The Combination of GCN and CNN in Spatial-Temporal Domain

Xu Guangning, Ph.D. Student

Computer Science & Technology

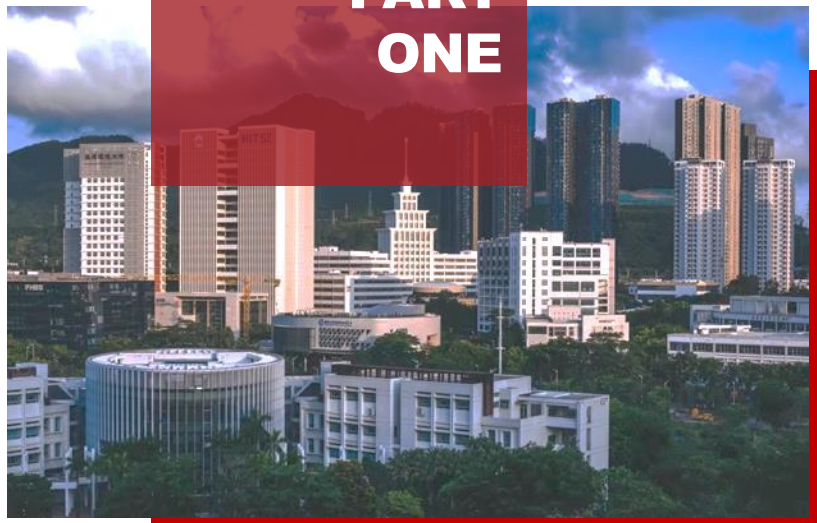
Harbin Institute of Technology, Shenzhen

目录

CONTENTS

01

PART ONE

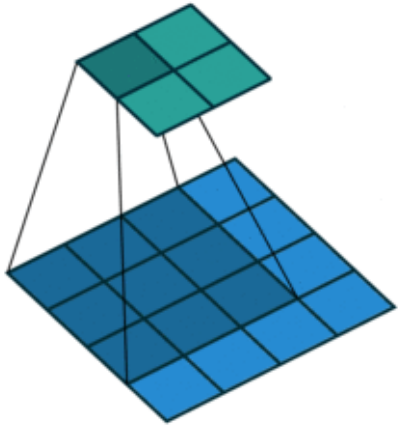


Background Knowledge

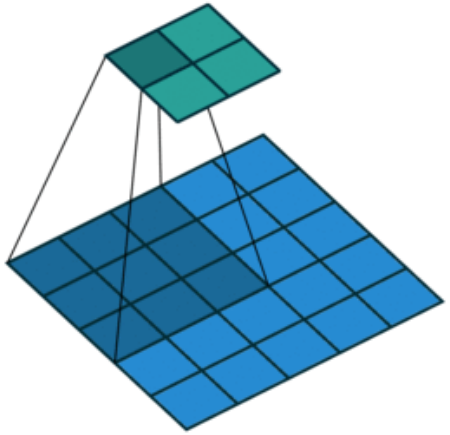
The principle of the Convolution Neural Network (CNN)

Euler spatial domain model

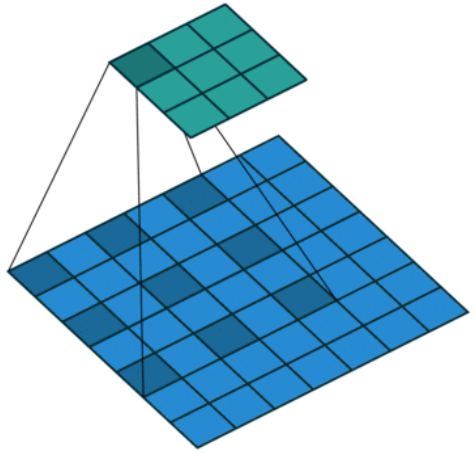
Technically, the convolution in deep learning is actually a "cross-correlation" operation, which is different from the convolution in signal processing. The main parameters of the convolution operation are **kernel**, **stride**, **padding** and **output channel**. Among them, the kernel is learnable and the receptive field of the convolution operation; stride is the number of pixels moved per step when the kernel traverses the input; padding adds a circle of 0s around the input feature map; output channel refers to the number of kernels.



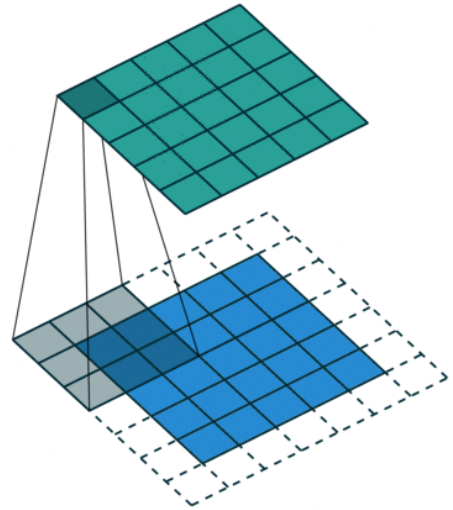
Non-padding
Stride = 1



Non-padding
Stride = 2



Non-padding
Stride = 1 dilation = 2



padding
Stride = 1

The gif images are from [1][2]

$$Y(i,j) = \sum_{p=1}^{k} \sum_{q=1}^{k} W(p,q) X(i-p, j-q)$$

The Matrix Form of Convolution in PyTorch CPU implementation

Patch: It is the area the input and traversing kernel perform multiplication, which mean a step stride can generate a patch.

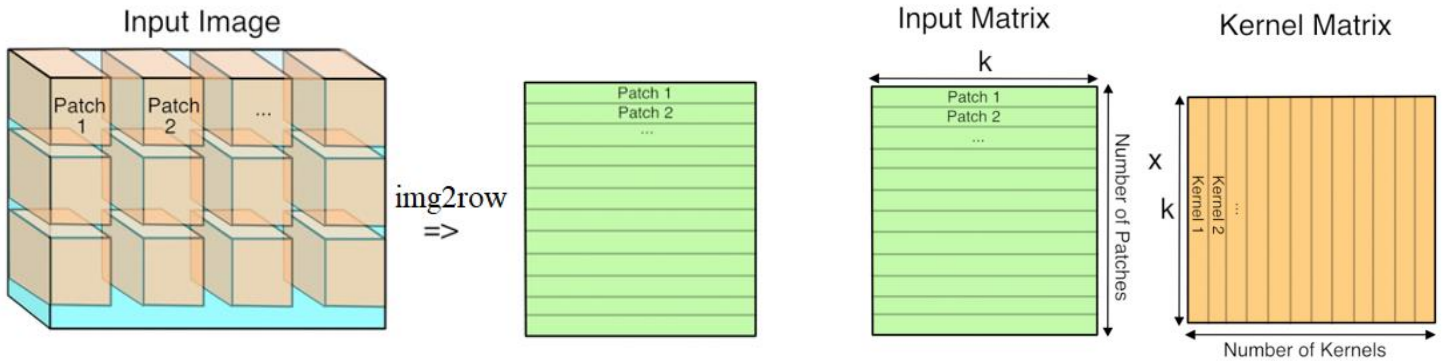


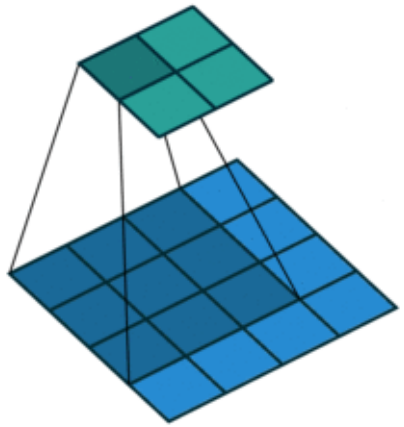
Image is from [14]

The Generation of the matrix form in CPU PyTorch source code

First, All the patches will form the corresponding row of the matrix, which is so call image to rows.
Second, the flattened kernels are also formed as a matrix.
Finally, the convolution is formed as a matrix multiplication between the patching image and unfolding kernels.

Here is the source code in CPU PyTorch for this implementation. For more detailed implementation information please kindly refer to the link.

https://github.com/pytorch/pytorch/blob/e9902358df14dc4809e4f50b12088a5200a1862d/aten/src/ATen/native/ConvolutionMM2d.cpp



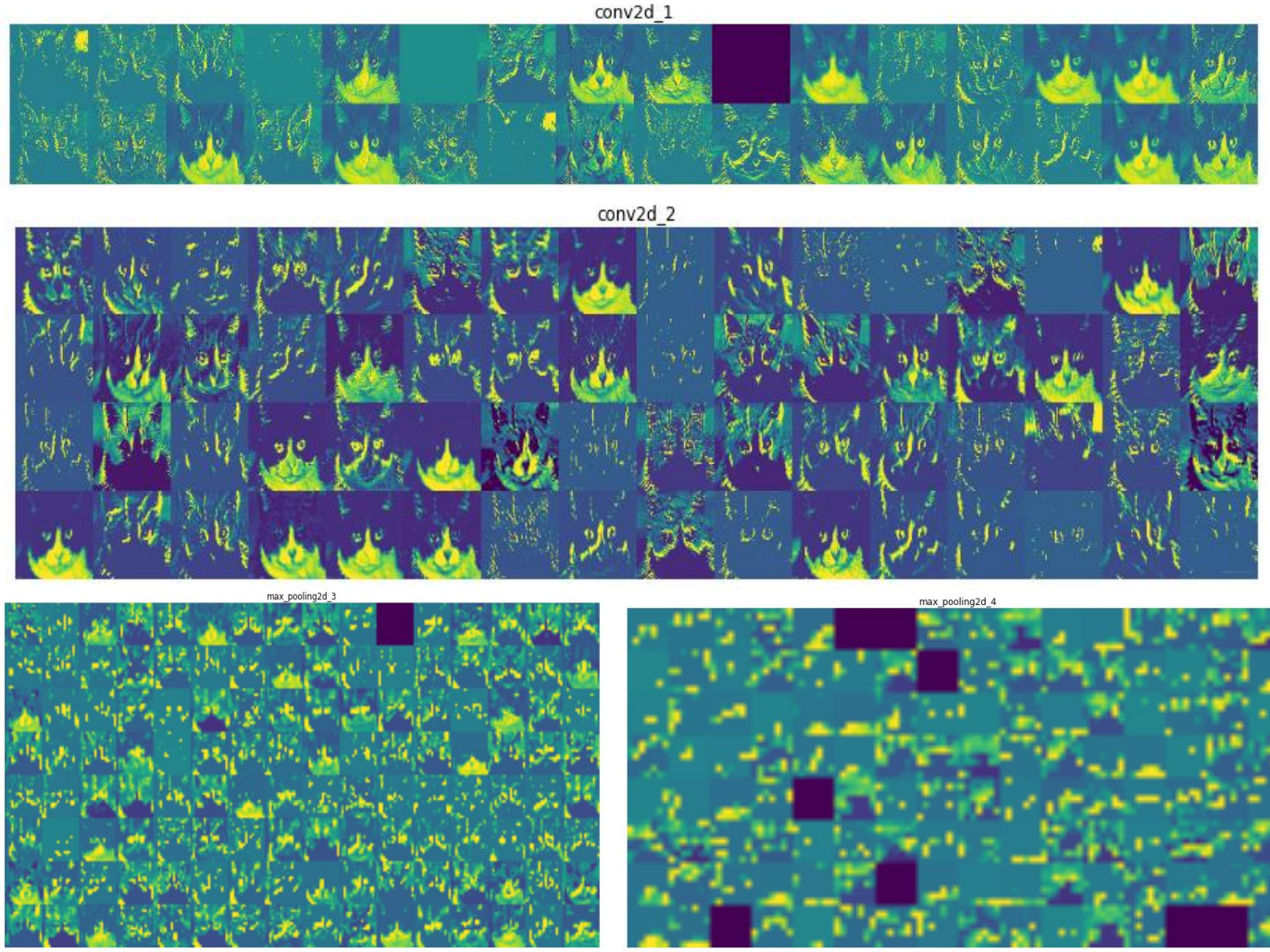
The feature maps of the CNN

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dropout_1 (Dropout)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 1)	513

Deeper layer can capture global features by stacking convolution layers, which might help the classification.

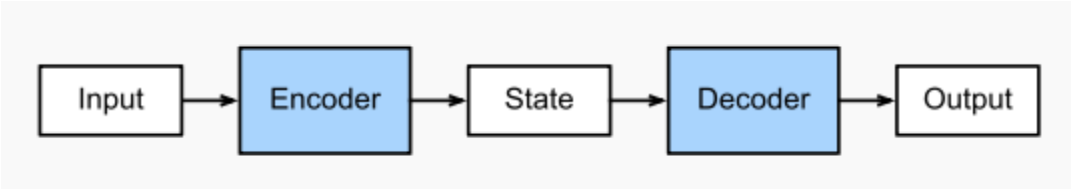
To visualize the features maps, a cat image with shape (256,256,3) is fed into the CNN model.

