Project Proposal - Attacks on Automated Vehicles

Connor Raymond Stewart
University of Waterloo
200 University Ave W, Waterloo, ON N2L 3G1
crstewar@uwaterloo.ca

Abstract

Many recent advances in machine learning are being used to develop AI-piloted, driver-less autonomous vehicles. Despite recent advances in machine learning and AI approaches to autonomous vehicle development, many issues related to autonomous vehicles remain, such as the problems related to the robustness of machine learning models. Planning and perception modules in autonomous vehicles can be fooled into making incorrect inferences about the environment given slight modifications to vehicular sensory input. Therefore, the project aims to research a set of methods to test and evaluate model robustness to conduct a theoretical evaluation to determine the best methods to develop robust autonomous vehicle models.

1 Background

Multiple advances in machine learning have been made, which has helped expand autonomous self-driving AI vehicles' ability to reach a level of autonomy where human intervention on the road is no longer needed. However, safety concerns, including adversarial attacks on both perception-based and planning-based models, have been observed. The following report introduces how machine learning modules are fooled to create a theoretical evaluation of such models.

1.1 Perception

Perception models are susceptible to attack through slight perturbations in image data, and small perturbation on image data has been shown to cause erroneous output from a trained DNN model [1]. Physical attacks like placing black lines on stop signs and roads have been shown to cause erroneous output - which leads to actual movements - in self-driving autonomous vehicles [1]. Bayesian Optimisation with a robust objective function can be used to hijack an autonomous vehicle. It can cause it to deviate from its path onto a path crafted by an attacker [1]. Methods to combat these problems can be explored by analyzing errors associated with the networks, and testing data can be analyzed to determine correct defences.

Submitted to Hongyang Zhang for CS886 Robustness of Machine Learning.

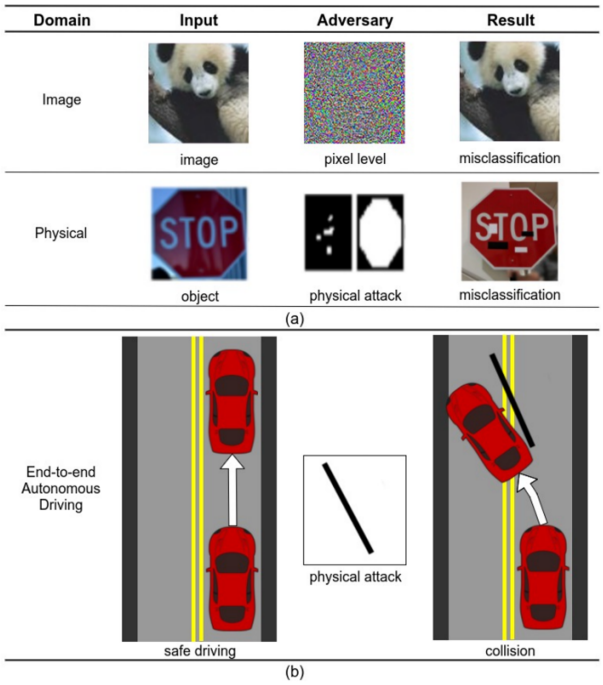


Figure 1: Vision-Based Perception can lead to physical attacks such as placing lines on objects - as seen in (a) - to fool vehicle perception modules into making inaccurate environmental classifications - as seen in (b) - which lead to collisions [1].

Adversarial attacks on sensor cameras and light detection and ranging systems are vulnerable to optimization methods for input perturbation [8]. Discussing defences against algorithmic methods and attempting to evaluate defences against formulas to fool vehicle sensors will be covered in the report.

Environment sensors such as dynamic vehicle sensors (Tire Pressure Monitoring Systems (TPMS), magnetic encoders, and inertial sensors) and environment sensors (Light Detection and Ranging (LiDAR), ultrasonic, camera, Radio Detection and Ranging (Radar) systems, and Global Positioning System (GPS) units) are attainable by cyber-attackers [9]. Attacks and defences for attacks are discussed in detail in the article published by *El-Rewin et al.*, and attack methods like spoofing and jamming are discussed along with machine-learning countermeasures [9].

Various solutions to prevent physical attacks on perception-based vehicle systems have been elaborated and include the development of active and passive algorithms to detect attacks and mitigate their effects on a system [6]. Exploring these topics is undertaken in the project to determine models that can be used to create machine learning models to control attacks [6].

Intelligently selected adversarial examples have been explored by *Chowdhury et al.* and shown to increase vehicle robustness [2]. Machine learning methods can monitor for abnormalities in-vehicle monitors, and robustness methods can defend against input irregularities [2].

Reasons behind perturbation attack effectiveness (ghost object detection) and mitigation strategies to prevent such attacks - such as using Spatial Pyramid Pooling - are elaborated upon [5].

1.2 Planning

PlanFuzz is a dynamic testing system used to discover vulnerabilities associated with vehicle planning module-based behaviours when presented with environmental obstacles on a roadway, such as bikes or boxes [10]. Adversarial examples such as roadside debris can cause a vehicle to accidentally detect a roadblock or road sign on the reads [10]. A system of constraints for the attacker-introduced physical objects (the planning invariant) is introduced [10]. Furthermore, a planning invariant aware physical object generation system to enforce driving rules and a behavioural planning vulnerability

distance system to measure how close the current planning decision is to violating PI and thus trigger a vulnerability in the run time is introduced [10]. Considering that the vehicle's planning system can be attacked by placing otherwise benign objects on the road in strategic locations, developing a robust learning system which omits benign objects or plans around their existence is necessary to prevent adversarial attacks on the planning module [10].

Fine-tuning the victim against the adversary by training against adversaries that engage in seemingly random or bizarre behaviour can benefit self-driving cars. Adversarial examples on deep reinforcement learning models show that zero-sum games can be lost by confusing an AI agent by having an adversary play seemingly random choices [3]. If extended to autonomous vehicles, it is essential to realize the problems with a driver on the road who intentionally behaves irrationally. A vehicle cannot make irrational driving decisions in the face of an adversarial driver. The system presented in the paper lays the groundwork for robustness against poor drivers on the road.

