Seoul Woman's University
Hyemin Lee

CANTransfer

Transfer Learning based Intrusion Detection on a Controller Area Network using Convolutional LSTM Network

Shahroz Tariq, Sangyup Lee, Simon S. Woo

Introduction

1. The problem of network attacks arising from the spread of autonomous driving in the automobile industry
2. CAN Protocol provides stable and economical communication between ECUs.
3. Re-learning of ConvLSTM-based model using one-shot learning to solve CAN vulnerability problems.

What is CAN(Controller Area Network)?

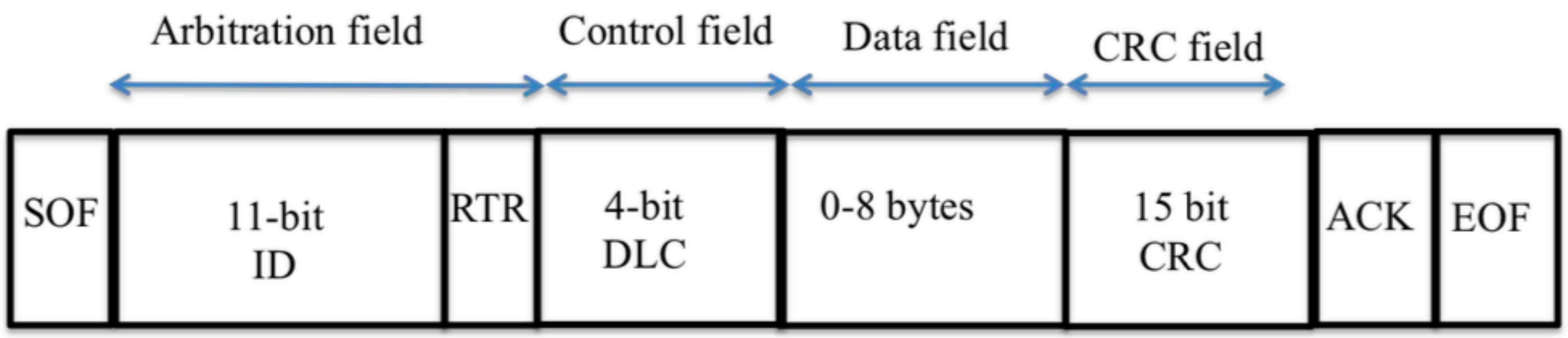


Figure 1: CAN Bus Frame Format

- **CAN** is a de-facto standard for **serial communication.**

- It has been widely used for in-vehicle communication, **Stable and economic link** between electronic control units (ECUs).

CAN Traffic Analysis

Table 1: Sample CAN packets, showing different ID and DLC fields

Timestamp	ID	DLC	Data
1479246664.162438	0153	8	00 21 10 ff 00 ff 00 00
1479246664.164967	02b0	5	bd ff 00 07 dc
1479246664.165822	043f	8	00 40 60 ff 58 28 08 00
1479246664.171065	05f0	2	f4 00
1479246664.171695	0002	8	00 00 00 00 00 06 01 a2

- To understand the **behavioral patterns** in CAN traffic, two different cars (KIA Motors, Hyundai Sonata) were used.
- Each car uses a **different frame ID**, It can be confirmed manufactured by a **different company.**
- **High correlation** occured the precedence bits and sequences between Car1 and Car2.

Problem

Problem Statement

-The **receiving node does not verify** the source of a CAN message.