Code and results are from [3]



The Encoder-Decoder Architecture in CNN

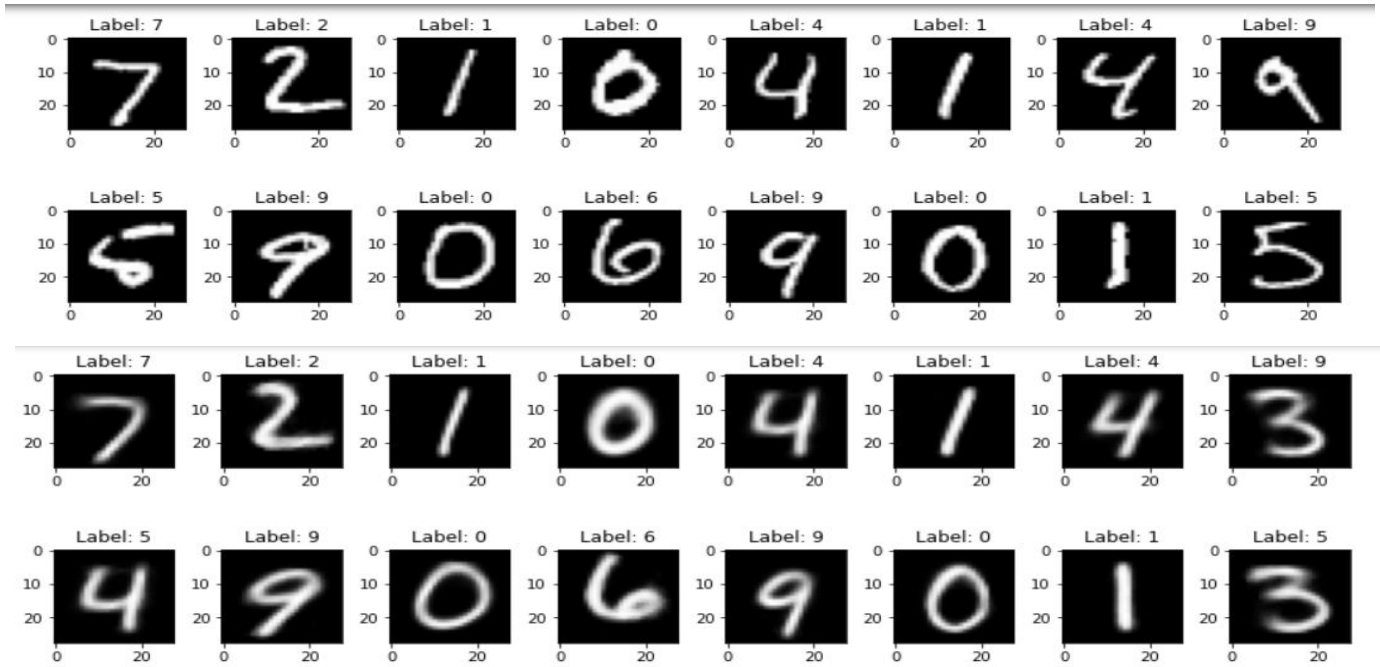
Here is a conventional architecture of encoder-decoder, which can be used for supervised learning and unsupervised learning. If the output is the same as the input, we call it as Autoencoder:



Code and results are from [4]

Here is one computer vision domain example for autoencoder

```
import torch.nn as nn
class AutoEncoder(nn.Module):
  def __init__(self):
    super(AutoEncoder, self).__init__()
    self.encoderConv = nn.Sequential(
      nn.Conv2d(1, 16, 3, stride=3, padding=1), # [16, 10, 10]
      nn.ReLU(True),
      nn.MaxPool2d(2), # [16, 5, 5]
      nn.Conv2d(16, 8, 3, stride=1, padding=1), # [8, 5, 5]
      nn.ReLU(True),
      nn.MaxPool2d(2), # [8, 2, 2]
    )
    self.decoderConv = nn.Sequential(
      nn.ConvTranspose2d(8, 16, 3, stride=2),  # [16, 5, 5]
      nn.ReLU(True),
      nn.ConvTranspose2d(16, 8, 5, stride=3, padding=1),  # [8, 15,
        15]
      nn.ReLU(True),
      nn.ConvTranspose2d(8, 1, 2, stride=2, padding=1),  # [1, 28, 28]
      nn.ReLU(True)
    )
  def forward(self, x):
    x1 = self.encoderConv(x)
    x = self.decoderConv(x1)
    return x1, x
```



Input

output

As we can see, the reconstructed outcomes look the same as the input in most cases, which means that the latent state in the lower dimension space can somehow represent the corresponding image.

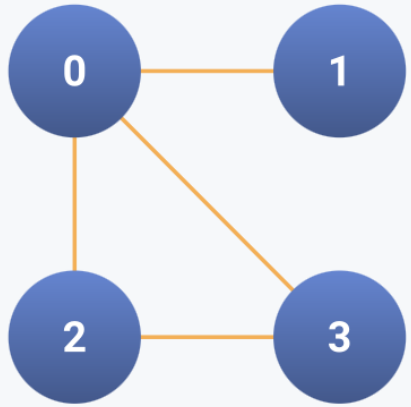
Graph Convolution Neural Network (GCN)

Non-Euler spatial domain model

The GCN is similar to CNN since it is obtained from the CNN. For a convolution in deep learning, two main steps should be considered: i) how to identify the neighbors and ii) how to aggregate the neighbors.

For the first point, how the GCN find the neighbors? We use adjacency matrix A to identify neighbors.

	0	1	2	3
0	0	1	1	1
1	1	0	0	0
2	1	0	0	1
3	1	0	1	0



For the second point, how the GCN aggregate the neighbors? The GCN use sum method to aggregate the neighbors which is same as CNN.

$$H^{l+1} = \sigma(AH^lW^l + B)$$

where A is the adjacent matrix denoting the neighbors, $H^{(l)}$ is the nodes' attribute and $W^{(l)}$ is the kernel and B is the bias.

GCN V.S CNN

To empress understanding, we compare GCN and CNN: the CNN uses the around pixels as neighbors while the neighbors in GCN should be identified by an adjacency matrix. For the aggregation method, both GCN and CNN sum up neighbors together as the final result.

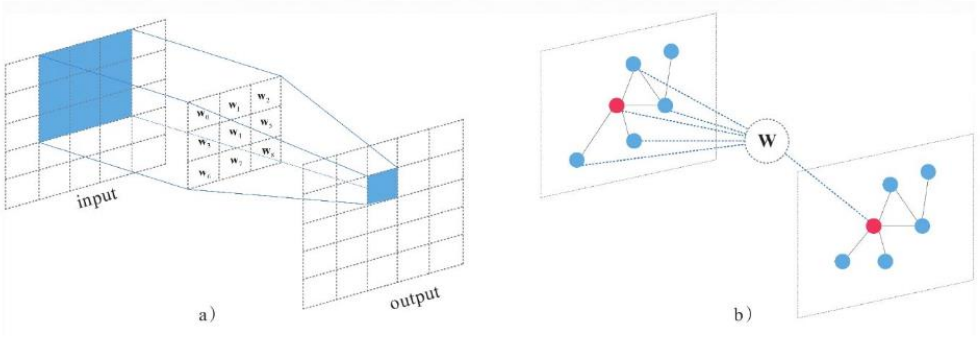


Image is from [5]

Like the CNN, the stacking convolution layer can enlarge the receptive field of the model, the GCN can be stacked to capture multi-hop neighbor features

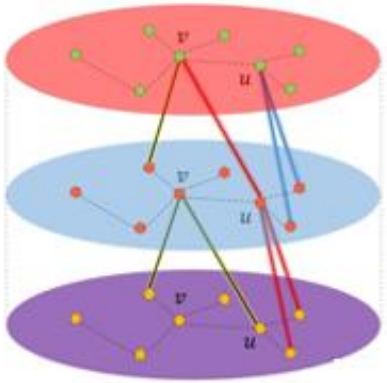


Image is from [6]

Besides, the Encoder-Decoder Architecture can be also applied into the graph structure data.

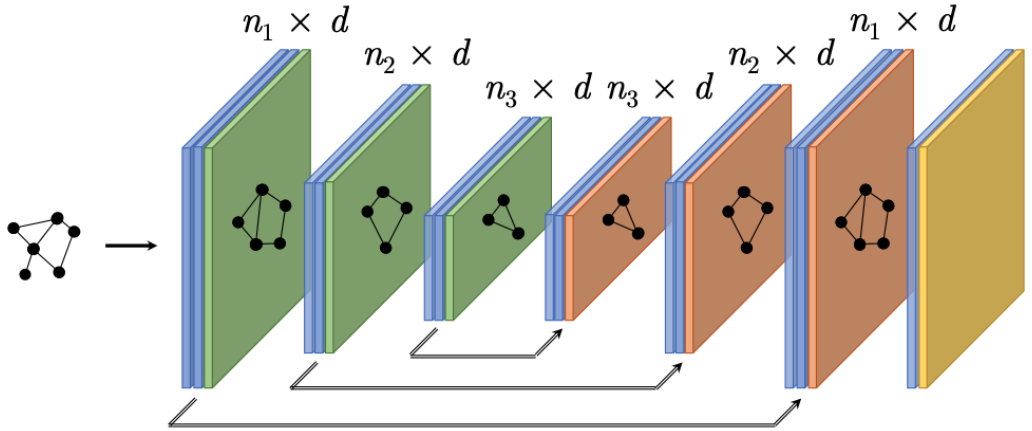


Image is from [7]

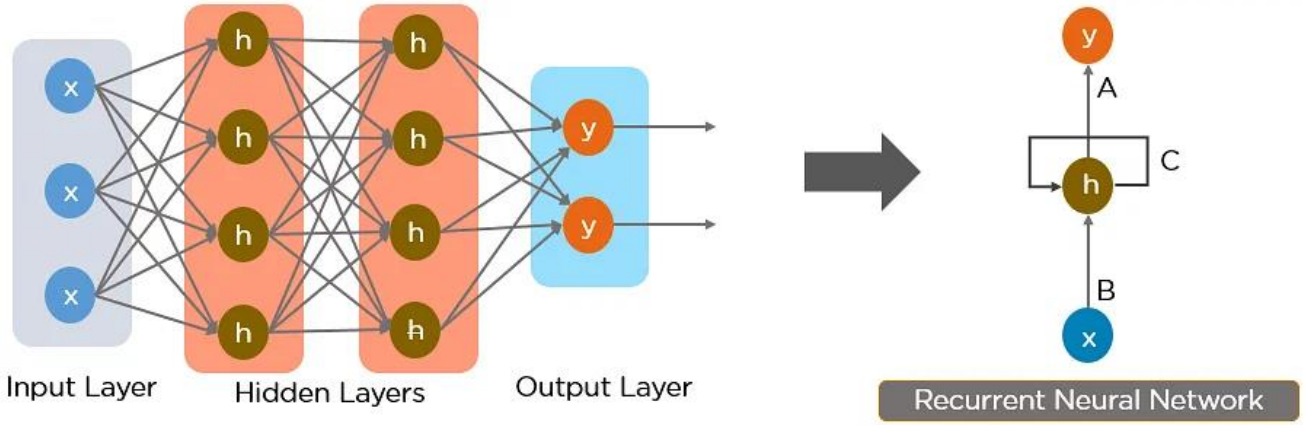
The principle of the Recurrence Neural Network (RNN)

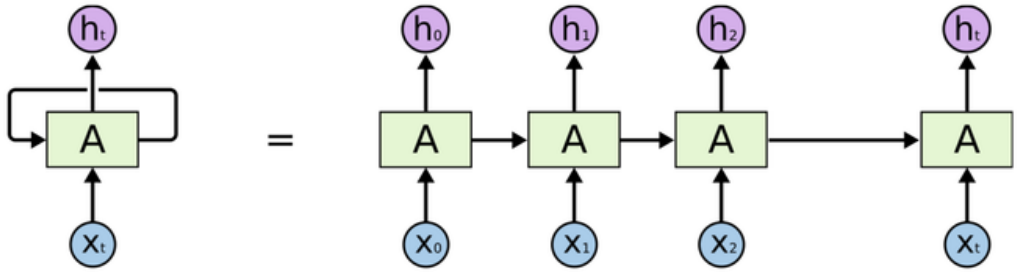
Temporal domain model

Recurrent Neural Networks (RNNs) are a class of neural network that are helpful in modeling sequence data. They specialize in processing sequences and are often used in NLP because of their effectiveness in handling text. The most common used RNN models are Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU).

A type of fully connective neural network recurrent in the temporal domain

- Parameters are shared in temporal domain
- The current output is mainly relied on the previous output.





