Intelligent Corner Synthesis via Cycle-Consistent Generative Adversarial Networks for Efficient Validation of Autonomous Driving Systems

Handi Yu[1] and Xin Li[1,2]

[1] ECE, Duke University, Durham, NC, USA

[2] iAPSE, Duke Kunshan University, Jiangsu, P. R. China

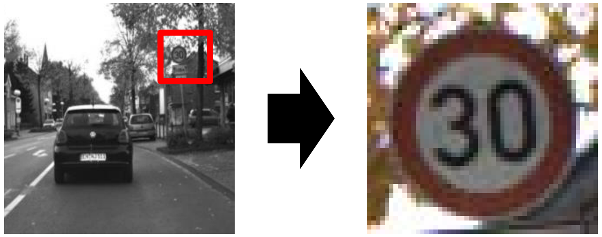
Duke UNIVERSITY

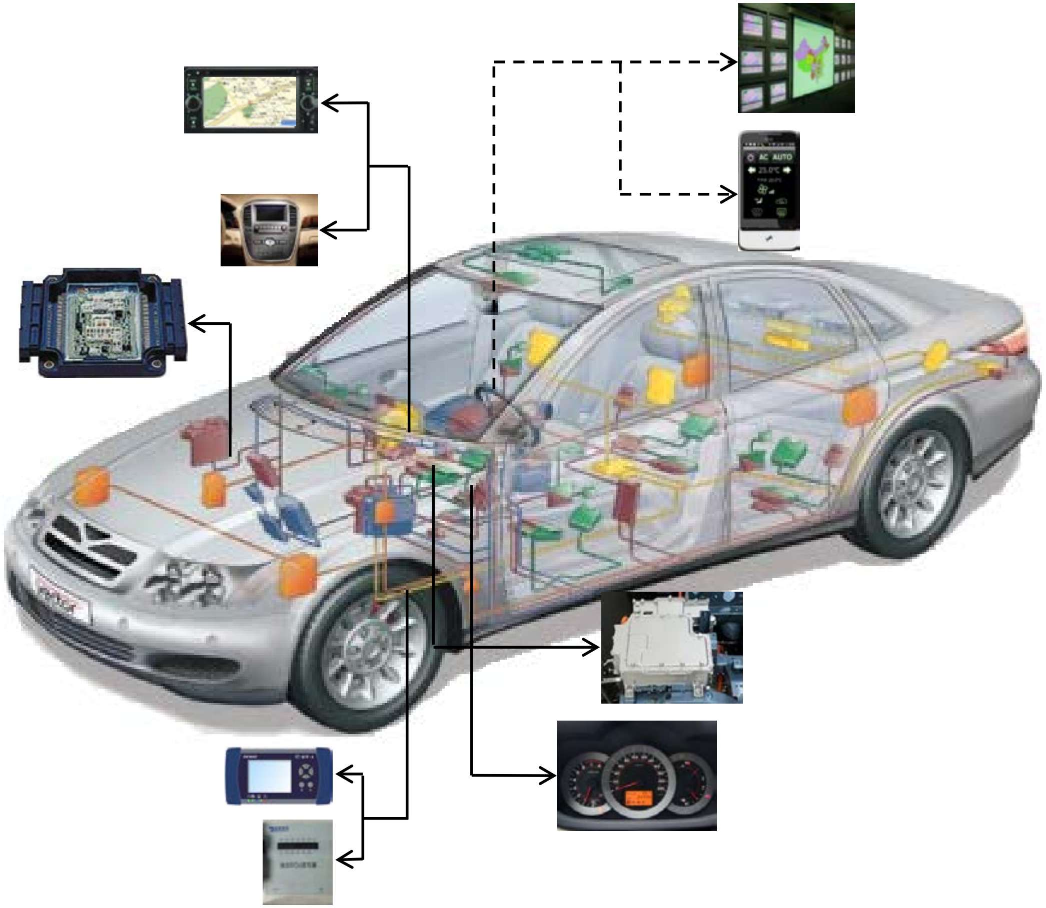
昆山杜克大学 DUKE KUNSHAN UNIVERSITY

Autonomous Driving Systems

- Autonomous driving relies on a large number of machine learning algorithms for perception, planning and control
  - A machine learning algorithm can NEVER be 100% accurate