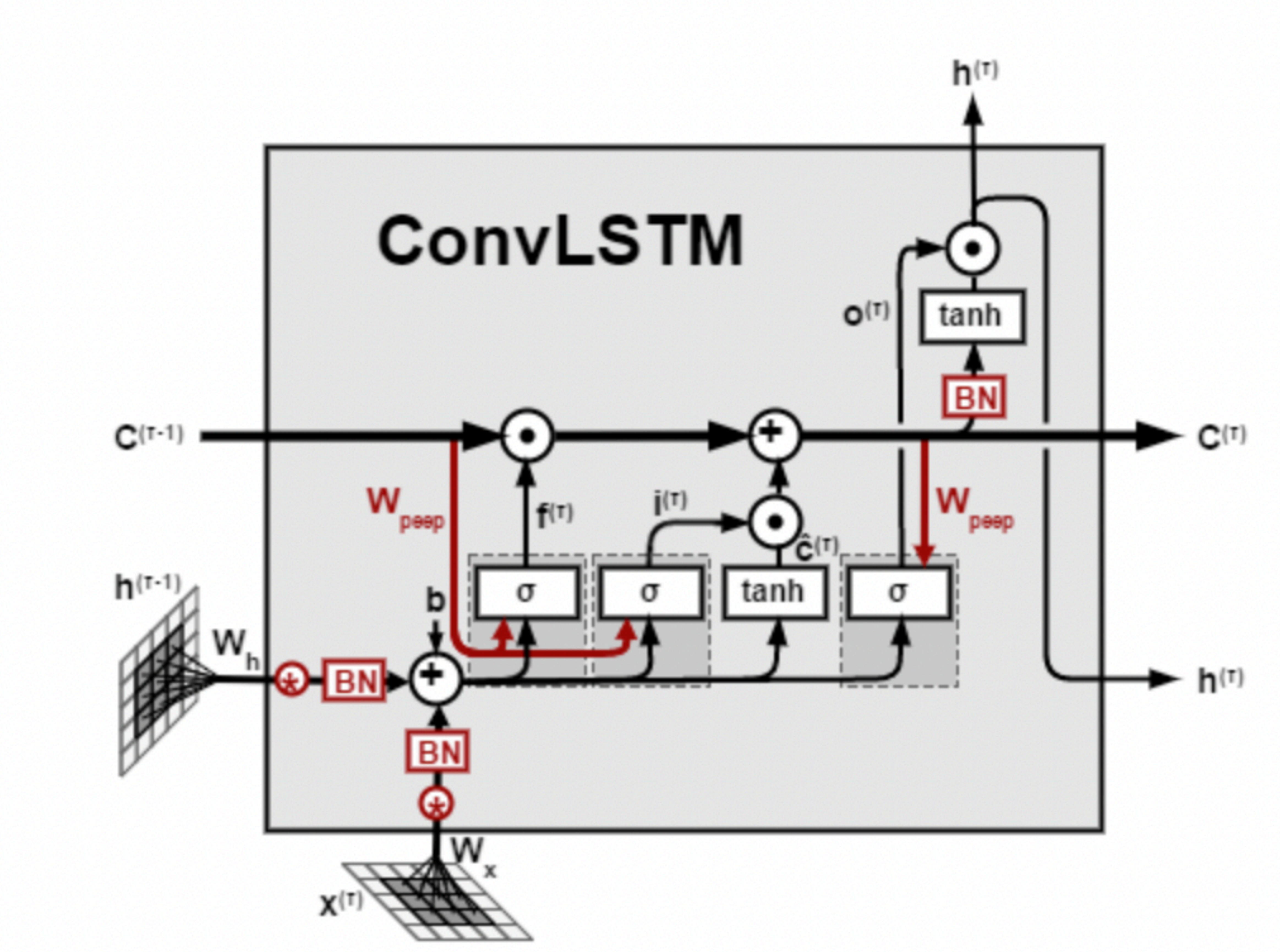
- Numerous network attacks can be easily carried out and practically deployed on the CAN bus.

- These attacks seem to be legitimate-looking traffic sequences.

Problem Solving

-> This paper propose **CANTransfer**, intrusion detection method on Controller Area Network using Transfer Learning based on Convolutional **LSTM model !**

ConvLSTM (Covolutional LSTM)



1. **Image feature vector** is input of **LSTM**
   - **Extract the image vector** with **CNN**
2. Convolution is input the **LSTM internal operation**

Preprocessing

Table 1: Sample CAN packets, showing different ID and DLC fields

Timestamp	ID	DLC	Data
1479246664.162438	0153	8	00 21 10 ff 00 ff 00 00
1479246664.164967	02b0	5	bd ff 00 07 dc
1479246664.165822	043f	8	00 40 60 ff 58 28 08 00
1479246664.171065	05f0	2	f4 00
1479246664.171695	0002	8	00 00 00 00 00 06 01 a2

→

Table 2: Representation of the dataset after preprocessing.Labels are normal (0) and intrusion (1).

Time Diff.	ID	DLC	Data								*Label*
			D1	D2	D3	D4	D5	D6	D7	D8	
0.6	497	8	0	0.501	0.062	1	0	1	0.815	0.368	**0**
0.4	544	8	0	0	0	0	0	0	0	0	**0**
0.6	848	8	0.874	0.011	0.996	0.011	0.035	0	0.227	0.062	**0**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.2	399	8	0.019	0.141	0.368	0.054	0.141	0.101	0	0.498	**1**
0.1	128	8	0	0.109	0.141	0	0	0.356	0	0.062	**1**

- The original 4-feature dataset is splited and transformed into an 11 features dataset.

- The ‘Time Diff.’ column is the time difference between two consecutive timestamps.