Images are from [8][9]

What about Attention V.S RNN?

- Attention needs to calculate the correlation at each time step, the time complexity of the simple attention is $O(n^2)$.
- The time complexity of RNN is O(n), where n is the length of the sequence.

The introduction of the LSTM

Long Short-Term Memory (LSTM) is a type of RNN that is specifically designed to handle sequential data, such as time series, speech, and text. LSTM networks are capable of learning order dependence in sequence prediction problems.

Image is from [10]

Input Gate $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

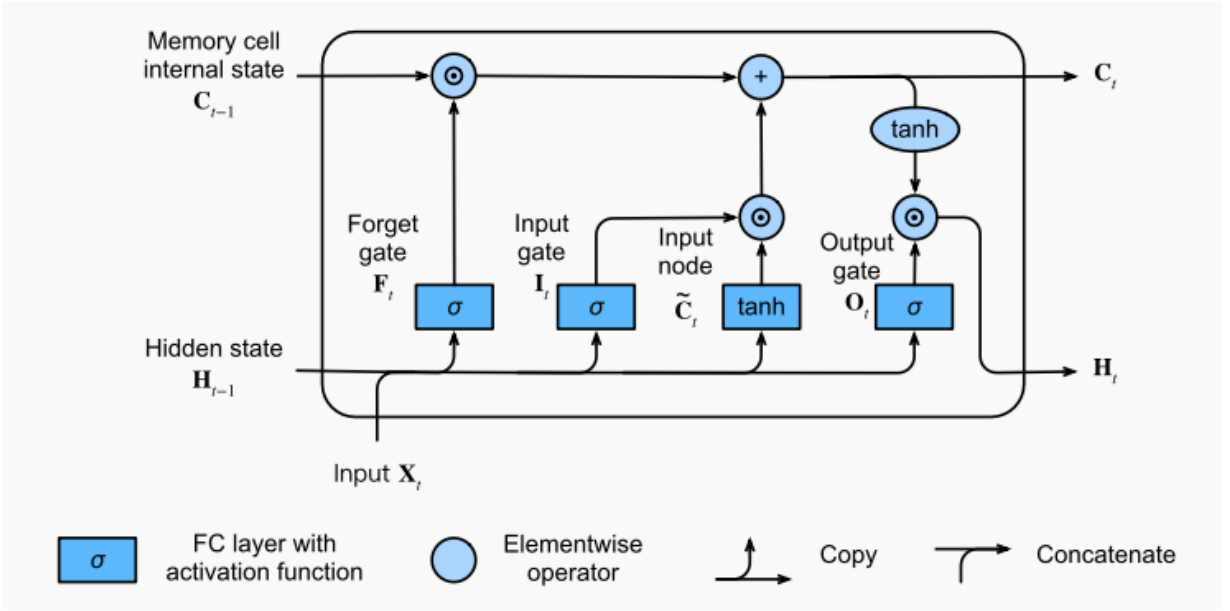
Forget Gate $$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Memory Cell $$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate $$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

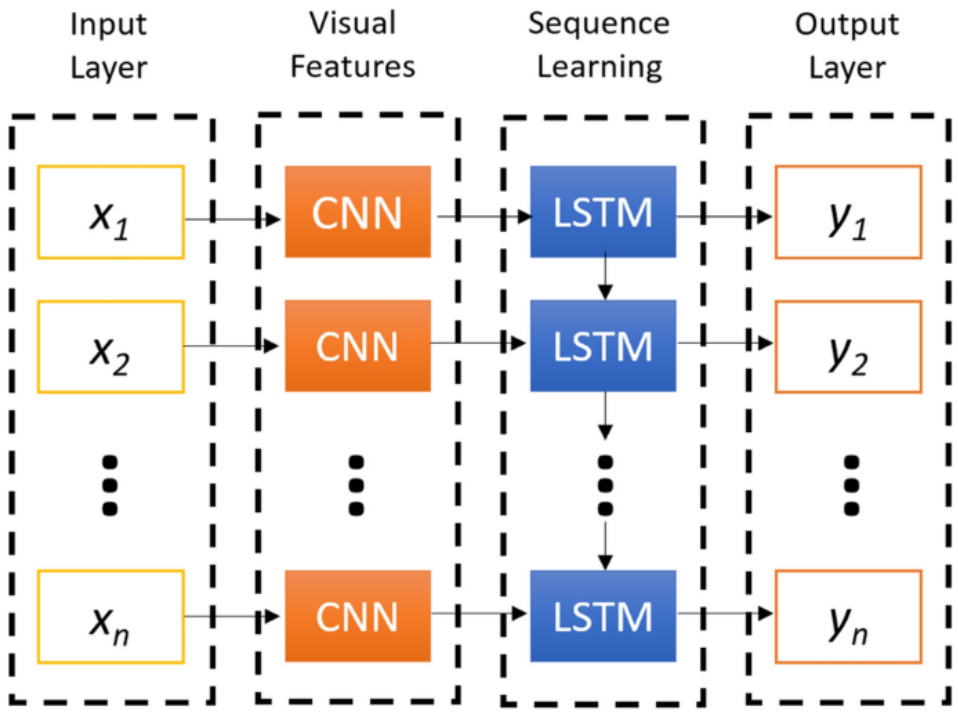
Output $$h_t = o_t * \tanh(C_t)$$



Usually, σ is chosen as a sigmoid function S(x). $$S(x) = \frac{1}{1 + e^{-x}}$$

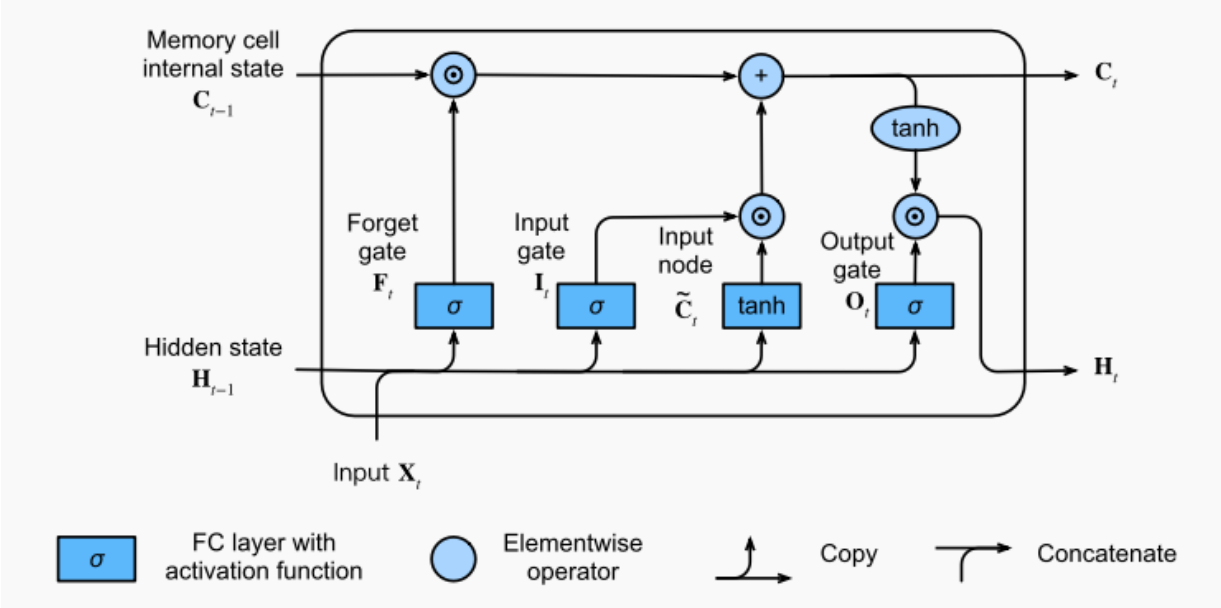
How we handle the video data which are time-series images? **Spatial-temporal domain model**

An intuitive way is that we first apply the CNN to the images. After that, we flatten the CNN result as a vector and then feed into the RNN model. However, this manner can not handle spatial-temporal feature at the same time since it handles spatial and temporal features separately.



The intuitive way

Images are from [11][10]



Considering the LSTM, can we integrate the convolution into the LSTM? Rather than just conducting CNN and LSTM separately

The ConvLSTM for video prediction

ConvLSTM is a type of recurrent neural network for spatial-temporal prediction that has convolutional structures in both the input-to-state and state-to-state transitions. It determines the future state of a certain cell by the inputs and past states of its local neighbors.

Note that:

- The input-to-state transition means the input X_t transits to H_t
- The state-to-state transition means the H_{t-1}, C_{t-1} transits to H_t , C_t

Images is from [12]

the ConvLSTM is similar to the LSTM while the matrix multiplication is replaced as convolution

Input Gate $i_t = \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i)$

Forget Gate $f_t = \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f)$

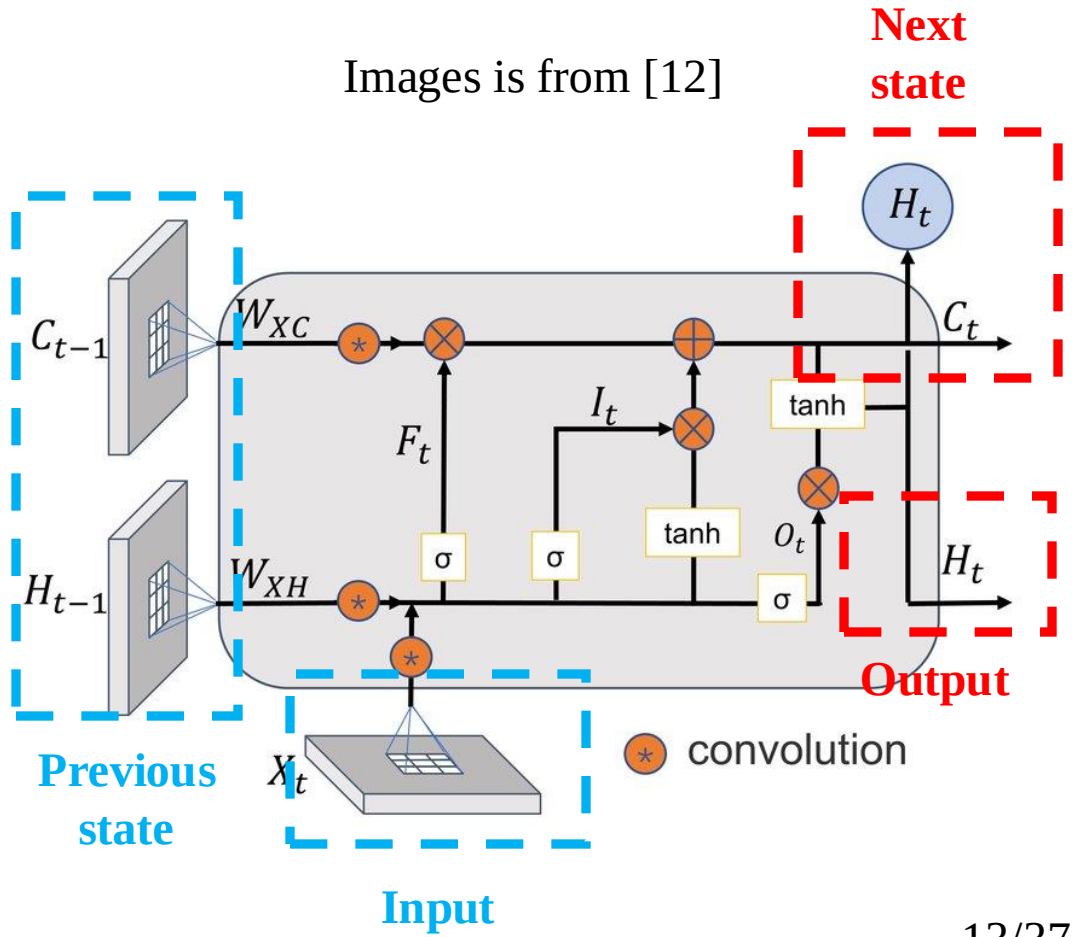
Memory Cell $\tilde{\mathcal{C}}_t = \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c)$

$\mathcal{C}_t = f_t \circ \mathcal{C}_{t-1} + i_t \circ \tilde{\mathcal{C}}_t$

Output Gate $o_t = \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o)$

Output $\mathcal{H}_t = o_t \circ \tanh(\mathcal{C}_t)$

where * is the convolution operation for spatial feature. ° is the element-wise multiplication. As the spatial-temporal feature can be capture in one operator, the ConvLSTM can handle video prediction task.



