Supplementary Material

No Author Given

No Institute Given

A Detection F1 scores

This table presents the complete detection F1 scores for various attacks and detection approaches presented in the evaluation section of the main paper.

Table 1: Intrusion Detection F1 Scores (mean and standard deviation) for Different Attacks with Different Detection Approaches

Attack	**Defense**	**Mean**	**Std.**
Average	D1: LSTM	0.73	0.00
	D2: CNN	0.71	0.00
	D3: AE	0.69	0.00
	DAD	**0.86**	0.00
Average (adv. masked)	D1: LSTM	0.40	0.00
	D2: CNN	0.26	0.00
	D3: AE	0.32	0.00
	DAD	**0.78**	0.00
m-FDI	D1: LSTM	0.73	0.04
	D2: CNN	0.66	0.06
	D3: AE	0.66	0.03
	DAD	**0.77**	0.05
m-FDI (adv. masked)	D1: LSTM	0.49	0.03
	D2: CNN	0.08	0.08
	D3: AE	0.00	0.00
	DAD	**0.78**	0.04
v-FDI-Acce.	D1: LSTM	0.81	0.02
	D2: CNN	**0.84**	0.03
	D3: AE	**0.84**	0.01
	DAD	0.82	0.03
v-FDI-Acce. (adv. masked)	D1: LSTM	0.54	0.06
	D2: CNN	0.85	0.02
	D3: AE	0.55	0.07
	DAD	**0.87**	0.03
v-FDI-Speed	D1: LSTM	0.88	0.01
	D2: CNN	0.81	0.01
	D3: AE	**0.92**	0.01
	DAD	**0.92**	0.01

v-FDI-Speed (adv. masked)	D1: LSTM	0.78	0.01
	D2: CNN	0.35	0.06
	D3: AE	0.70	0.01
	DAD	**0.83**	0.01
v-FDI-Loc.	D1: LSTM	0.78	0.02
	D2: CNN	0.80	0.03
	D3: AE	0.80	0.02
	DAD	**0.85**	0.03
v-FDI-Loc. (adv. masked)	D1: LSTM	0.59	0.01
	D2: CNN	0.00	0.00
	D3: AE	0.52	0.02
	DAD	**0.86**	0.03
v-FDI-Acce.Speed	D1: LSTM	**0.89**	0.01
	D2: CNN	0.88	0.01
	D3: AE	0.87	0.01
	DAD	**0.89**	0.01
v-FDI-Acce.Speed (adv. masked)	D1: LSTM	0.82	0.02
	D2: CNN	0.81	0.01
	D3: AE	0.80	0.01
	DAD	**0.85**	0.01
v-FDI-Acce.Loc.	D1: LSTM	0.78	0.01
	D2: CNN	0.75	0.03
	D3: AE	0.77	0.03
	DAD	**0.89**	0.04
v-FDI-Acce.Loc. (adv. masked)	D1: LSTM	0.00	0.00
	D2: CNN	0.00	0.00
	D3: AE	0.00	0.00
	DAD	**0.86**	0.08
v-FDI-Speed.Loc.	D1: LSTM	0.33	0.04
	D2: CNN	0.47	0.12
	D3: AE	0.34	0.05
	DAD	**0.80**	0.03
v-FDI-Speed.Loc. (adv. masked)	D1: LSTM	0.00	0.01
	D2: CNN	0.00	0.00
	D3: AE	0.00	0.00
	DAD	**0.75**	0.05
v-FDI-Acce.SpeedLoc.	D1: LSTM	0.68	0.03
	D2: CNN	0.44	0.06
	D3: AE	0.30	0.04
	DAD	**0.90**	0.06
v-FDI-Acce.SpeedLoc. (adv. masked)	D1: LSTM	0.00	0.00
	D2: CNN	0.00	0.00
	D3: AE	0.00	0.00
	DAD	**0.90**	0.06

B Derivations of Physical Consistency Checker

Vehicle's relative distance and speed can be computed directly based on the measurements or indirectly based on the kinematic model. The detailed derivations are as follows.

