Deep Learning and AI: Approaches and Applications in Traffic & Transportation

- ---A Comprehensive Tutorial to Review Past, Present and Trends

A*STAR
Institute of High Performance Computing

Outline

- Some Questions about Deep Learning
- Methodology of Deep Learning
- Some Basic Deep Learning Architecture
- Applications in Traffic & Transportation
- Useful Tools for Deep Learning Research & Implementation

Some Questions about Deep Leaning

1. **What is the essence behind deep learning?**

Deep learning is multiple layers of representation, obtained by composing non-linear modules that each transform the representation at one level into a representation at a higher, more abstract level. With enough such transformations, very complex functions can be learned.

2. **Why use multiple layers in deep learning instead of 2 layers network? Since 2 layers can approximate the universal functions.**

Deeper network requires exponentially smaller size to approximate the same function.

3. **Why use non-linear activation functions in neural network?**

If not use non-linear activation functions, the whole neural network is a simple linear transformation.

4. **Is deep learning superior in every tasks?**

No free lunch theorem in machine learning, there is no single best machine learning algorithm.

5. **Deep learning is non-convex optimization, SGD algorithm cannot guarantee the global minimum. Why is it still popular?**

Because of the complexity of deep network, local minimum can also provide good solution.

Methodology of Deep Leaning

Deep learning approaches can be divided into four broad categories:

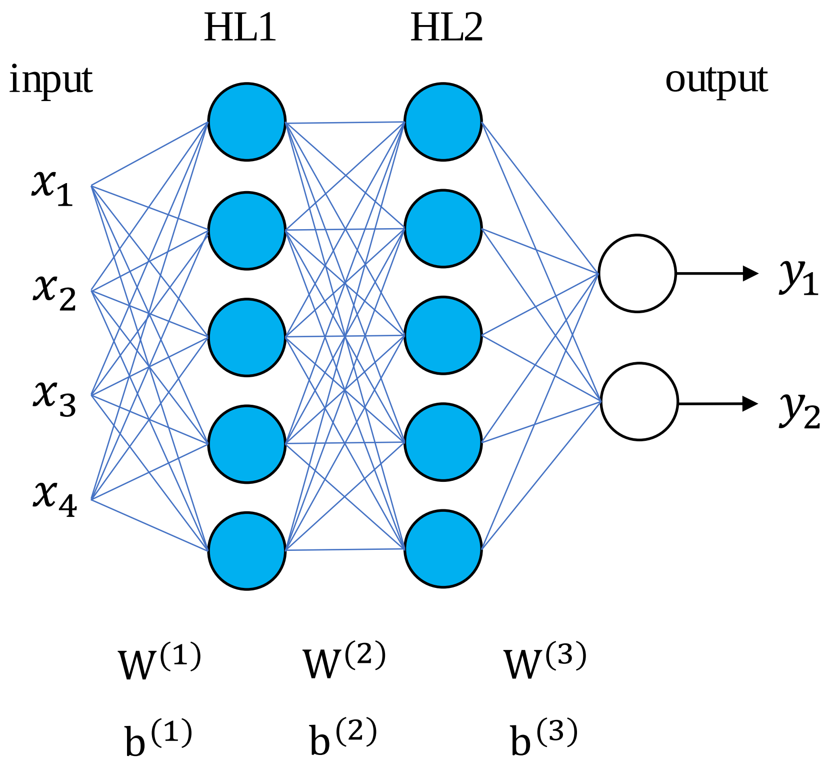
Supervised Learning	With only labeled training data, e.g., classification and regression
Semi-Supervised Learning	A small amount of labeled data and a large amount of unlabeled data
Unsupervised Learning	With no labeled training data. E.g., dimensionality reduction, feature extraction, etc.
Reinforcement Learning	Self-teaching system, agent interacts with environment to take the best action

Methodology of Deep Leaning

There are other branches of deep learning approaches which have been extensively investigated (not limited):

Ensemble Learning	Use multiple learning algorithms to get better performance than single learning algorithm
Multi-Task Learning	Multiple learning tasks are solved at the same time
Imbalanced Learning	Learning from imbalanced data for classification algorithm
Transfer Learning	Model learn from one type of problem and then the learning is applied to solve a different but related problem

A Simple Deep Neural Network



1. Terminology: layer, hidden layer, node (neuron), parameter, activation function, weight (W), bias (b)
2. Input: $\boldsymbol{x}$, output: $\boldsymbol{y} = \emptyset(\mathbf{W}, \mathrm{b}, \boldsymbol{x})$, label (target): $\hat{\boldsymbol{y}}$
2. Loss function $\boldsymbol{L}(\mathbf{W}, \mathbf{b}; \boldsymbol{x}, \hat{\boldsymbol{y}}) = \boldsymbol{Loss}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ + Regularization + Sparsity…
3. Minimize loss function $\mathbf{W}, \mathbf{b} = \underset{\mathrm{W,\,b}}{\boldsymbol{arg\ min}}\ \boldsymbol{L}$
4. Stochastic Gradient Descent (SGD) and Back-propagation (BP) algorithm are commonly employed for optimization

Loss Function

- **MSE Loss**

$$MSE = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2$$

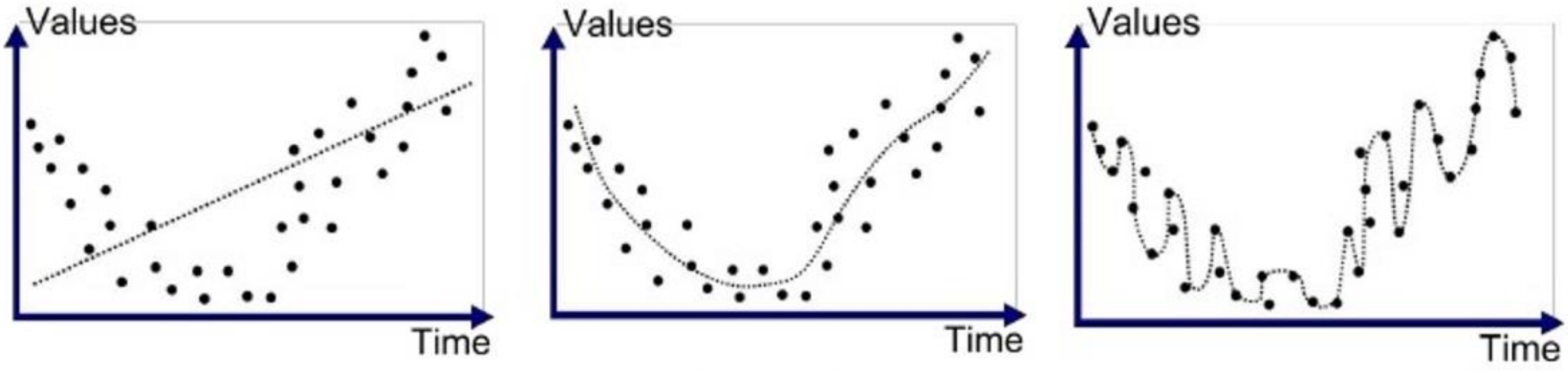
- **Cross Entropy Loss**

$$CE = -\sum_{i=0}^{n} \hat{y}_i * log(y_i)$$

- **Hinge Loss**

$$HL = \sum_{i=0}^{n} max(0,1 - \hat{y}_i * y_i)$$

Underfitting & Overfitting



Underfitted

Good Fit/Robust

Overfitted

- **Deep learning employs some strategies to prevent overfitting, such as regularization, dropout, data augmentation, early stopping, etc.**

Regularization Penalty

- **Why adding a regularization penalty can prevent overfitting?**

Regularization penalty can make weight ($\mathbf{W}$) smaller. The derivative of an overfitting function is normally large, because the function fluctuates greatly.