AVMaestro presents an efficient and effective policy framework to allow autonomous vehicles' safe execution [11]. Systems to allow for detecting anomalies on the roads, along with defensive methods to avoid bad decision-making in their presence, are discussed in detail [11].

1.3 Testing and Evaluation

The appropriate methods to plan, test, and debug autonomous vehicles with machine learning concepts are characterized by a lack of agreement in the industry. A framework for testing and evaluating machine learning systems is presented by *Tuncali et al.* by converting arrays to test combinations of discrete parameters and simulating annealing to find corner cases [4]. Understanding more methods to test robustness is vital to evaluating the performance of machine learning models to determine the best methods to make vehicles more robust.

General defences against adversarial examples for deep neural networks and algorithms to defend against adversarial attacks can be read in more detail in the paper written by *Papernot et al.* [7].

2 Introduction

In the project, theoretical analysis and or an algorithmic design are proposed. An theoretical analysis inspired by and using the data from the proposed literature sources, along with an algorithmic analysis consisting of a explanation of the various methods, will be written up and tested for the paper summary and presentation. The project aims to study then which methods are most effective and provide quantitative and qualitative analysis to determine how models operate. Results from papers will be evaluated, and theoretical and algorithmic explanations of the results from authors will be created.

Therefore, the project's purpose is not to produce a working adversarial defence system but to evaluate existing defence systems to determine how well they perform in general. Code sources from GitHub produced by article authors will be evaluated and reproduced, and tests will be run to determine approaches to machine learning robustness evaluation.

Results will include explanations to empirical data generated from literature sources to determine how machine learning robustness methods compare to one another in the autonomous vehicle system context. Algorithms are elaborated on past how they are explained in articles, and an algorithmic analysis may be undertaken to elaborate on the best process discovered for evaluating or enforcing robustness.

In the project herein, three problems will be evaluated. The first is the reproduction of and theoretical analysis of previously created empirical results of perception-based attacks. The second problem is an algorithmic analysis and explanation of methods for fooling self-driving vehicle planning modules and testing their robustness rate. Finally, the third problem is a theoretical analysis of the different systems for testing adversarial robustness.

The next section of the paper focuses on detailed information regarding the previous works used for the project, with the following section focusing on the methodology behind the project's three problems. Results of the evaluation performed from the empirical analysis will be presented in a final section after the methodology.

3 Previous Works

Multiple topics are covered in the project, including perceptron module attacks, planning module attacks, and testing and evaluation of attacks. Perceptron module attacks focus exclusively on the perception, classification, and identification of objects by autonomous vehicle sensors. Topics include vision-based perceptron attacks, perturbation attacks, physical sensor attacks, and supporting topics. Planning module attacks focus on planning module vulnerabilities which can lead to abnormal driving decisions based on adversarial vehicle behaviour, planning around physical obstacles in an autonomous vehicle driving environment, and adversarial policy attacks. Finally, testing and evaluation of attacks include the methodology behind the planning, testing, and debugging of attacks on autonomous vehicles and general defences for neural networks against adversarial inputs.

3.1 Perceptron Module Attacks

The perceptron module can be attacked to fool a vehicle into misinterpreting sensory input. Such misinterpretations can cause vehicles to make situationally inappropriate choices. We will investigate algorithms and data evaluating such attacks.

Vision-Based Perceptron Attack Several objective functions are used to evaluate the algorithms presented in the vision-based perception system presented by *Boloor et al.* Essentially, attacks are a set of length, width, and rotation angles, and attacks are generated through random or grid search functions which are evaluated with objective functions.

Algorithm 1 Adversary Search Algorithm

Require: Strategy $\in$ Random, Grid
  $i \leftarrow 0$
  MetricsList $\leftarrow$ [ ]
  loop
    $\delta_i \leftarrow$ GenerateAttack(Strategy)
    results $\leftarrow$ RunScenario(δ_i)
    $y_i \leftarrow$ CalculateObjectiveFunction(results)
    MetricsList.append(y_i)
    $i \leftarrow i + 1$
  end loop
  return arg max MetricsList

Figure 2: The RunScenario function runs a simulation and returns vehicle data, and the CalculateObjectiveFunction calculates an objective function with the results of the RunScenario function [1]

Larger pattern spaces benefit enormously from Bayesian optimization since it does not require gradient information.

$$z = \frac{\mu_{f|\mathcal{D}}(\delta) - y_{\max}}{\sigma_{f|\mathcal{D}}(\delta)};$$
$$u(\delta) = (\mu_{f|\mathcal{D}}(\delta) - y_{\max})\Phi(z) + \sigma_{f|\mathcal{D}}(\delta)\phi(z),$$

Figure 3: The authors presented the following closed-form solution. [1]

Bayesian optimization's lower runtime complexity leads the authors to create a second algorithmic method for determining an adversary search.

Algorithm 2 Bayesian Adversary Search Algorithm

$i \leftarrow 0$
MetricsList $\leftarrow$ [ ]
loop
  $\delta_i \leftarrow \arg\max u(\delta)$
  results $\leftarrow$ RunScenario(δ_i)
  $y_i \leftarrow$ CalculateObjectiveFunction(results)
  MetricsList.append(y_i)
  Update Gaussian Process and $\mathcal{D}$ with (δ_i, y_i)
  $i \leftarrow i + 1$
end loop
return $\arg\max$ MetricsList

Figure 4: The optimized adversarial search algorithm [1]

The authors use objective functions based on several parameters. The objective function used in the algorithmic implementations by the authors are presented below:

$$\vec{\Theta}_\delta = [\theta_{F_l}, \theta_{F_{l+1}}, \cdots, \theta_{F_{l+\Delta}}]$$

Figure 5: Predicted Steering Vector [1]

$$\vec{\mathrm{p}}_\delta = [(x_l, y_l), (x_{l+1}, y_{l+1}), \cdots, (x_{l+\Delta}, y_{l+\Delta})]$$

