Learning from Computer Simulations to Tackle Real-World Problems

Hoon Kim

Joint work with Kangwook Lee, Gyeongjo Hwang, and Changho Suh

KAIST

About myself

Hoon Kim

Korea Advanced Institute of Science and Technology (KAIST)

- ***M.S.*** in School of Electrical Engineering (2017 Mar. – Current)
- Advisor: Prof. Changho Suh

- ***B.S.*** in School of Electrical Engineering (2012 Feb. – 2017 Feb.)
- Double major Computer Science

- Interested in utilizing simulators to tackle real world problems!

Motivation

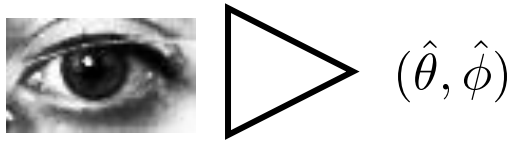
- Deep learning has achieved exceptional performances in many tasks
- Training sets need to be large, diverse, and accurately annotated

1.2 million images / 1000 Object Classes

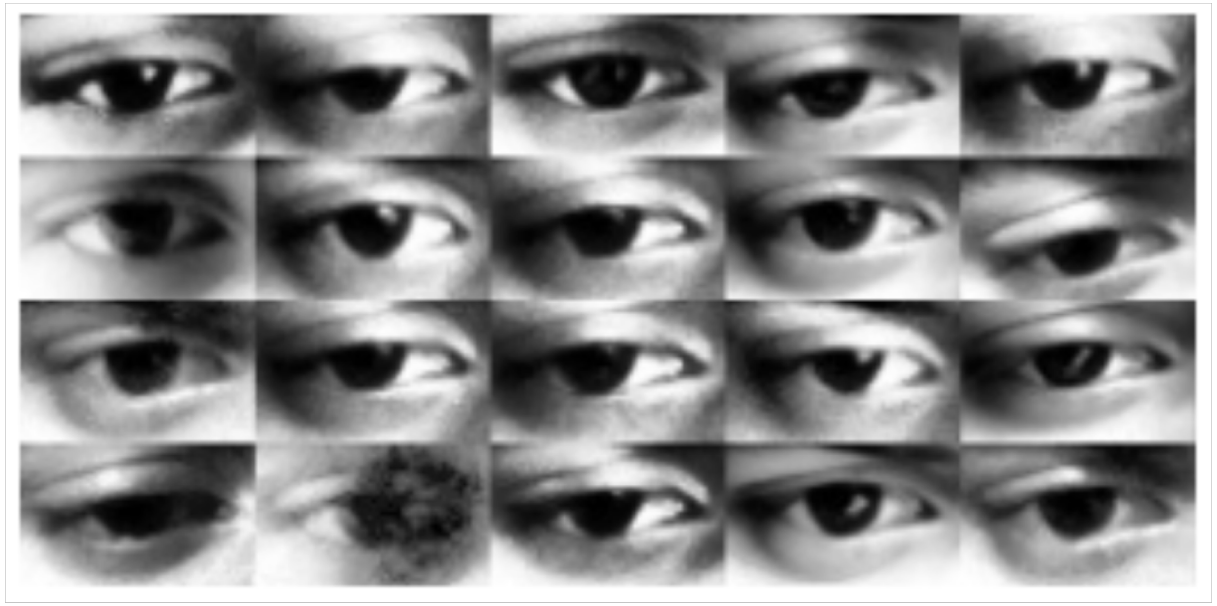
→ Such datasets are not always viable in the real world

Motivation

Application 1: ”Predicting Eye Gaze Vector”



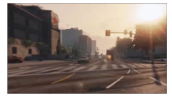
→ Difficult to annotate the eye gaze vector



E.g., eyegaze dataset

Motivation

Application 2: ”Predicting Accidents”



Accident / Non-accident

→ **Difficult to acquire the data itself in the real world**

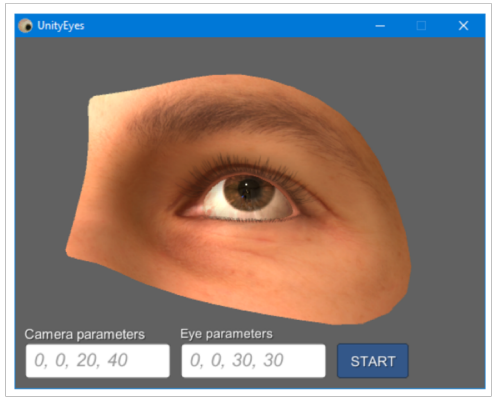
E.g., accident dataset

Motivation

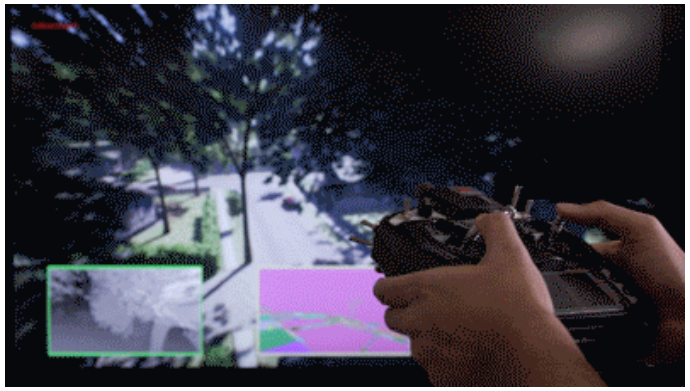
Recently, many researchers proposed utilizing simulator to tackle data scarcity

→ Generate arbitrary type & amount of data

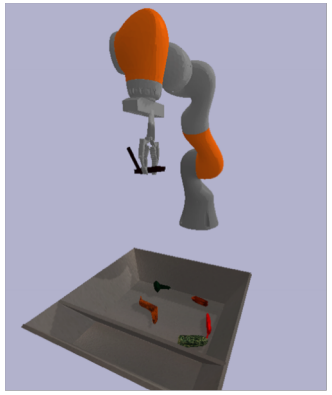
→ No labeling cost



UnityEyes - Eye simulator



AirSim - Drone simulator



Robot Arm Simulator

CARLA - Driving simulator

However, synthetic data may not be realistic enough, resulting in poor performance in the real world

Overview of this talk

1. Simulated+Unsupervised Learning with Adaptive Data Generation and Bidirectional Mapping

   - International Conference on Learning Representation (ICLR) 2018

2. Crash to not Crash: Learn to Identify Dangerous Vehicles Using a Simulator

   - Association for the Advancement in Artificial Intelligence (AAAI) 2019

Overview of this talk

1. **Simulated+Unsupervised Learning with Adaptive Data Generation and Bidirectional Mapping**

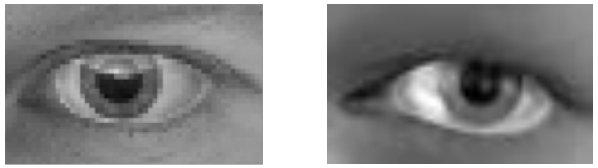
   - **International Conference on Learning Representation (ICLR) 2018**

2. Crash to not Crash: Learn to Identify Dangerous Vehicles Using a Simulator

   - Association for the Advancement in Artificial Intelligence (AAAI) 2019

Problem Formulation

Source domain (simulator)

X_s 

Y_s (-5, 5) , (25, 15)

Target domain (real world)

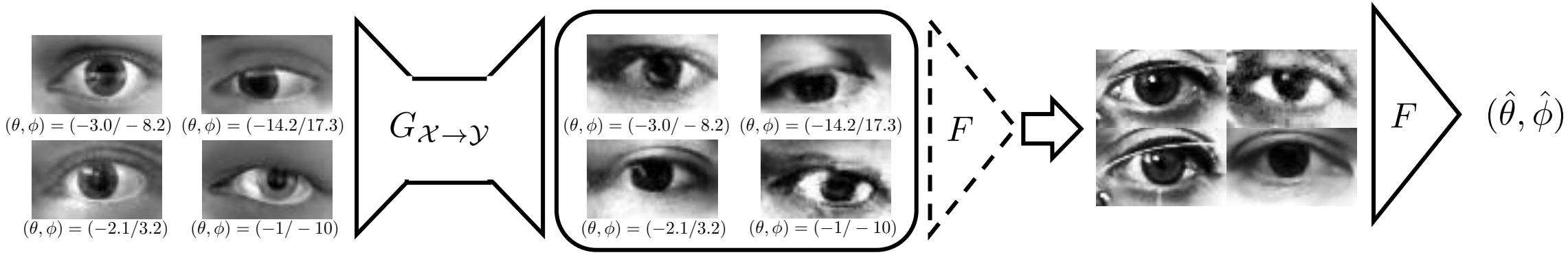
X_t 

Given { labeled source domain data (X_s, Y_s)
unlabeled target domain data (X_t)

→ Train a predictor that works well in the target domain

Domain Adaptation from Simulation to Real World

[Shrivastava et al., CVPR 2017] proposed "Simulated+Unsupervised learning"



(1) Forward/translate images (2) Train predictor on translated images (3) Predict using the original images

They showed,

1. Refining *synthetic* images with GAN makes them look *real* while preserving the labels
2. Training models on these refined images leads to significant improvements in accuracy

Domain Adaptation from Simulation to Real World

Comparison of a gaze estimator trained on Synthetic vs Refined Synthetic

Training Data	% of images within 7 degree error
Synthetic Data	62.3
Synthetic Data 4x	64.9

Domain Adaptation from Simulation to Real World

Comparison of a gaze estimator trained on Synthetic vs Refined Synthetic

Training Data	% of images within 7 degree error
Synthetic Data	62.3
Synthetic Data 4x	64.9
Refined Synthetic Data	69.4
Refined Synthetic Data 4x	**87.2**

Can we do better?