Example: stop sign detection

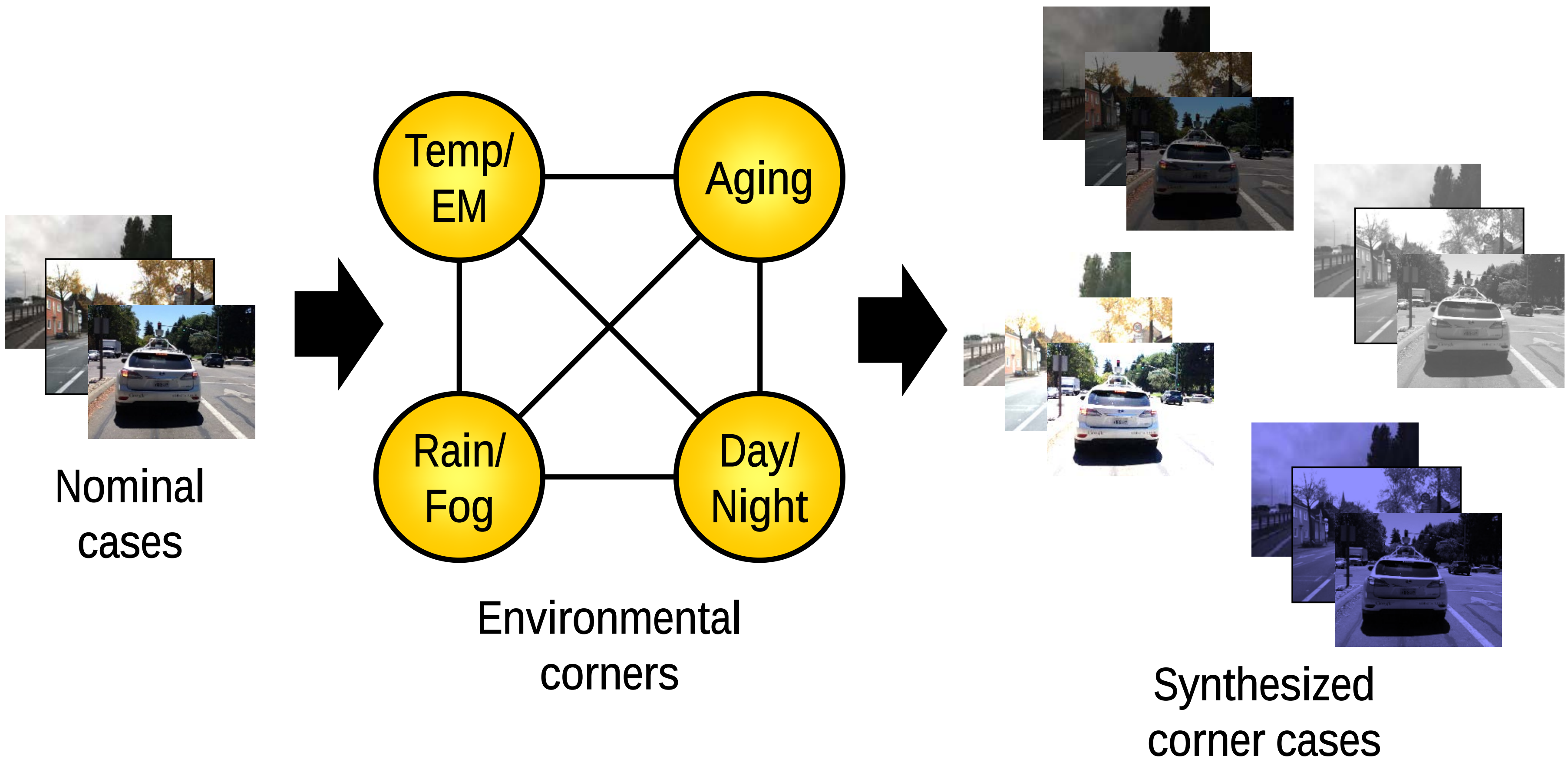




- System validation is necessary over a large set of test cases

Test Case Generation

- Test cases must broadly cover all possible scenarios
  - Extreme corners are difficult or expensive to observe physically
  - Test cases must be artificially synthesized with high accuracy



Test Case Generation

- State-of-the-art methods are task-specific and rely on physical models that may not be highly accurate in practice
  - [Yu 2017]: model and synthesize circuit-level non-idealities
  - [Hospach 2016]: model and synthesize rain drops
  - [Gallen 2015]: model and synthesize fog
  - Etc.
- Proposed work
  - A general generator for corner synthesis is developed by using cycle-consistent generative adversarial network (Cycle-GAN)
  - High-fidelity corner cases are efficiently generated by the proposed Cycle-GAM model

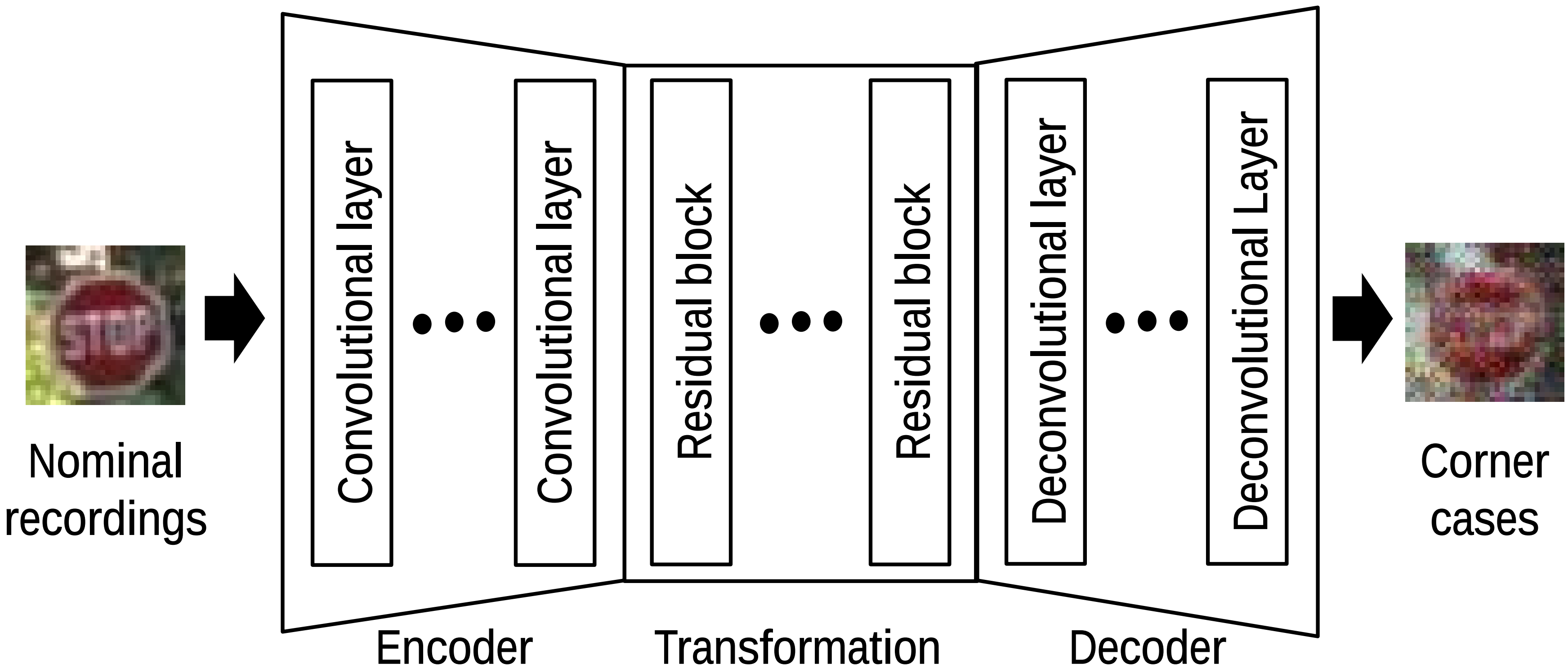
[Yu 2017]: Impact of circuit-level non-idealities on vision-based autonomous driving systems, *ICCAD*, 2017
[Hospach 2016]: Simulation of falling rain for robustness testing of video-based surround sensing systems, *DATE*, 2016
[Gallen 2015]: Nighttime visibility analysis and estimation method in the presence of dense fog, *IEEE Trans. Intell. Transp. Syst.*, 2015