- **L^1 Regularization**

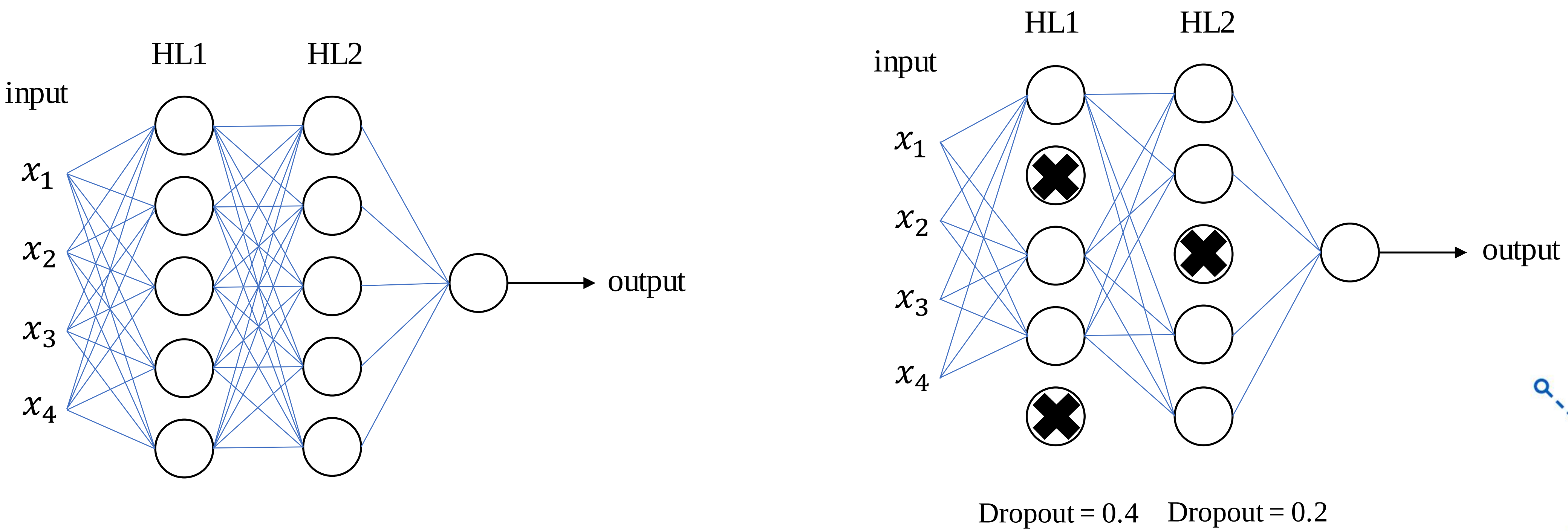
$$L(\mathbf{W}; x, \hat{y}) = loss(y, \hat{y}) + \frac{\propto}{2}\|\mathbf{W}\|_1 = loss(y, \hat{y}) + \frac{\propto}{2}\sum_i |w_i|$$

- **L^2 Regularization**

$$L(\mathbf{W}; x, \hat{y}) = loss(y, \hat{y}) + \frac{\propto}{2}\|\mathbf{W}\|_2^2 = loss(y, \hat{y}) + \frac{\propto}{2}\sum_i {w_i}^2$$

Dropout

- *Dropout* is to drop out some nodes randomly when training while keeping the parameters.
- A kind of regularization to prevent deep neural network from overfitting. Equivalent to training different neural networks with different architecture (ensemble of different architecture).
- More suitable for wider network (i.e., more nodes in the hidden layers).



Sparsity

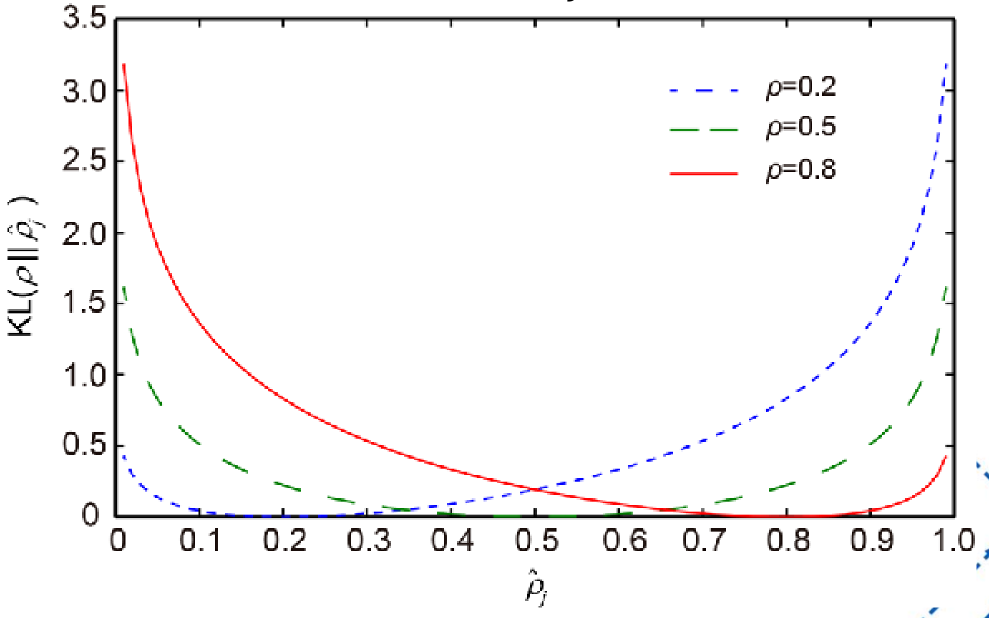
- Sparsity: majority is 0 or near 0, and minority is not 0.
- Sparse feature is informative, outstanding and descriptive, which is better for patterns recognition, especially the feature amount is large.
- Generally, sparsity is employed when the network is wide.
- KL Divergence is appended to the loss function to make the hidden layer features sparse.
- Loss function with sparsity penalty:

$$L(\mathbf{W}; x, \hat{y}) = loss(y, \hat{y}) + \frac{\propto}{2}\|\mathbf{W}\|_2^2 + \gamma \sum_{j=1} \mathrm{KL}(\rho||\hat{\rho}_j)$$

$$\mathrm{KL}(\rho||\hat{\rho}_j) = \rho \log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}$$

- ρ is set as a value close to 0 (e.g., 0.05) for sigmiod function $[0,1]$. For tanh function $[-1,1]$, do scale $\hat{\rho}_j$ to $[0,1]$.

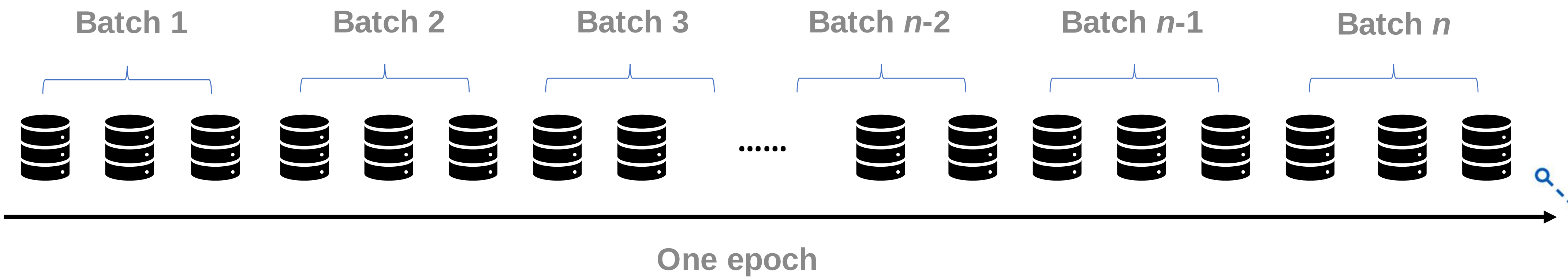
$\mathrm{KL}(\rho||\hat{\rho}_j)$



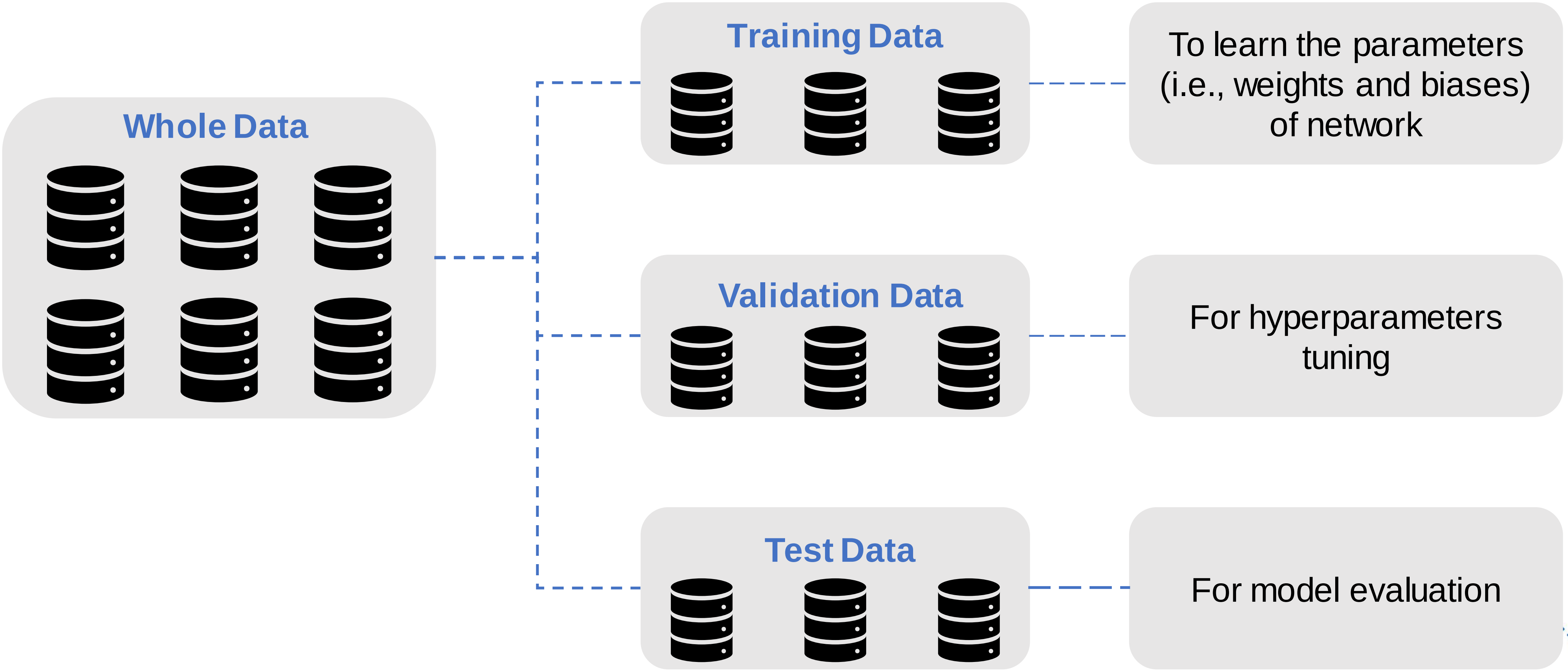
Mini-Batch Stochastic Gradient Descent (SGD)