02

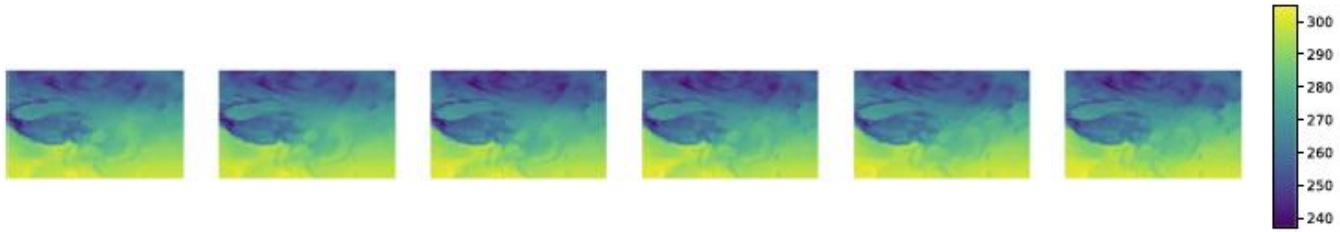
PART
Two

The Motivation of the Combination

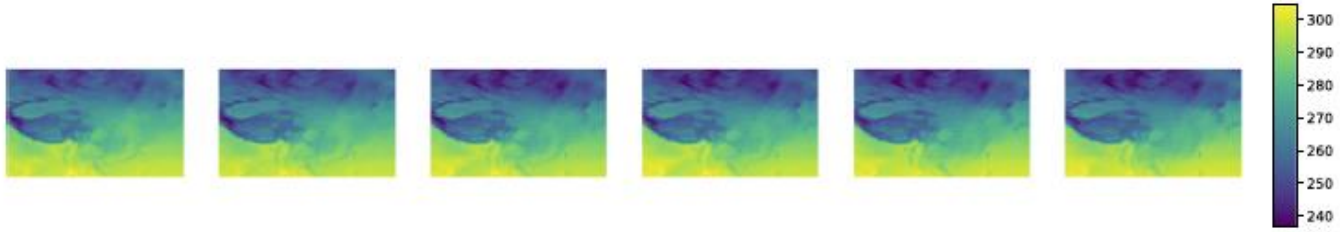
Introduction of the spatial-temporal task for Near-surface Temperature

The Near-surface Temperature prediction (NTP) is one of the application task of weather forecast in spatial-temporal domain. The model can be formulated as follows:

$$\tilde{\mathcal{X}}_{t+1}, \ldots \tilde{\mathcal{X}}_{t+\beta} = \arg\max p(\mathcal{X}_{t+1}, \ldots \mathcal{X}_{t+\beta} | \mathcal{X}_t, \ldots \mathcal{X}_{t-\alpha+1}),$$



(a) An input near-surface temperature sequence.



(b) An output near-surface temperature sequence.

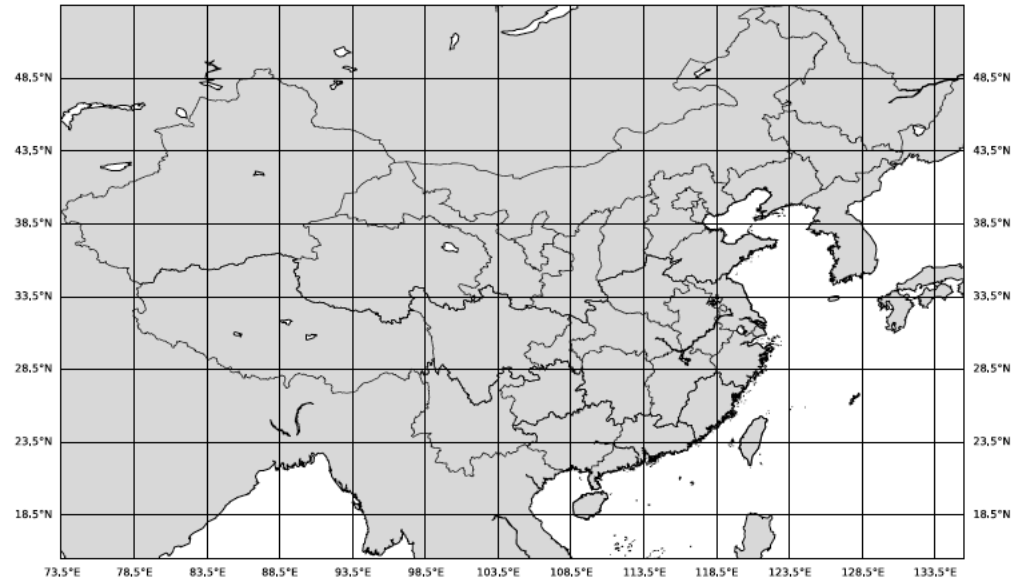


Fig. 6. Visualization of the selected zone focused in this study.

PS: The unit of the temperature is Kelvin. The formula of the transform is K=℃+273.15

All images which do not mention are from [13]

Observation in real-world application

Pearson Coefficient

$$\rho(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

Difference distances identify by different colors

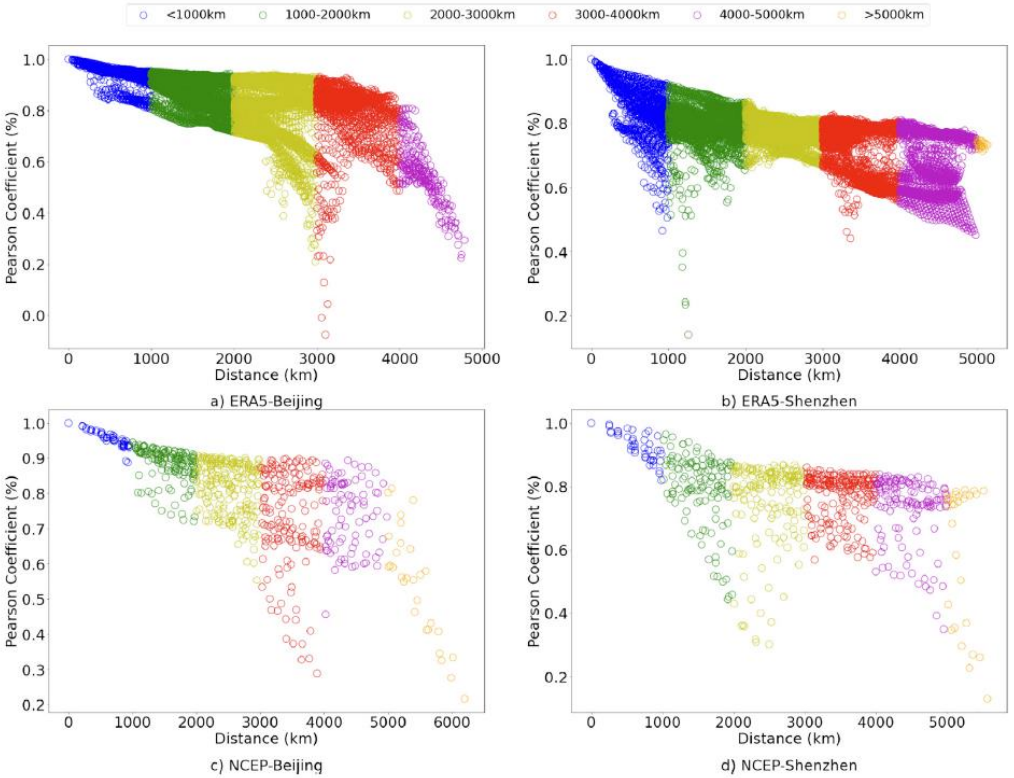


Fig. 1. An illustrative example of long- and short-range spatial correlations for near-surface temperature.

we can make three key observations:

(i) the grid pairs with a close distance have a high correlation, which reflects the short-range spatial correlations; suitable for CNN

(ii) the grid pairs with a long-distance may still have a strong correlation, which indicates the existence of long-range spatial correlations; suitable for GCN

(iii) the grid pairs with the same distance may have varying correlations, which shows that the spatial correlations are quite complex. suitable for spatial attention, a technique to reweight pixels based on their importance.

The observations imply that both long- and short-range spatial correlations would be beneficial for near-surface temperature prediction and grids with the same distances may have different contributions.

03

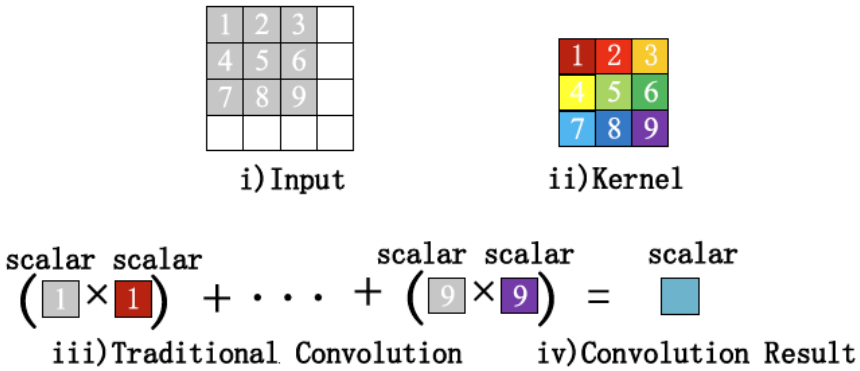
PART
Three

Methodology

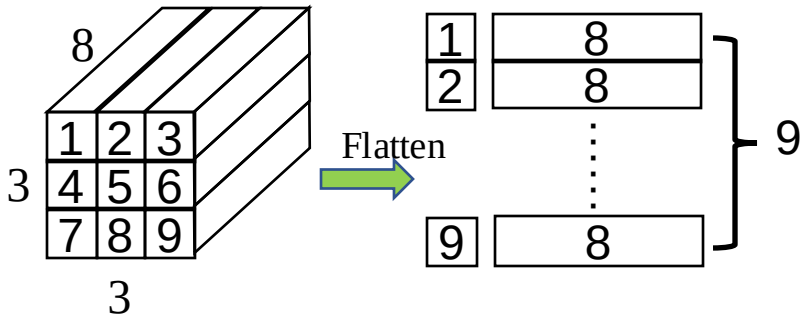
The Main Idea of the **L**ong- and **S**hort range **Conv**olution LS-Conv

The LS-Conv is design to capture long- and short- range spatial correlation in a convolution operator by unifying GCN and CNN.

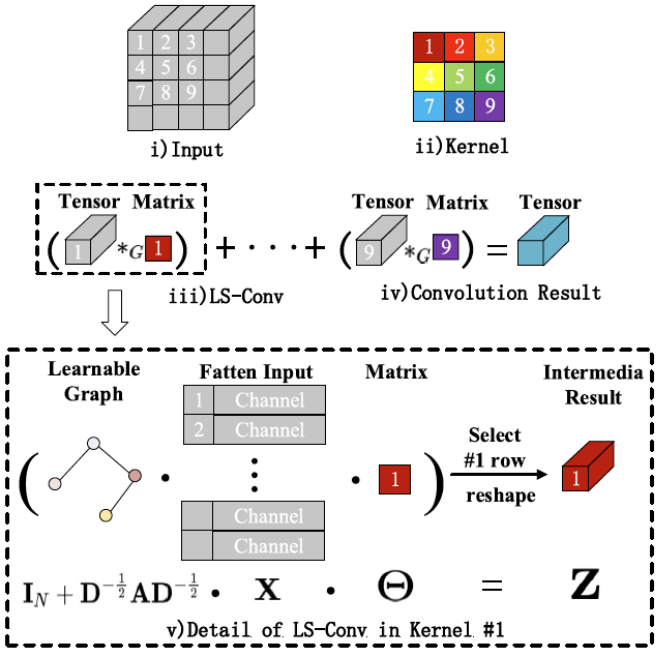
Here is the conventional convolution procedure



(a) The traditional convolution manner.



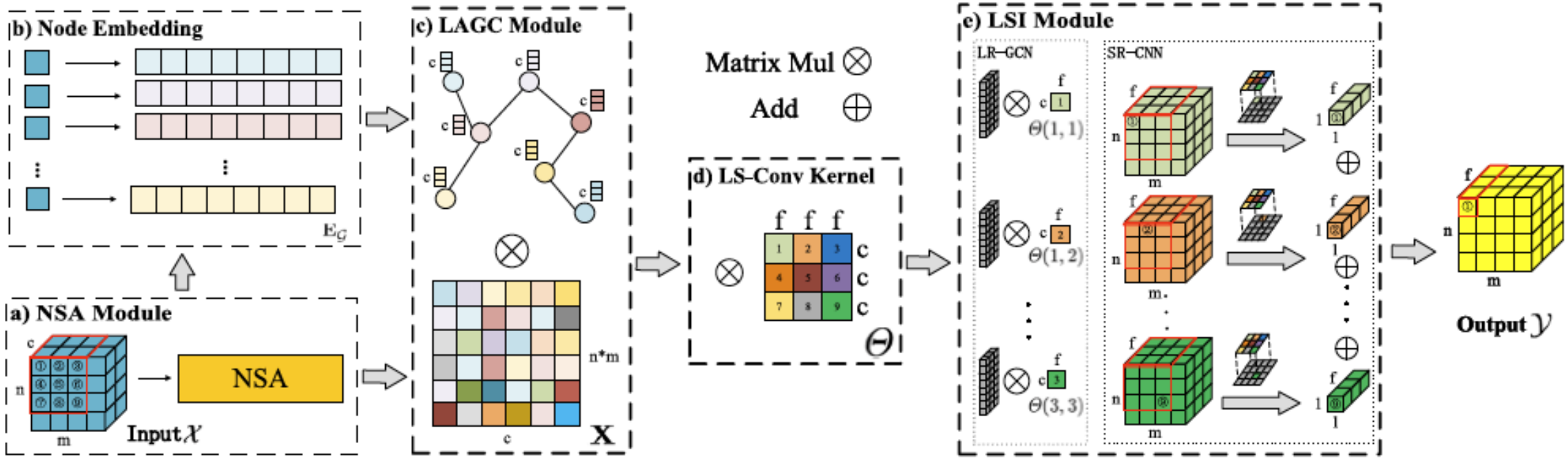
the LS-Conv replaces scalar multiplication with graph convolution



(b) The LS-Conv manner.