Figure 6: Change in position (x, y) across frames in a given attack [1]

$$\begin{aligned}
\text{Collide Right}: &\max_{l,\delta} \sum_{i=0}^{\Delta} \vec{\Theta}_{\delta_i} \\
\text{Collide Left}: &\min_{l,\delta} \sum_{i=0}^{\Delta} \vec{\Theta}_{\delta_i} \\
&\text{subject to}: \quad l \in L, \quad \delta \in S.
\end{aligned}$$

Figure 7: Steering angle summation objectives for the min-max sum of steering angles [1]

$$\begin{aligned}
&\max_{l,\delta} ||\vec{\Theta}_\delta - \vec{\Theta}_{\text{baseline}}||_1 \\
&\text{subject to}: \quad l \in L, \quad \delta \in S.
\end{aligned}$$

Figure 8: Absolute Steering Angle Differences for the most significant change in steering angles [1]

$$\begin{aligned}
&\max_{l,\delta} ||\vec{p}_\delta - \vec{p}_{\text{baseline}}||_2 \\
&\text{subject to}: \quad l \in L, \quad \delta \in S.
\end{aligned}$$

Figure 9: Path Deviation for optimizing deviation from baseline path making [1]

We make a note of the following empirical data as presented by the authors. When modifying the sum of steering angles, we notice that the safe driving rate only reduces significantly for left turns.

Path deviation works best at reducing safe driving for straight car movements, and only halves safe driving for left and right turns. The absolute steering difference parameter almost removes all safe driving for left and right movements and strongly reduces safe driving for right movements. The absolute standard steering difference generates a very high collision rate for right-hand moves.

	Left				Straight				Right			
Metric	safe	collision	offroad	opp. lane	safe	collision	offroad	opp. lane	safe	collision	offroad	opp. lane
$\sum$ st. angles	18.2	0	0	81.8	99	0	0	1	72.2	9.5	13.8	24.5
path deviation	64.6	0	0	35.4	23.8	2.5	2.8	76.2	57.2	24.0	28.3	40.2
abs. st. diff.	**0.2**	0	0	**99.8**	**22.7**	**7.5**	**9.3**	**77.3**	**0**	**95.2**	**99.2**	**100**

Figure 10: Comparing candidate objective functions shows that absolute standard difference works the best [1]

Results show that Bayesian optimization finds a more significant number of unique successful adversaries in the same number of iterations as random and naive grid searches but also finds these adversarial attacks faster.

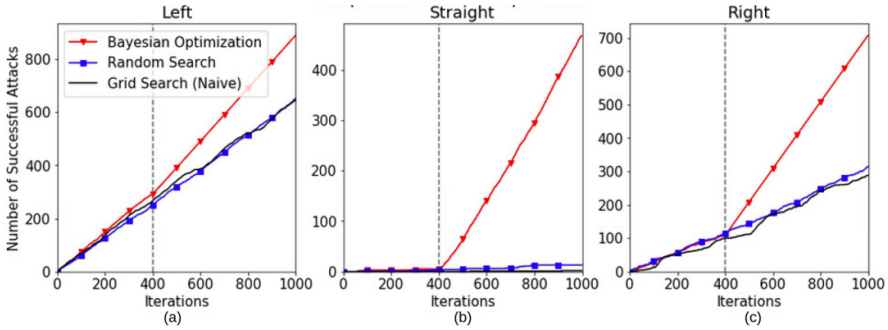


Figure 11: Bayesian optimization (red) shows strong results when compared to other search optimizers [1]

Finally, when measuring adversaries against right-turn driving, the number of infractions is proportional to the change in the summation of steering angles. Furthermore, we notice patterns that cause the minimum steering sum and maximum collision to look similar. Therefore, there is a pattern between minimizing the steering sum and causing the autonomous vehicle to make an infraction.

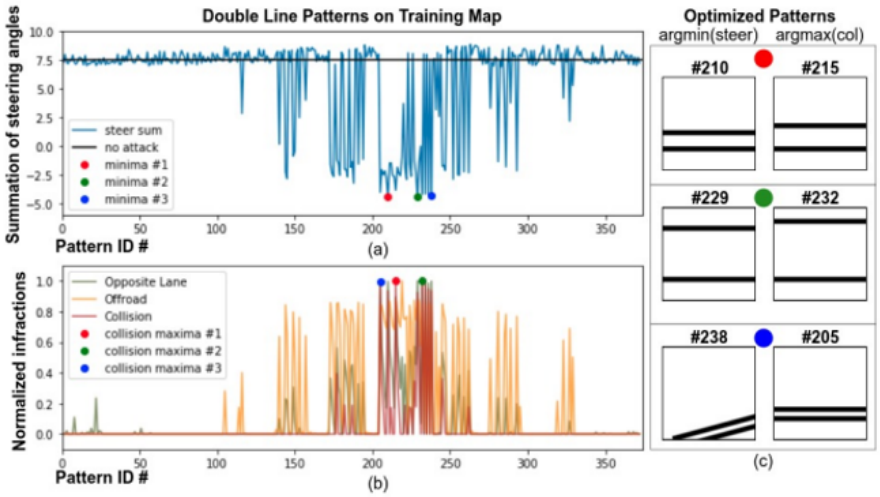


Figure 12: Adversary analysis against right-turn driving to compare summation of steering angles with the normalized infractions shows that patterns which cause minimum steering sum and maximum collisions are similar [1]

Perturbation Attacks The algorithm presented by Zhang *et al.* attempts to add pixel sized changes to make deep learning models dysfunctional is the purpose of the system [5]. We can consider a system where a perturbation is updated using projected loss gradient of a 3D object detector via multiple iterations.

$$\delta_n^{\text{per}} = \text{Clip}_\epsilon\{\alpha \times \text{sign}(\nabla_{(I_l,I_r)} L(O_\theta(\tilde{I}_{l,n}^{\text{per}}, \tilde{I}_{r,n}^{\text{per}}), b^{\text{true}}))\}$$
$$(\tilde{I}_{l,n+1}^{\text{per}}, \tilde{I}_{r,n+1}^{\text{per}}) = (\tilde{I}_{l,n}^{\text{per}} + \delta_n^{\text{per}}, \tilde{I}_{r,n}^{\text{per}} + \delta_n^{\text{per}})$$