Domain Adaptation from Simulation to Real World

2 limitations of existing approach

1. **[S] How can we specify synthetic label distribution?**

   → Can we utilize the flexibility of the simulator to generate better *synthetic* data?

2. **[U] Not taking advantage of the asymmetry between the *real* and *synthetic* dataset**

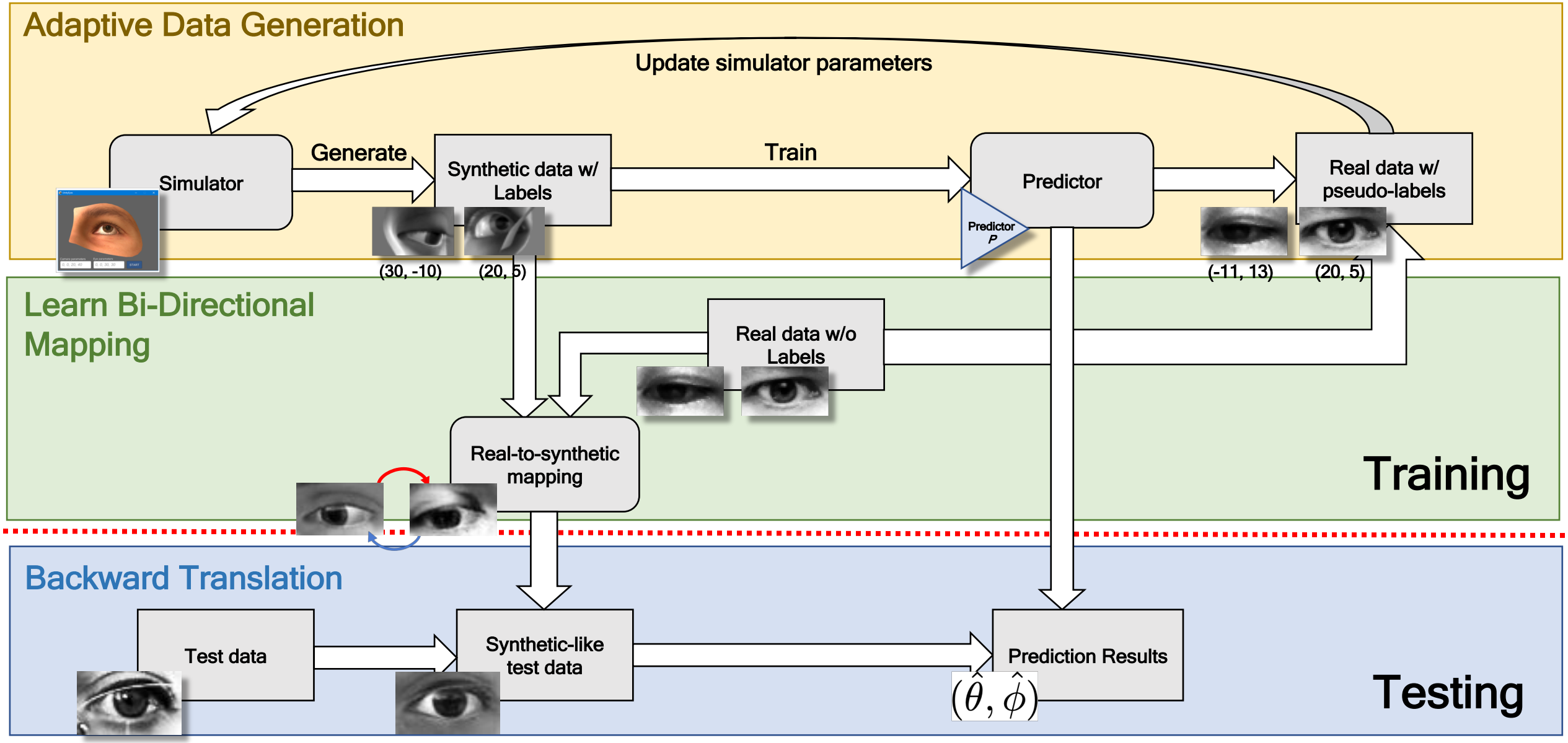
   - The amount of available *synthetic* data is much larger than that of *real* data
   - *Synthetic* data is usually much cleaner (less noise) compared to *real* data

   → Can we devise a better unsupervised domain adaptation algorithm?

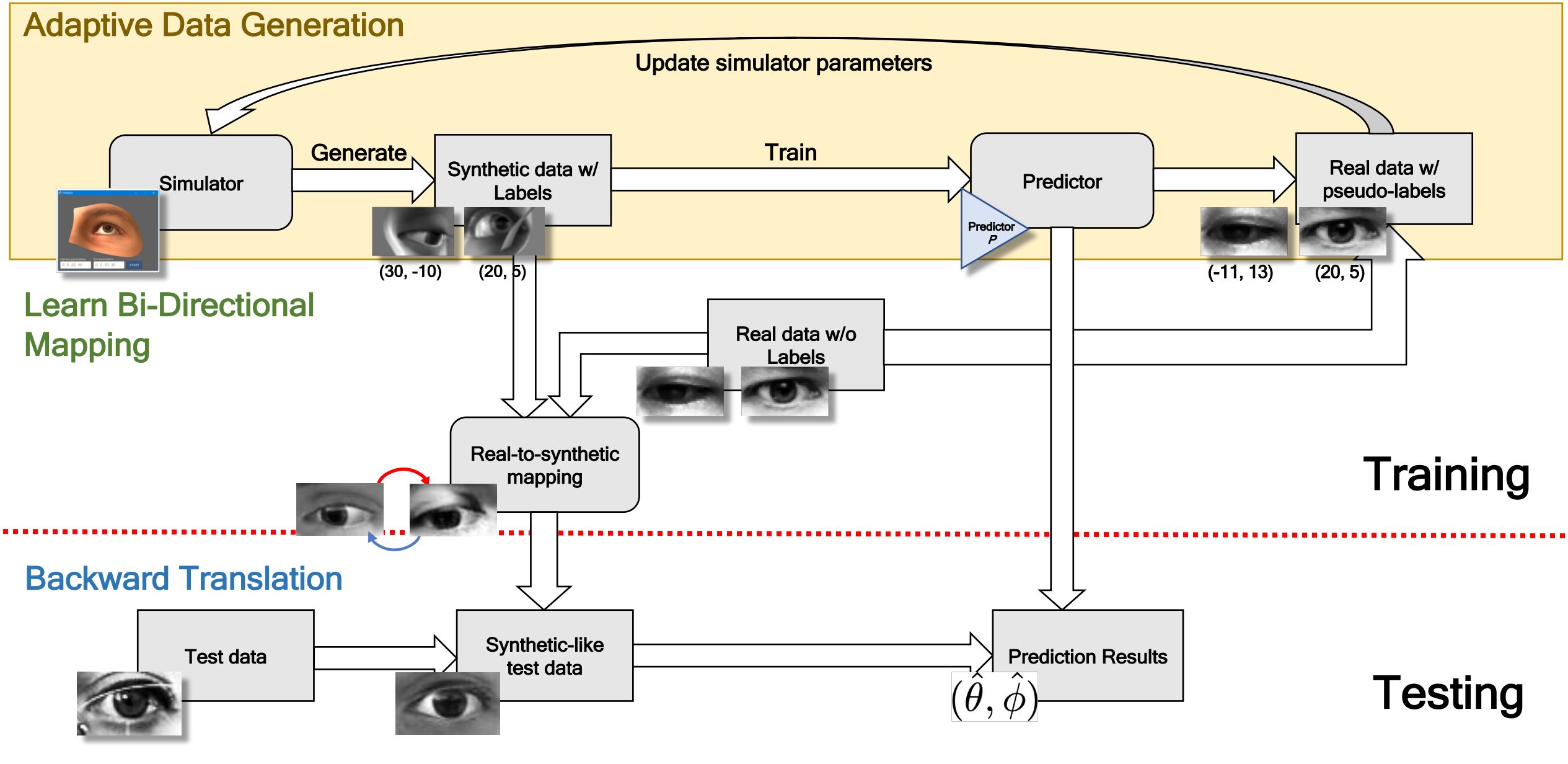
Our proposed algorithm - Overview

1. **Adaptive Data Generation**
   - Match distributions between *synthetic* / *real* labels
2. **Learn Bi-Directional Mapping**
   - Learn mapping between *synthetic* / *real* data with modified CycleGAN
3. **Backward Translation**
   - Make prediction with *real* ➡ *synthetic* translated image