Time series Transformation

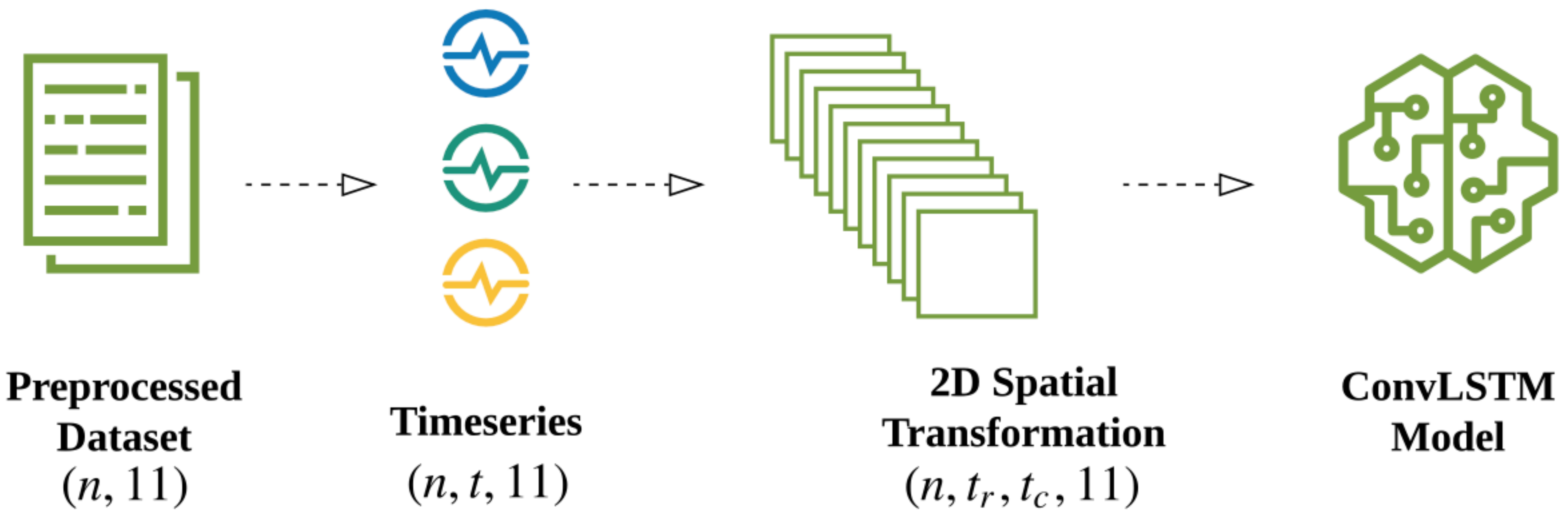


Figure 2: Dataset is transformed to be processed with the Convolutional LSTM model.

- **It is transformed the original dataset and constructed a multivariate time series.**
  - **A multivariate time series data means It has multiple observations for each time-step.**
- **This paper split into samples, maintaining the order of observations across the 11 feature's input sequences.**

Time series Transformation

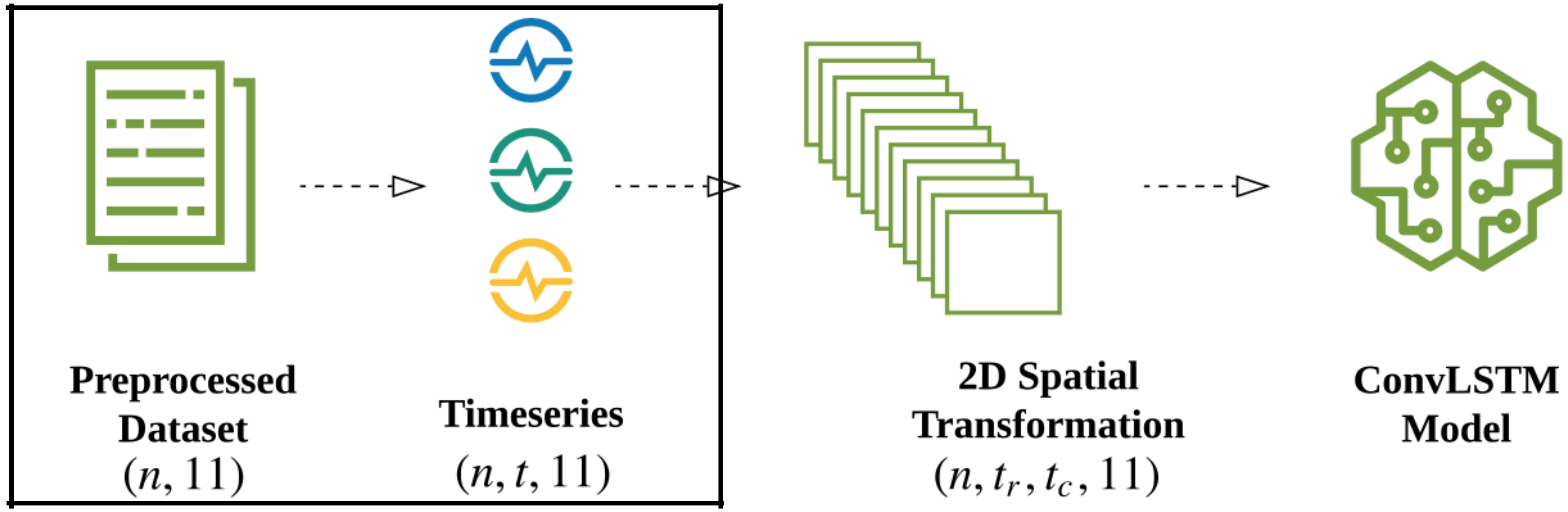


Figure 2: Dataset is transformed to be processed with the Convolutional LSTM model.