Figure 13: Projected loss gradient of a 3D object detector via multiple iterations [5]

Furthermore, we can represent a patched image pair as follows:

$$\underset{\delta^{\text{pat}}}{\text{argmin}}\, \mathbb{E}_{(I_l,I_r)\sim\mathcal{I},(loc_l,loc_r)\sim\mathcal{L},\tau\sim\mathcal{T}} L(O_\theta(\tilde{I}_l^{\text{pat}}, \tilde{I}_r^{\text{pat}}), b^*)$$

Figure 14: Patched image pair minimization [5]

As can be seen in the below table, the driving safety performance metric under the perturbation attack reveals that the average success rate is lower as the iteration value and alpha value increase. Stereo R-CNN models have a noticeably more significant change of a collision on the right turns when attached when compared with the DSGN model, when alpha equals zero-point-four. We also notice that the Stereo R-CNN model has a noticeably greater chance of collision on the right, left, and straight vehicle movements when attacked compared with the DSGN model, when alpha equals one.

We note that alpha represents the attack intensity, n represents the number of iterations, the iteration counter represents the planned trajectory, DGSN means Deep Stereo Geometry Network, and R-CNN means Region-Based Convolutional Neural Network [5]. Furthermore, we notice that the success rate represents the percent chance of success that a vehicle trajectory is generated, the collision rate represents the percent chance that a trajectory leads to a collision, and the safe driving rate is the percentage chance that a trajectory leads to a collision-free outcome [5].

DRIVING SAFETY PERFORMANCE METRICS UNDER THE PERTURBATION ATTACK ($\alpha = 1$)

	Model	DSGN					Stereo R-CNN				
	Iteration	Unattacked	1	2	3	4	Unattacked	1	2	3	4
Success rate (%)	Left	89.6	90.0	89.3	89.8	89.5	89.8	90.2	88.4	89.8	84.6
	Straight	96.5	96.7	96.5	96.5	96.9	96.4	95.8	94.2	95.5	90.6
	Right	84.4	84.4	84.4	84.6	84.8	84.9	85.7	85.1	86.7	82.4
Collision rate (%)	Left	2.2	2.6	2.8	3.4	3.4	3.1	3.3	3.2	4.7	6.4
	Straight	0.7	0.7	0.9	1.3	1.3	1.1	1.5	1.2	2.1	3.5
	Right	1.7	3.2	3.8	5.3	5.9	3.3	4.1	5.1	8.6	11.3
Safe driving rate (%)	Left	87.7	87.7	86.7	86.7	86.4	87.0	87.2	85.5	85.5	79.2
	Straight	95.8	96.0	95.6	92.9	95.6	95.3	94.3	93.1	93.4	87.4
	Right	83.0	81.7	81.1	80.1	79.7	82.1	82.1	80.7	79.2	73.0

DRIVING SAFETY PERFORMANCE METRICS UNDER THE PERTURBATION ATTACK ($\alpha = 0.4$)

	Model	DSGN					Stereo R-CNN				
	Iteration	Unattacked	1	2	3	4	Unattacked	1	2	3	4
Success rate (%)	Left	89.6	89.6	89.5	89.3	89.5	89.8	89.8	90.4	90.0	88.8
	Straight	96.5	96.7	96.7	96.7	96.9	96.4	96.0	95.1	95.1	94.5
	Right	84.4	84.2	84.4	84.6	84.8	84.9	85.3	84.4	85.9	85.2
Collision rate (%)	Left	2.2	2.2	2.4	2.6	2.8	3.1	2.7	3.7	4.1	3.7
	Straight	0.7	0.7	0.7	0.9	1.1	1.1	1.1	0.9	1.3	1.7
	Right	1.7	2.3	2.7	3.6	3.8	3.3	3.5	3.5	5.4	7.2
Safe driving rate (%)	Left	87.7	87.7	87.3	86.9	86.9	87.0	87.4	87.0	86.3	85.4
	Straight	95.8	96.0	96.0	95.8	95.8	95.3	94.9	94.1	93.8	92.9
	Right	83.0	82.2	82.0	81.5	81.5	82.1	82.3	81.4	81.2	79.1

Figure 15: Driving Safety Rates when under Perturbation Attacks [5]

In the tables below, it is essential to note that AP is an abbreviation for average precision [5]. The scenario represents the difficulty of a benchmark, random attacks are performed by placing trained patches at a random position in the image, and specific attacks are performed by placing trained patches randomly within a specific region of the image [5].

AVERAGE PRECISION FOR 3D OBJECT DETECTION UNDER PATCH ATTACK

	Model	DSGN					Stereo R-CNN				
	Scenario	Unattacked	Random Attack	Specific Attack			Unattacked	Random Attack	Specific Attack		
				Left	Straight	Right			Left	Straight	Right
3D Detection AP (%)	easy	70.94	63.85	65.72	64.96	64.87	56.47	53.17	48.14	47.82	50.07
	moderate	52.98	48.20	51.47	50.77	51.63	38.20	37.07	36.27	35.23	38.02
	hard	47.29	44.30	46.80	46.15	46.48	32.66	31.88	31.21	30.60	32.45

DRIVING SAFETY PERFORMANCE METRICS UNDER THE PATCH ATTACK

	Model	DSGN			Stereo R-CNN		
	Scenario	Unattacked	Random Attack	Specific Attack	Unattacked	Random Attack	Specific Attack
Success rate(%)	Left	89.6	90.0	90.0	89.8	70.3	83.7
	Straight	96.5	96.5	96.7	96.4	81.2	57.5
	Right	84.4	84.6	84.8	84.9	68.7	74.4
Collision rate(%)	Left	2.2	2.6	2.8	3.1	1.9	2.1
	Straight	0.7	0.7	0.9	1.1	1.4	4.7
	Right	1.7	2.3	2.1	3.3	3.9	4.2
Safe driving rate(%)	Left	87.7	87.7	87.5	87.0	68.9	81.9
	Straight	95.8	95.8	95.8	95.3	80.1	54.7
	Right	83.0	82.6	83.0	82.1	65.9	71.2