Our proposed algorithm - Overview

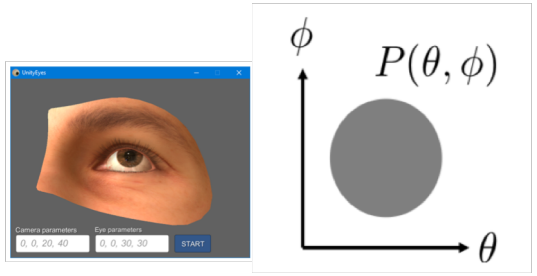


Our proposed algorithm - Overview



Step 1 - Adaptive Data Generation

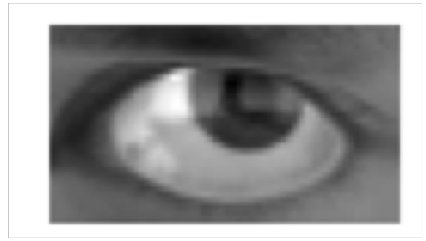
How Simulator works



Specify Label Distribution

$$(\theta, \phi) = (12.5, -7.8)$$

Draw random label instance



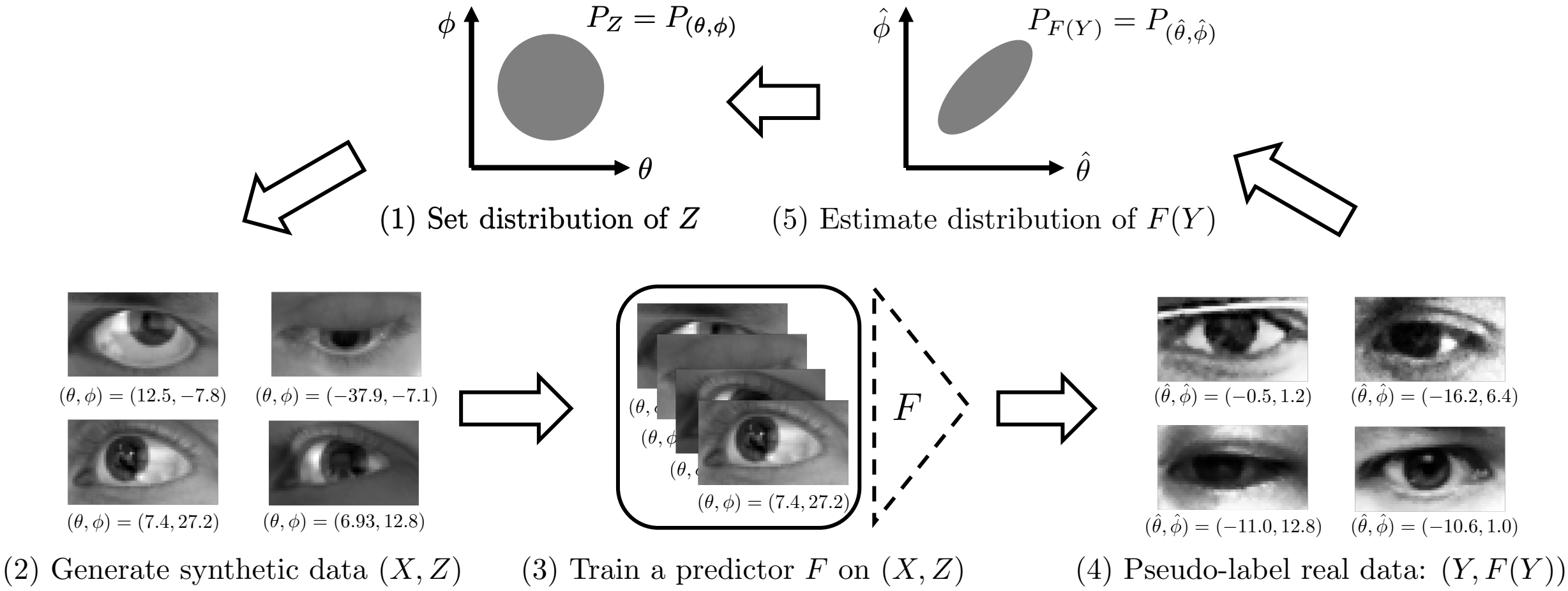
Generate image

→ Label distributions can be different in *synthetic* & *real* datasets

→ Fortunately, the *synthetic* distribution can be easily modified

Q. How can we systematically adapt the label distribution?

Step 1 - Adaptive Data Generation



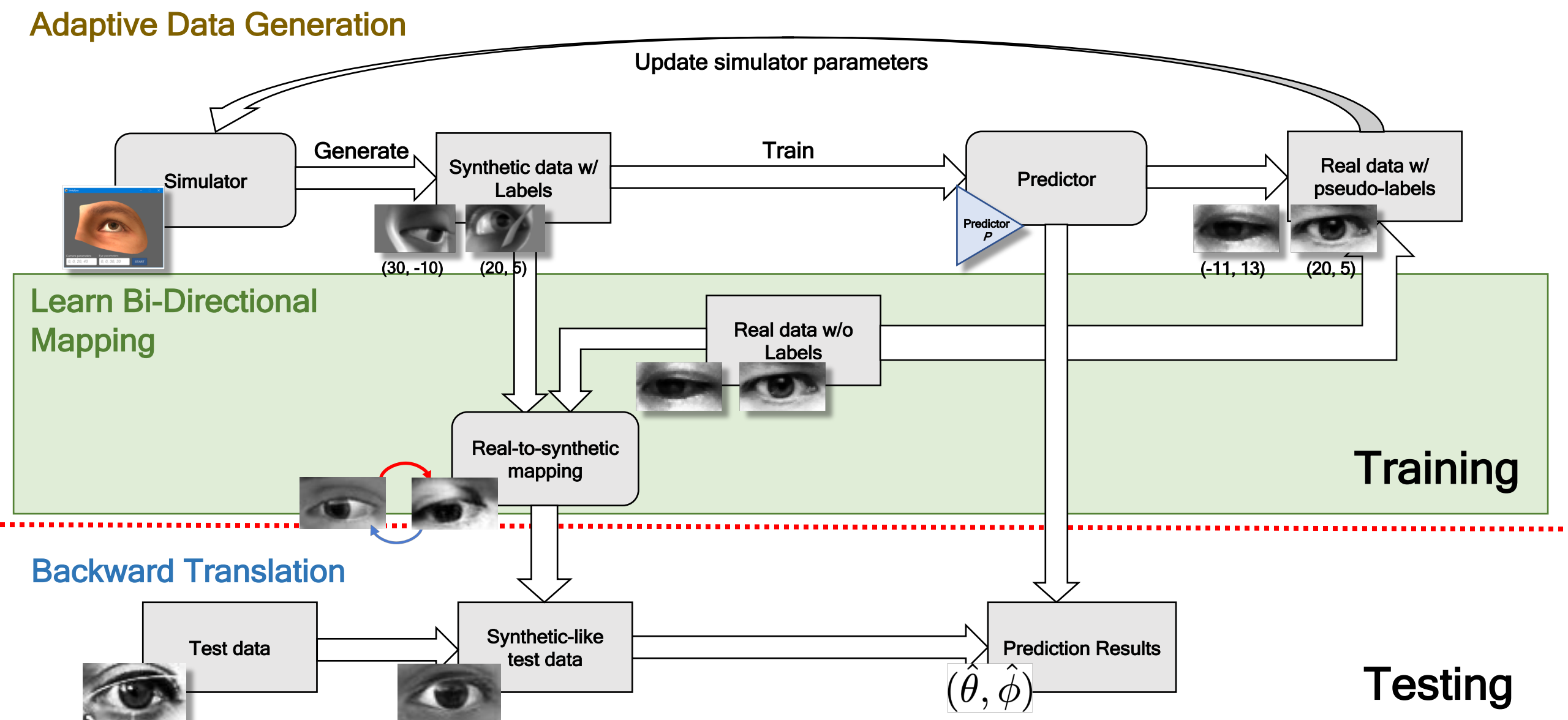
Overview of our adaptive data generation process

Step 1 - Adaptive Data Generation

→ **Experimental results of the adaptive data generation algorithm**

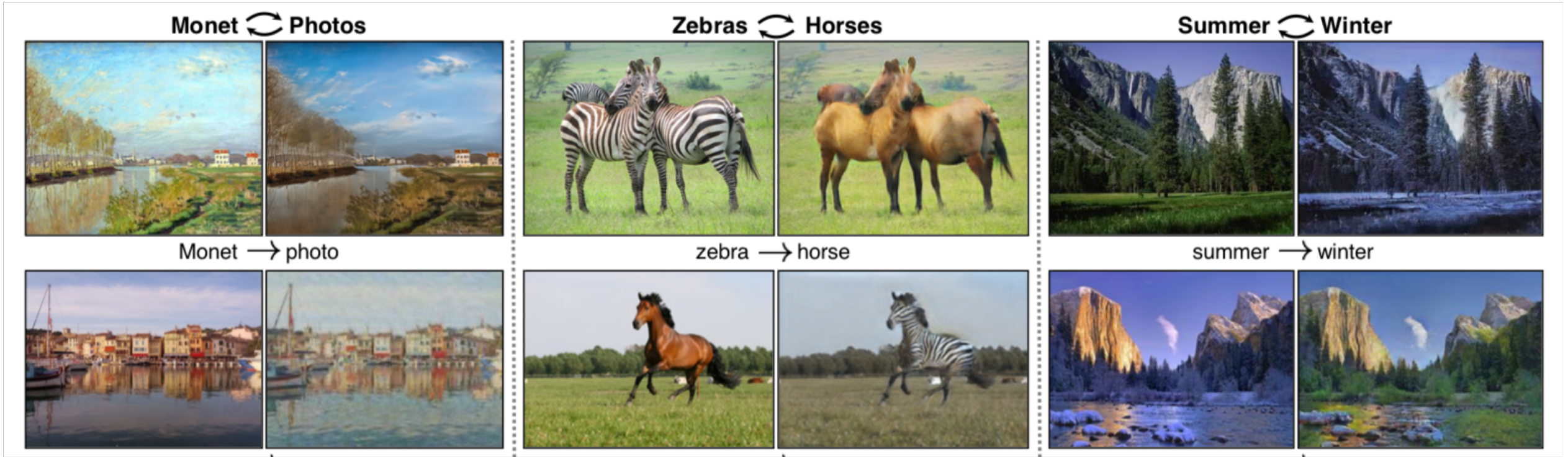
	Iteration 0	Iteration 1	Iteration 2
$\lvert\mathbb{E}[\Theta^{(\ell)}] - \mathbb{E}[\hat{\Theta}^*]\rvert$	8.54	3.00	**1.04**
$\lvert\mathbb{E}(\Phi^{(\ell)}) - \mathbb{E}(\hat{\Phi}^*)\rvert$	0.31	0.02	**0.00**

Step 2 - Learn Bi-Directional Mapping



Step 2 - Learn Bi-Directional Mapping

- **CycleGAN**



[Zhu et al., ICCV 2017]

Step 2 – Learn Bi-Directional Mapping

(a) (Input images) y | (b) (Cycle only) $G_{\mathcal{Y}\to\mathcal{X}}(y)$ | (c) (Cycle only) $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$ | (d) (Cycle&feature) $G_{\mathcal{Y}\to\mathcal{X}}(y)$ | (e) (Cycle&feature) $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$