Suppose the sample size of training data is m

- **Batch Gradient Descent (BGD):** **Feed all the training data at one time, i.e., $Batch\ Size = m$. Require huge memory.**
- **Stochastic Gradient Descent (SGD):** **Feed one sample at one time, i.e., $Batch\ Size = 1$. Difficult to converge, slow training.**
- **Mini-Batch Stochastic Gradient Descent:** **$1 < Batch\ Size < m$.**



Training Processing

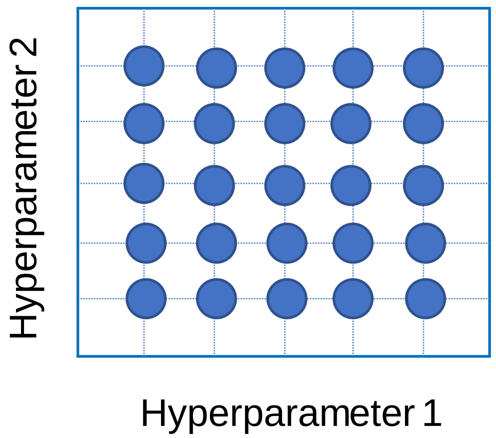


Institute of High Performance Computing
A*STAR

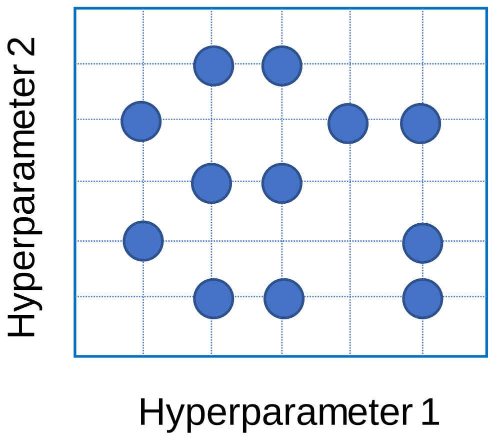
Hyperparameters

- Hyperparameters include model hyperparameters and algorithm hyperparameters. Note the difference to network's parameters
- Model hyperparameters determine the network topology, e.g., input number, hidden layer number, nodes in each layer
- Algorithm hyperparameters include maximum epoch, batch size, learning rate, momentum, dropout, regularization, activation function, etc.
- Hyperparameters tuning is crucial for deep learning to give better results
- Tuning methods include Random Search, Grid Search, Bayesian Optimization, Differential Evolution Optimization, etc.

Grid Search



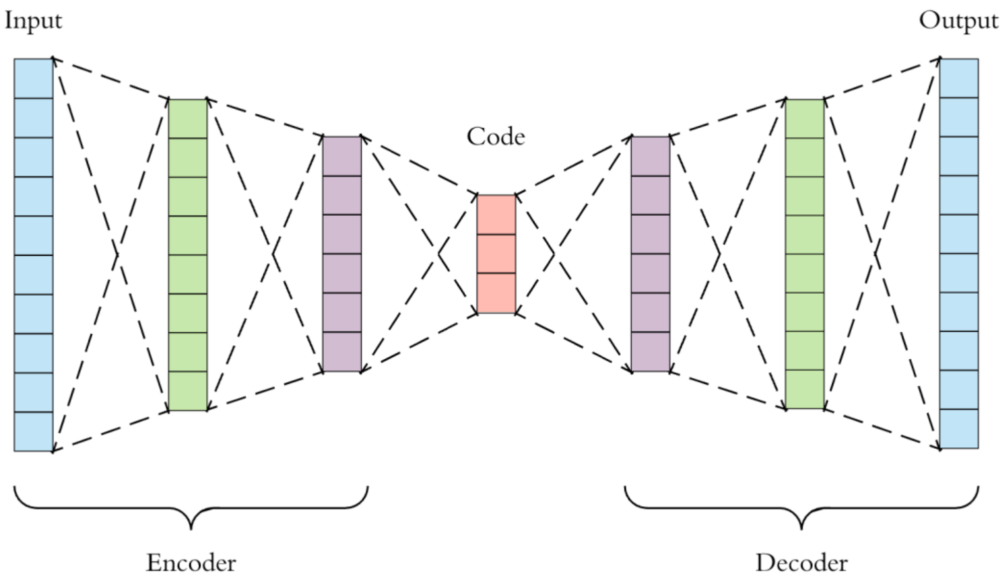
Random Search

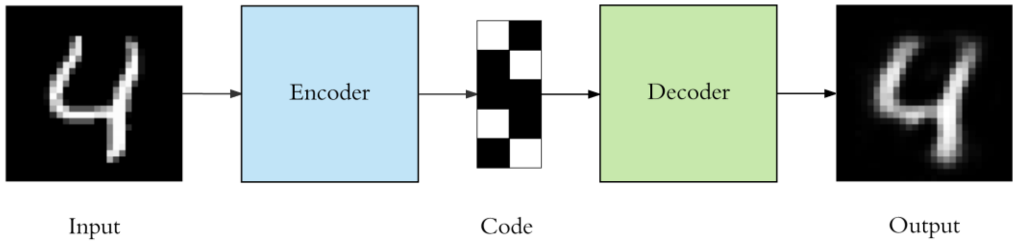


Some Basic Deep Learning Architecture

- **Stacked Autoencoder**
- **Convolutional Neural Network (CNN)**
- **Graph Neural Network (GNN)**
- **Reinforcement Learning**

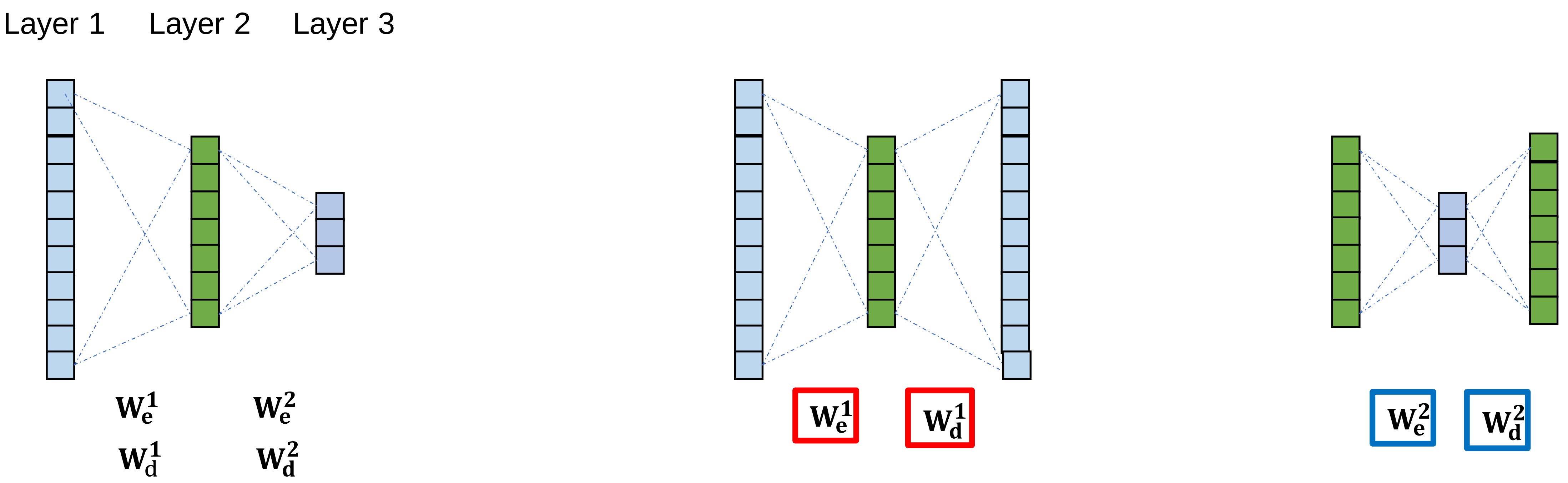
Stacked Autoencoder





- Unsupervised learning (or supervised learning, but the label is also the input)
- Mainly used in dimensionality reduction, feature extraction, information retrieval, etc.
- Greedy layer-wise pretraining is proposed for network's parameters initialization closer to good solutions.