Outline

- Motivation
- Proposed approach
- Experimental results
- Conclusions

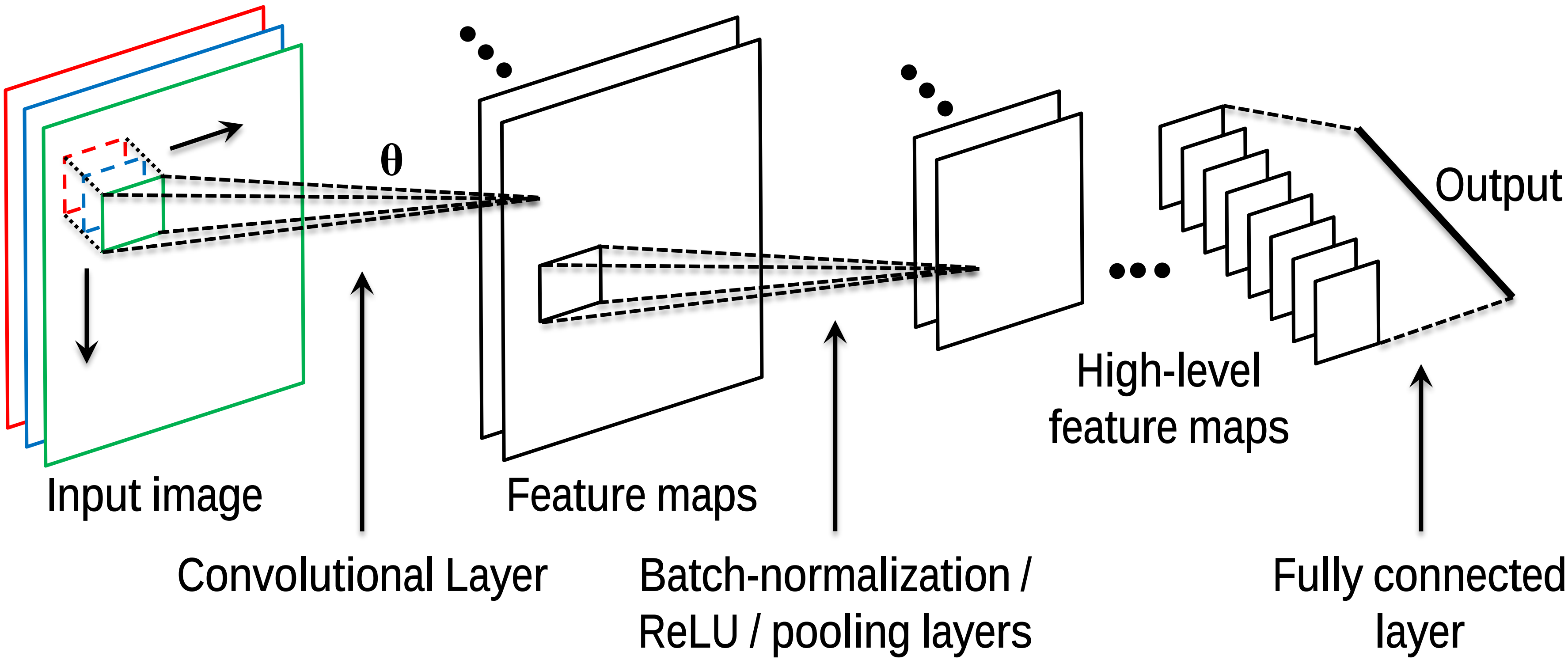
Generator Structure

- A generator synthetically maps nominal recordings to corner cases
  - Encoder: extract features from a given image
  - Transformation: modify extracted features
  - Decoder: generate corner cases from modified features