Step 2 - Learn Bi-Directional Mapping

(a) (Input images) y (b) (Cycle only) $G_{\mathcal{Y}\to\mathcal{X}}(y)$ (c) (Cycle only) $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$ (d) (Cycle&feature) $G_{\mathcal{Y}\to\mathcal{X}}(y)$ (e) (Cycle&feature) $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$

Step 3 - Backward Translation

Adaptive Data Generation

Update simulator parameters

Simulator

Generate

Synthetic data w/ Labels

(30, -10)

(20, 5)

Train

Predictor

Predictor *P*

Real data w/ pseudo-labels

(-11, 13)

(20, 5)

Learn Bi-Directional Mapping

Real data w/o Labels

Real-to-synthetic mapping

Training

Backward Translation

Test data

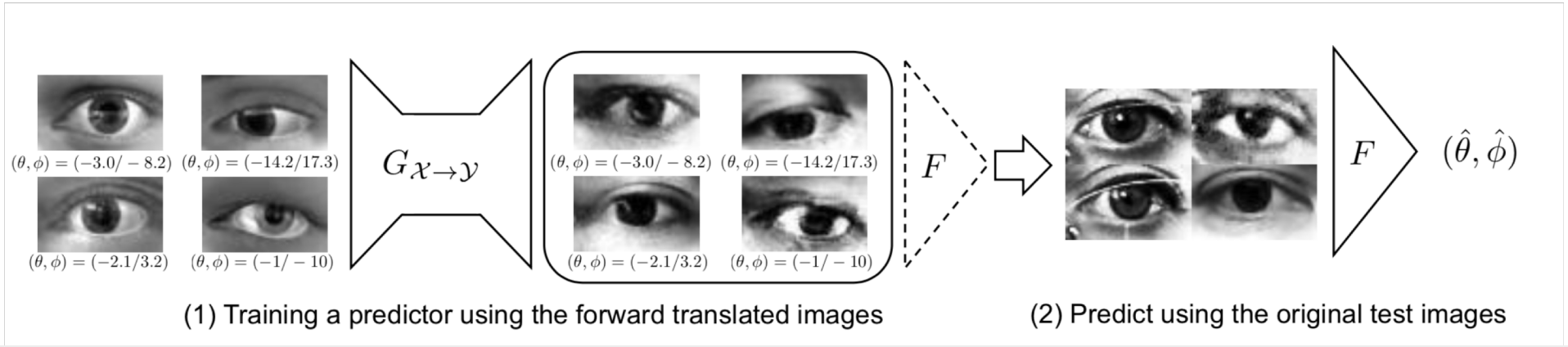
Synthetic-like test data

Prediction Results

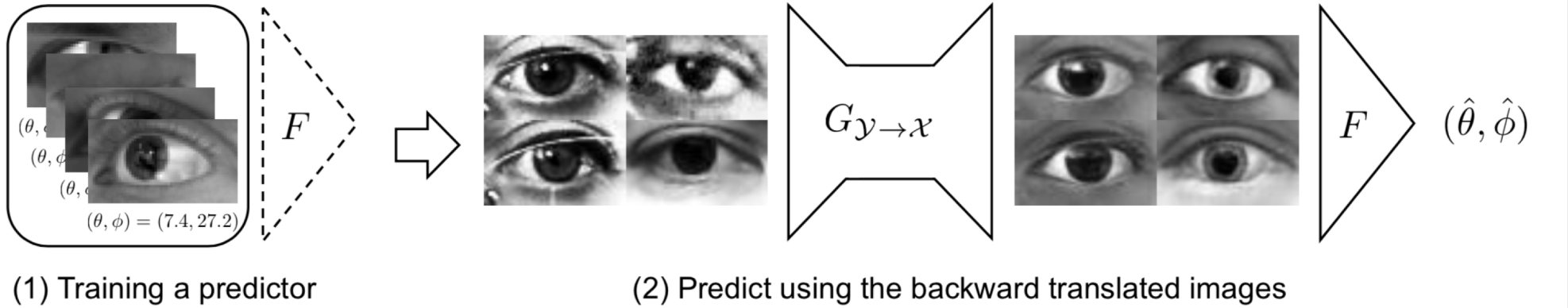
$(\hat{\theta}, \hat{\phi})$

Testing

Step 3 - Backward Translation



Forward/Translation: training/test methodology proposed by [Shrivastava et al., 2017]



Backward/Translation: Our proposed training/test methodology

Error with Forward/Translation: 8.4 ➡ 7.71

Error with Backward/Translation: 8.4 ➡ <u>7.60</u>

Results

Comparison of the state of the arts on the MPIIGaze dataset (Error: mean angle error in degrees)

Method	Trained on	Tested with	Error
KNN w/ UT Multiview (Zhang et al., 2015)	R	R	16.2
CNN w/ UT Multiview (Zhang et al., 2015)	R	R	13.9
CNN w/ UnityEyes (Shrivastava et al., 2017)	S	R	11.2
KNN w/ UnityEyes (Wood et al., 2016)	S	R	9.9
CNN w/ UnityEyes (Shrivastava et al., 2017)	RS	R	7.8
Ours w/ feature loss	**S**	**RR**	**7.6**

Conclusion

- **In this work, we proposed a new S+U learning algorithm**
  - Fully exploits the flexibility of simulators
  - Improved bidirectional mapping
  - Utilizes the data asymmetry between *real*/*synthetic* dastaset
  - Achieved the state-of-the-art result on MPIIGaze dataset

Overview of this talk

1. Simulated+Unsupervised Learning with Adaptive Data Generation and Bidirectional Mapping

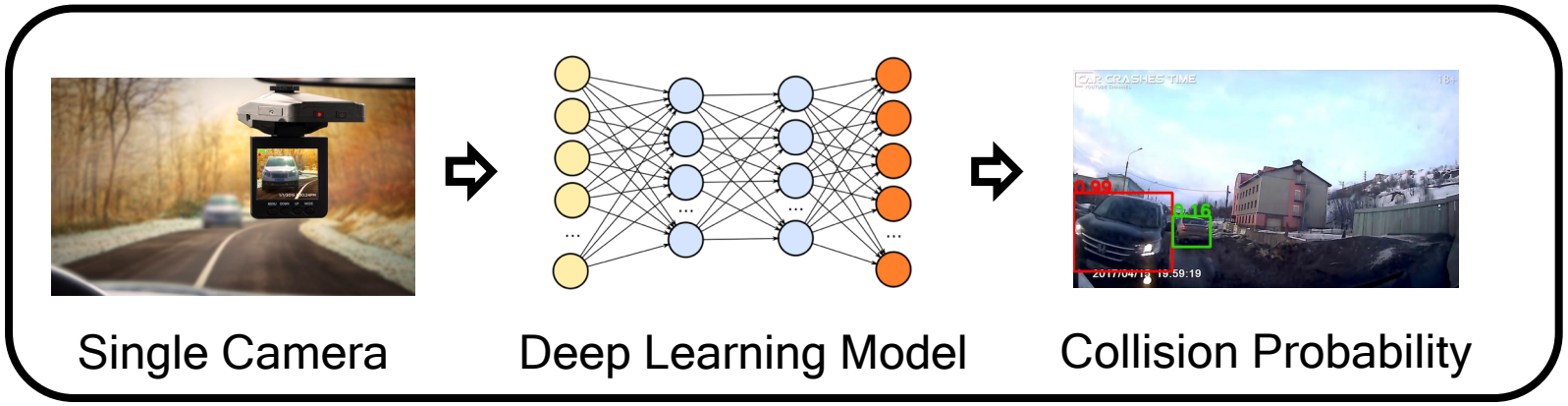
   - International Conference on Learning Representation (ICLR) 2018

2. **Crash to not Crash: Learn to Identify Dangerous Vehicles Using a Simulator**

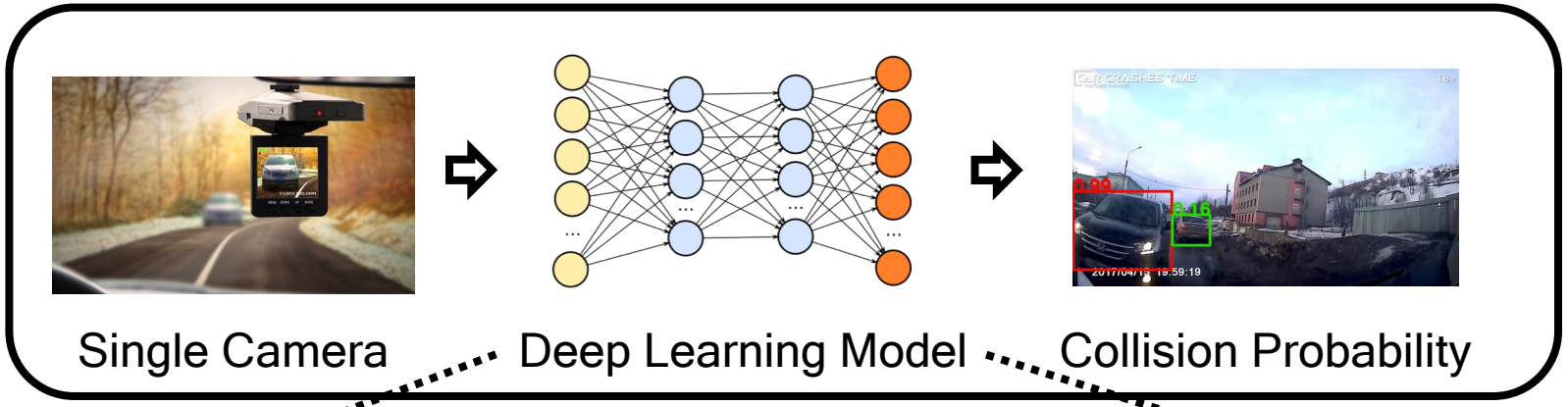
   - Association for the Advancement in Artificial Intelligence (AAAI) 2019