Figure 16: Driving Safety Performance Metrics under Patch Attacks [5]

We notice that random and specific attacks affect the average precision for 3D object detection under patch attacks. Hard scenarios appear to be much more robust than easy and moderate scenarios. DGSN models appear to be more robust to specific attacks, whereas stereo R-CNN models are more robust to random attacks. Random and specific attacks do not affect the safe driving rate of DSGN models very much, but random and specific attacks appear to increase the collision rate for left and right turns when used against DSGN models. Notably, Stereo R-CNN models show a dramatic reduction in safe driving when turning left or right under random attacks or turning right during specific attacks. We also notice a significant drop in safe driving rate under both random and specific attacks for straight vehicle movements. A moderate reduction in safe driving when left turning under specific attacks is also noted. The Stereo R-CNN model shows a significant increase in collision rate under specific attacks when moving straight or right. Interestingly, under both attacks, the R-CNN model's collision rate drops. The DSGN model is comparatively more resistant to random attacks than specific attacks, whereas the Stereo R-CNN model is comparatively more resistant to specific attacks when compared to random attacks.

In the top table below, specific attack scenarios are very close to the same intention cases, whereas different intentions are close to attacked scenarios [5]. In the middle diagram, ground truth indicates if an object is moving [5]. A patch attack significantly affects a small number of pixels, whereas a perturbation attack slightly affects many pixels [5]. In the mid-table, DSGN models are very robust under perturbation and patch attacks, showing only slight drops in safe driving rates under all attacks. The Stereo R-CNN model shows large drops in safe driving rates under perturbation attacks and substantial drops in safe driving rates under patch attacks. In the final table, we notice that greedy breadth-first searches (GBFS) have lower ground truth safe driving rates than A* planning algorithms when planning left, and right turns. Ground truth data of 3D object detection is used as inputs for motion planning algorithms [5]. Therefore, the planning algorithm can affect the self-driving vehicle's safe driving rate and robustness.

DRIVING SAFETY PERFORMANCE METRICS OF STEREO R-CNN UNDER THE PATCH ATTACK WITH VARIOUS INTENTIONS

	Specific attack	Random attack		
	—	Same intentions[1]	Different intentions[2]	Unattacked
Success rate (%)	71.9	76.2	91.6	90.4
Collision rate (%)	3.5	2.3	1.5	2.4
Safe driving rate (%)	69.3	74.4	90.1	88.1

[1] "Same intentions" refers to cases where the attack intention and the driving intention are the same.
[2] "Different intentions" refers to cases where the attack intention differs from the driving intention.

SAFE DRIVING RATE WITH DIFFERENT INPUTS

	Model	—	DSGN			Stereo R-CNN		
	Scenario	Ground Truth	Unattacked	Perturbation Attack	Patch Attack	Unattacked	Perturbation Attack	Patch Attack
Safe driving rate (%)	Left	89.7	87.7	86.4	87.7	87.0	79.2	68.9
	Straight	98.0	95.8	95.6	95.8	95.3	87.4	80.1
	Right	85.2	83.0	79.7	82.6	82.1	73.0	65.9

SAFE DRIVING RATE USING DIFFERENT PLANNING ALGORITHMS

	Planning algorithm	GBFS	A*
	Scenario	Ground Truth	
Safe driving rate (%)	Left	87.9	89.7
	Straight	98.0	98.0
	Right	82.3	85.2

Figure 17: Safe Driving Rates under Various Types of Attacks [5]

Physical Sensor Attacks Understanding the types of defences against algorithmic methods and attempting to evaluate defences against formulas to fool vehicle sensors is required to prevent sensory-based attacks. We note that Cao *et al.* and Raj Gautam Dutta (2018) use complex derivations and linear algebraic models for camera input to model attacks in their algorithmic systems. Complex algorithms for attack generation and defence are created to prevent attacks on vehicular sensors.

Algorithm 1: Generating adversarial examples by leveraging global spatial transformation

input: Target model: $\boldsymbol{M}$;
3D point cloud $\boldsymbol{X}$;
3D spoofed 3D point cloud $\boldsymbol{T}$;
Optimizer $\boldsymbol{opt}$;
Max iteration $\boldsymbol{N}$;
output: 3D adversarial 3D point cloud $\boldsymbol{X'}$;

Initialization: $\theta \leftarrow 0$, $\tau_x \leftarrow 0$, $s_h \leftarrow 1$, $l_{min} = +\text{inf}$, $x = \Phi(X), t = \Phi(T)$;
/* Initiate parameters by sampling around the transformation parameters $Target_\theta$, $Target_{\tau_x}$ that transforms t to the target position (px, py) of the attack */
for $i\tau_x \leftarrow -L_t$ **to** L_t **do**
  for $i\theta \leftarrow -L_\theta$ **to** L_θ **do**
    /* Initialize parameter . */;
    $\theta \leftarrow Target_\theta + i\theta$, $\tau_x \leftarrow Target_{\tau_x} + \tau_x i$;
    for $iter \leftarrow 1$ **to** N **do**
      /* Calculate adversarial loss */;
      $l_{\text{adv}} \leftarrow$ **Equation** 7.;
      /* Update the parameters θ, τ_x, s_h based on optimizer opt and loss l_{adv} */
      ;
      $\theta, \tau_x, s_h \leftarrow opt(l_{\text{adv}}; \theta, \tau_x, s_h)$
      if $l_{min} < l_{adv}$ **then**
        $\theta^{final}, \tau_x^{final}, s_h^{final} \leftarrow \theta, \tau_x, s_h$
      end
    end
  end
end
$T' \leftarrow G_T(\theta^{final}, \tau_x^{final}, s_h^{final}; T)$;
$X' \leftarrow X + T'$;
Return: T'

Figure 18: Algorithmic method for generating adversarial examples by Coe *et al.*

