FORECASTING LOCAL BEHAVIOR OF MULTI-AGENT SYSTEM AND ITS APPLICATION TO FOREST FIRE MODEL

Beomseok Kang, Minah Lee, Harshit Kumar, Saibal Mukhopadhyay

School of Electrical and Computer Engineering, Georgia Institute of Technology, GA 30332, USA
{beomseok, minah.lee, hkumar64, smukhopadhyay6}@gatech.edu

ABSTRACT

In this paper, we study a CNN-LSTM model to forecast the state of a specific agent in a large multi-agent system. The proposed model consists of a CNN encoder to represent the system into a low-dimensional vector, a LSTM module to learn the agent dynamics in the vector space, and a MLP decoder to predict the future state of an agent. A forest fire model is considered as an example where we need to predict when a specific tree agent will be burning. We observe that the proposed model achieves higher AUC with less computation than a frame-based model and significantly saves computational costs such as the activation than ConvLSTM.

Index Terms— Multi-agent system, Self-organized system, Forest fire, Convolutional neural network, Long short-term memory.

1. INTRODUCTION

Artificial intelligence (AI) in multi-agent systems has been widely used in both synthetic environments and the real world [1 2 3 4]. A key problem to implement the intelligent machines for the systems is to discover hidden interaction rules between agents from sensory signals. Recently, deep neural networks have been studied as powerful nonlinear tools to model the complex interaction rules that govern the systems [5]. However, learning the dynamics of large-scale multi-agent systems is still a challenging problem. Designing the efficient deep learning models will accelerate the promising multi-agent applications such as game AI, autonomous driving, and robotics.

Many prior works have studied multi-agent interactions among few (∼10s) agents assuming all agents are interacting with each other [6]. In contrast, this paper is focused on multi-agent systems with many (1000s∼) agents where agents are locally interacting following non-linear dynamics. The global state of the system is defined by collection of such local interactions. However, our interest is in predicting localized behavior of a few agents in this multi-agent system. One example of such a system is fire propagation in a forest where each tree can act as an agent, and our interest is finding whether a specific tree catches fire. In summary, we aim to predict local behavior of a single agent in multi-agent dynamical system with local interactions.

Such system emerged from the local dynamics of agents is often called self-organization. Self-organization in multi-agent systems is also closely related to cellular automata. Recently, neural cellular automata has shown that complex self-organized systems can be modeled by convolutional neural networks (CNN) [7][8]. The proposed approaches are similar to the autoencoder as they reconstruct the full image of the system to learn the transition of agent states. While the state of all agents are easily accessible, the computational cost of the reconstruction is expensive in the large systems. Moreover, if we only want to predict the local behavior of an agent, the reconstruction may not be necessary. An interesting question is whether we can predict the local behavior of the multi-agent systems without the reconstruction.

In this paper, we propose a CNN-LSTM model to predict an agent state at a region of interest (ROI) without the...
reconstruction. To be specific, the proposed model repeatedly forecasts the state each time step in a prediction window after observing a few sequential frames showing the state of all agents. The model is trained and evaluated on NetLogo, a widely used multi-agent programming environment [9]. We take the forest fire model, where a lot of tree and fire agents are interacting during the evolution of the system, as an example because it is a well-known large multi-agent system with self-organization. Fig. 1 shows two frames in an observation and prediction window. We observe that the proposed model makes the agent state more predictable with less computation than a frame-based model designed with the same encoder and prediction module. Also, we demonstrate that separately learning the spatial and temporal feature significantly saves computational costs such as the activation than ConvLSTM.

2. PROPOSED APPROACH

2.1. Model architecture

Fig. 2 describes the model architecture. A CNN encoder transforms the forest image to a context vector, and then LSTM learns fire dynamics in the latent space. The predicted context vector by LSTM is decoded by two-layer MLP to output the burning probability of a ROI agent. The encoder consists of three convolution blocks. Each block includes a convolutional layer, batch normalization, ReLU activation, and max pooling. The convolutional layers have (7,7), (3,3), and (3,3) kernels with stride 2. LSTM is designed with a single LSTM cell in which the number of hidden states are 64. The input and output of LSTM is processed by a fully-connected layer without an activation function to change the dimension of the vector. The decoder has two full-connected layers with ReLU and Sigmoid activation.