Collision Prediction System (CPS)

- Goal of CPS: To predict vehicle collisions ahead of time
- Existing CPSs are
  - based on rule-based algorithms
  - require expensive devices (LIDAR, communication chips, …)
- Our goal is to build a CPS that is
  - based on data-driven algorithms
  - require commodity devices (camera sensor)



Problem Formulation



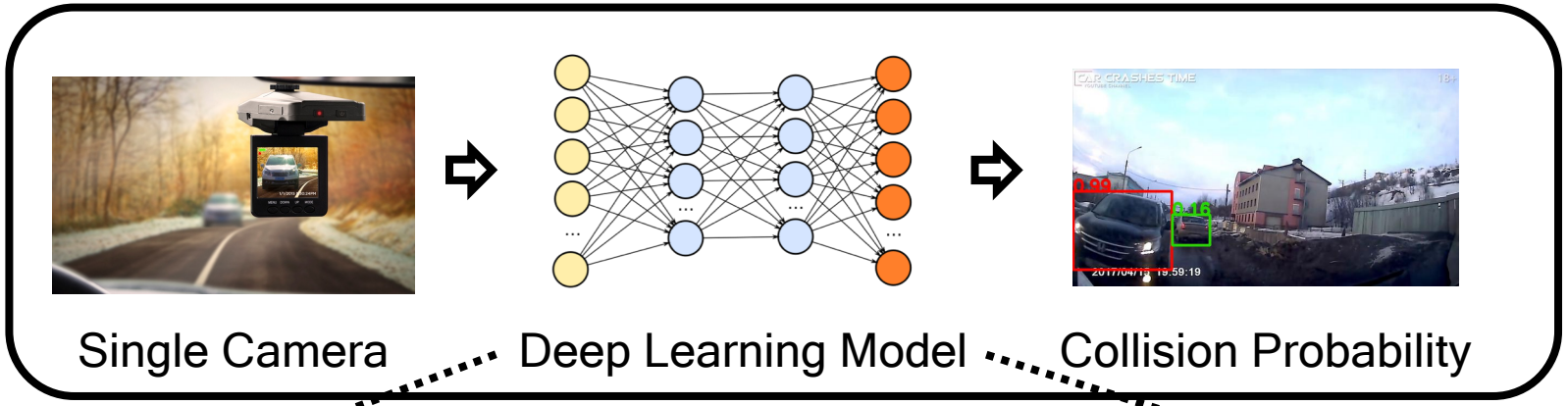
1. Object Tracking

detects/tracks vehicles
in the input image(s)

2. Object Classification

classifies whether or not the
given vehicle is dangerous

Problem Formulation



1. Object Tracking

detects/tracks vehicles
in the input image(s)

2. Object Classification

classifies whether or not the given vehicle is dangerous

→ Our main interest!

CPS in Action

Single Camera

CPS

- Safe vehicle - Dangerous vehicle

Challenges in building a data-driven CPS: Data Scarcity

- Existing driving dataset (KITTI, Cityscapes) has no accident data
- Collecting accident data in the real world is very expensive
- We can take advantage of driving simulator

Grand Theft Auto V

CARLA - Driving simulator

How do we collect accident dataset from a simulator?

Q1. How do we reproduce accident scenes in a simulator?

How do we collect accident dataset from a simulator?

Q1. How do we reproduce accident scenes in a simulator?

→ we modify driver behavior via simulator hacking

Data Collection

- *ScriptHookV* - A third party script that allows for an access to game engine
- On top of ScriptHookV, we implemented the following helper functions
  - **Start_Cruising(s)** – makes the player's car cruise along the lane at a constant speed s
  - **Get_Nearby_Vehicles()** – returns the list of vehicles in the current scene of the game
  - **Set_Vehicle_Out_Of_Control(v)** – makes the vehicle v drive out of control
  - **Is_My_Car_Colliding()** – checks whether my car is colliding or not
- Our accident simulator generates accident data in a random manner by setting nearby vehicles out of control while cruising and saving past 20 frames when collision occurs

Data Collection

Original GTA V

Driver behavior modification
via simulator hacking

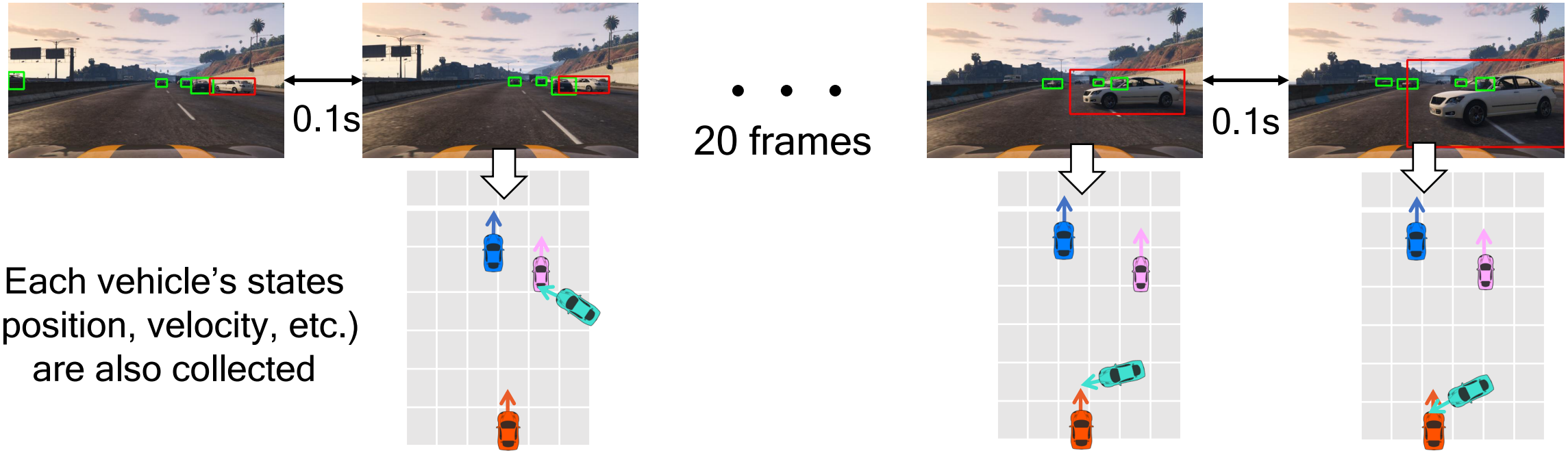
Hacked GTA V

Normal Driving Scene

Accident Driving Scene

Data Collection - *GTACrash*

- 3661 from original GTA V (Non-accident scenes)
- 7720 from hacked GTA V (Accident scenes)
- Each scene is composed of following frames:



Each vehicle's states (position, velocity, etc.) are also collected

Dataset is available @ https://sites.google.com/view/crash-to-not-crash

How can we collect accident dataset from a simulator?

Q2. How can we make them more realistic?

How can we collect accident dataset from a simulator?

Q2. How can we make them more realistic?

→ Feature (Image): Style transfer via GAN

→ Label: label adaptation via motion model

Domain Adaptation - Feature Adaptation

Synthetic train data

Real test data

Domain difference between

train and test images

results in poor test performance

Domain Adaptation - Feature Adaptation

→ We apply backward translation approach (Lee, Kim, and Suh 2018)

Synthetic train data

Real test data

Map images to synthetic domain at test time

Backward-translated test data

How can we collect accident dataset from a simulator?

Q2. How can we make them more realistic?

→ Feature (Image): Style transfer via GAN

→ Label: label adaptation via motion model

Synthetic labels are biased

Synthetic Label = 1

Desired Label = 0

Synthetic Label = 1

Desired Label = 0

Synthetic Label = 1

Desired Label = 1

Synthetic Label = 1

Desired Label = 1

Synthetic Label = 1

Desired Label = 1

Synthetic Label = 1

Desired Label = 1

Synthetic Label = 1

Desired Label = 1

Synthetic Label = 1

Desired Label = 1

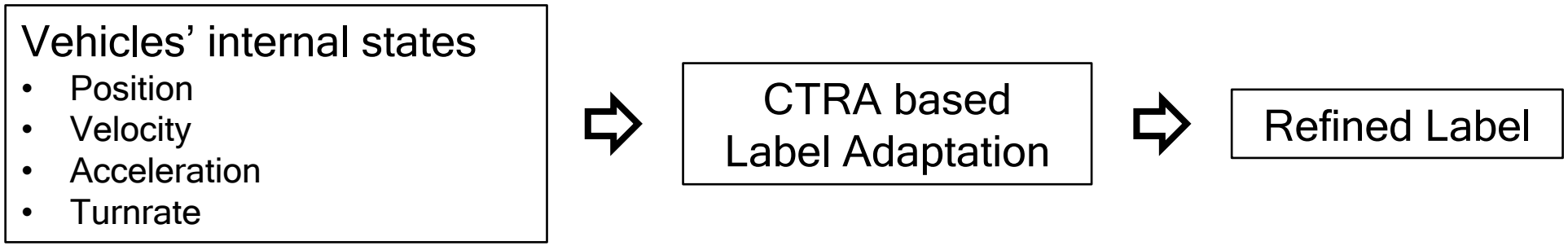
→ Synthetic labels are biased!