- Choose input -> t
- Sample shape -> (t,11)
- n Sample -> time series (n, t, 11)

2D spatial Transformation

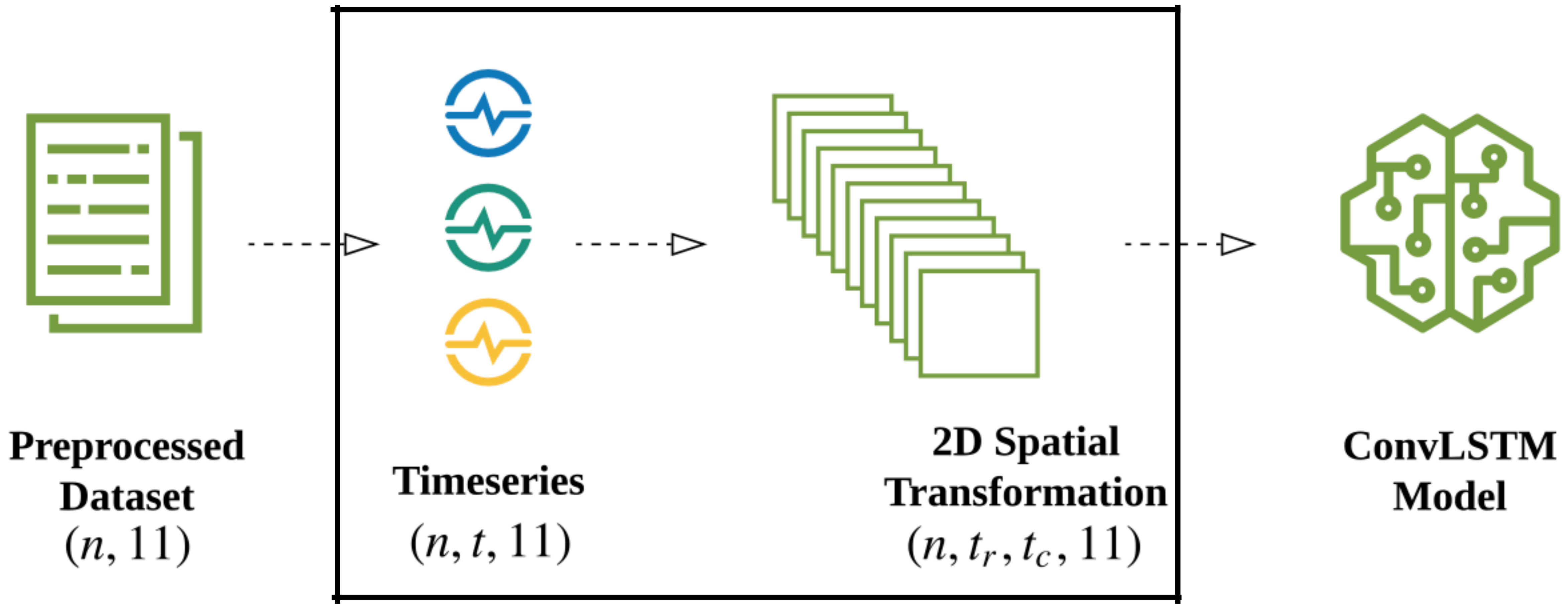


Figure 2: Dataset is transformed to be processed with the Convolutional LSTM model.

- Time stamp t
  -two-dimensional multivariate time series $< t_r, t_c >$

- t_r, t_c have equal size.