Algorithm 3 Attack Detection & Resilient State Estimation

Input: Observation $\{y'_k\}_{k\geq 1} \in \mathbb{R}^q$; detection threshold η; model parameters $A, B, C, \Sigma_v, \Sigma_w, \{u_k\}_{k\geq 0}$.
Output: Estimated values $\hat{x}_k, k \geq 1$
Initialize: $\hat{x}_0 \in \mathbb{R}^n$ and $P_0 \in \mathbb{R}^{n\times n}$;

for $k = 0, 1, 2, \cdots$ **do**
  Calculate $\hat{x}_{k+1|k}$ and $P_{k+1|k}$ using (4.5).
  Apply χ^2 detector to
  $g_{k+1} = (y_{k+1} - C\hat{x}_{k+1|k})^T (CP_{k+1|k}C^T + \Sigma_v)^{-1}(y_{k+1} - C\hat{x}_{k+1|k})$
  if $g_{k+1} > \eta$ **then** $P_{k+1} = P_{k+1|k}$ and
  $\hat{x}_{k+1} = \hat{x}_{k+1|k}$.
  else Calculate P_{k+1} and $\hat{x}_{k+1}$ using (4.6) and (4.7)
  end if
  Return: $\hat{x}_{k+1}$.
end for

Figure 19: Raj Gautam Dutta's (2018) attack detection method for adversarial attack generation

Results from LiDAR spoofing attacks show that the number of points increases robustness in machine learning models [8]. As can be seen in the image below, the number of spoofed points decreases robustness (as it results in an eighty-two percent success rate). When forty points are spoofed, fewer spoofed points lead to an eighty-seven percent success rate and sixty spoofed points leads to a ninety percent success rate [8]. LiDAR is used to make 3D representations of an area using data points collected using laser beams. We can inject spoofed data points into a LiDAR by firing lasers into the LiDAR's 2D camera's field of view [8]. As can be seen, the system can be fooled, and models should be trained with spoofed data points to increase model robustness.

Targeted position	# Spoofed points		
	20	40	60
2-8 meters	87%	82%	90%

Figure 20: Number of spoofed points over the accuracy rate for LiDAR spoofing attacks [8]

3.2 Planning Module Attacks

The planning module can be attacked to fool a vehicle into making poor decisions. Poor decisions making can prove fatal for an autonomous vehicle. We will investigate algorithms and data evaluating such attacks.

Planning Module Vulnerabilities In the paper by Wan *et al.*, Simplified pseudo-code is analyzed and tested to find vulnerabilities in the planning module.

```
# Iterate over the obstacle list
for (each obs : obstacle_list):
  # Judge based on the lateral position of a veh.
  # start_l, end_l are veh's left/right boundaries
  if (obs.end_l < -2.5 or obs.start_l > 2.5):
    continue
  # The backward safe buffer
  BackwardSafeBuffer ← 4.0f
  # Check whether the veh is close
  if (ego_start_s-obs.end_s<BackwardSafeBuffer):
    IsClearChangeLane ← false
```

Figure 21: The authors iterate over the obstacle list and test to find obstacles with incorrect behavioural responses [10]

A vulnerability occurs when any attack target position is triggered [10]. As can be seen, triggering objects can influence agent behaviour to cause agents to execute incorrect planning behaviour. The violated PI is the violated planning invariant (which is the output behaviour given a set of vehicle, obstacle, and pedestrian positions.

Vuln #	Driving Scenario	Software	Violated PI #	Attack-influenced Planning Behavior	Triggering Objects
V1	Lane following	Apollo 3.0/5.0	PI1 (PI-C1)	Permanent stop	Static obstacle
V2	Lane changing	Apollo 3.0/5.0	PI3 (PI-C2, 3)	Fail to change lane and never reach destination	Vehicle
V3	Lane borrow	Apollo 3.0/5.0	PI4 (SP-PI-C1, 2)	Fail to borrow the lane and permanent stop	Vehicle or static obstacles in front of blocking vehicle
V4	Lane borrow	Apollo 3.0/5.0	PI4 (PI-C1, 4, 5)	Fail to borrow the lane and permanent stop	Off-road static obstacle
V5	Intersection w/ traffic signal	Apollo 3.0/5.0	PI6 (PI-C4)	Fail to pass intersection and permanent stop	Static pedestrian
V6	Intersection w/ traffic signal	Apollo 3.0/5.0	PI6 (PI-C5)	Emergency stop; possible to cause permanent stop and thus fail to pass intersection	Pedestrian who is leaving the intersection
V7	Intersection w/ stop sign	Apollo 3.0/5.0	PI5 (PI-C1)	Fail to pass intersection and permanent stop	Static bicycle off the road
V8	Lane following	Autoware	PI1 (PI-C1)	Permanent stop	Static obstacle off the lane
V9	Lane following	Autoware	PI1 (PI-C3)	Emergency stop	Moving vehicle off the lane

Figure 22: A list of driving scenarios violated planning invariants (PI's) based on a given triggering object which results in new attack-influenced planning behaviour [10]

Adversarial Policy Attacks The policy consists of a player playing against an opponent in a two-player Markov game. Adversaries are given unlimited black-box actions. Attackers must solve the given single-player Markov Decision Process [3]:

$$T_\alpha\left(s, a_\alpha\right) = T\left(s, a_\alpha, a_\nu\right) \qquad \text{and} \qquad R'_\alpha(s, a_\alpha, s') = R_\alpha(s, a_\alpha, a_\nu, s')$$

Figure 23: Single-Player Markov Decision Process formulation for the AI two-player zero-sum games presented [3].

The goal of an attacking agent is to find adversarial policies such that the sum of discounted rewards is maximized, as we can see below [3]:

$$\sum_{t=0}^{\infty} \gamma^t R_\alpha(s^{(t)}, a_\alpha^{(t)}, s^{(t+1)}), \quad \text{where } s^{(t+1)} \sim T_\alpha(s^{(t)}, a_\alpha^{(t)}) \text{ and } a_\alpha \sim \pi_\alpha(\cdot \mid s^{(t)})$$

Figure 24: Maximization for the sum of discounted rewards [3].