B.1 Distance Checker

$$\Delta\epsilon_{i,i-1}(t) = \epsilon_{i,i-1}(t) - \epsilon_{i,i-1}(t-1) \tag{1}$$

$$\begin{aligned}
\Delta\tilde{\epsilon}_{i,i-1}(t) &= x_i(t) - x_{i-1}(t) - (x_i(t-1) - x_{i-1}(t-1)) \\
&= (x_i(t) - x_i(t-1)) - (x_{i-1}(t) - x_{i-1}(t-1)) \\
&= \Delta x_i(t) - \Delta x_{i-1}(t-1) \\
&= (v_i(t-1) - v_{i-1}(t-1)) \cdot \Delta t \\
&+ \frac{1}{2}\Delta t^2 \left(a_i(t-1) - a_{i-1}(t-1)\right) \\
&= \dot{\epsilon}_{i,i-1} \cdot \Delta t + \frac{1}{2}\Delta t^2 \left(a_i(t-1) - a_{i-1}(t-1)\right)
\end{aligned} \tag{2}$$

B.2 Speed Checker

$$\Delta\dot{\epsilon}_{i,i-1}(t) = \dot{\epsilon}_{i,i-1}(t) - \dot{\epsilon}_i(t-1) \tag{3}$$

$$\begin{aligned}
\Delta\tilde{\dot{\epsilon}}_{i,i-1}(t) &= v_i(t) - v_{i-1}(t) - (v_i(t-1) - v_{i-1}(t-1)) \\
&= v_i(t) - v_i(t-1) - (v_{i-1}(t) - v_{i-1}(t-1)) \\
&= \Delta v_i(t) - \Delta v_{i-1}(t) \\
&= (a_i(t-1) - a_{i-1}(t-1))\, \Delta t
\end{aligned} \tag{4}$$

C Data Preparation

Three scenarios are considered to generate training data:

1. normal driving scenario with different traffic conditions;
2. repeating the process of accelerating from 0 to 70/90 km/h then decelerating to 0 km/h;
3. randomly changing speed between 70-90 km/h.

D Implementation Details of Baseline Defense Methods

The baseline LSTM predictor (D1:LSTM) consists of three stacked LSTM layer with 100 units each followed by followed by a dropout layer (rate=0.3). The last dropout layer is connected with two fully connected layers with 50 and 1 hidden units respectively. The CNN predictor (D2:CNN) consists of three stacked 1D convolutional layers with 512, 256, 128 neurons respectively and followed by a 1D max pooling layer respectively. The last pooling layer is connected with two fully connected layers with 50 and 1 hidden units respectively. The autoencoder (D3:AE) consists of two stacked convolutional layers (each with 256 and 128 neurons respectively) as the encoder and two transposed convolutional layers (with 128 and 256 neurons) followed by the same fully connected layers for time-series prediction. In terms of adversarial training, all of the adversarial examples with the same ϵ are used to fine-tune the trained LSTM model. Both the learning rate and the number of epochs are reduced as one tenth of the original training parameters.

E Basic Iterative Method

Let $\boldsymbol{x}$ denote the clean input, y_{true} denote its corresponding ground-truth output and y_{target} denote the attacker's targeted output value. Given a trained prediction model $f(\boldsymbol{x})$, the goal of an *untargeted* attack is to find adversarial examples $\boldsymbol{x}^{adv}$ by solving the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{x}^{adv}} \ & \left\| f(\boldsymbol{x}^{adv}) - y_{true} \right\|_2 \\ s.t. \ & \left\| \boldsymbol{x}^{adv} - \boldsymbol{x} \right\|_\infty < \epsilon \ , \end{aligned} \tag{5}$$

where ϵ defines the attack budget, $\|\cdot\|_2$ and $\|\cdot\|_\infty$ are L_2 and L_∞ norms respectively. In this work, we use these two distance measures but other distance measures, such as L_0 norm, can also be used for different threat models. For a *targeted* attack, the problem becomes:

$$\begin{aligned} \min_{\boldsymbol{x}^{adv}} \ & \left\| f(\boldsymbol{x}^{adv}) - y_{target} \right\|_2 \\ s.t. \ & \left\| \boldsymbol{x}^{adv} - \boldsymbol{x} \right\|_\infty < \epsilon \ . \end{aligned} \tag{6}$$

In general, a deep learning training process aims to minimize a loss function $J(\boldsymbol{x}, y_{true})$. Inspired by the linear behaviour of modern machine learning models, the basic iterative method (BIM) [**?**] uses the first-order information of the loss function and generates adversarial examples iteratively. The update procedure of *untargeted* BIM is as follows:

$$\boldsymbol{x}_{t+1}^{adv} = Clip_x^\epsilon \left\{ \boldsymbol{x}_t^{adv} + \alpha \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^{adv}, y_{true}) \right) \right\} , \tag{7}$$

where $\boldsymbol{x}_0^{adv} = \boldsymbol{x}$, α is the step size, $\operatorname{sign}(\cdot)$ is the sign operator and $Clip_x^\epsilon\{\cdot\}$ clips the adversarial example within the ϵ max-norm ball of the clean input.

Similarly, *targeted* BIM can be described as:

$$\boldsymbol{x}_{t+1}^{adv} = Clip_x^\epsilon \left\{ \boldsymbol{x}_t^{adv} - \alpha \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^{adv}, y_{target}) \right) \right\} . \tag{8}$$