

# Provably Safe Prediction for Autonomous Vehicles Using Uncertainty Estimations



Michael OBroin and Adrian Kager

# Motivation for Our Project

- Wanted to explore the combination of ML and verifying CPS
- Didn't have necessary background to dive deeper into research grade topics
- Scaled back to a simplified model and system

# Widespread Adoption Self Driving Technology could have Enormous Impact

- \$800 Billion dollar impact in US alone<sup>1</sup>
- 35k deaths on US roads per year
- Could revolutionize transportation, shipping industries



1: <https://aworkforce.secureenergy.org/>

# But Fears about Safety Can Hold Back Adoption

- Safety critical
- Boost public opinion
- 50 percent of Americans somewhat or very worried<sup>2</sup>



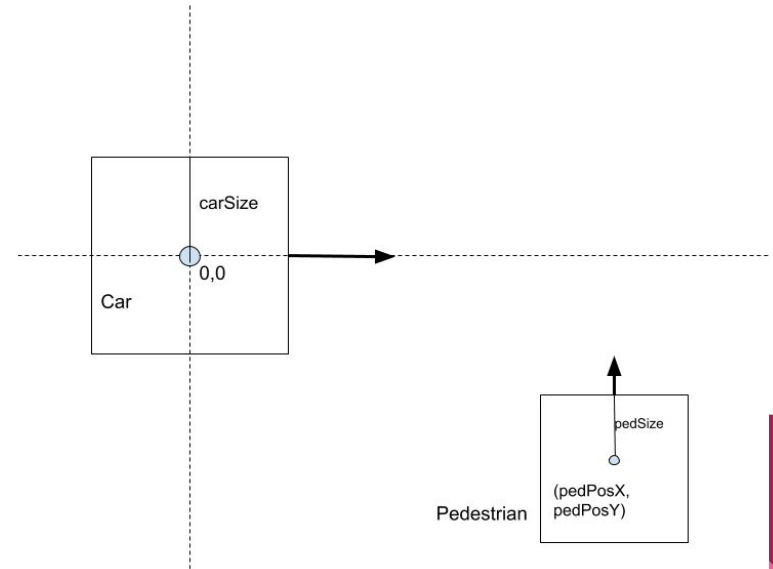
# Our Approach

# We Proved the Safety of a Specific AV Prediction Task with Assumptions about Uncertainty

- Created simple model of pedestrian
- Learned ML prediction model with uncertainty estimate
- Proved safety given assumptions on uncertainty

# Modeling the System

- 2D plane with straight line motion
- Car and Pedestrian represented by squares
- Car makes decision to brake or not based on estimated crossing time and uncertainty



# Modeling the Pedestrian

- Considered “important characteristics”
  - Distance From Curb
  - Speed
  - Direction
  - Red Shirt
- Generated a label using a function we wrote



# Label Generator Function

```
def time_to_cross(stopwalk_dist, direction, speed, red_clothes):  
    """  
    Gives time until a pedestrian with these attributes will cross the street.  
    This is the function we want to learn with our neural networks.  
    """  
    scale = 1  
  
    if speed == 0:  
        speed = 0.1  
    time_to_cross = scale * stopwalk_dist / speed  
  
    if direction == 1:  
        time_to_cross /= 2  
    elif direction == 2:  
        time_to_cross *= 3  
  
    if red_clothes:  
        time_to_cross /= 1.5  
  
    return time_to_cross
```

# Learning a Model of The Pedestrian and Estimating Prediction Uncertainty

- Trained Neural Network on the data generated from the label generation function
- Attempted two methods for uncertainty estimation
  - Ensemble of randomly initialized networks
  - Dropout at runtime to get Monte Carlo samples

# Network Architectures

- Experimented with different architectures a little bit
- Settled on ones that trained reasonably quickly and didn't overfit

Linear(7,50), ReLU, Linear(50,1)

Linear(7,50), ReLU, Dropout(p=.05), Linear(50, 1)

# Experimental Results

# Networks learned the Function, but Uncertainty Estimates were Poor for Dropout Method

- Were able to learn function with low Mean Squared Error
  - $\sim 0.03$  for ensemble method on test set (Appendix I)
- Dropout method was inconsistent
- Ensemble produced accurate uncertainty estimates (Appendix II)

# General Takeaways

- Was difficult to come up with a project idea that blended ML / CPS with realistic complexity
- Building out infrastructure from scratch was a challenge
- Doing something new (for class projects) is hard

# Questions?

## Modeling the Pedestrian

- Considered “important characteristics”
  - Distance From Curb
  - Speed
  - Direction
  - Red Shirt
- Generated a label using a function we wrote



8

## We Proved the Safety under Assumptions of a Specific Autonomous Vehicle Prediction Task in a Simplified Model of the Environment

- Created simple model of pedestrian
- Learned ML prediction model with uncertainty estimate
- Proved safety given assumptions on uncertainty

3

## Networks learned the Function, but Uncertainty Estimates were Poor for Dropout Method

- Were able to learn function with low Mean Squared Error
  - $\sim 0.03$  for ensemble method on test set (Appendix I)
- Dropout method was inconsistent
- Ensemble produced accurate uncertainty estimates (Appendix II)

13

# Sources

Charles Carson, Susan Helper, Erica Groshen, and John Paul Macduffie. America's workforce and the self driving future, Jun 2018. URL <https://avworkforce.secureenergy.org/>.