Greedy layer-wise pretraining



- **Pretraining is to initialize the starting values of parameters $\mathbf{W}_1$ and $\mathbf{W}_2$**
- **It is conducted layer by layer**
- **When conducting this layer, the parameters in the previous layers are fixed**

Convolutional Neural Network (CNN)

Convolution

5×5 $\quad$ 2×2 Conv Kernel (Filter)

$$\begin{bmatrix} 0 & 3 & 1 & 0 & 1 \\ 2 & 0 & 2 & 4 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 4 & 3 & 5 & 2 & 2 \\ 0 & 3 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \boxed{2} & 5 & 7 & 4 \\ 4 & 1 & 2 & 4 \\ 8 & 9 & 7 & 4 \\ 7 & 6 & 7 & 3 \end{bmatrix}$$

$$2 = 0 * 1 + 3 * 0 + 2 * 1 + 0 * 1$$

Max-Pooling

$4 \times 4 \times 1$ $\quad$ 2×2 Max-Pooling

$$\begin{bmatrix} \boxed{2} & 5 & 7 & 4 \\ 4 & 1 & 2 & 4 \\ 8 & 9 & 7 & 4 \\ 7 & 6 & 7 & 3 \end{bmatrix} \qquad \begin{bmatrix} \boxed{5} & 7 \\ 9 & 7 \end{bmatrix}$$

$$5 = \max(2, 5, 4, 1)$$

Convolutional Neural Network (CNN)

Multi-Channel Convolution

$4 \times 4 \times 3$ $\qquad$ $2 \times 2 \times 3$

R $$\begin{bmatrix} 0 & 3 & 1 \\ 2 & 0 & 2 \\ 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 4 & 1 \end{bmatrix}$$

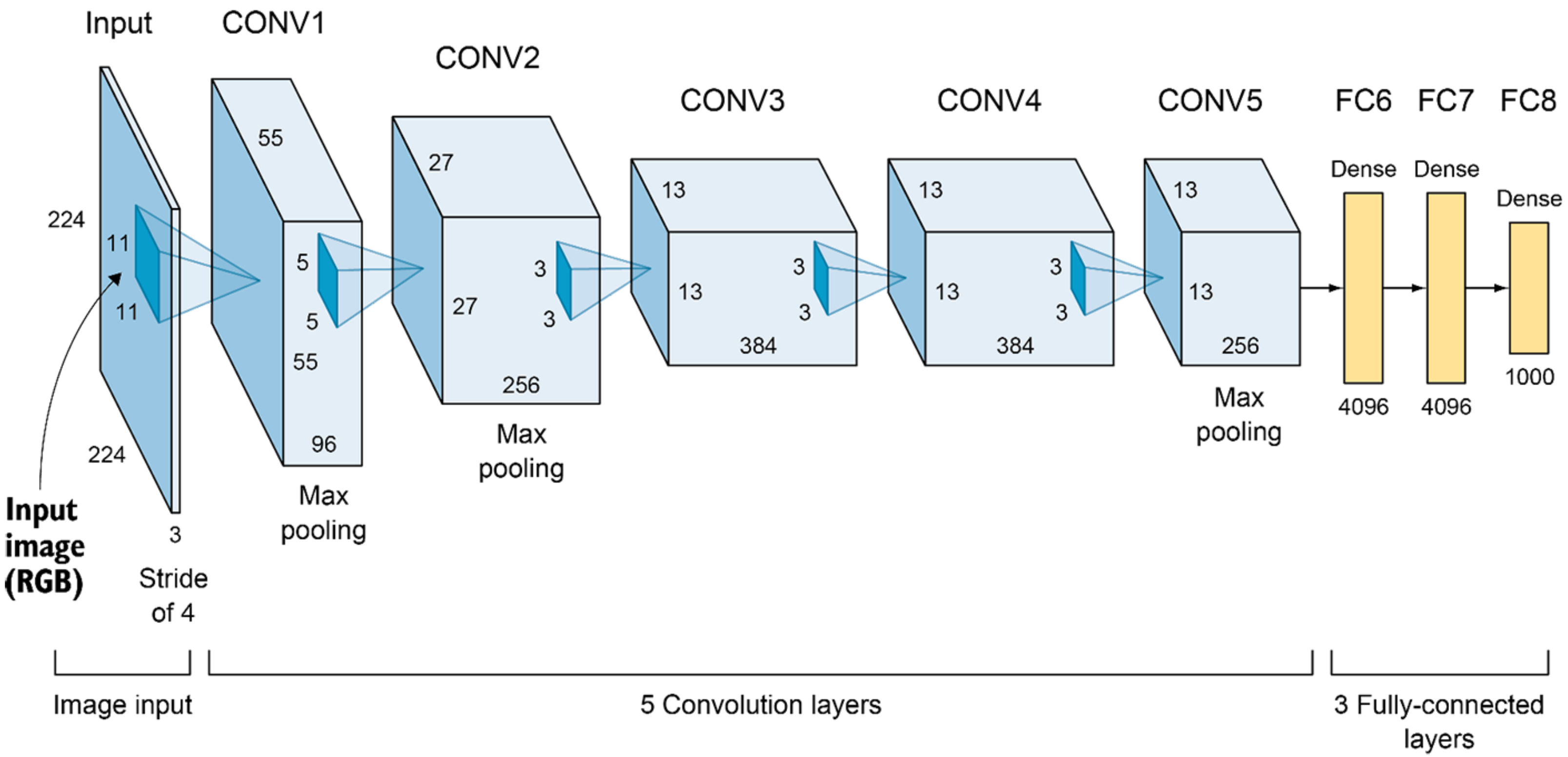
G $$\begin{bmatrix} 0 & 3 & 1 \\ 2 & 0 & 2 \\ 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}$$

sum $$\begin{bmatrix} 2 & 12 \\ 8 & 1 \end{bmatrix}$$

B $$\begin{bmatrix} 0 & 3 & 1 \\ 2 & 0 & 2 \\ 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 5 \\ 3 & 0 \end{bmatrix}$$

Convolutional Neural Network (CNN)

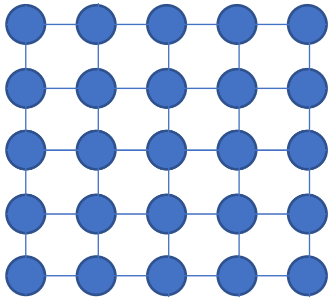
Architecture of AlexNet



- AlexNet is considered one of the most influential work in image classification and computer vision.
- AlexNet consists of 8 layers, namely 5 convolution layers and 3 fully connected layers. 3 convolution layers are followed by max-pooling layers.

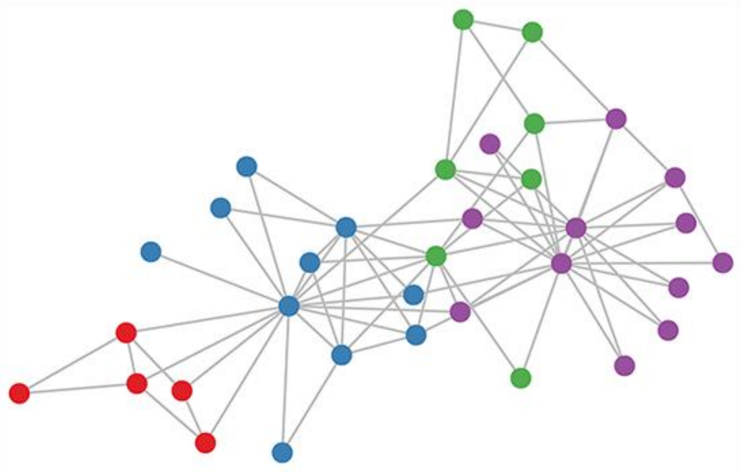
Graph Neural Network (GNN)

Euclidean Structure



$$\begin{bmatrix} a_{0,0} & \cdots & a_{0,n} \\ \cdots & \cdots & \cdots \\ a_{m,0} & \cdots & a_{m,n} \end{bmatrix}$$