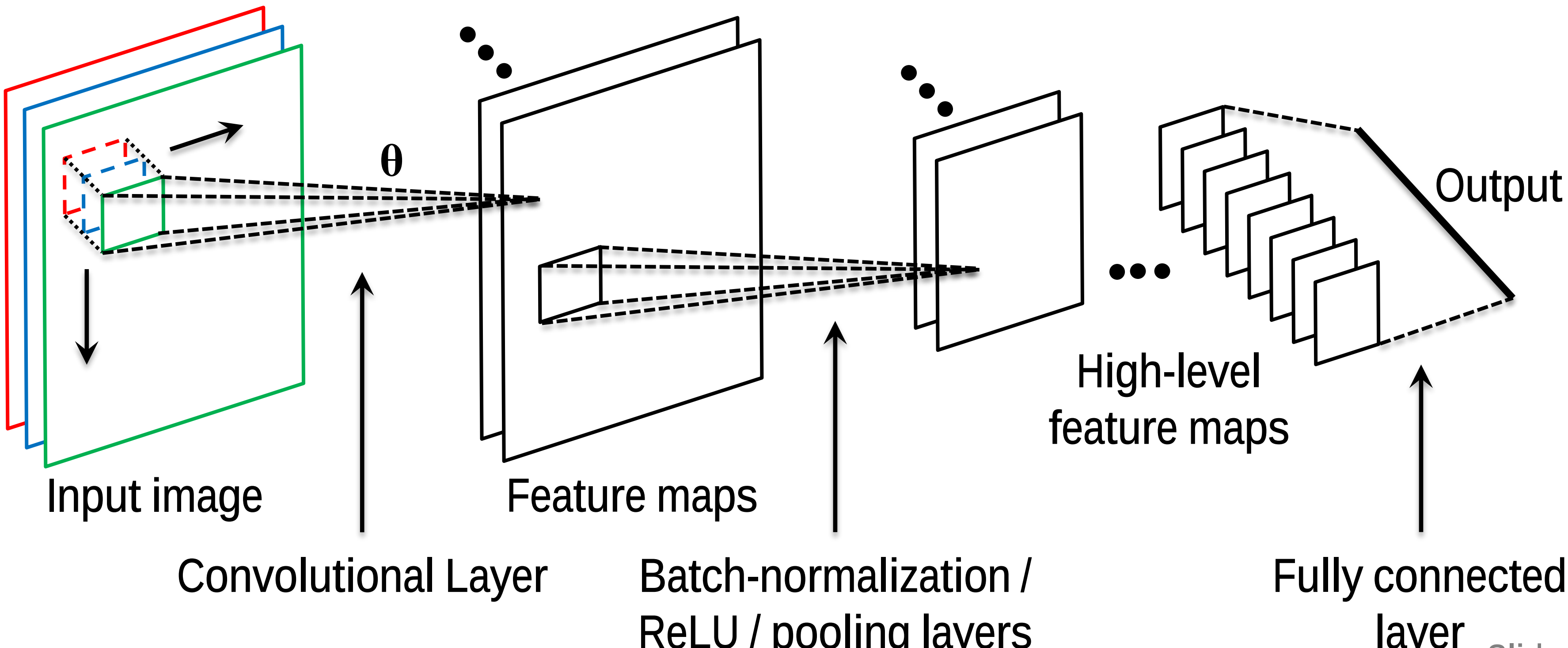
Generator: Encoder

- Encoder is composed of a convolutional neural network (CNN)
- Each convolution: convolute input data with a small-size weight filter θ, resulting in a feature map
  - Extracted features are locally correlated and spatially invariant (consistent with the characteristics of real-world images)



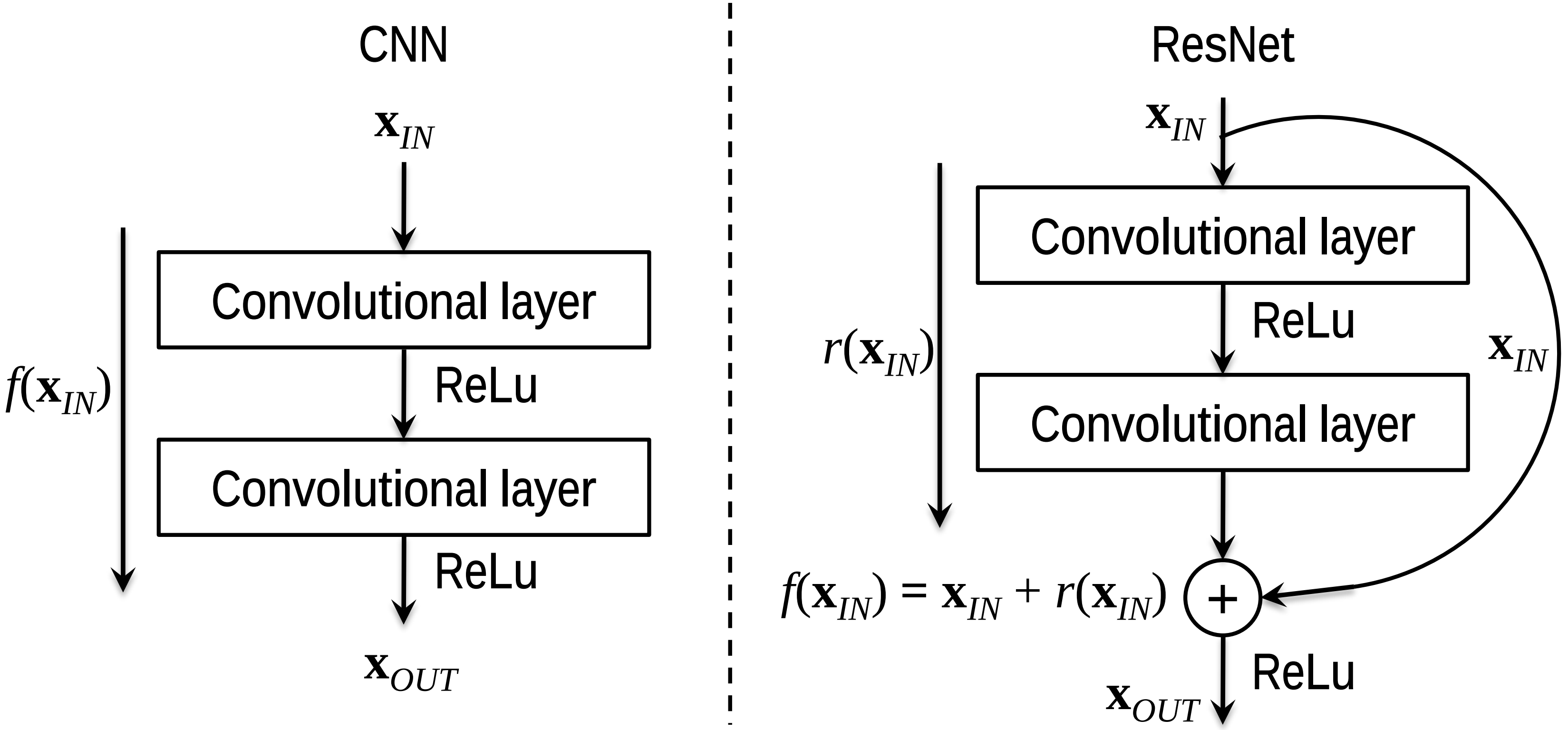
Generator: Encoder

- Batch-normalization layer: improve learning speed and accuracy
- ReLU layer: perform nonlinear transformation
- Pooling layer: reduce dimension
- Fully connected layer: generate classification result
- CNN extracts high-level features for complex objects in an image