Convolutional LSTM

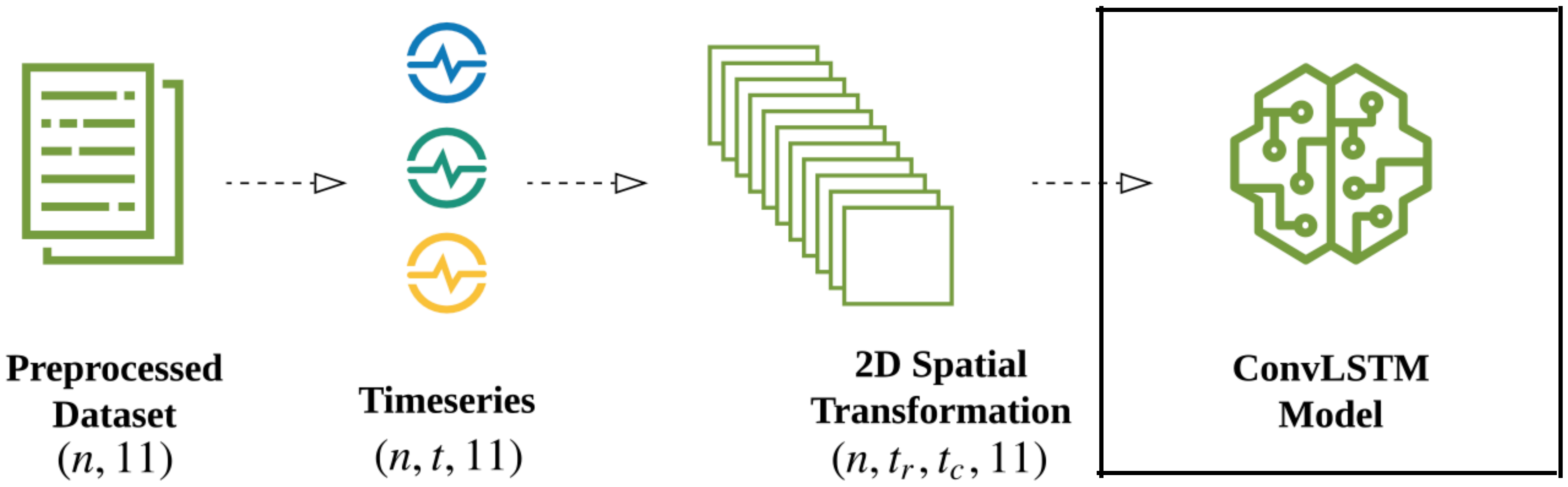


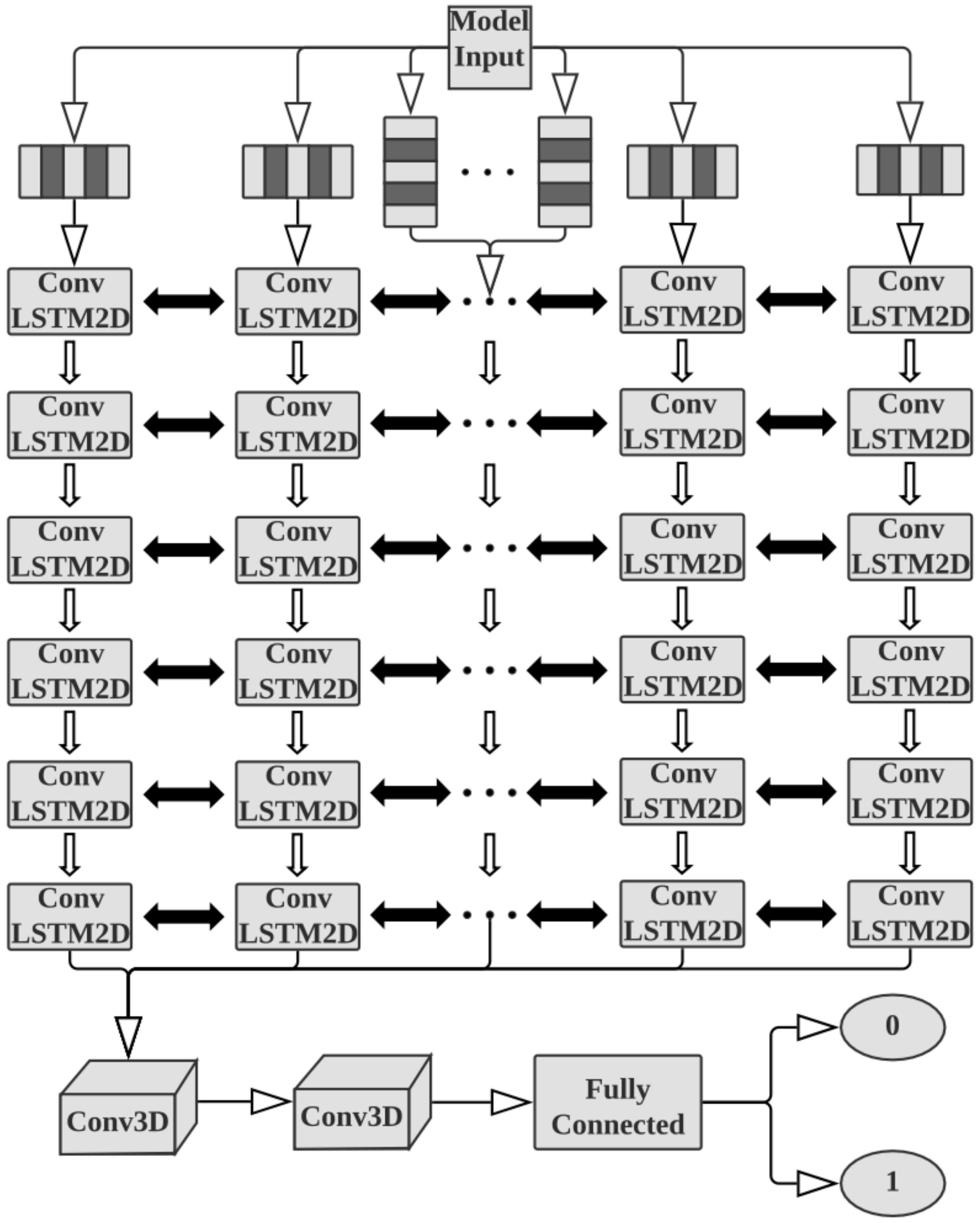
Figure 2: Dataset is transformed to be processed with the Convolutional LSTM model.