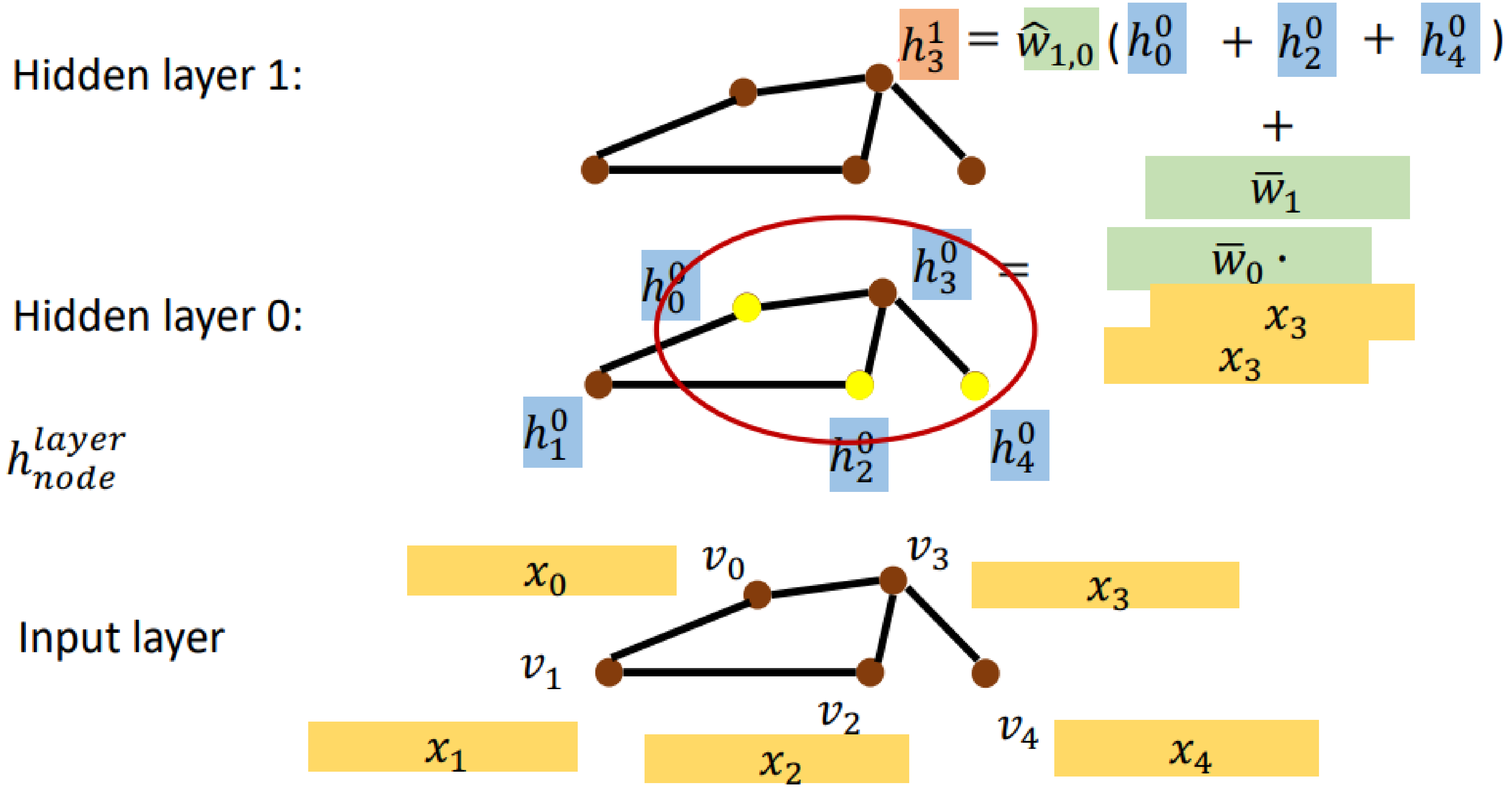
Non-Euclidean Structure



- Non-Euclidean structured graph doesn't have Euclidean properties.
- It is ubiquitous in social networks, knowledge graphs, protein-interaction networks, etc.

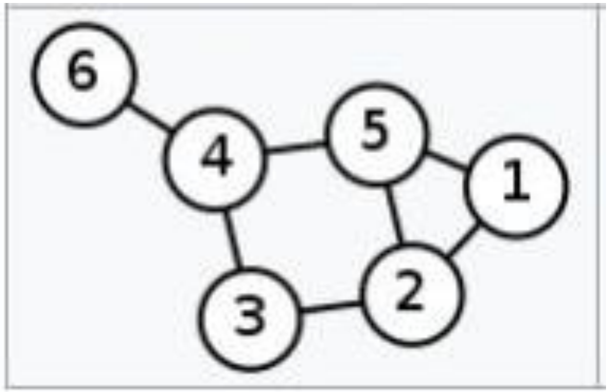
Graph Neural Network (GNN)

Spatial-Based Convolution (NN4G Model)



Graph Neural Network (GNN)

Spectral-Based Convolution (Graph Convolutional Network (GCN) model)



Adjacency Matrix $\mathbf{A}$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Sum by row

Degree Matrix $\mathbf{D}$

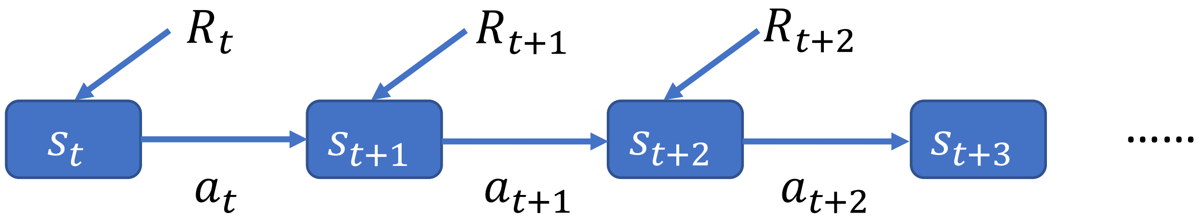
$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The universal formulation of graph convolution is $\mathbf{H}^{l+1} = f(\mathbf{H}^{l}, \mathbf{A})$

Frequently used solution: $\mathbf{H}^{l+1} = \sigma(\breve{\mathbf{D}}^{-\frac{1}{2}} \breve{\mathbf{A}} \breve{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^{l} \mathbf{W}^{l})$

$\breve{\mathbf{A}} = \mathbf{I} + \mathbf{A}$ or $\breve{\mathbf{A}} = \mathbf{I} - \mathbf{A}$

Reinforcement Learning (RL)



- s --- state
- a --- action
- R --- reward
- $\pi(s|a)$ --- policy

RL is to learn the policy function $\pi(s|a)$

Cumulative future reward: $U_t = R_t + \lambda R_{t+1} + \lambda^2 R_{t+2} + \lambda^3 R_{t+3} + \cdots + \lambda^n R_{t+n}$

Reward function: $R_t = f(s_t, a_t)$

Objective: $\underset{\pi}{maximize}\ U_t$

$subject\ to\ 0 < \lambda < 1$

Expectation of U_t: $Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}(U_t)$

$$Q^*(s_t, a_t) = \max_{\pi}(Q_\pi(s_t, a_t))$$

Deep Q-Network (DQN)

- DQN uses deep learning architecture to approximate the $\boldsymbol{Q(s_t, a_t)}$ function
- DQN uses **Temporal Differential Learning** algorithm for training the network

$$U_t = R_t + \lambda U_{t+1}$$

$$Q(s_t, a_t) = R_t + \lambda Q(s_{t+1}, a_{t+1})$$

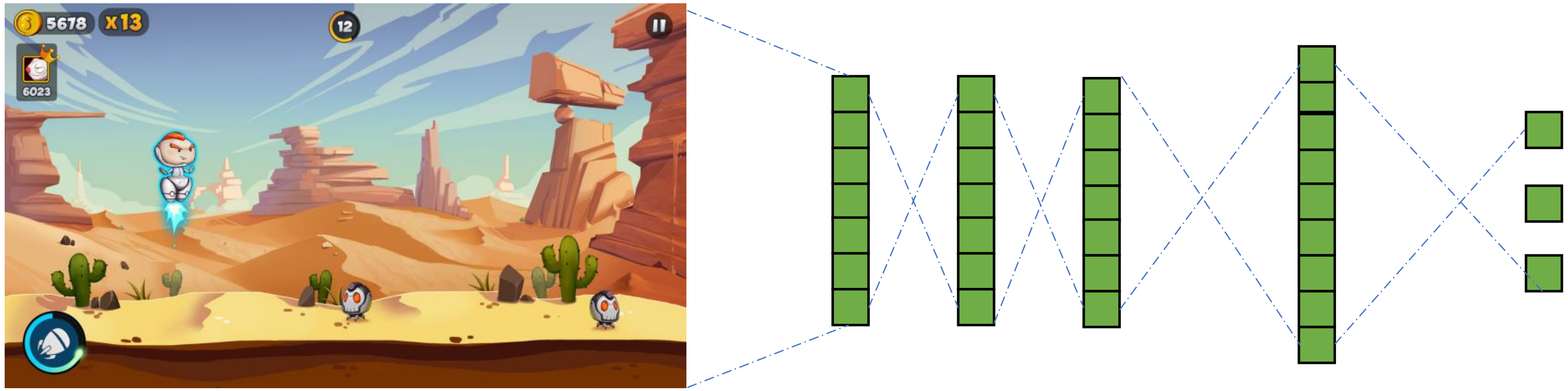
$$Q(s_t, a_t) = R_t + \lambda \cdot \max_a Q(s_{t+1}, a)$$

$Q(s_t, a_t)$ as the predicted

$R_t + \lambda \cdot \max_a Q(s_{t+1}, a)$ as the groundtruth of DQN

- An example---video game

State (s_t) | Convolution Layers | Dense Layer | Output (Q values for each action a)