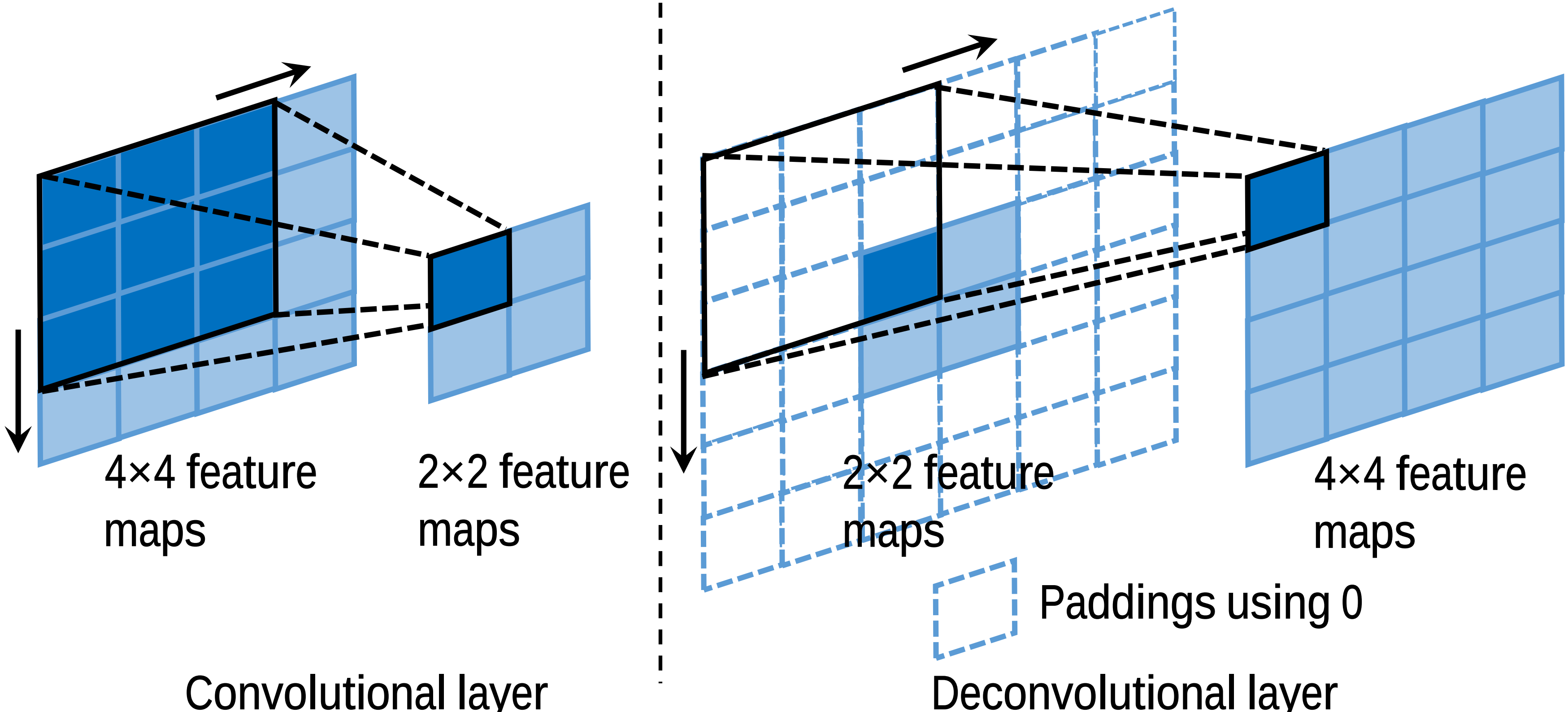
Generator: Transformation

- Residual network (ResNet) is adopted to implement transformation
  - CNN learns a full mapping $f(\mathbf{x}_{IN})$ directly
  - ResNets learns a residual function $r(\mathbf{x}_{IN})$ that is often easy to implementation for transformation



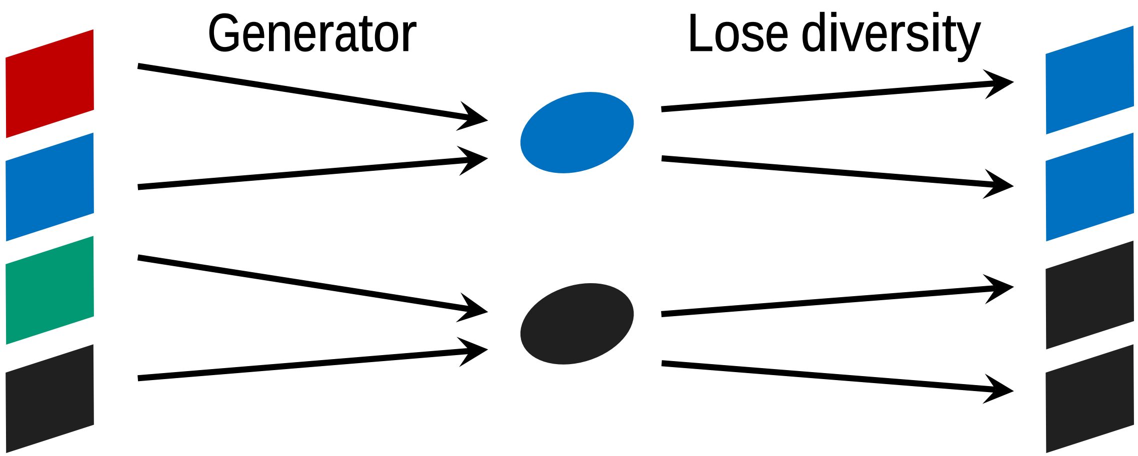
Generator: Decoder

- Deconvolution layers are applied to invert the convolution operations adopted by encoder
  - A deconvolutional layer enhances the resolution of feature maps
  - Modified features are transformed back to an image with its original resolution



Generator Training

- Training a generator is a critical yet challenging task
  - No prior knowledge is known for corner cases
  - Nominal and corner cases are unpaired in training dataset
  - Mode collapse often occurs if a training algorithm is not appropriately designed