We choose a data-driven method to generate the learnable adjacency matrix. After that, we flatten the input image into a matrix for long-range spatial correlation capturing. After the graph convolution, we fold the result back to an image and select the corresponding row to integrate the short-range spatial correlation.

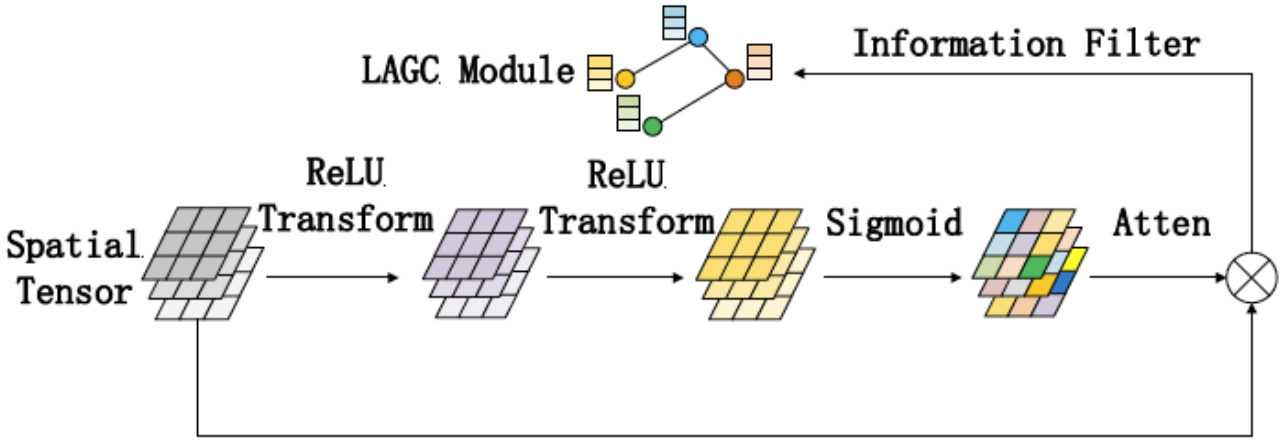
The Overview of the LS-Conv



Here is the whole overview of the LS-Conv. Given an input X, The Node-based Spatial Attention is first applied to evaluate the importance of each grid of the X aiming to distinguish important nodes for long-range spatial correlations construction. Meanwhile, the learnable node embedding is initiated, which denotes the channel information of the corresponding grid. After the NSA, the LAGC module learns an adaptive adjacent matrix to model the long-range spatial correlations by using graph convolution kernel from LS-Conv kernel. Finally, in the LSI, these GCN outputs are aggregated in a CNN manner, where each GCN result is similar to the element-wise product result between kernel and input in conventional CNN. Consequently, these results will be summed together and placed at the corresponding position of the output Y.

The Mechanism of Node-based Spatial Attention (NSA)

Spatial attention is a form of attention mechanism that focuses on specific regions of an image. It is used in computer vision tasks to improve the performance of CNNs by allowing them to focus on important regions of an image. As the input image can be viewed as nodes of the adaptive adjacency matrix, so we also call this module Node-based spatial attention (NSA).



$$\mathcal{X}_1 = \mathrm{ReLU}\Big(\mathrm{GroupNorm}\big(\mathrm{Conv1@1}(\mathcal{X})\big)\Big),$$

$$\mathcal{X}_2 = \mathrm{ReLU}\Big(\mathrm{GroupNorm}\big(\mathrm{Conv1@1}(\mathcal{X}_1)\big)\Big),$$

$$\mathrm{Atten} = \mathrm{Sigmoid}(\mathcal{X}_2),$$

$$\mathrm{NSA}(\mathcal{X}) = \mathcal{X} \cdot \mathrm{Atten},$$

PS:
Conv1@1 means the Conv2d operation with 1*1 kernel size

Long-range adaptive graph constructor (LAGC)

we review the convention GCN, which is shown as follows:

$$(\mathbf{X} *_G \Theta) = \left(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{X}\Theta + \mathbf{b},$$

where D is the degree matrix, $D[i,i] = \sum_{j=1}^{N} A[i,j]$ and $D[i,j] = 0$.The term $D^{-1/2}AD^{-1/2}$ is called adjacency matrix normalization. I_N is the identity matrix also called the self-loop factor which means that the GCN consider both neighbors and itself.

Many graph-based task can easyly find neighbors. However, in meteorological task, it is hard for us to identify the neighbors. So we propose a data-driven method to learn the neighbors:

$$\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathrm{SoftMax}\left(\mathrm{ReLU}\left(\mathbf{E}_G\mathbf{E}_G^T\right)\right),$$

$$\mathbf{Z} = \left(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{X}\Theta + \mathbf{b},$$

E_g is the learnable node embedding corresponding to the grid in the input X. The multiplication of E_g and E_g^T can be viewed as measuring the similarity of other node embedding vector. The ReLU activation function aims to guarantee the positive value in the learnable adjacent matrix and the SoftMax activation function is designed for the normalization. In this way, we design a data-driven adjacency matrix generation method to capture the meteorological long-range spatial correlation.

Long- and short-range integrator (LSI)

We first conduct the graph constraint for the input X, which has shape (c, n×m). Then, a graph convolution is performed with the help of the LS-Conv kernel aiming to capture long-range spatial correlation. After the graph convolution, the result is reshaped into a new shape (n, m, f). Finally, we select the corresponding result based on the location of the LS-Conv kernel, which is same as CNN.

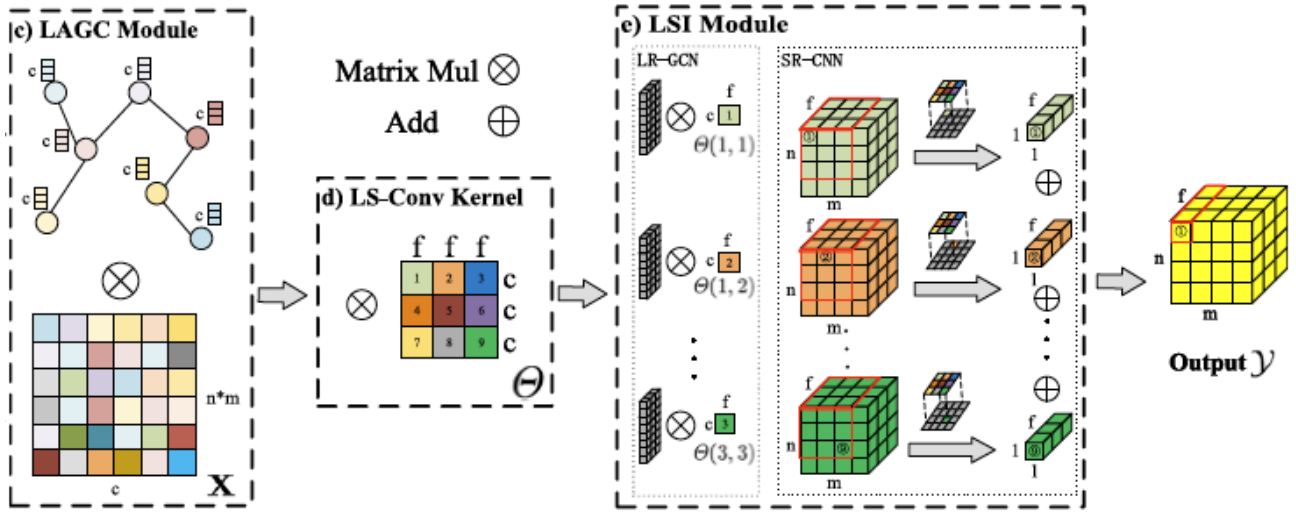
graph constraint

$$\mathbf{L} = \left(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{X},$$

CNN manner in aggregation

$$(\mathbf{X} \star \boldsymbol{\Theta})(i, j, :) = \sum_{p=1}^{k}\sum_{q=1}^{k}\left[\mathbf{L}\boldsymbol{\Theta}_{p,q,:,:}\right]_{(n,m,f)}(i-p, j-q, :),$$

graph convolution



- Light green outcomes is generated by the light green LS-Conv kernel, so we pick up the first row in the first column as the final result.
- For the orange outcomes, which is generated by the second LS-Conv kernel. So we pick up the first row in the second column.
- Finally, we conduct sum aggregation in a CNN manner to capture short-range spatial correlation.

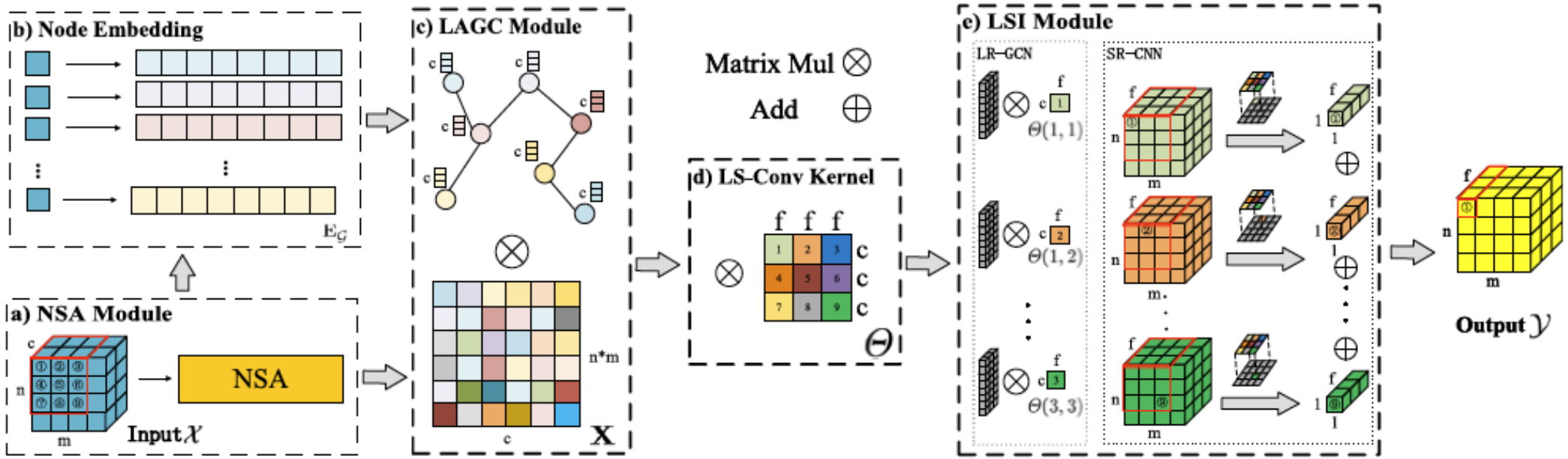
Long- and short-range Convolution (LS-Conv)

$$\mathbf{X} = [\mathrm{NSA}(\mathcal{X})]_{(nm,c)}, \quad \text{Identify the important nodes for long-range spatial correlations}$$

$$\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathrm{SoftMax}\left(\mathrm{ReLU}\left(\mathbf{E}_{\mathcal{G}}\mathbf{E}_{\mathcal{G}}^{T}\right)\right), \quad \text{Adjacency matrix is constructed by a data-driven method}$$

$$\mathbf{L} = \left(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{X}, \quad \text{Graph constrain is applied to the input}$$

$$(\mathbf{X} \star \Theta)(i,j,:) = \sum_{p=1}^{k}\sum_{q=1}^{k} [\mathbf{L}\Theta_{p,q,:,:}]_{(n,m,f)}(i-p, j-q, :), \quad \text{Aggregate the GCN result in a CNN manner}$$



ConvLSTM equipped with LS-Conv (LS-NTP)

Upon the LS-Conv, we embed this new operator into the ConvLSTM (we replace the convolution operator with the LS-Conv operator.)