$Q(s_t, up) = 500$

$Q(s_t, left) = 800$

$Q(s_t, right) = 1200$

Applications in Traffic & Transportation

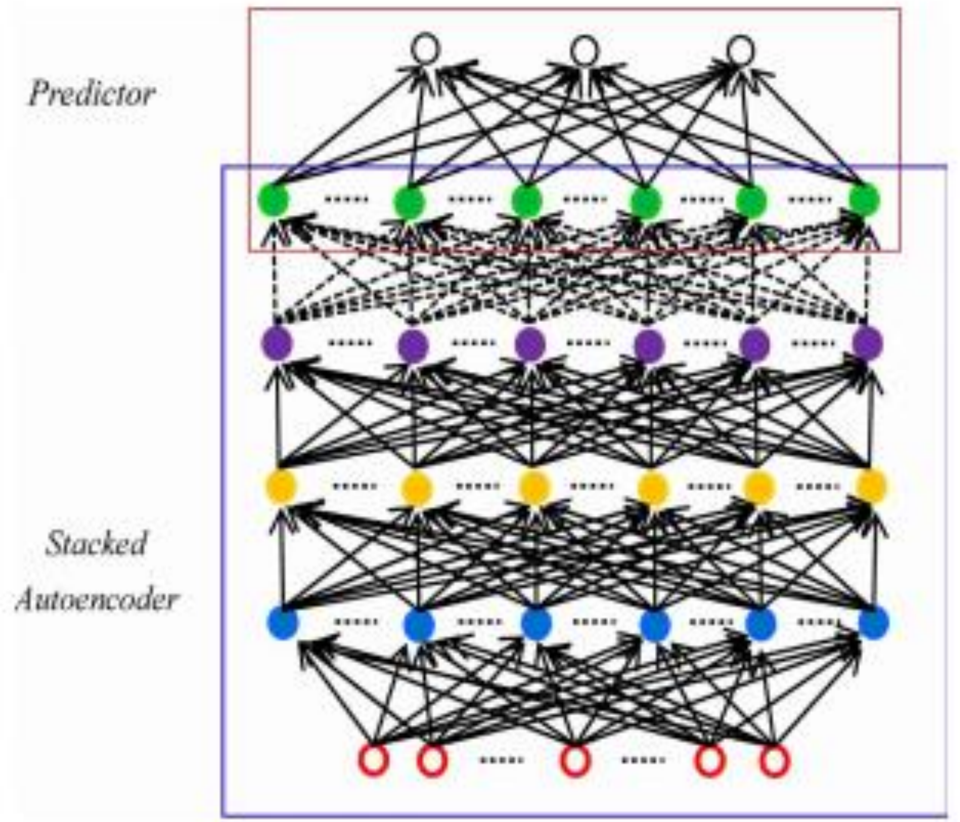
The applications of deep learning models in traffic & transportation are mainly reviewed as the following 5 aspects:

- Time-Series Prediction
- Classification Problem
- Unsupervised Learning
- Transfer Learning
- Reinforcement Learning

Time-Series Prediction

Temporal Prediction of Traffic Flow

Output



Input

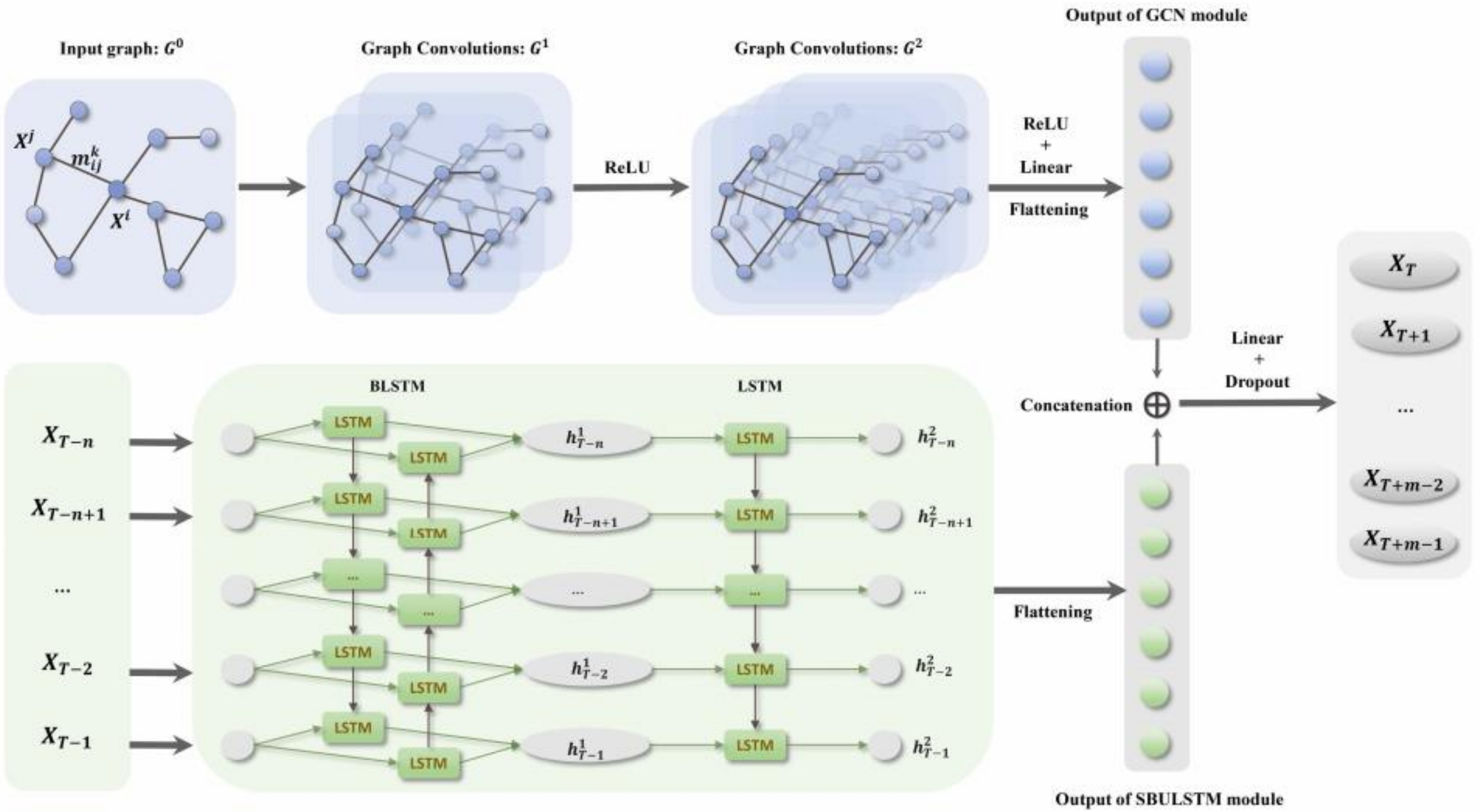
- Input: the historical m time series points
- Output: the future n time series points
- Stacked Autoencoder for feature extraction
- KL Divergence is used to make the hidden feature sparse
- A logistic regression is appended at the top for traffic flow regression
- Pretraining is performed for stacked Autoencoder parameters initialization

Time-Series Prediction

Spatio-Temporal Metro Ridership Prediction

Graph for Metro Stations

Time Series of Ridership in All Metro Stations



Predicted Future Ridership in All Metro Stations

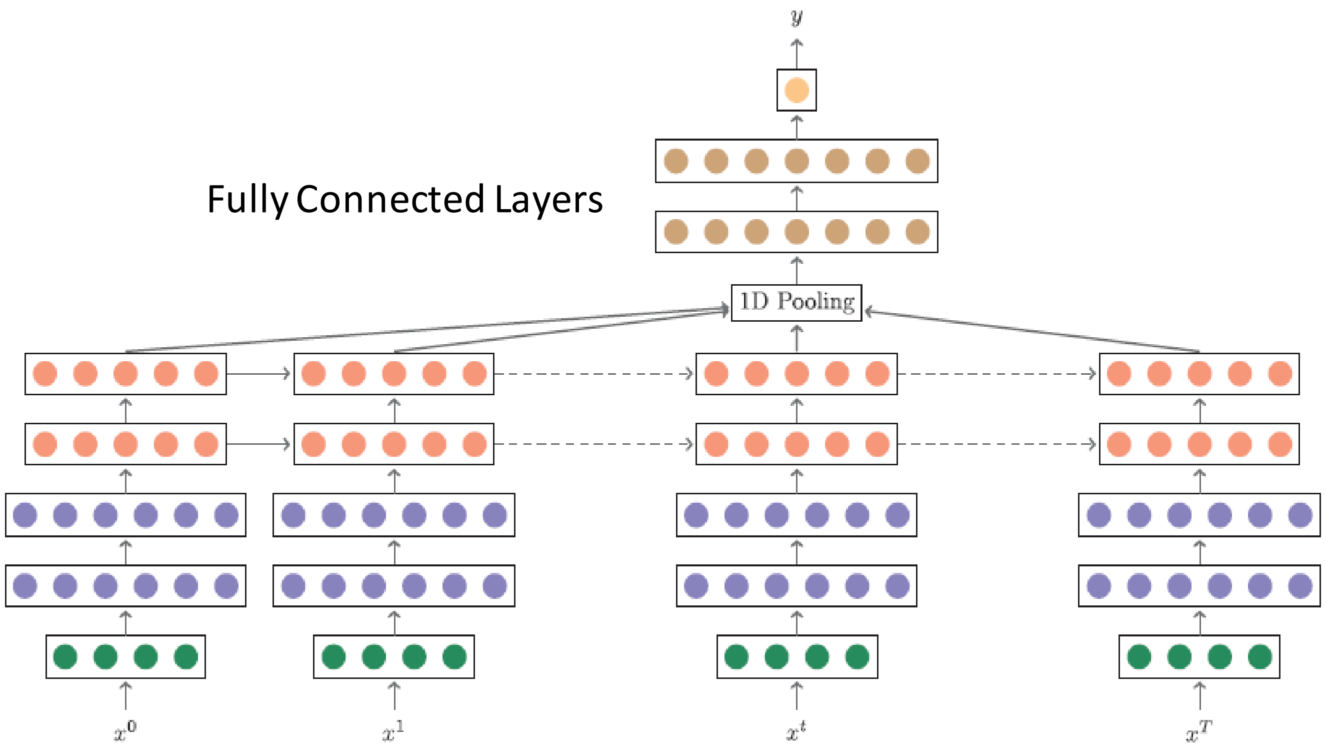
Institute of High Performance Computing

A*STAR

Classification Problem in Traffic

Vehicle Type Classification based on Low-Frequency GPS Data

Outputs---small-duty, median-duty or heavy-duty vehicle?