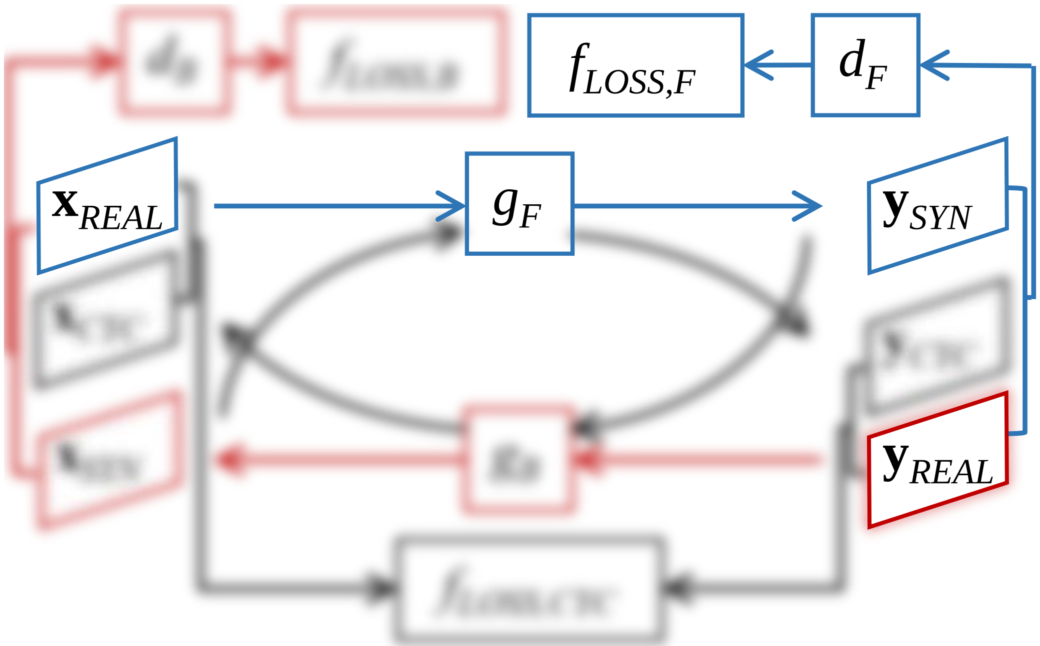


- A robust general-purpose training method is required
  - Cycle-consistent generative adversarial network (CycleGAN)

Tr

Cycle-Consistent Generative Adversarial Network (CycleGAN)

- Forward generator $\boldsymbol{g_F}$ generates synthetic corner case $\mathbf{y}_{SYN}$ from real nominal recording $\mathbf{x}_{REAL}$
- Forward discriminator $\boldsymbol{d_F}$ (implemented with CNN) judges whether a given corner case is "real" or "synthetic"
- $\boldsymbol{d_F}$ learns to be a good judger, while $\boldsymbol{g_F}$ learns to fool $\boldsymbol{d_F}$
  - g_F and d_F compete and improve each other during training



$$\max_{g_F} \min_{d_F} \quad f_{LOSS,F}(g_F, d_F)$$

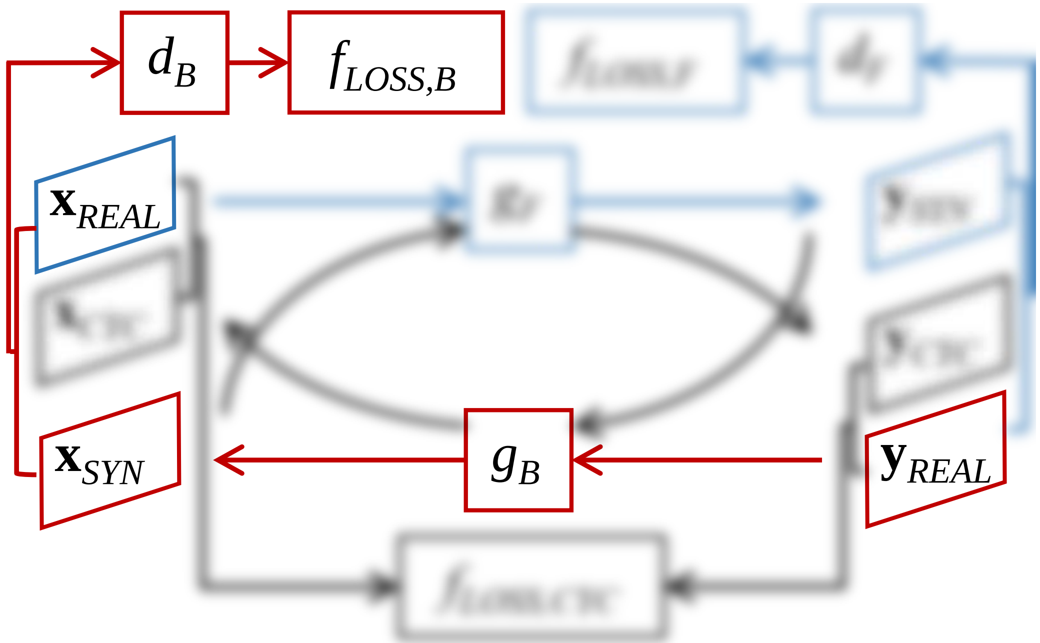
$\boldsymbol{f_{LOSS,F}}$: Forward adversarial loss function

Good $\boldsymbol{g_F} \rightarrow$ large $\boldsymbol{f_{LOSS,F}}$

Good $\boldsymbol{d_F} \rightarrow$ small $\boldsymbol{f_{LOSS,F}}$

Cycle-Consistent Generative Adversarial Network (CycleGAN)

- Backward generator g_B synthesizes nominal image $\mathbf{x}_{SYN}$ from real corner case $\mathbf{y}_{REAL}$
- Backward discriminator d_B judges whether a given nominal image is "real" or "synthetic"
- $\boldsymbol{g_B}$ and $\boldsymbol{d_B}$ compete and improve each other during training



$$\max_{g_B} \min_{d_B} \quad f_{LOSS,B}\left(g_B, d_B\right)$$

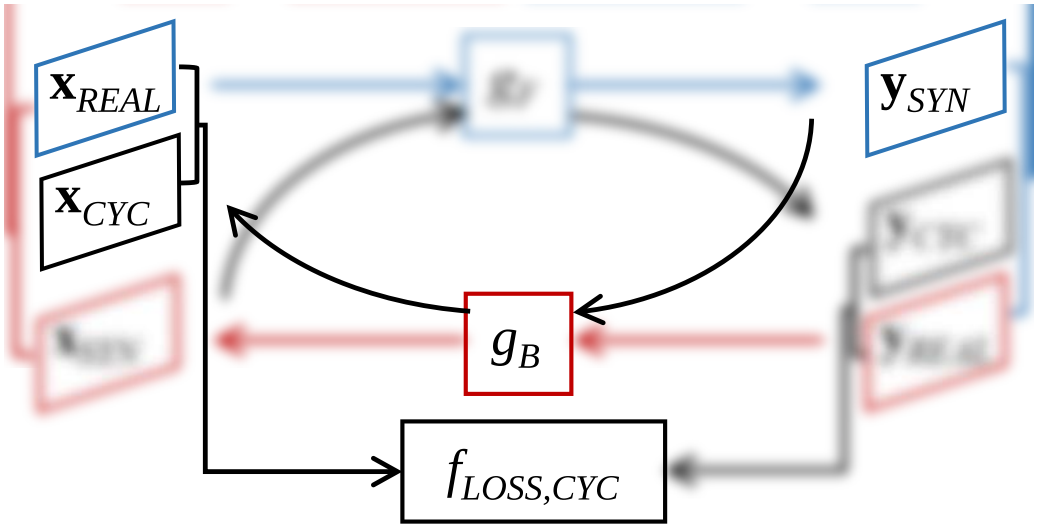
$\boldsymbol{f_{LOSS,B}}$: Backward adversarial loss function