Input Gate
$$i_t = \sigma\left(W_{xi} \star \mathcal{X}_t + W_{hi} \star \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i\right),$$

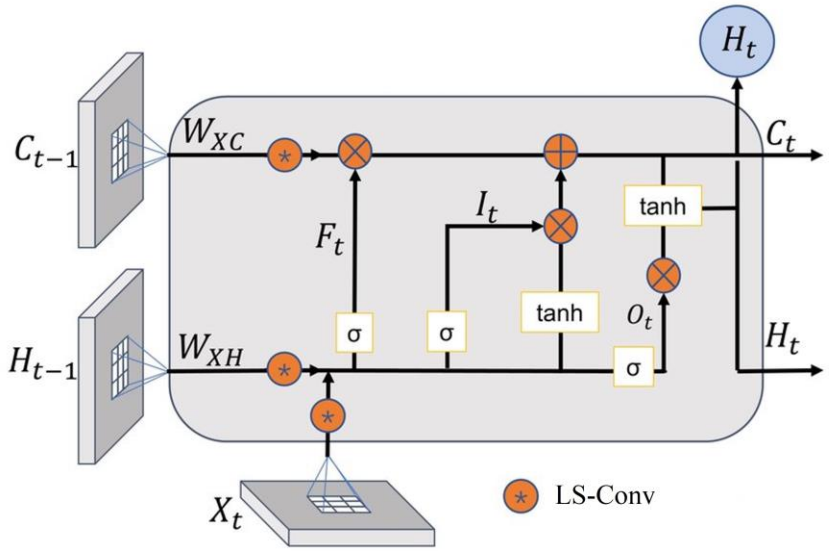
Forget Gate
$$f_t = \sigma\left(W_{xf} \star \mathcal{X}_t + W_{hf} \star \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f\right),$$

Memory Cell
$$\tilde{\mathcal{C}}_t = \tanh\left(W_{xc} \star \mathcal{X}_t + W_{hc} \star \mathcal{H}_{t-1} + b_c\right),$$
$$\mathcal{C}_t = f_t \circ \mathcal{C}_{t-1} + i_t \circ \tilde{\mathcal{C}}_t,$$

Output Gate
$$o_t = \sigma\left(W_{xo} \star \mathcal{X}_t + W_{ho} \star \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o\right),$$

Output
$$\mathcal{H}_t = o_t \circ \tanh\left(\mathcal{C}_t\right),$$

where $X_t \in R^{n \times m \times c}$ is the input tensor at time-step t, $\star$ is the LS-Conv operator, and the $\circ$ is the element-wise multiplication.



The main difference from ConvLSTM is we replace the convolution operator with the LS-Conv operator.

The framework of the final prediction model

After defining the spatial-temporal model, we use encoder-decoder architecture for the final prediction.

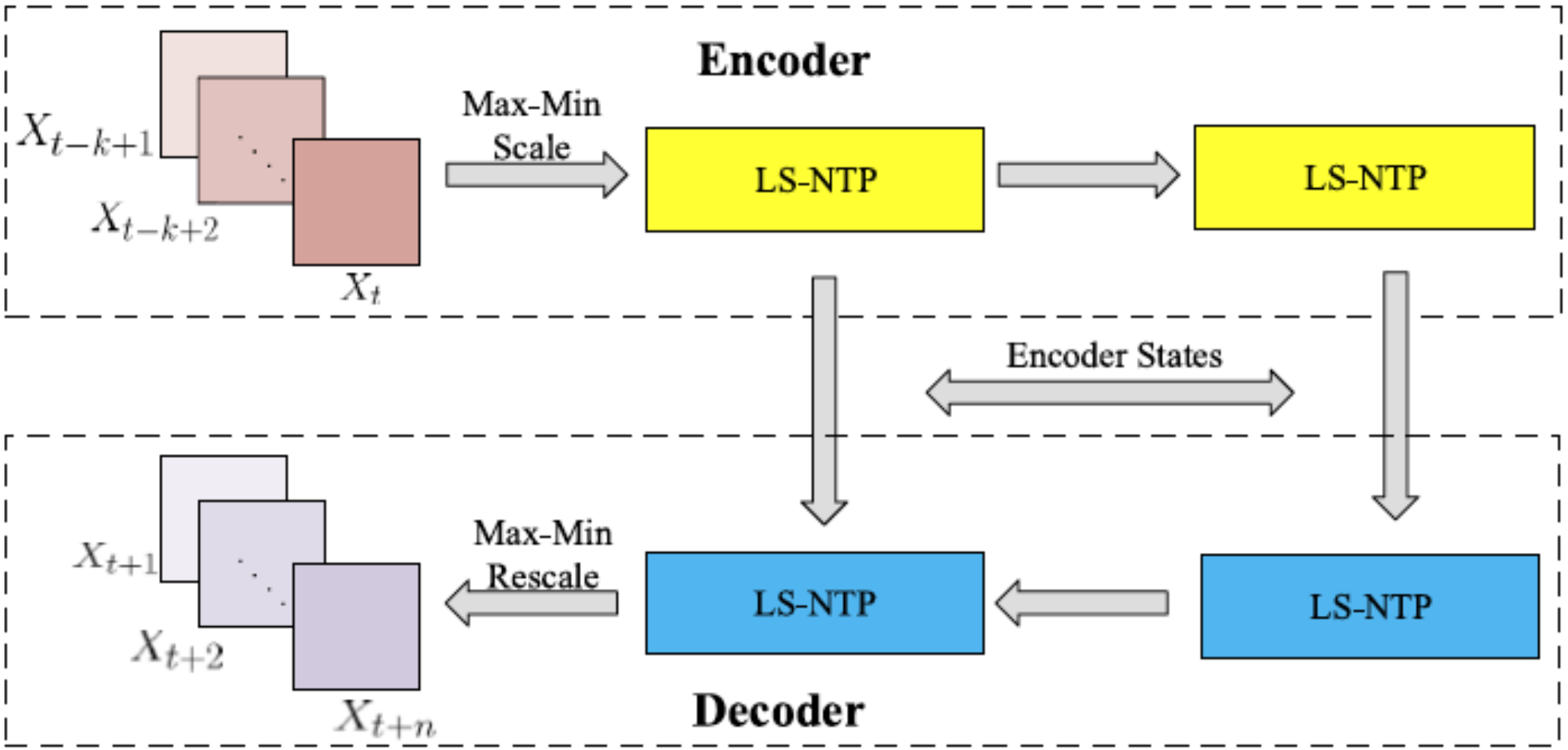


Fig. 5. The framework of encoder–decoder upon LS-NTP.

04

PART
Four

Experiments

Main Comparison

Table 3
The overall comparisons results for the near-surface temperature prediction.

Method	ERA5 (per 1 h)					NCEP (per 6 h)				
	#Params	MAE↓	RMSE↓	SSIM↑	PSNR↑	#Params	MAE↓	RMSE↓	SSIM↑	PSNR↑
NCEP-GFS	–	–	–	–	–	–	3.998		0.696	33.421
ConvLSTM	2.88M	1.054	2.247	0.854	45.220	2.88M	1.819	2.413	0.909	40.313
ConvGRU	2.16M	0.938	1.263	0.876	46.237	0.82M	1.832	2.441	0.905	40.242
TrajGRU	1.59M	0.945	1.238	0.865	46.197	0.97M	1.751	2.333	0.913	40.578
PredRNN	7.12M	1.303	1.663	0.878	44.088	11.53M	1.870	2.525	0.911	40.098
PredRNN++	3.25M	1.406	1.676	0.859	43.523	3.19M	2.153	2.483	0.915	39.994
MIM	12.32M	1.339	1.662	**0.884**	43.957	30.05M	2.183	2.441	0.915	40.264
PhyDNet	23.01M	1.375	1.760	0.849	43.605	7.41M	2.042	2.667	0.898	39.508
CrevNet	20.90M	1.574	1.898	0.867	42.665	20.90M	2.049	2.351	**0.916**	39.486
AGCRN	15.31M	0.972	1.288	0.882	46.006	19.21M	1.970	2.609	0.901	39.658
SA-ConvLSTM	5.83M	1.270	1.628	0.881	44.296	6.48 M	1.974	2.581	0.912	39.849
D-ConvLSTM	1.04M	0.999	1.311	0.857	45.753	1.04M	1.962	2.589	0.893	39.671
LS-NTP	5.99M	**0.915**	**1.199**	0.883	**46.507**	19.48M	**1.741**	**2.318**	**0.916**	**40.655**

PS:

- SSIM is an index for measuring the similarity between two images.
- PSNR uses to measure image quality for image distortion.

The comparison on each predictive time-step

We also compare the LS-NTP with other methods in the four mentioned indexes on each predictive timestep.

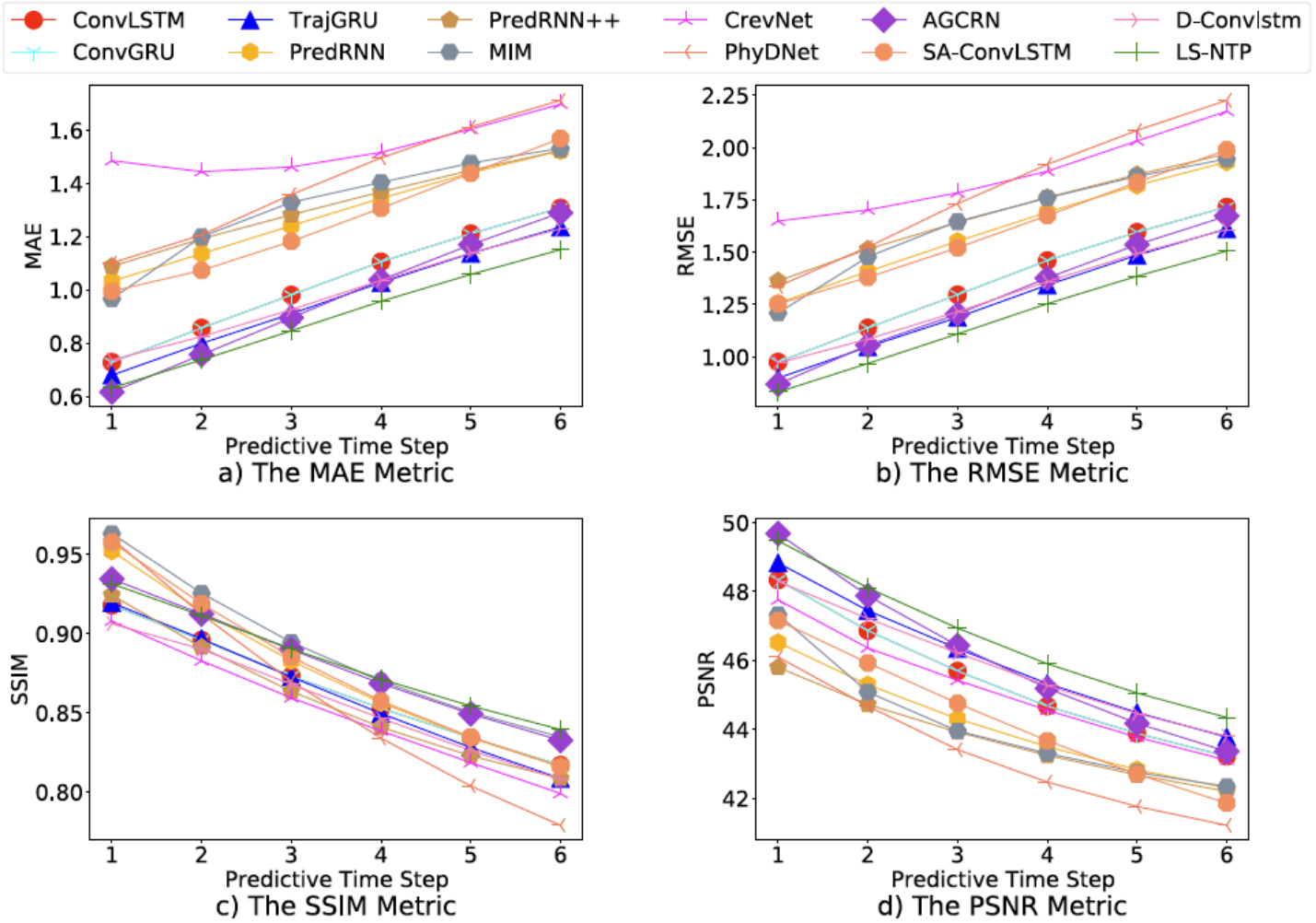


Fig. 8. Performance against different forecast lead time steps on ERA5.