We note that the figure below shows the win rates against the medium victim in each AI-competitive environment [3]:

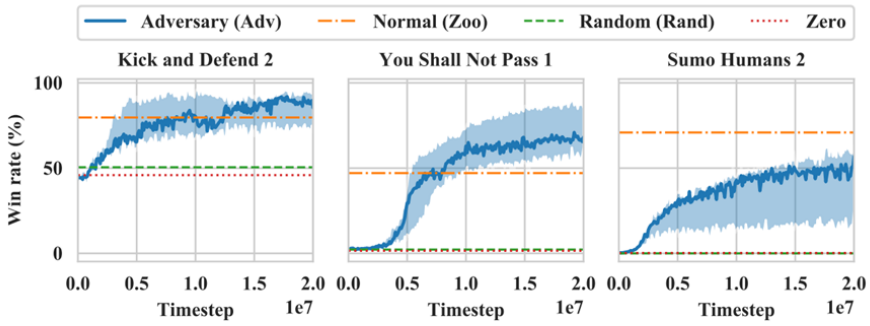


Figure 3: Win rates while training adversary `Adv` against the median victim in each environment (based on the difference between the win rate for `Adv` and `Zoo`). The adversary outperforms the `Zoo` baseline against the median victim in *Kick and Defend* and *You Shall Not Pass*, and is competitive on *Sumo Humans*. For full results, see figure 4 below or figure C.1 in the supplementary material.

Figure 25: Win-Rates while training adversary against the median victim in each zero-sum game [3]

Finally, the figure below shows the percentage of games won by an opponent given the policy used by the opponent [3]. Please note that zoo refers to the pre-trained baseline, zero refers to the lifeless policy with zero control and refers to the random action policy, and adv refers to the adversarial policy:

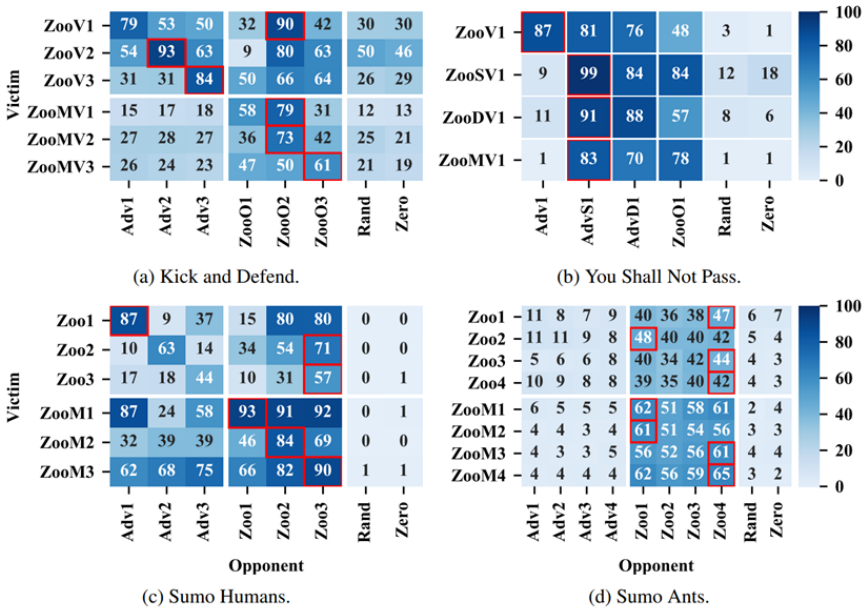


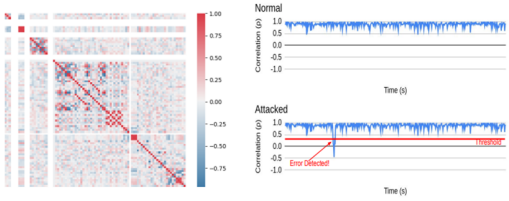
Figure 26: As can be seen, carefully calculated mistakes can be used to fool AI models to eliminate good agents from competitive tasks [3]

Planning Around Physical Obstacles The table below shows the selected variables for attack injection. Correlated variables indicate a connection between properties measured by physical sensors and expected vehicle behaviour. If a vehicle's machine learning based planning-module is trained using sensor data, then correlations between sensor values are mapped within the machine learning neural network. Changing a sensor value via an adversarial attack can trigger invalid inferences from the neural network since it expects sensor values to be correlated for given planning-module-based decisions. Note that FPR denotes the false positive rate for spoofing detection, and correction denotes the sensor correlation (a correlation is significant if it has a value greater than a half) [11].

Variable 1	Variable 2	Correlation	FPR
position.y	linear_velocity.y	-0.741	8.77%
position.z	orientation.qz	0.801	12.65%
orientation.qy	euler_angles.y	0.877	3.18%
position_std_dev.x	orientation_std_dev.x	0.845	2.22%
orientation.qw	euler_angles.y	-0.801	6.54%
linear_velocity.x	euler_angles.z	-0.964	2.55%
linear_acceleration.x	linear_acceleration_vrf.x	0.899	4.66%
angular_velocity.z	angular_velocity_vrf.z	0.999	0.3%
orientation_std_dev.z	linear_velocity_std_dev.z	0.699	9.20%
linear_velocity_std_dev.x	linear_velocity_std_dev.y	0.915	0.83%

Figure 27: Correlation values between vehicular sensor variables [11]

Figure (a) below shows 184 variable pairwise correlations in a matrix. Figure (b) shows we can cause errors in run-time correlation, which can compromise a vehicle's planning module. Therefore, there is empirical evidence of neural network models being compromised when correlated variable pairs have a variable which is attacked [11].



(a) (b)

Figure 28: Correlation between variables (a) and resulting compromised vehicular planning (b) during adversarial sensor attacks [11]

3.3 Testing and Evaluation of Attacks

We explore various methods centred around the testing and evaluation of attack methods for autonomous vehicles. Understanding what makes a vehicle vulnerable to attack, how to plan around attacks, test vulnerabilities, and debug vulnerabilities is essential to understand attacks on autonomous vehicle models.