Jaime F. Fisac, Andrea Bajcsy, Sylvia L. Herbert, David Fridovich-Keil, Steven Wang, Claire J. Tomlin, and Anca D. Dragan. Probabilistically safe robot planning with confidence-based human predictions. *CoRR*, abs/1806.00109, 2018. URL <http://arxiv.org/abs/1806.00109>.

Shashank Ojha and Yufei Wang. Verified cruise control system on rc vehicle, Dec 2019. URL [https://lfcps.org/course/lfcps19/projects/shashano\\_yufeiw2.pdf](https://lfcps.org/course/lfcps19/projects/shashano_yufeiw2.pdf).

Ishan Pardesi and Dhruv Mahajan. Formal verification of v2i aided autonomous driving: A hybrid systems approach, Dec 2018. URL <https://lfcps.org/course/lfcps18/projects/dmahajan-ipardesi.pdf>.

André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Cham, 2018. ISBN 978-3-319-63587-3. doi: 10.1007/978-3-319-63588-0. URL <http://www.springer.com/978-3-319-63587-3>.

Mattia Segù, Antonio Loquercio, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *CoRR*, abs/1907.06890, 2019. URL <http://arxiv.org/abs/1907.06890>.

# Appendix I - Experimental Results

Avg. Ensemble Std. Dev.	0.149
Ensemble MSE	0.030
Avg. Dropout Std. Dev.	2.442
Dropout MSE	0.253

## Appendix II - Standard Deviation Statistics

Std. Dev.	Ensemble	Dropout
1	93.20%	63.53%
2	99.53%	82.47%
3	99.93%	86.00%
4	99.93%	86.60%
5	100%	86.80%

```

def gen_data(num_samples, seed=None, print_sample=False):
    """
    Returns a tuple of (data, labels) of our synthetically generated data
    """
    features = ["dist_to_curb", "direction", "speed", "red"]
    label = ["time_to_cross"]

    cols = features + label
    if seed:
        gen = default_rng(seed)
    else:
        gen = default_rng()

    # in meters, m/s
    max_dist = 3
    max_speed = 2
    prob_red = .2
    shape = (num_samples, 1)

    # one hot encoding of cardinal direction of pedestrian
    # 0 = facing down street, away from us
    # 1 = facing towards street
    # 2 = facing away from street
    # 3 = facing towards us
    num_dirs = 4
    directions = gen.integers(0, num_dirs, num_samples)
    directions_onehot = get_one_hot(directions, num_dirs)

    distances = gen.beta(2, 2, shape) * max_dist
    speeds = gen.beta(2, 2, shape) * max_speed

    red = gen.random(shape) >= (1 - prob_red)

    labels = np.empty(shape)
    for i in range(num_samples):
        labels[i] = (time_to_cross(distances[i], directions[i], speeds[i], red[i]))

    labels = labels.astype(np.single)
    samples = np.concatenate((distances, directions_onehot, speeds, red), axis=1).astype(np.single)

```

# Appendix III - Data Generation Function

# Appendix IV - Sample Data

```
['dist_to_curb', 'direction', 'speed', 'red', 'time_to_cross']
dist_to_curb:[1.86011801] direction:0 speed:[0.47321935] red:[ True] time_to_cross:[2.6205156]
dist_to_curb:[2.86271022] direction:0 speed:[0.93775998] red:[False] time_to_cross:[3.052711]
dist_to_curb:[0.63058479] direction:3 speed:[1.47866603] red:[False] time_to_cross:[0.42645517]
dist_to_curb:[1.35991577] direction:1 speed:[1.31267374] red:[False] time_to_cross:[0.5179946]
dist_to_curb:[1.07116477] direction:0 speed:[1.41596904] red:[False] time_to_cross:[0.75648886]
dist_to_curb:[0.48128463] direction:0 speed:[1.17329629] red:[False] time_to_cross:[0.41019872]
dist_to_curb:[0.09551664] direction:0 speed:[0.41434567] red:[False] time_to_cross:[0.23052405]
dist_to_curb:[0.99714647] direction:2 speed:[0.47440003] red:[False] time_to_cross:[6.305732]
dist_to_curb:[0.94040105] direction:3 speed:[0.78095979] red:[ True] time_to_cross:[0.8027738]
dist_to_curb:[0.99818491] direction:0 speed:[1.26511839] red:[False] time_to_cross:[0.78900516]
dist_to_curb:[2.55843258] direction:2 speed:[0.94679047] red:[False] time_to_cross:[8.106648]
```

# Appendix V - KeYmara Model

```
(canPassSafely(carVel) | canStop(-B, carVel)) &
validStartingPosition(carPosX, pedPosY)

->
[
  /*assign a random accleration and check if it is in correct bounds
  and is safe*/
  {{carAcc := *; ?(canStop(carAcc, carVel) & carAcc < 0 & carAcc >= -B);}
  ++ {carAcc := 0; ?(canPassSafely(carVel));}}

  {carPosX' = carVel, carVel' = carAcc,
  pedPosY' = pedVel, t' = 1 & carVel >= 0}
]
didntCrash(carPosX, pedPosY)
```

Full Model  
Available on  
[Github](#)

# Appendix MMXX - Our use of ML