Ablation Study

To measures the effectiveness of the proposed components, we conduct ablation study

- w/o NSA, which does not consider spatial attention in LS-Conv operator;
- w/o LAGC, which does not consider adaptive graph in LS-Conv operator;
- w/o NSA & LAGC, which ablates both the NSA and LAGC modules;
- w/o LSI, which means we do not apply CNN after the adaptive graph convolution;
- DAGG-ConvLSTM, which leverages the DAGG [15] to perform the adaptive graph convolution.

Table 4
The ablation study results for near-surface temperature prediction.

Method	ERA5 (per 1 h)				NCEP (per 6 h)			
	MAE↓	RMSE↓	SSIM↑	PSNR↑	MAE↓	RMSE↓	SSIM↑	PSNR↑
LS-NTP	**0.897**	**1.176**	**0.883**	**46.639**	**1.740**	**2.318**	**0.915**	**40.646**
w/o NSA	1.033	1.358	0.862	45.535	1.867	2.479	0.902	39.981
w/o LAGC	0.911	1.188	0.868	46.482	1.760	2.339	0.913	40.573
w/o NSA&LAGC	1.071	1.433	0.854	45.107	1.830	2.425	0.909	40.284
w/o LSI	0.961	1.279	0.881	46.124	2.132	2.817	0.882	38.870
DAGG-ConvLSTM	0.966	1.277	0.880	46.060	2.228	2.957	0.883	38.666

The ablation study results are shown in Table. As we can see that the performance suffer different degree decreases by ablating different components.

05

PART FIVE

Analysis & Result

Q1: How does a long distance affect the performance?

We set a distance to filter out the entries in the adjacency matrix as zeros, whose distances to the corresponding grid are larger than a threshold d. Here d varies from 0 km to 6000 km (the largest distance of the selected zone).

- When d = 0, the setting means no long-range spatial correlations are considered;
- When d = 6000 the setting is equivalent to the original LS-NTP.

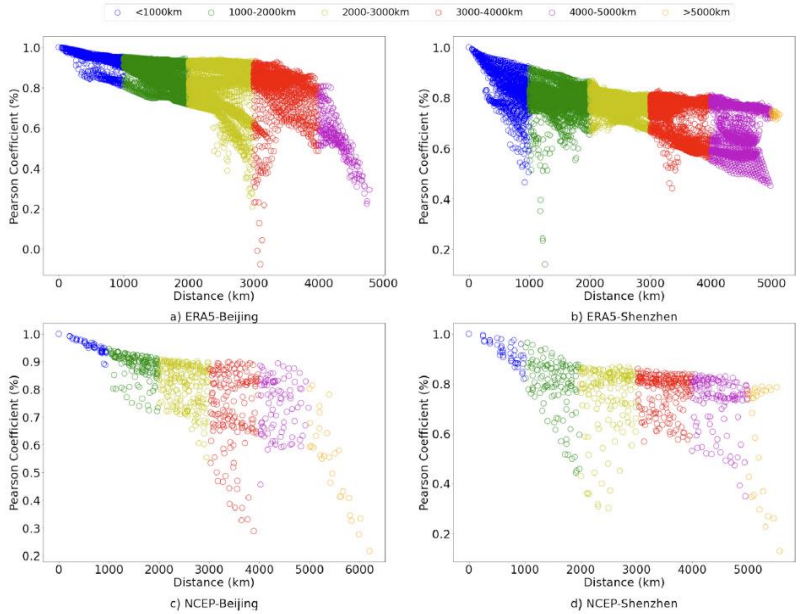
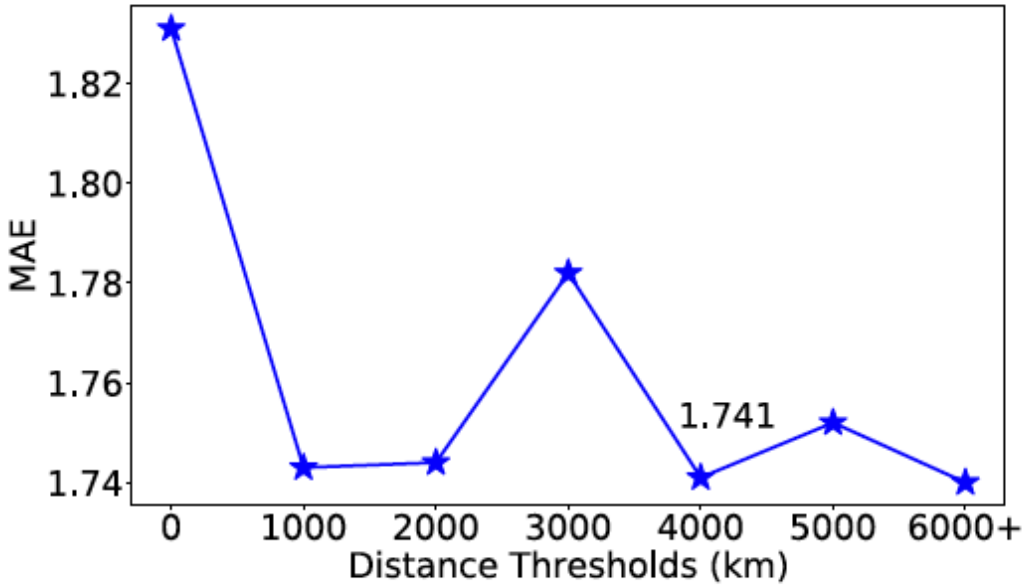
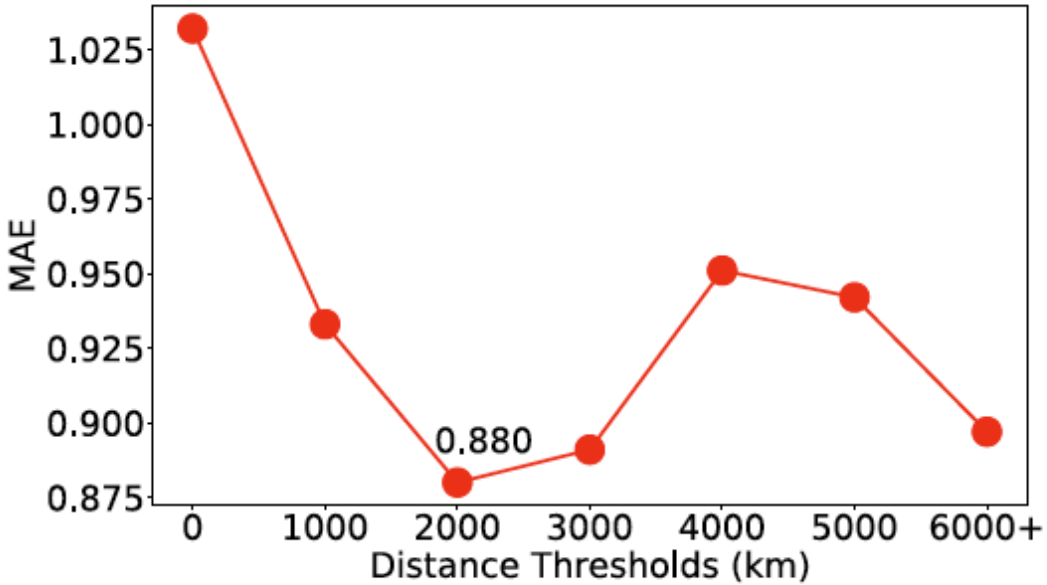


Fig. 1. An illustrative example of long- and short-range spatial correlations for near-surface temperature.



a) The MAE on NCEP



b) The MAE on ERA5

Q2: How does the LS-Conv kernel affect the short-range spatial correlations?

we vary the LS-Conv kernel from 1×1 to 7×7

- the kernel of LS-Conv is 1×1 means we omit the short-range spatial correlations
- the kernel of LS-Conv is 3×3 means we consider short-range in 3×3 area and consider long-range spatial correlation outside this area.

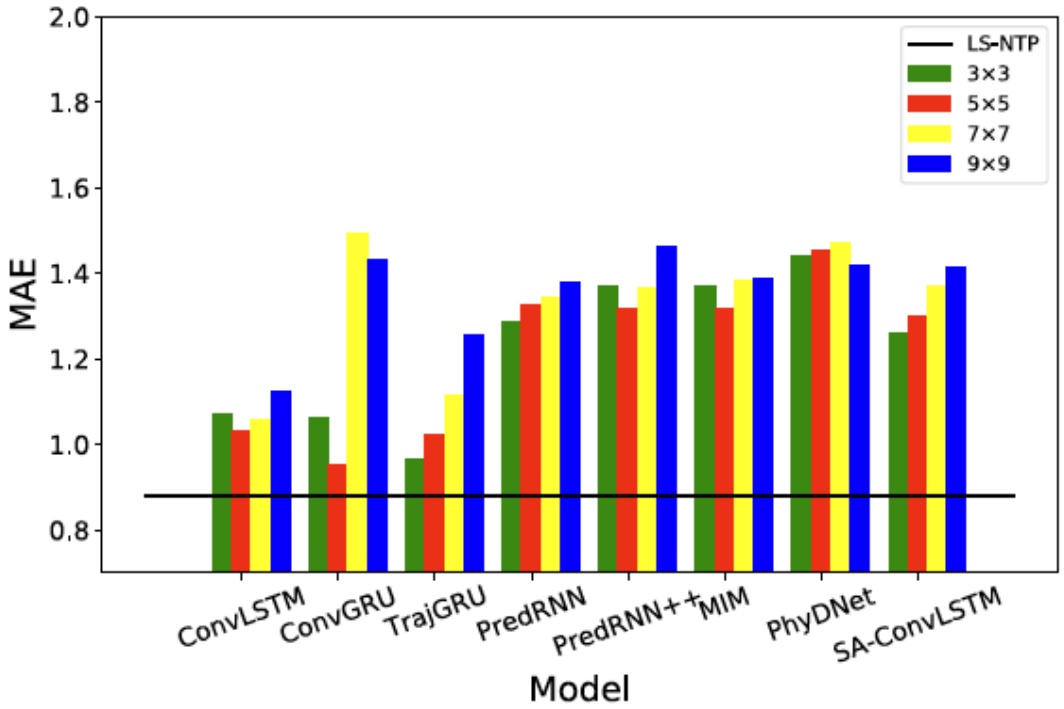
Table 5
Analysis of LS-NTP kernel size K@K.

K@K	ERA5 (per 1 h)				NCEP (per 6 h)			
	MAE↓	RMSE↓	SSIM↑	PSNR↑	MAE↓	RMSE↓	SSIM↑	PSNR↑
1@1	0.939	1.233	0.878	46.308	1.799	2.386	0.910	40.380
3@3	**0.897**	**1.176**	**0.883**	**46.639**	**1.740**	**2.318**	**0.915**	**40.646**
5@5	1.034	1.357	0.869	45.585	1.792	2.397	0.908	40.377
7@7	1.118	1.455	0.845	44.937	1.860	2.465	0.908	40.123

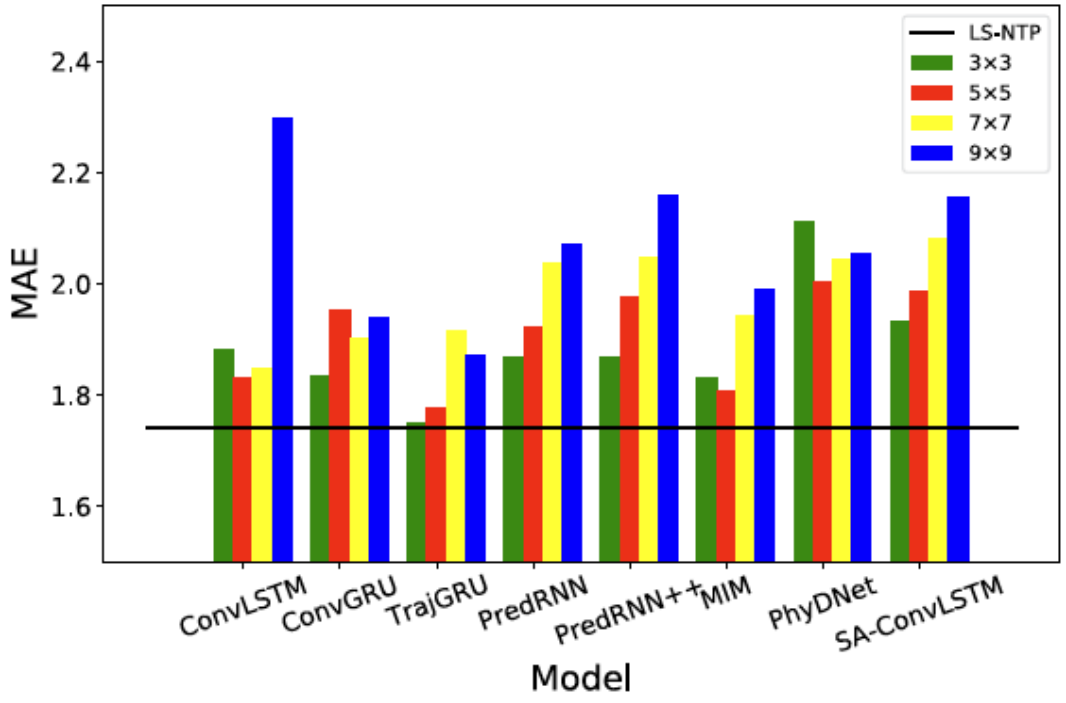
A 3×3 kernel size reaches the best performance, which means considering non short-range spatial correlation or too large short-range spatial correlation helps little of the performance improvement.