2.2. Forest fire model and dataset

The forest fire model has the three agents: fire, ember, and tree. The interaction of the agents with initial fire seeds gives rise to the evolution of the forest fire. A single simulation has the following key phases: (1) fire seeds start at random locations, (2) the fire evolves from the seeds, and (3) the fire is no longer spreading. The tree distribution and location of initial fire seeds are randomly selected for every simulation. We modify the pre-defined model with a set of interaction rules inspired by the Rothermel equations [10]. We also have different parameters that impact the evolution of the fire such as a forest density. The code can be found online: https://github.com/harshitk11/NetLogo-Forest-Fire-evolution.

We generate chunk-based training and test datasets. Each chunk includes successive 60 frames of the forest, and multi chunks for each simulation are generated with a 10-frame time difference. For example, the first and second chunk defines the 0-th to 59-th frames and 10-th to 69-th frames, respectively. Two density parameters 76 and 72 are considered to study the model performance in the different forest schemes. The higher density parameter indicates there are more trees in the forest. Our datasets for the density 76 include 970 chunks (70 simulations) for train and 1386 chunks (100 simulations) for test. For the density 72, 1255 chunks (70 simulations) for train and 912 chunks (50 simulations) for test.

2.3. Training procedure

We define the first 10 and next 50 frames as an observation and prediction window, respectively. In other words,
3. EXPERIMENTAL RESULT

3.1. Burning probability prediction

The two models are trained for the density 72 and 76 to predict the burning probability of the ROI (125, 125). Fig. 3 shows training and test loss for 50 epochs. The minimum test loss is noted on both the graphs, indicating the low-density forest is less predictable. Fig. 4(b) displays the predicted and ground truth burning probabilities ($p_{\text{burn}}$) at the ROI as the observation and prediction window shift. The predicted probabilities gradually grow when the fire event at the ROI occurs later in the prediction window. Fig. 5 shows receiver operating characteristics (ROC) curves at the first 4 chunks in the different densities. A positive case indicates a ROI is burning and $p_{\text{burn}}$ is higher than 0.5. The last frame in the prediction window such as the 59-th frame for the first chunk is considered in the evaluation. The model is sensitive to probability thresholds in the density 72 and mainly fails at the first chunk.

The proposed model is trained and evaluated in multi ROIs. We individually train the different model for each ROI. Fig. 6 shows the F1 score in the multi ROIs. Note, the coordinate of the left top corner in the forest is (0, 0) and the right bottom corner is (250, 250). We observe that the performance of the proposed models largely depend on the ROIs in both the densities.

3.2. Comparison with ConvLSTM

We compare the proposed model to ConvLSTM with the similar amount of trainable parameters. The ConvLSTM model generates an probability map of the forest, instead of a specific ROI. We implement an encoder with the first convolutional block in the proposed model, a single-layer ConvLSTM cell with (3,3) kernel and (64, 61, 61) hidden and cell states, and a decoder with three upconvolutional blocks with (3,3) kernels. Fig. 7 shows the F1 score at a ROI (125, 125) versus the total activation during the prediction. The scatters in the figure indicate the F1 score for the last frame in the different prediction windows. The ConvLSTM model shows the
Fig. 6. **F1 score in multi ROIs.** (a) F1 score at ROIs of north area, (b) middle area, (c) and south area in both the density 76 and 72.