Good $\boldsymbol{g_B} \rightarrow$ large $\boldsymbol{f_{LOSS,B}}$

Good $\boldsymbol{d_B} \rightarrow$ small $\boldsymbol{f_{LOSS,B}}$

Cycle-Consistent Generative Adversarial Network (CycleGAN)

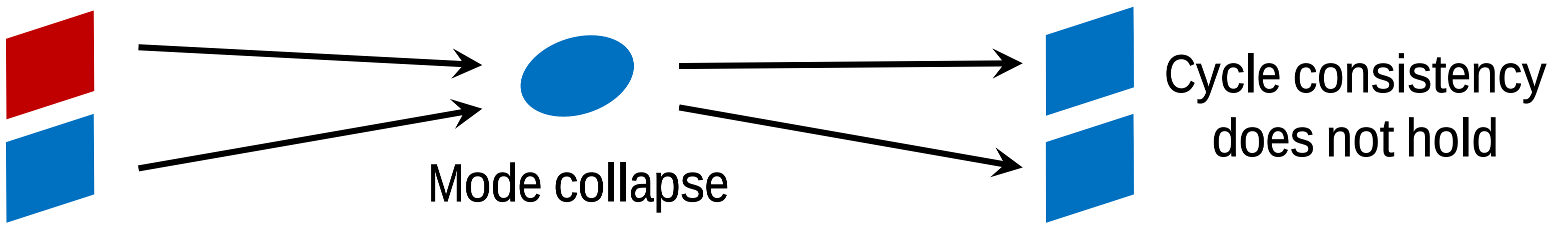
- Cycle-consistency is introduced to avoid mode collapse
  - $\mathbf{y}_{SYN}$ synthesized from $\mathbf{x}_{REAL}$ is mapped back to $\mathbf{x}_{CYC}$ by g_B
  - $\mathbf{x}_{CYC}$ and $\mathbf{x}_{REAL}$ should satisfy cycle consistency: $\mathbf{x}_{CYC} \approx \mathbf{x}_{REAL}$



$$\min_{g_F, g_B} \quad f_{LOSS,CYC}(g_F, g_B)$$

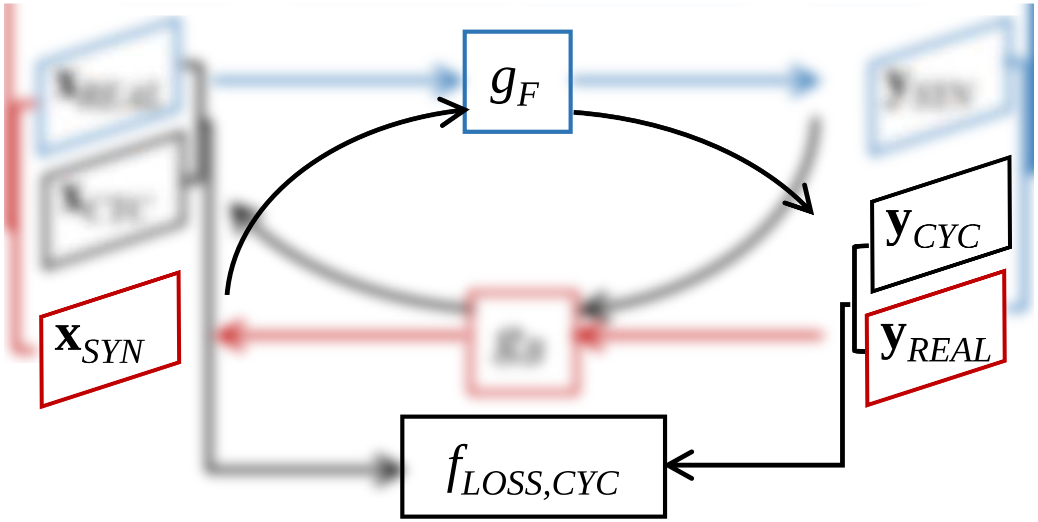
$f_{LOSS,CYC}$: Cycle-consistent loss function

- If cycle consistency holds, mode collapse cannot occur



Cycle-Consistent Generative Adversarial Network (CycleGAN)

- Cycle consistency between $\mathbf{y}_{CYC}$ and $\mathbf{y}_{REAL}$ is similarly defined
  - $\mathbf{x}_{SYN}$ synthesized from $\mathbf{y}_{REAL}$ is mapped back to $\mathbf{y}_{CYC}$ by g_F
  - $\mathbf{y}_{CYC}$ and $\mathbf{y}_{REAL}$ should satisfy cycle consistency: $\mathbf{y}_{CYC} \approx \mathbf{y}_{REAL}$



$$\min_{g_F, g_B} \quad f_{LOSS,CYC}\left(g_B, g_F\right)$$

$f_{LOSS,CYC}$: Cycle-consistent loss function

- A full loss function is formed to train CycleGAN composed of two generators and two discriminators

$$\max_{g_F, g_B} \min_{d_F, d_B} \quad f_{LOSS}\left(g_F, d_F, g_B, d_B\right) = \begin{bmatrix} f_{LOSS,F}\left(g_F, d_F\right) + \\ f_{LOSS,B}\left(g_B, d_B\right) \end{bmatrix} - \underset{\text{Weight}}{\underset{\downarrow}{\lambda}} \cdot \begin{bmatrix} f_{LOSS,CYC}\left(g_F, g_B\right) + \\ f_{LOSS,CYC}\left(g_B, g_F\right) \end{bmatrix}$$

