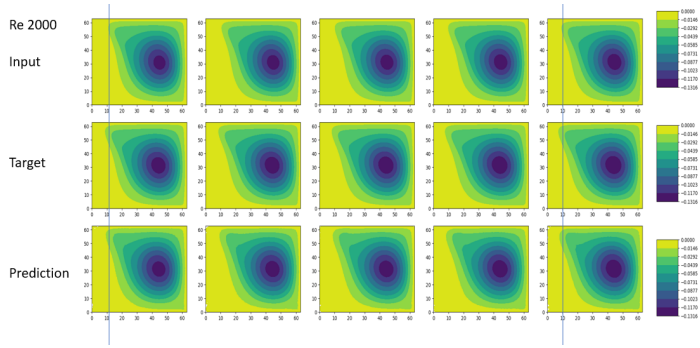



**FIGURE 4.** *Evolution of stream-lines predicted using neural network for the case: lid driven cavity*

### Flow around cylinder

Another fluid problem tested was flow around cylinder where the data consisted of 2000 frames, obtained using five simulations for Reynolds between 250 and 750. The data is obtained from [1].

It can be observed from figure 5, that the flow is predicted quite well for all the three configurations as shown in table 1. Here, the network is trained for 25 epochs. The prediction results shown in figure 5, indicate that the auto-encoder ConvLSTM neural network is excellent in predicting the outcome. The prediction, which is the output of the neural network, is then fed back to obtain the future sequence of time series as shown in figure 6.

**TABLE 1.** *Configuration and parameters*

| Configuration(input channel -> output channel)(Kernal size) | Parameters | MSE error |
|---|---|---|
| (1->16)(3) to (16->32)(3) to (32->32)(3) to (32->64)(3) to (64->64)(3) to (64->64) (3) | 1457825 | 7E-3 |
| (1->16)(3) to (16->64)(3) to (64->64)(3) to (64->96)(3) to (96->96)(3) to (96->96)(3) | 3527201 | 6E-3 |
| (1->16)(3) to (16->64)(5) to (64->64)(3) to (64->96)(5) to (96->96)(3) to (96->96)(5) | 9032225 | 4E-3 |

## Conclusion and discussion

The above results show that the data-driven models work well against the classical numerical simulations for fluid flow problems. Although the network had too many parameters given the complexity of the discussed problem, a smaller neural network with fewer parameters might work well.

Novel neural network architectures such as transformers have worked better and can be used to more solve complex dynamical systems. Although, the applicability of these data-driven models to predict complex flow problems such as second-order statistics in the turbulent regime needs to be studied further.

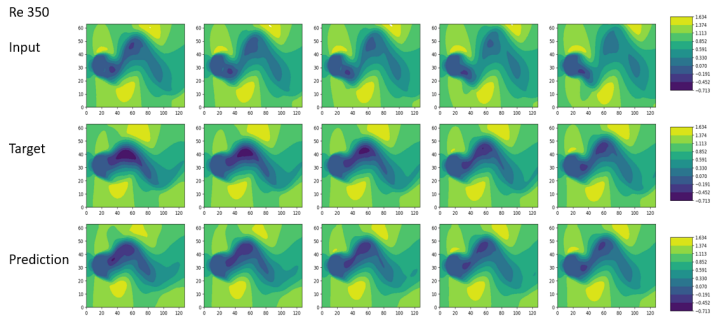
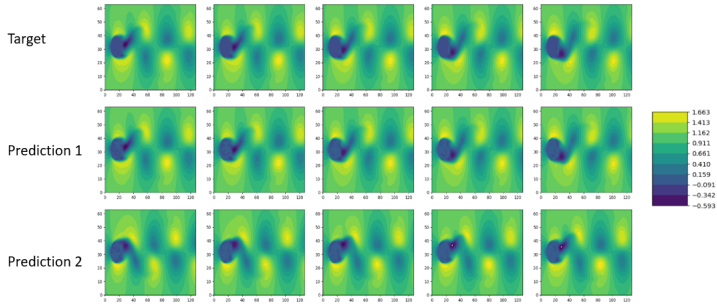



**FIGURE 5.** *Output prediction of stream-wise flow velocity around a cylinder.*

## References

[1] Nicholas Geneva and Nicholas Zabaras. Transformers for Modeling Physical Systems. In: *arXiv preprint arXiv:2010.03957* (2020).




*Notes:* Row one is the target, row two is the prediction and the last row is output predicted based on the first prediction as input to nueral network

**FIGURE 6.** *Stream-wise velocity prediction is fed into the network to obtain future sequence of time steps.*

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In: *Neural computation* 9.8 (1997), pages 1735–1780.

[3] Adam Paszke et al. Automatic differentiation in PyTorch. In: (2017).

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. Paper presented at the: International Conference on Medical image computing and computer-assisted intervention. Springer. 2015, pages 234–241.

[5] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[6] Xingjian Shi et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: *arXiv preprint arXiv:1506.04214* (2015).