Label Adaptation

- **Vehicle path prediction is a well studied field**
- **Constant Turn Rate and Acceleration (CTRA) model** – Schubert et al. 2008
  - state-of-the-art path prediction performance in the real world
  - Requires vehicle states such as position, velocity, acceleration, and turn rate
  - By accessing internal states we can access all the required information

→ **Our proposed label adaptation**

1. Compute the predicted path of each vehicle as per the CTRA model
2. Assign a positive label if and only if the predicted path intersects with that of the ego-vehicle

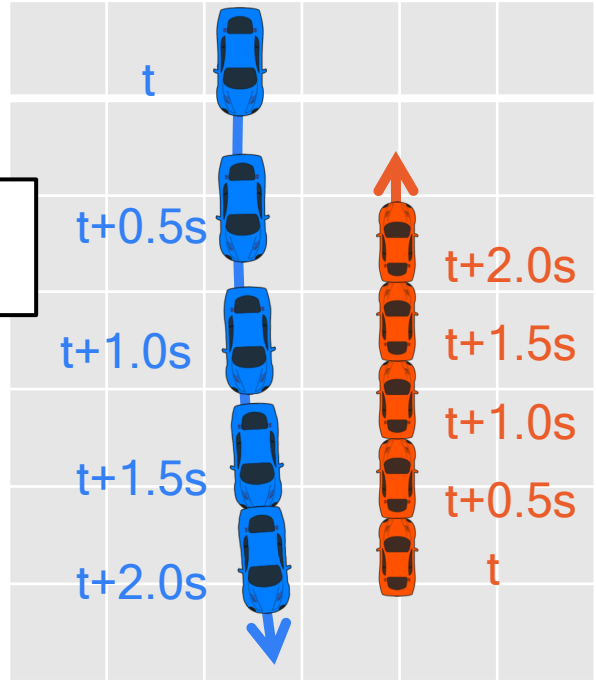


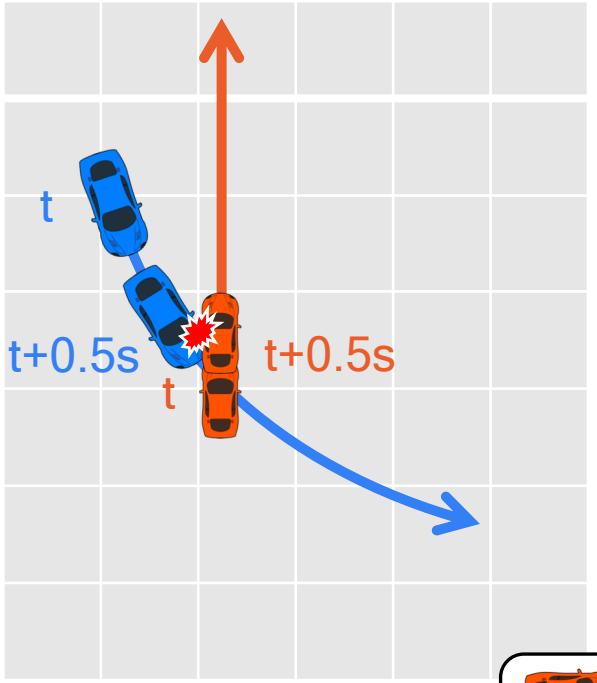
Label Adaptation

→ We adapt labels based on a well-studied vehicle motion model

1. Retrieve internal states

2. Predict future path via motion model





3. Refine Labels

Refined Label = 0

Refined Label = 1

Experiments

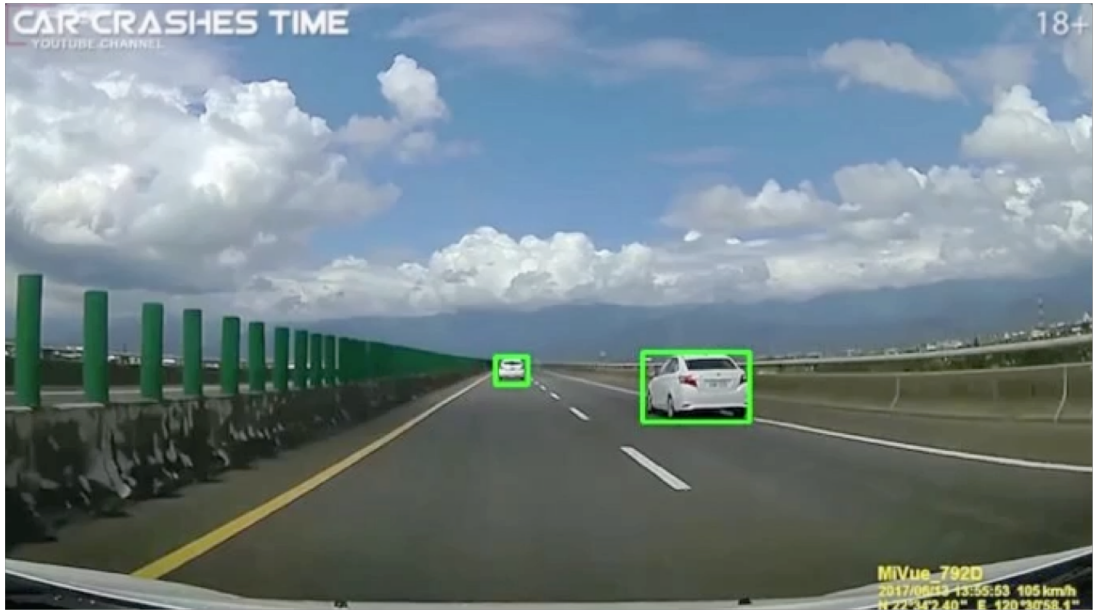
	Trained with Small real dataset	Trained with Large synthetic data
Pros	Dataset is unbiased	Large amount
Cons	Small amount	Dataset is biased

Q. Which approach will perform better on real data?

Experiments - *YouTubeCrash*

- 100 Non-accident Scenes / 122 Accident Scenes
- Manual annotation of bounding box and vehicle IDs
- A half of *YouTubeCrash* is used as training dataset for baseline
- Another half is used for test dataset for both baseline and our approach

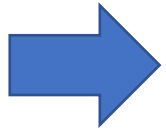
Accidents



Non-Accidents

Experiments - Dataset Comparison

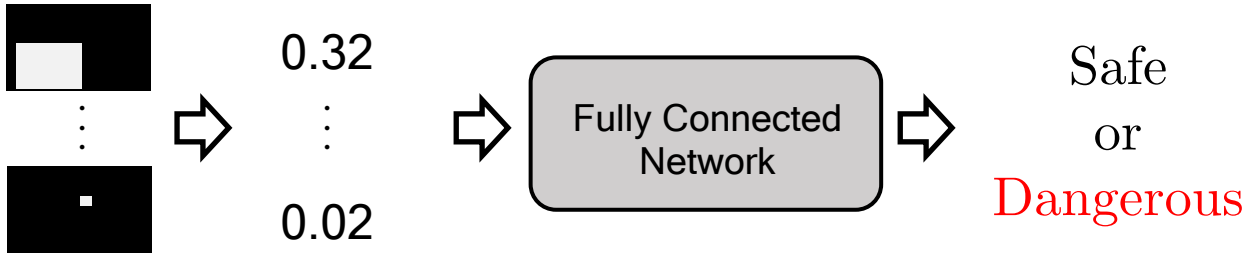
	YouTubeCrash	*GTACrash*
Accident scenes	61	~7000
Non-accident scenes	50	~3500
Positive samples	~ 1000	~130000
Negative samples	~ 5500	~600000



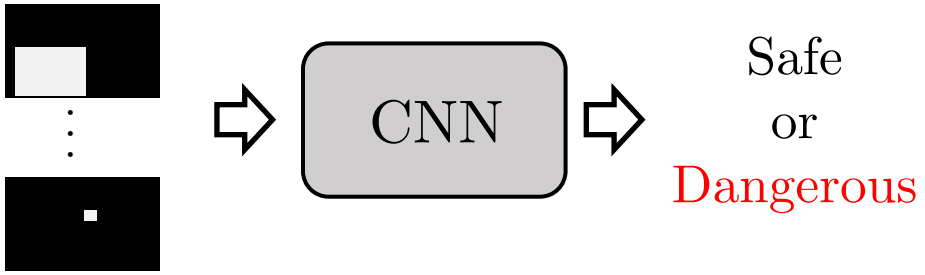
~ 100x

Experiments - Classification Algorithms

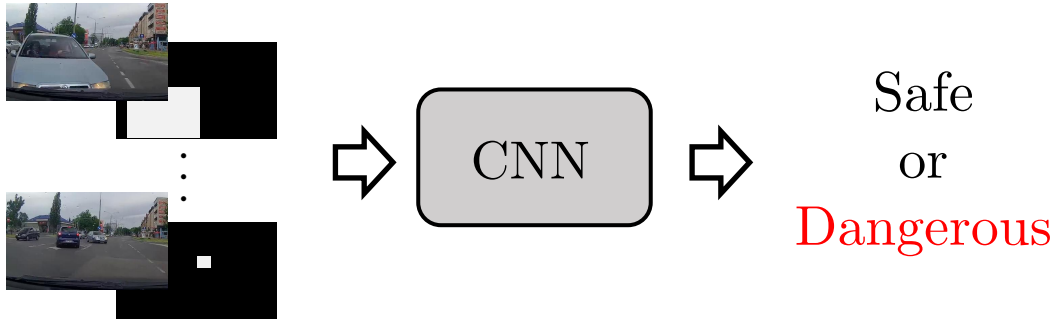
Alg1. 5-layer NN with bounding box sizes



Alg2. CNN with bounding boxes



Alg3. CNN with RGBs & bounding boxes



Experiments – Test AUC on *YoutubeCrash*

Alg/Training data	**Trained with real data**	**Trained with synthetic data**
Size of training data	61 accidents 50 non-accidents	7000 accidents 3500 non-accidents
Alg1	0.8766	
Alg2	0.8708	
Alg3	0.8730	

Experiments - Test AUC on *YoutubeCrash*

Alg/Training data	**Trained with real data**	**Trained with synthetic data**
Size of training data	61 accidents 50 non-accidents	7000 accidents 3500 non-accidents
Alg1	0.8766	0.8812
Alg2	0.8708	0.8992
Alg3	0.8730	**0.9086**