- This paper proposes **multivariate ConvLSTM-based** intrusion detection model

  \- **CANTransfer** model is to predict between normal and abnormal sequences effectively via **transfer learning.**

CANTransfer

- Multivariate Prediction



$$T(f_i) = \{f_i(t_1), f_i(t_2), f_i(t_3), \ldots, f_i(t_n)\}$$

- X : Two-dimensional spatial-temporal data

- contain t (historical values of each feature f)

$$T(f_{1,\ldots,n}) = \{T(f_1), T(f_2).T(f_3), \ldots, T(f_n)\}$$

- f (tj) : jth step in the time series T(fi)

CANTransfer

- Multivariate Prediction

$$X = \left\{ \begin{matrix} \begin{bmatrix} f_1(t_{r_1,c_1}) & \dots & f_1(t_{r_1,c_n}) \\ \vdots & \vdots & \vdots \\ f_1(t_{r_n,c_1}) & \dots & f_2(t_{r_n,c_n}) \end{bmatrix} \\ \begin{bmatrix} f_2(t_{r_1,c_1}) & \dots & f_2(t_{r_1,c_n}) \\ \vdots & \vdots & \vdots \\ f_2(t_{r_n,c_1}) & \dots & f_2(t_{r_n,c_n}) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} f_{11}(t_{r_1,c_1}) & \dots & f_{11}(t_{r_1,c_n}) \\ \vdots & \vdots & \vdots \\ f_{11}(t_{r_n,c_1}) & \dots & f_{11}(t_{r_n,c_n}) \end{bmatrix} \end{matrix} \right\}, \ y = \left\{ \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \right\}$$

$$T\left(f_{1,\dots,n}\right) = \{T\left(f_1\right), T\left(f_2\right).T\left(f_3\right), \dots, T\left(f_n\right)\}$$

- X : timeseries for all 11 features can be written
- X is the input for CAN-Transfer model

Data Preprocessing

Table 2: Representation of the dataset after preprocessing.Labels are normal (0) and intrusion (1).

Time Diff.	ID	DLC	Data								*Label*
			D1	D2	D3	D4	D5	D6	D7	D8	
0.6	497	8	0	0.501	0.062	1	0	1	0.815	0.368	**0**
0.4	544	8	0	0	0	0	0	0	0	0	**0**
0.6	848	8	0.874	0.011	0.996	0.011	0.035	0	0.227	0.062	**0**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.2	399	8	0.019	0.141	0.368	0.054	0.141	0.101	0	0.498	**1**
0.1	128	8	0	0.109	0.141	0	0	0.356	0	0.062	**1**

Dataset

- Performed preprocessing and normalization. (11 features dataset)
- Converted timestamp into time difference. (time interval between the current and previous packet)
- Labeled to timestep t.

Data Preprocessing

Table 2: Representation of the dataset after preprocessing.Labels are normal (0) and intrusion (1).

Time Diff.	ID	DLC	Data								*Label*
			D1	D2	D3	D4	D5	D6	D7	D8	
0.6	497	8	0	0.501	0.062	1	0	1	0.815	0.368	**0**
0.4	544	8	0	0	0	0	0	0	0	0	**0**
0.6	848	8	0.874	0.011	0.996	0.011	0.035	0	0.227	0.062	**0**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.2	399	8	0.019	0.141	0.368	0.054	0.141	0.101	0	0.498	**1**
0.1	128	8	0	0.109	0.141	0	0	0.356	0	0.062	**1**

Dataset

- Train Dataset : 70% (Car1 : 2,887,500 / Car2 : 2,625,554)
- Validation Dataset : 25% (Car1 : 1,031,250 / Car2 : 937,698)
- Test Dataset : 5% (Car1 : 206,250 / Car2 : 187,539)