Fully Connected Layers

LSTM Block

Fully Connected Layers

Inputs

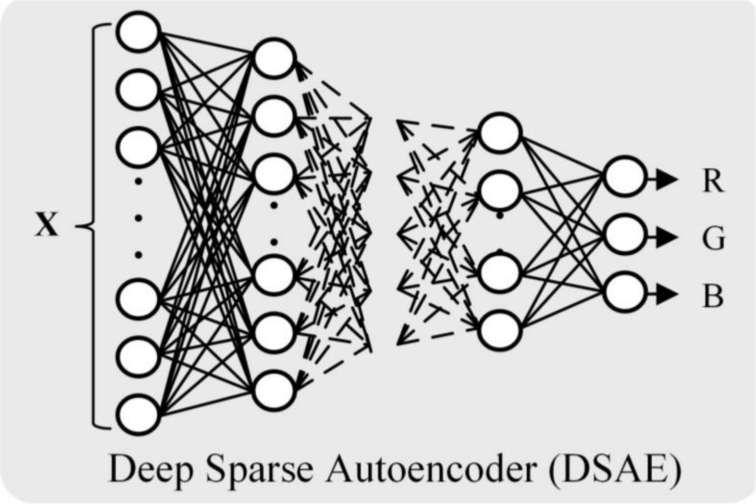
- Input includes 5 types, namely distance from previous point, time from previous point, speed, acceleration and road type.
- Result: accuracy for light, mid and heavy-duty vehicle are 85%, 48% and 93%, respectively.

Predicted

Real		Light	Mid	Heavy
	Light	**85%**	13%	2%
	Mid	41%	**48%**	11%
	Heavy	1%	6%	**93%**

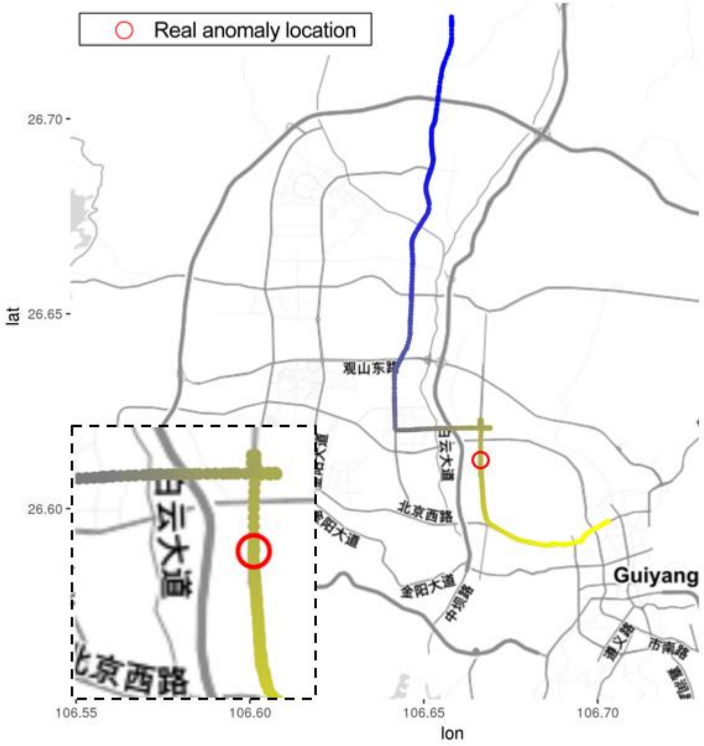
Unsupervised Learning in Traffic

Traffic Anomaly Detection Based on Vehicle Trajectories



Deep Sparse Autoencoder (DSAE)

map fusion

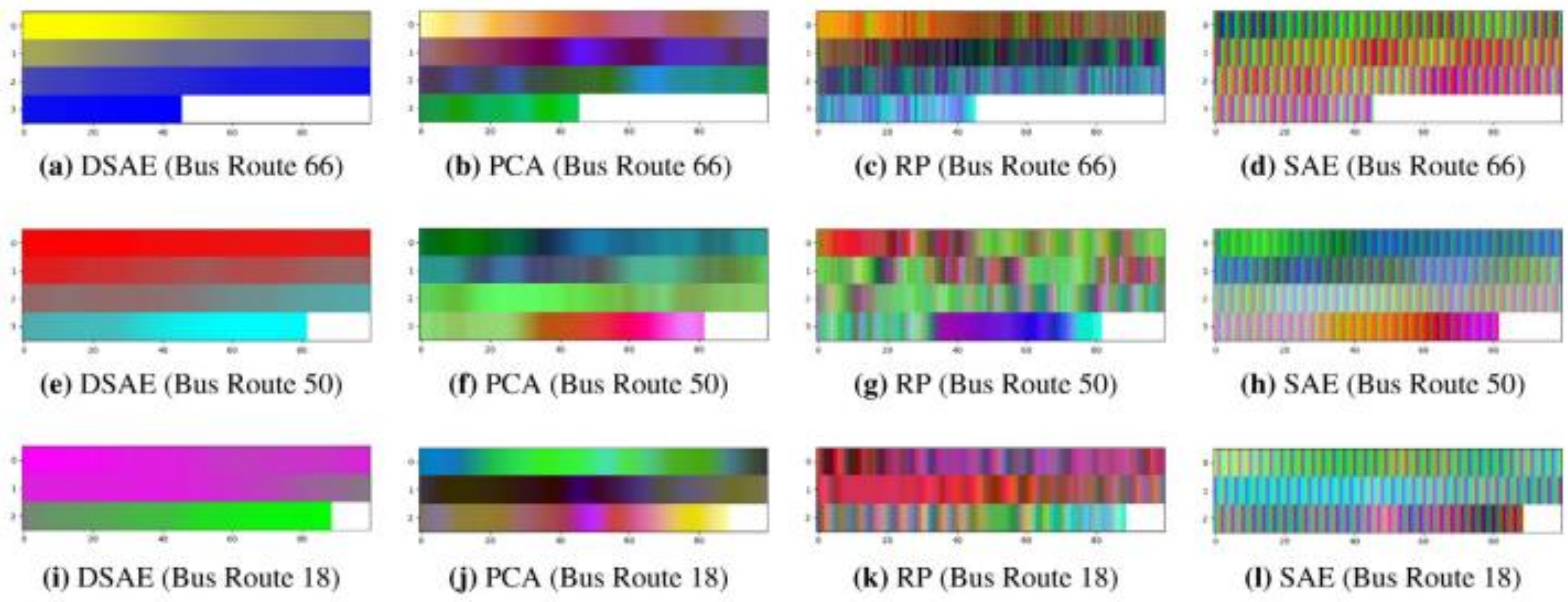


- Deep Sparse Autoencoder is employed for bus trajectories visualization and feature extraction.
- 3 channels output corresponding to R, G, B channels in color space.
- Input (X) is time series includes 4 types, namely latitude, longitude, speed and weather data (i.e., rainfall).