Performance improvement with all tested algorithms

Experiments - Test AUC on *YoutubeCrash*

Alg/Training data	**Trained with real data**	**Trained with synthetic data**
Size of training data	61 accidents 50 non-accidents	7000 accidents 3500 non-accidents
Alg1	0.8766	0.8812
Alg2	0.8708	0.8992
Alg3	0.8730	**0.9086**

Increasing model complexity

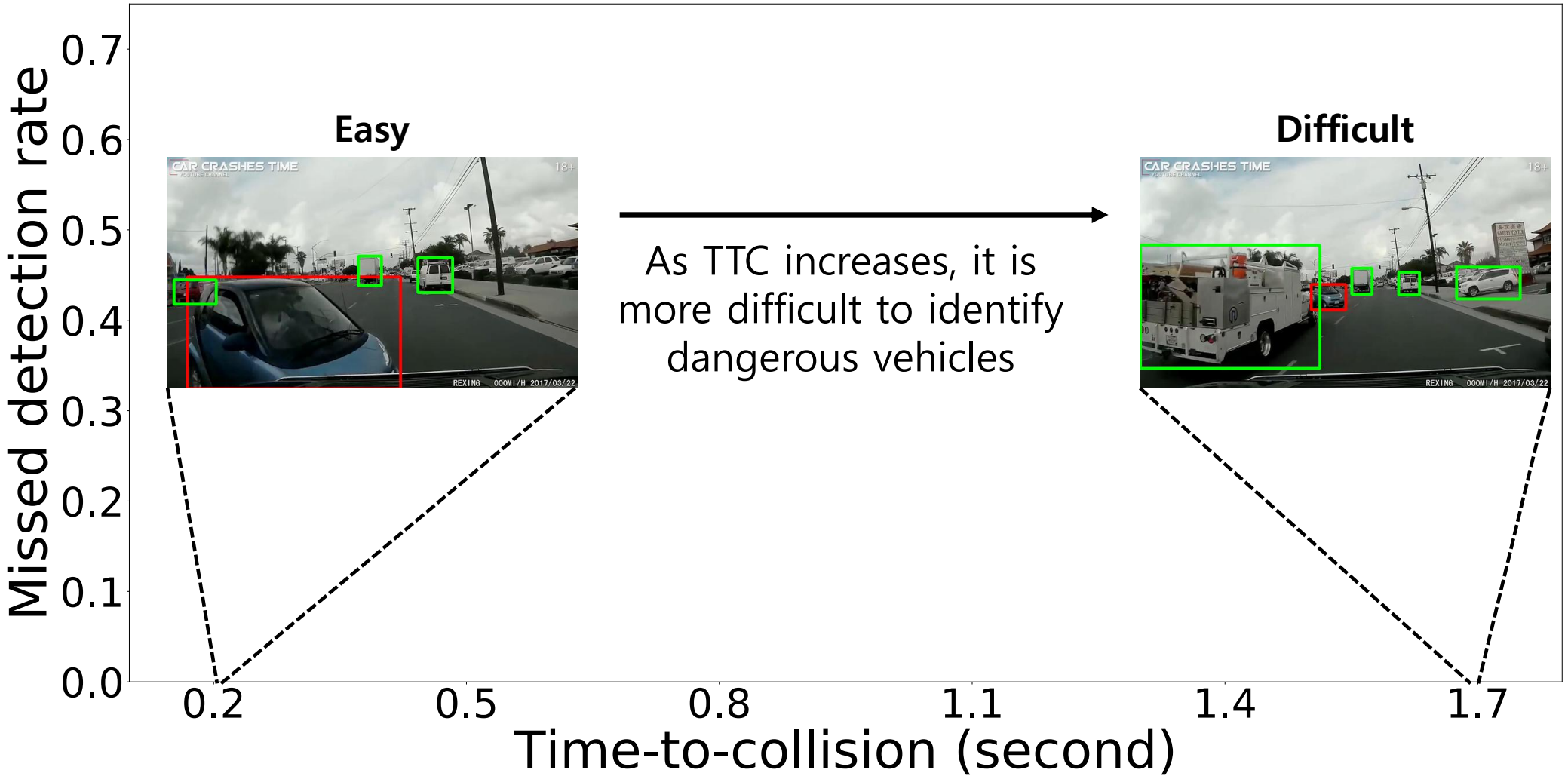
A complex model can be trained only with large data

Experiments - Test AUC on *YoutubeCrash*

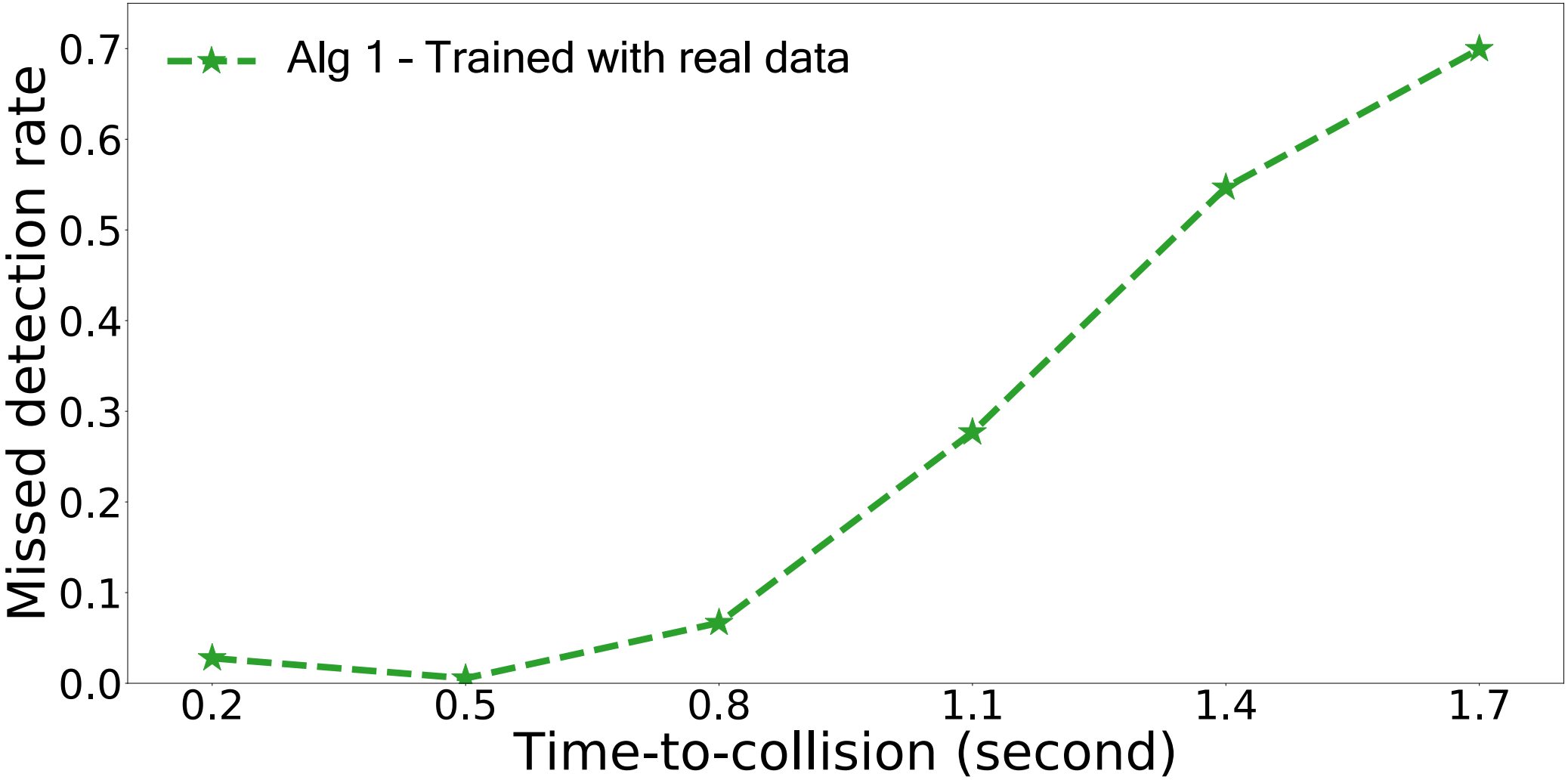
Alg/Training data	Trained with real data	**Trained with synthetic data**
Size of training data	61 accidents 50 non-accidents	7000 accidents 3500 non-accidents
Alg1	0.8766	0.8812
Alg2	0.8708	0.8992
Alg3	0.8730	**0.9086** **→ 0.9164**