Experimental Setup

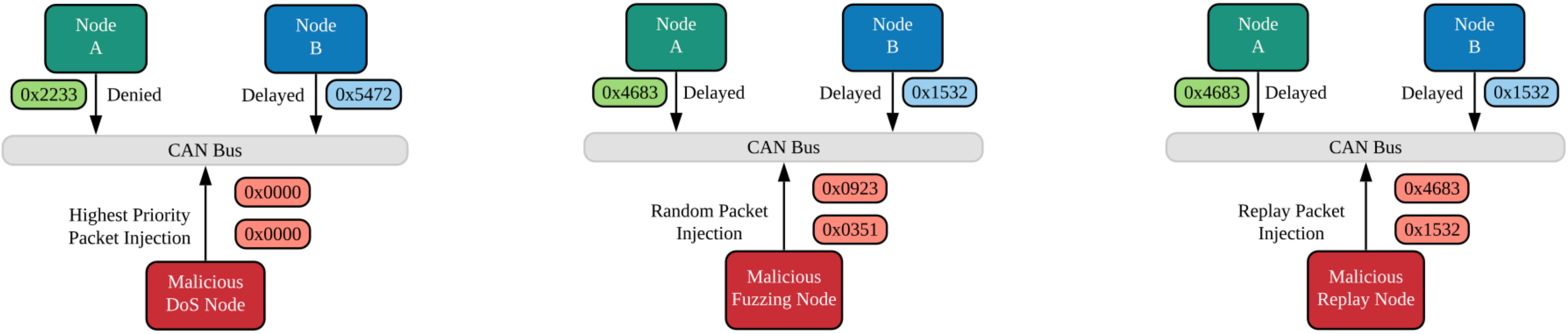
Baseline Model

- Threshold-based Models
  - Detect attacks using the offset ratio and time interval between CAN messages. (OTIDS)

- Classification-based models
  - Use One-Class SVM as a One-Class classification method and Isolation Forest. (OCSVM)

- Ensemble-based model
  - Use Heuristics based approach. (RNN + Heuristics)

Experiment



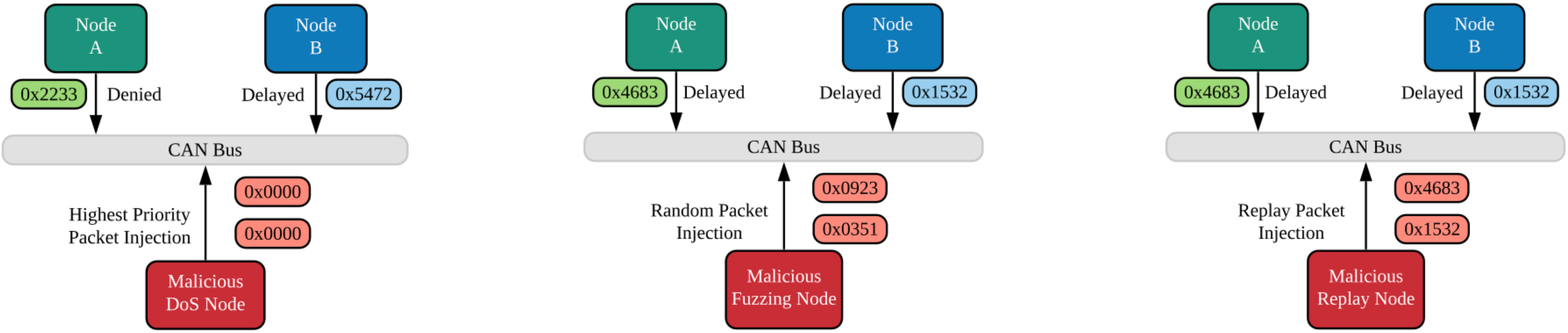
a) DoS Attack **b) Fuzzy Attack** **c) Replay Attack**

a) **DoS Attack** : High Priority Packet Injection
   - DoS attack is the most common intrusion and relatively straightforward for the model

b) **Fuzzy Attack** : Random Packet Injection

c) **Replay Attack** : Replay Packet Injection

Experiment



a) DoS Attack **b) Fuzzy Attack** **c) Replay Attack**

1) Conv-LSTM based CANTransfer model is trained with normal (class 0) & known intrusion (class 1)

2) Transfer learning is performed by training with only one instance of a new intrusion **(one-shot learning)**

Result

(RQ1) Intrusion detection performance

Table 3: Performance comparison of *CANTransfer* with the baseline methods. The best performer from baseline methods for each metrics is underlined and overall best performers are shown in bold.

Method	**Precision (%)**		**Recall (%)**		**F1-Score (%)**	
	Known	*New*	*Known*	*New*	*Known*	*New*
OCSVM	35.43	10.83	71.15	35.12	47.30	16.55
IF	43.58	15.62	73.42	31.56	54.69	20.90
OTIDS	**99.82**	70.81	71.68	42.01	83.44	52.73
RNN+Heuristics	98.69	70.25	**99.49**	50.53	**99.09**	58.78
CANTransfer	94.93	**87.97**	95.57	**88.97**	95.25	**88.47**
Gain	-4.89	**17.16**	-3.92	**38.44**	-3.84	**26.69**

Result

(RQ2) Importance of preprocessing

1. It didn’t perform any **preprocessing or spatial transformation** on the raw dataset and allowed to model to train on raw data.

2. This paper preprocessed the dataset and the applied **2D spatial transformation.**

-> The results are different in favor of the second method **by 72.54%** in gain on F1-score.

Result

(RQ3) Transfer Learning

Table 3: Performance comparison of *CANTransfer* with the baseline methods. The best performer from baseline methods for each metrics is underlined and overall best performers are shown in bold.