Institute of High Performance Computing
A*STAR

Unsupervised Learning in Traffic

Traffic Anomaly Detection Based on Vehicle Trajectories



(a) DSAE (Bus Route 66) (b) PCA (Bus Route 66) (c) RP (Bus Route 66) (d) SAE (Bus Route 66)

(e) DSAE (Bus Route 50) (f) PCA (Bus Route 50) (g) RP (Bus Route 50) (h) SAE (Bus Route 50)

(i) DSAE (Bus Route 18) (j) PCA (Bus Route 18) (k) RP (Bus Route 18) (l) SAE (Bus Route 18)

- **DSAE generates better visualization than PCA, Random Projection (RP), and Single Sparse Autoencoder (SAE).**

Transfer Learning in Traffic

Vehicle Type Recognition Using Deep Transfer Learning

Source domain (web nature labeled images)

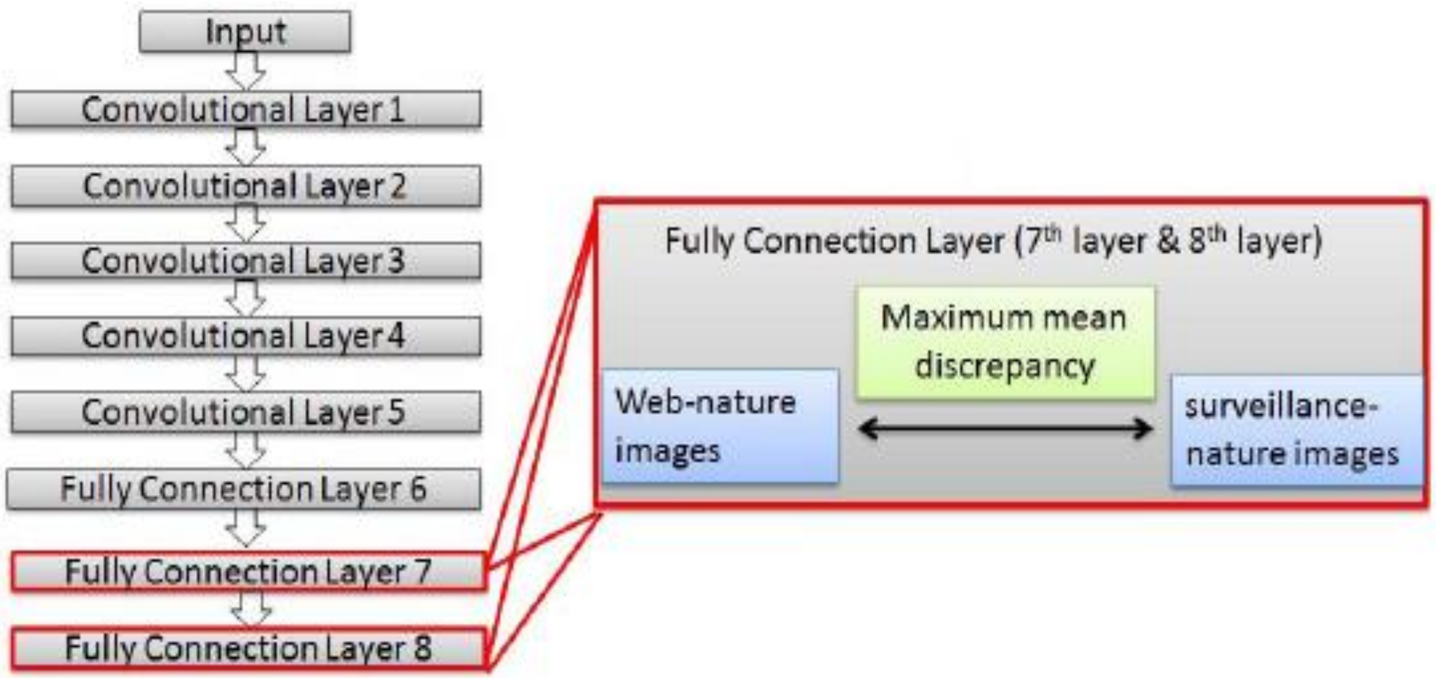
Target domain (unlabeled images from traffic surveillance)

- Labeled images of vehicle are easy to obtain from web.
- Labeling for camera surveillance images are very exhaustive.
- Transfer learning uses the images in source domain for training a deep learning model.

Transfer Learning in Traffic

Vehicle Type Recognition Using Deep Transfer Learning

AlexNet



- The main difference is the loss function

$$\sum L(\theta(xs), ys) + \gamma \sum_{l=1}^{2} \mathbf{MMD}(\boldsymbol{\theta_l}(\mathbf{XS}), \boldsymbol{\theta_l}(\mathbf{XT}))$$

- θ – model, xs – batch input (source domain), ys – batch label (source domain), XS – all images (source domain), XT – all images (target domain)
- The main idea to minimize the model's outputs of the source domain input and target domain input
- All images in source domain as input, get hidden feature at the last l layer $\theta_l(\mathrm{XS})$
- All images in target domain as input, get hidden feature at the last l layer $\theta_l(\mathrm{XT})$
- Maximum mean discrepancy (MMD) of the values in the last two layers

Reinforcement Learning in Traffic

Intelligent Traffic Light Control

Design of State, Action and Reward:

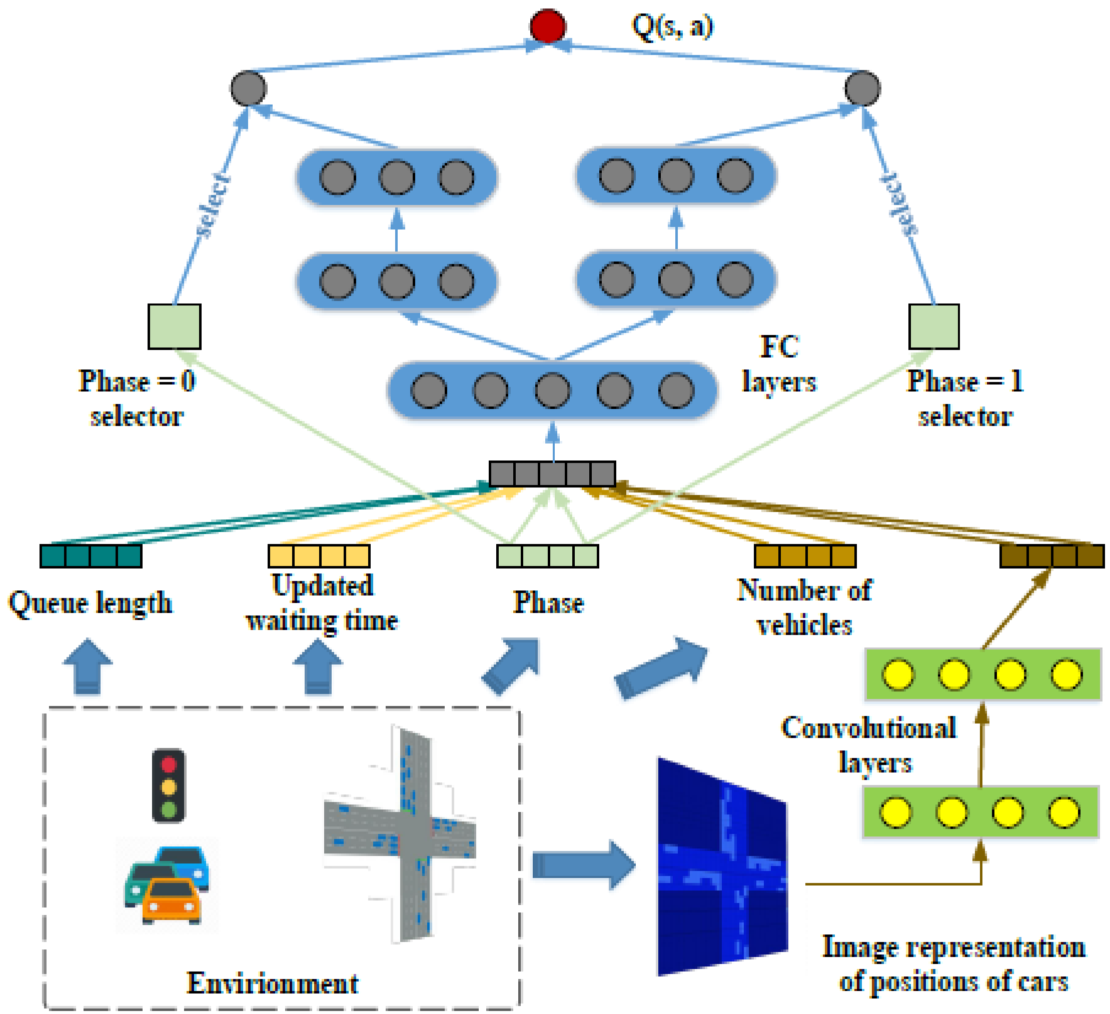
- **State (s):** one state is defined for one intersection. State components include the queue length, number of vehicle, waiting time for each lane, current light phase and traffic image (vehicles position images)

- **Action (a):** $a = 1$, change light to next phase; $a = 0$, keep current light phase

- **Reward (r):** a weighted sum of factors such as the queue length, delay, waiting time, vehicle number passing, travel time, etc.

Reinforcement Learning in Traffic

Intelligent Traffic Light Control



$$Q(s_t, a_t) = r_t + \lambda \max_a Q(s_{t+1}, a)$$

- The input of DQN is the state, queue length, waiting time, light phase, vehicle number and traffic image
- The output is the Q values for each action
- r_t is a known value
- $Q(s_t, a_t)$ is the predicted value, while $r_t + \lambda \max_a Q(s_{t+1}, a)$

Institute of High Performance Computing

A*STAR

Useful Tools for Deep Learning Research and Implementation

- **Framework**

TensorFlow

PyTorch

theano

Keras

Caffe

- **Specialized Package**
    - ❑ Deep Graph Library ----- GNN Implementation
    - ❑ OpenAI Gym ---- Reinforcement Learning Algorithm Development and Comparison

- **Repository**
    - ❑ Kaggle ----- Public Data Set
    - ❑ UCI Machine Learning Repository ----- Public Data Set
    - ❑ GitHub ---- Various Deep Leaning Model Implementation Codes

Thanks

Q&A