Q3: Can a larger convolution kernel capture the long-range spatial correlations?

To answer this question, we enlarge the kernel size of the baseline models from 3×3 to 9×9 in baseline model which kernel uses for the spatial correlation capturing.



(a) The ERA5

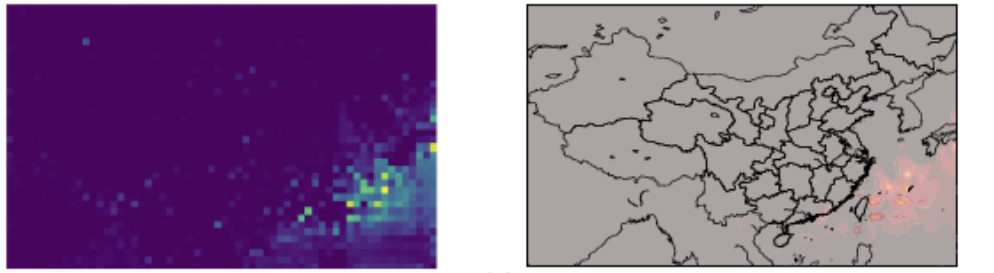


(b) The NCEP

simply enlarge the convolution kernel size might introduce noisy information leading to a performance decreasing.

Q4: Can we visualize the long-range spatial correlations?

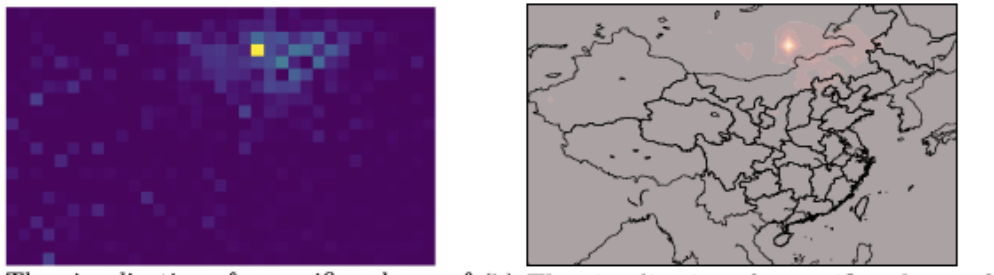
As we have used a data-driven method to learn the long-range spatial correlations, we select the column of the adjacency matrix which means the long-range spatial correlation learned by the model. We visual two cases in two datasets.



(a) The visualization of a specific column of the learned adaptive graph matrix

(b) The visualization of a specific column of the learned adaptive graph matrix on the map

Fig. 11. The validation of long-range spatial correlation on the ERA5 dataset.



(a) The visualization of a specific column of the learned adaptive graph matrix

(b) The visualization of a specific column of the learned adaptive graph matrix on the map

Fig. 12. The validation of long-range spatial correlation on the NCEP dataset.

the adaptive learning long-range spatial correlations are very obvious and reasonable

Q6: Can we visualize the predictive error performance

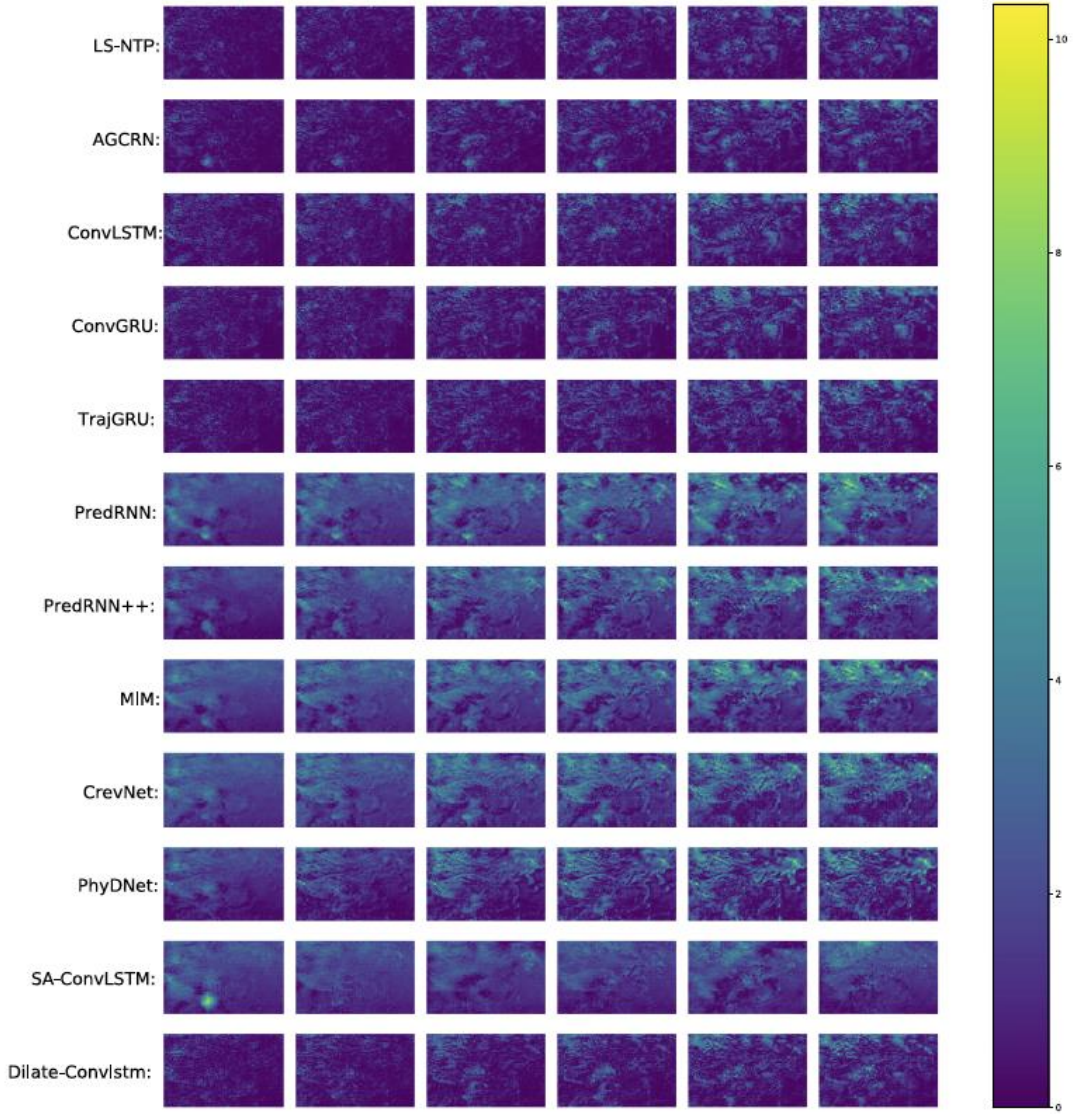


Fig. 15. The MAE metric of the results on the ERA5 dataset.

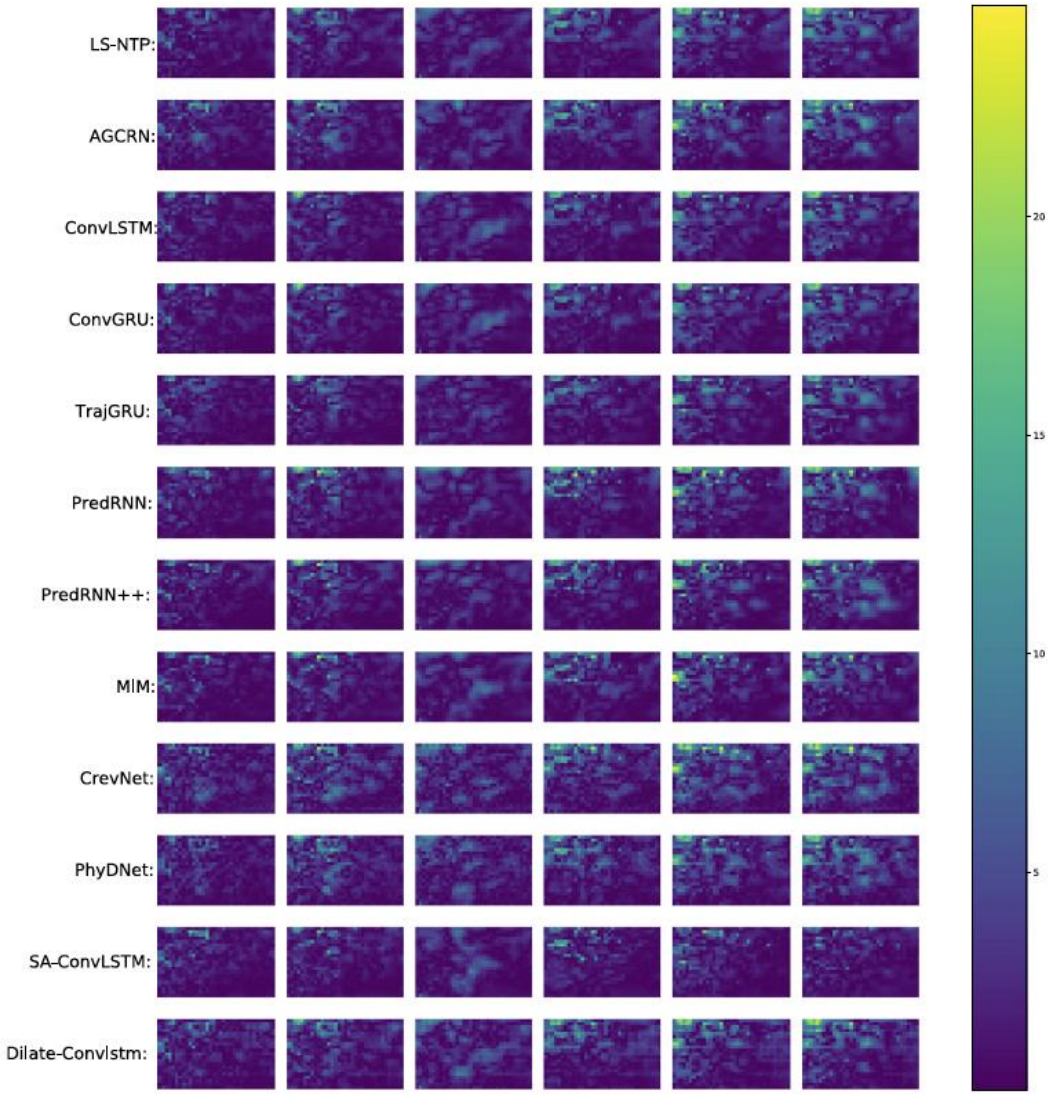


Fig. 16. The MAE metric of the results on the NCEP dataset.

Thank you for listening

Q&A

Reference

1. Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning[J]. arXiv preprint arXiv:1603.07285, 2016.
2. https://github.com/vdumoulin/conv_arithmetic
3. https://blog.csdn.net/shenziheng1/article/details/85058430
4. https://github.com/sixitingting/Adversarial_Autoencoder_Tina
5. https://aistudio.baidu.com/aistudio/projectdetail/5231866?channelType=0&channel=0
6. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30.
7. Hu F, Zhu Y, Wu S, et al. Hierarchical graph convolutional networks for semi-supervised node classification[J]. arXiv preprint arXiv:1902.06667, 2019.
8. https://zhuanlan.zhihu.com/p/65139447
9. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
10. https://en.d2l.ai/chapter_recurrent-modern/lstm.html
11. Tasdelen A, Sen B. A hybrid CNN-LSTM model for pre-miRNA classification[J]. Scientific reports, 2021, 11(1): 1-9.
12. Shi C, Zhang Z, Zhang W, et al. Learning multiscale temporal–spatial–spectral features via a multipath convolutional LSTM neural network for change detection with hyperspectral images[J]. IEEE Transactions on Geoscience and Remote Sensing, 2022, 60: 1-16.
13. Xu G, Li X, Feng S, et al. LS-NTP: Unifying long-and short-range spatial correlations for near-surface temperature prediction[J]. Neural Networks, 2022, 155: 242-257.
14. https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
15. Bai L, Yao L, Li C, et al. Adaptive graph convolutional recurrent network for traffic forecasting[J]. Advances in neural information processing systems, 2020, 33: 17804-17815.