Method	Precision (%)		Recall (%)		F1-Score (%)	
	Known	*New*	*Known*	*New*	*Known*	*New*
OCSVM	35.43	10.83	71.15	35.12	47.30	16.55
IF	43.58	15.62	73.42	31.56	54.69	20.90
OTIDS	**<u>99.82</u>**	<u>70.81</u>	71.68	42.01	83.44	52.73
RNN+Heuristics	98.69	70.25	**<u>99.49</u>**	<u>50.53</u>	**<u>99.09</u>**	<u>58.78</u>
CANTransfer	94.93	**87.97**	95.57	**88.97**	95.25	**88.47**
Gain	-4.89	**17.16**	-3.92	**38.44**	-3.84	**26.69**

Table 4: Test result comparison of known vs. new intrusion with and without using Transfer Learning in *CANTransfer*, where class 0 is normal and class 1 is new attack.

		Known Intrusion			New Intrusion		
Zero-shot Learning	**Class**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
	0	99.0%	99.0%	99.0%	68.0%	100.0%	81.0%
	1	99.0%	99.0%	99.0%	67.0%	0.00%	1.00%
One-shot Learning	**Class**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
	0	99.0%	91.0%	95.0%	92.0%	91.0%	91.0%
	1	90.0%	99.0%	94.0%	81.0%	83.0%	82.0%
Gain	**Class 0**	0.00%	-8.00%	-4.00%	24.0%	-9.00%	10.0%
	Class 1	-9.00%	0.00%	-5.00%	14.0%	83.0%	81.0%

- F1-score of **CANTransfer** increased 26.69% after applying **Transfer learning.**
  - Precision and recall in one-shot learning increased to **81.0%, 83.0%**

Result

(RQ4) Improvement from zero-shot

- One-shot learning : F1-Score (91.0%) / Class 1 (82.0%)
- Zero-shot learning : F1-Score (81.0%) / Class 0(1.00%)

(RQ5) False-positive

In the case of new intrusion, the precision

- **One-shot learning : Class 0: 92.0% / Class 1: 81.0%**
- Zero-shot learning : Class 0: 68.0% / Class 1: 67.0%

(RQ6) Processing Time

- It is important to **minimize processing time** in a **real-time system.**
- Small **timestep, t** was experimented in several sizes.
- When **t is 16**, **CANTransfer** produces the **best performance** results.

Result

(RQ7) Loss and Accuracy

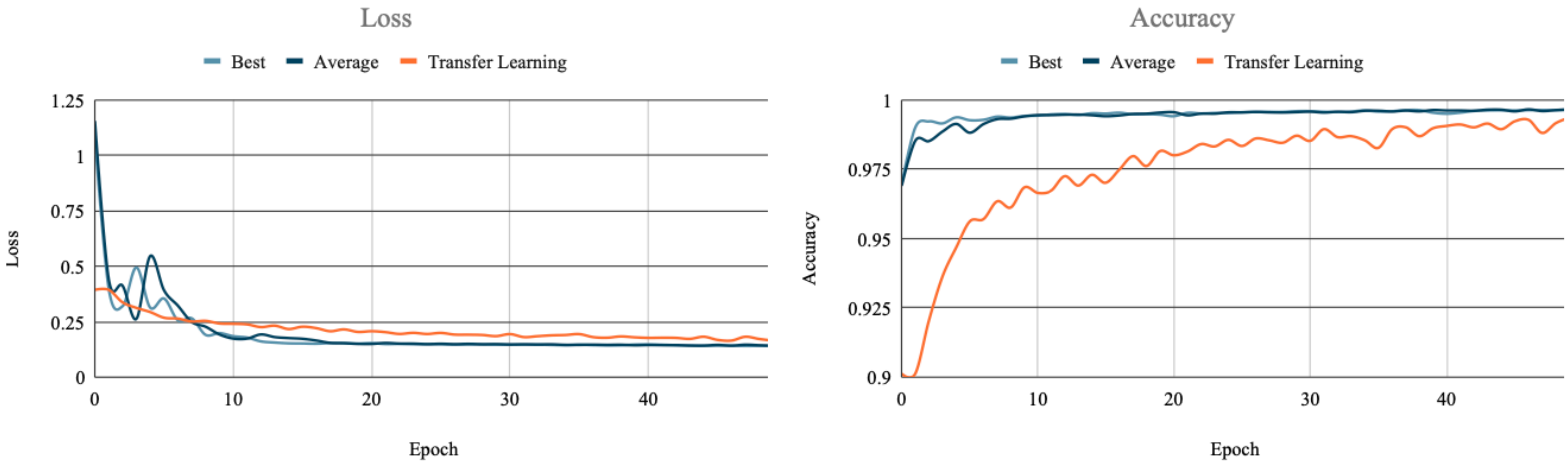


Figure 6: Loss (left) and Accuracy (right) of *CANTransfer* during training.

Conclusion

- During vehicle operation, **intrusion attacks** will cause **vehicle-related accidents** and have a serious impact.
- Train for **new intrusion detection** is **difficult** until a large amount of data is obtained.
- **CAN Transfer** was proposed for a new intrusion attack **using one-shot transfer learning.**