Domain adaptation further improves performance

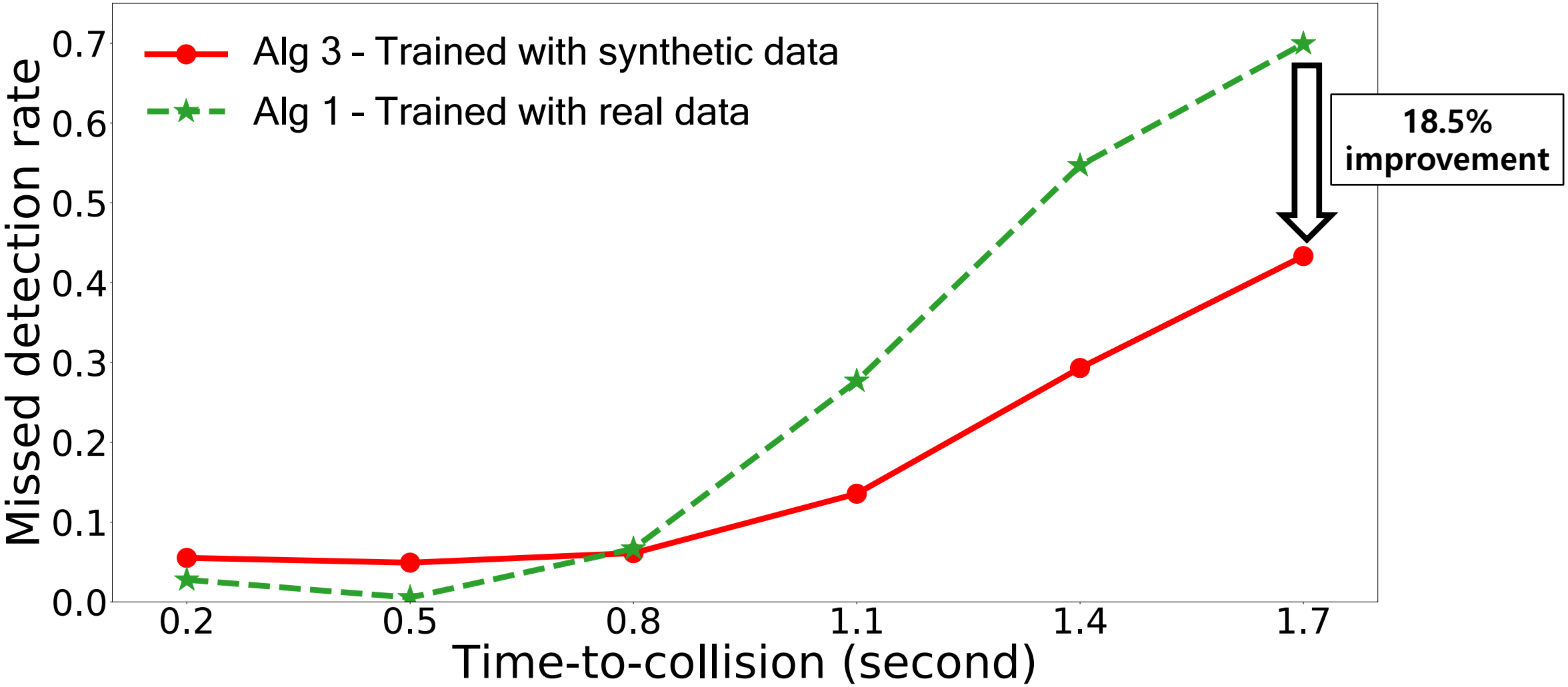
Experiments – Missed Detection vs Time-To-Collision



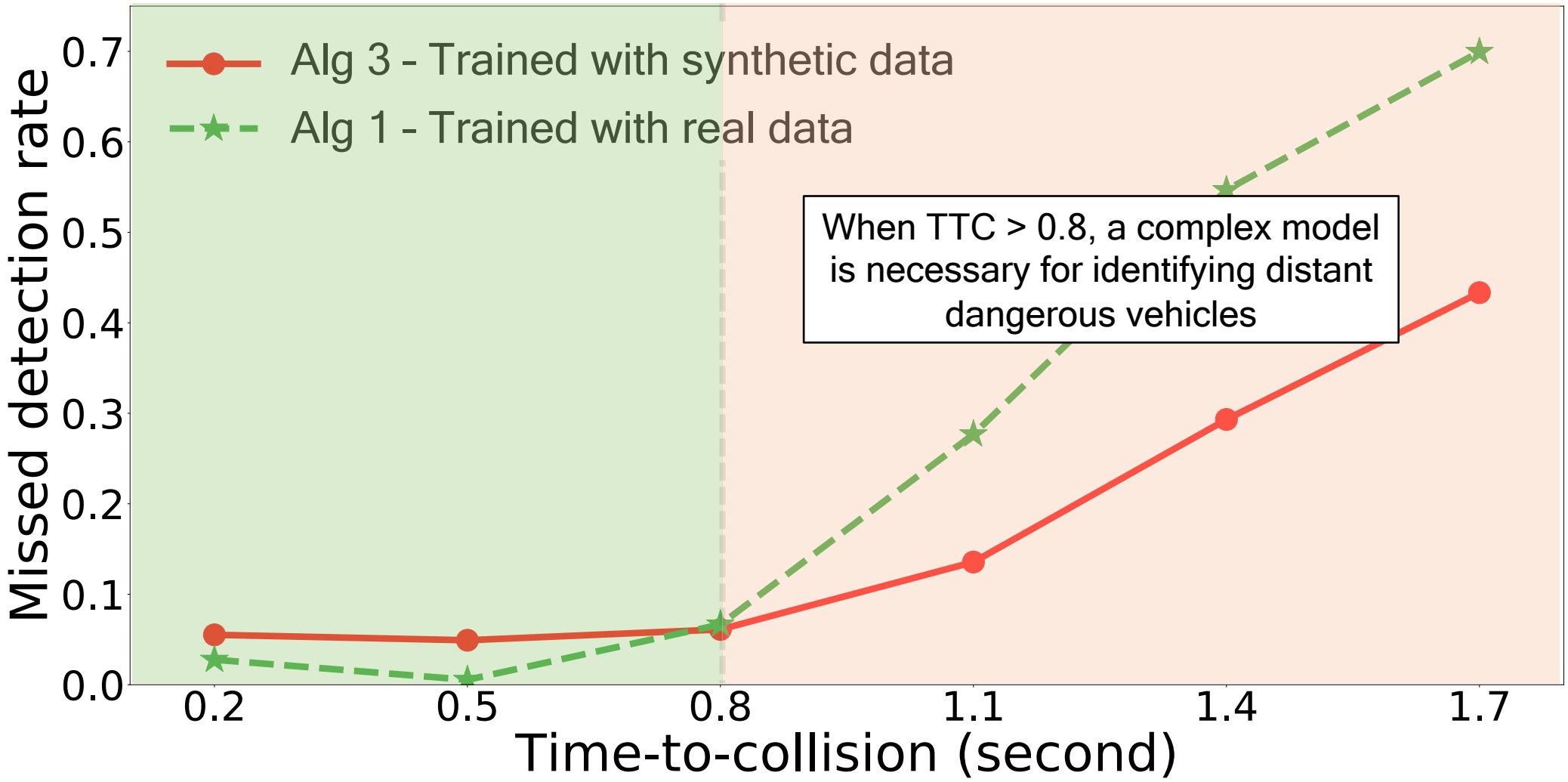
Experiments – Missed Detection vs Time-To-Collision



Experiments – Missed Detection vs Time-To-Collision



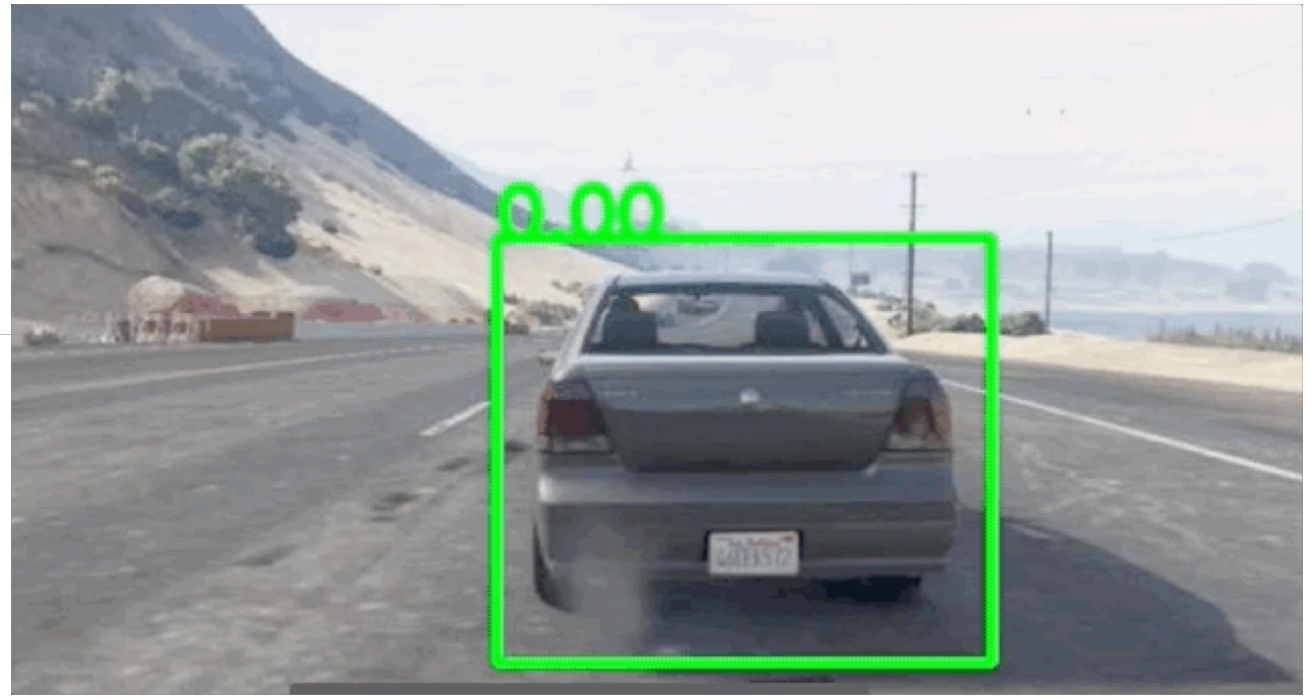
Experiment Results – Missed Detection vs Time-To-Collision



Experiment Results - Learning Key Visual Features

Wheel Angle

Vehicle Angle



Velocity

Checkout the website for the demonstration video

Demonstration Video

Each number represents the predicted probability of collision of the vehicle

: Ground truth dangerous vehicle

Predicted result of our algorithm

Safe **Dangerous**

Checkout the website for the demonstration video

Conclusion

- **In this work, we propose an efficient deep learning-based CPS**
  - Customized synthetic data generator
  - Novel label adaptation method
  - Our method outperforms those trained with real data
- **Our datasets and demo videos are available at**

https://sites.google.com/view/crash-to-not-crash