Outline

- Motivation
- Proposed approach
- Experimental results
- Conclusions

Experimental Setup

- Network settings
  - Each generator is composed of 12 CNN and ResNet blocks based on [Zhu 2017]
  - Each discriminator is implemented with a PatchGAN network composed of 4 blocks
- Experimental setup
  - Stop sign detection for autonomous driving
  - Consider nominal dataset from [BelgiumTS], [GTSDB] and [GTSRB] and high temperature corner based on physical modeling
  - A cascade classifier is trained by using nominal data and validated for both nominal and corner cases

[Zhu 2017]: Unpaired image-to-image translation using cycle-consistent adversarial networks, *ICCV*, 2017
[BelgiumTS]: Traffic sign recognition - how far are we from the solution, *IJCNN*, 2013
[GTSDB]: Detection of traffic signs in real-world images: the German traffic sign detection benchmark, *IJCNN*, 2013
[GTSRB]: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, *Neural Networks*, 2012

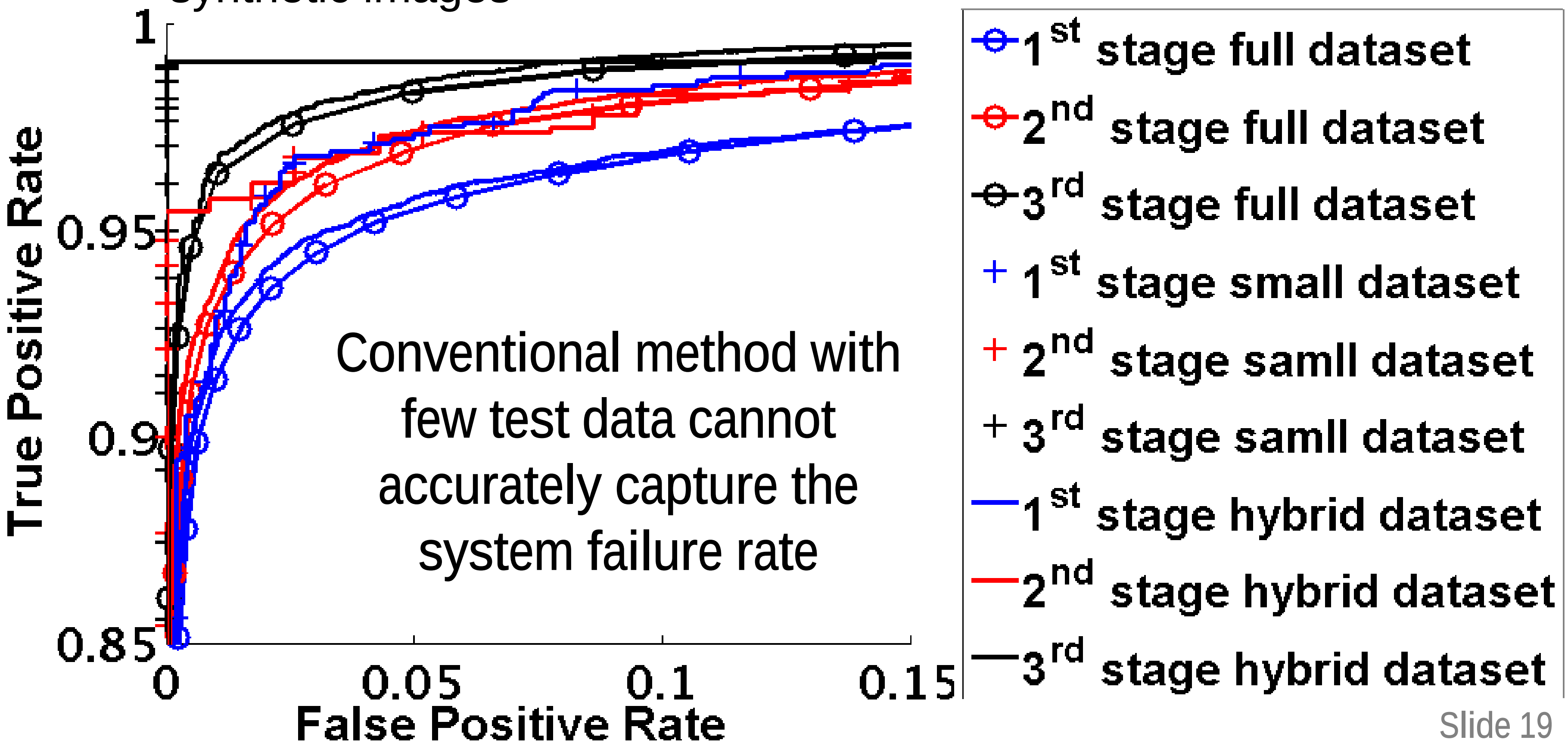
Temperature Variations

- Several examples of actual and synthetic images at high temperature are shown for comparison purposes

$\mathbf{x}_{REAL}$ (room temperature)	$\mathbf{y}_{SYN}$ (high temperature)	$\mathbf{y}_{REAL}$ (high temperature)	$\mathbf{x}_{CYC}$ (room temperature)

Temperature Variations

- STOP sign detection is validated at high temperature
  - Full dataset (golden): 164K high-temperature images
  - Small dataset (conventional): 2K high-temperature images
  - Hybrid dataset (proposed): 2K high-temperature images and 162K synthetic images



Temperature Variations

- Estimation error of false positive rate is reduced by 100× and 1.86× for the first- and second-stage classifiers
- Conventional method with small dataset results in zero false positive rate for the third-stage classifier

Failure rate		1st	2nd	3rd	Total
Full dataset	True positive	97.50%	97.50%	98.50%	93.64%
	False positive	14.67%	6.42%	5.85%	5.51×10^{-4}
Small dataset	True positive	97.50%	97.50%	98.50%	93.64%
	False positive	5.60%	8.57%	0.00	0.00
Hybrid dataset	True positive	97.50%	97.50%	98.50%	93.64%
	False positive	14.76%	5.27%	4.78%	3.72×10^{-4}

Conclusions

- Test cases must broadly cover all possible scenarios for accurate system validation
  - Physically observing extreme corners is difficult and expensive
- Propose a cycle-consistent generative adversarial network (CycleGAN) to generate synthetic test cases
  - Simultaneously train two generators and two discriminators
  - Accurately estimate true positive rate and false positive rate (up to 100× error reduction)
- Future work
  - Synthesize test cases for multiple scenarios