Plan, Test, and Debugging Robustness values close to zero represent cases where the vehicle collides with an object at low speeds. The probability distribution is a truncated normal distribution. Global Uniform Random Search (Global UR) randomly selects test cases for simulation and checks if traces satisfy constraints [4]. Covering Arrays and Uniform Random Search (CA+UR) creates a list of test cases and evaluations against specifications [4]. Covering Arrays and Simulated Annealing (CA+SA) combines covering arrays to evaluate discrete variables and simulates annealing, so a function is used to guide the search of continuous variables [4].

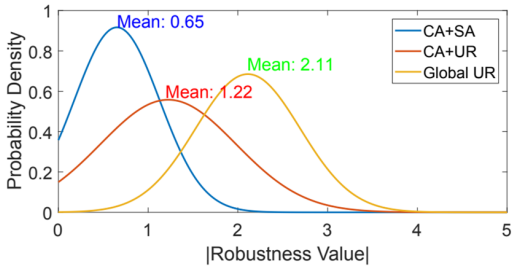


Figure 29: We can see that the CA+SA system shows the best results given its probability distribution clustering close to zero on the x-axes

Defences for Neural Networks Algorithms can generate adversarial examples to reduce confidence in classification, misclassify images, make targeted misclassifications, and cause source-target misclassifications [7].

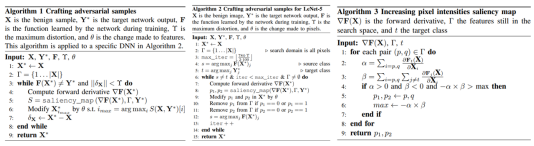


Figure 30: Algorithms to generate and test adversarial examples [7]

Adversarial examples can be used to create quantitative metrics for sampling robustness.

Source set of 10,000 original samples	Adversarial samples successfully misclassified	Average distortion	
		All adversarial samples	Successful adversarial samples
Training	97.05%	4.45%	4.03%
Validation	97.19%	4.41%	4.01%
Test	97.05%	4.45%	4.03%

Figure 31: Empirical results of adversarial testing [7]

4 Remarks

As previous authors have significantly elaborated on the data behind attacks, the remainder of the paper will focus on a theoretical conclusion of the meaning behind the attacks.

4.1 Perceptron Attacks

Perceptron models appear vulnerable to attacks since they lack a proper grounding in the meaning behind the object shapes they detect. Baloor *et al.* showed in their work that physical attacks such as placing lines on objects could be used to fool neural network models into making environmental misclassifications. Algorithms to detect attacks and train machine learning models under adversarial attacks have been proposed to prevent them, but they do not resolve the problems associated with perceptron models not understanding underlying properties to distinguish objects. Determining if a perceptron learning algorithm converges on a correct shape without defining the shape leaves it vulnerable to attacks based on point and line-based distortion. Vehicles driving on roads are vulnerable to attacks since they learn by approximating shape outlines, meaning they cannot become perfectly robust when using neural network models. Lines and patches added to roads and stop signs cause object misclassification by creating incorrect inferences for polygons within the perceptron model.

4.2 Planning Attacks

Planning attacks seem generalizable because they are vulnerable at points between vehicle goal setting, vehicle action planning, and vehicle action completion. Vehicles can incorrectly plan goals based on invalid perceptron input leading to inaccurate actions being made on the road. Furthermore, adversarial drivers can interfere with planning systems since they can provide otherwise unexpected input to driving vehicles, leading to undefined driving behaviours. Planning networks are trained by determining optimal moves given regular input, meaning training networks with a global set of rules to safeguard against adversarial drivers could be beneficial. Multiple network models for different aspects of the road, like distance from the curb, can be used to create a priority of perception-based plans. To ignore obstacles along the road, multiple sets of sensors to determine objects outside the road to be ignored can be added.

5 Conclusion

There are many ways to fool a planning module for a machine learning system. Determining how the planning module and perceptron module are interrelated and determining gives excellent insight into the meaning behind machine learning models. Determining how we test planning modules given the large number of choices they are presented with is an essential question for future researchers. Finally, evaluating if our methods are trustworthy and how we can collect more data on adversarial vehicle attacks is an important step forward for future research.

References

[1] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking vision-based perception in end-to-end autonomous driving models," Journal of Systems Architecture, vol. 110, p. 101766, 2020.

[2] A. Chowdhury, G. Karmakar, J. Kamruzzaman, A. Jolfaei, and R. Das, "Attacks on self-driving cars and their countermeasures: A survey," IEEE Access, vol. 8, pp. 207308–207342, 2020.

[3] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial Policies: Attacking Deep Reinforcement Learning," 2019.

[4] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," 2018 IEEE Intelligent Vehicles Symposium (IV), 2018.

[5] J. Zhang, Y. Lou, J. Wang, K. Wu, K. Lu, and X. Jia, "Evaluating adversarial attacks on driving safety in vision-based Autonomous Vehicles," IEEE Internet of Things Journal, vol. 9, no. 5, pp. 3443–3456, 2022.

[6] R. G. Dutta and Y. Jin, "Security of autonomous systems under physical attacks: With application to self-driving cars," dissertation, STARS, Orlando, FL, 2018.

[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," arXiv, 2015.

[8] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on Lidar-based perception in autonomous driving," Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019.

[9] Z. El-Rewini, K. Sadatsharan, N. Sugunaraj, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity attacks in vehicular sensors," IEEE Sensors Journal, vol. 20, no. 22, pp. 13752–13767, 2020.

[10] Z. Wan, J. Shen, J. Chuang, X. Xia, J. Garcia, J. Ma, and Q. A. Chen, "Too afraid to drive: Systematic discovery of semantic DOS vulnerability in autonomous driving planning under physical-world attacks," Proceedings 2022 Network and Distributed System Security Symposium, 2022.

[11] Z. Zhang, S. S. V. Singapuram, Q. Zhang, D. K. Hong, B. Nguyen, Z. M. Mao, S. Mahlke, and Q. A. Chen, "AVMaestro: A Centralized Policy Enforcement Framework for Safe Autonomous-driving Environments."

Checklist

See the questions and answers below:

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [N/A]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? **[TODO]**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[TODO]**