![F1 score in multi ROIs.](image)

Table 1. Computational cost of the models.

| Model          | Parameters | Activations |
|----------------|------------|-------------|
| Ours - ROI     | 262.7k     | 12.3M       |
| Ours - Frame   | 295.6k     | 12.8M       |
| ConvLSTM       | 250.6k     | 103.8M      |

descent performance throughout the overall chunks while the proposed models fail to achieve the high F1 score in the early prediction windows. However, it is important to note the activation of ConvLSTM is much larger than the proposed model. This is mainly because the large hidden state in the ConvLSTM cell dominates the activation while the hidden state of the proposed model is a small 1D vector.

### 3.3. Computational cost and AUC

Table 1 and 2 summarizes the computational cost and area under the ROC curve (AUC) of the models. We also design a frame-based model, that is implemented with the same encoder and prediction module but with a modified decoder, to study the performance of the CNN-LSTM model further. The decoder is designed with upconvolutional blocks, where each block consists of (2,2) upsample, convolutional layer, batch normalization, and ReLU activation. All the convolutional layers have (3,3) kernel with stride 1 and padding 1. The models have the similar amount of the parameters, but the CNN-LSTM models have the much lower activation by learning the dynamics using the small vector. The ROI-based model shows the lowest activation because as the decoder does not reconstruct the image and also achieves higher AUC in most of the prediction windows than the frame-based model. While the ConvLSTM model shows the highest AUC, the ROI-based model has the comparable performance in density 76.

| Model          | AUC (t=59) | AUC (t=69) | AUC (t=79) | AUC (t=89) |
|----------------|------------|------------|------------|------------|
| Ours - ROI     | 0.946      | 0.979      | 0.990      | 0.996      |
| Ours - Frame   | 0.730      | 0.906      | 0.952      | 0.985      |
| ConvLSTM       | 0.970      | 0.989      | 0.998      | 1.000      |

Table 2. AUC of the models at ROI (125, 125) in density 76 (top) and density 72 (bottom).

![F1 score with activation size.](image)

**Fig. 7. F1 score with activation size.** (a) Comparison of F1 score and activation size between the proposed model and ConvLSTM in density 76 and (b) density 72.

4. **CONCLUSION**

We present a CNN-LSTM model to predict the state of a ROI agent without the reconstruction. The proposed model is evaluated in NetLogo. The ROI-based model achieves the higher AUC with less computation than the frame-based model. Also, the proposed model significantly saves computational costs such as the activation than ConvLSTM by separating the spatial and temporal learning modules.
5. REFERENCES

[1] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al., “Grandmaster level in starcraft ii using multi-agent reinforcement learning,” Nature, vol. 575, no. 7782, pp. 350–354, 2019.

[2] Beomseok Kang, Harshit Kumar, Saurabh Dash, and Saibal Mukhopadhyay, “Unsupervised hebbian learning on point sets in starcraft ii,” in 2022 International Joint Conference on Neural Networks (IJCNN). IEEE, 2022, pp. 1–8.

[3] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li, “Multi-agent deep reinforcement learning for large-scale traffic signal control,” IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 3, pp. 1086–1095, 2019.

[4] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone, “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data,” in European Conference on Computer Vision. Springer, 2020, pp. 683–700.

[5] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al., “Interaction networks for learning about objects, relations and physics,” Advances in neural information processing systems, vol. 29, 2016.

[6] Priyabrata Saha, Arslan Ali, Burhan A Mudassar, Yun Long, and Saibal Mukhopadhyay, “Magnet: Discovering multi-agent interaction dynamics using neural network,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 8158–8164.

[7] William Gilpin, “Cellular automata as convolutional neural networks,” Physical Review E, vol. 100, no. 3, pp. 032402, 2019.

[8] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin, “Growing neural cellular automata,” Distill, vol. 5, no. 2, pp. e23, 2020.

[9] Uri Wilensky, “Netlogo. evanston, il: Center for connected learning and computer-based modeling, northwestern university,” 1999.

[10] Richard C Rothermel, A mathematical model for predicting fire spread in wildland fuels, vol. 115, Intermountain Forest & Range Experiment Station, Forest Service, US Department of Agriculture, 1972.