Robustness to Input Corruptions and Adversarial Examples in Steering Angle Prediction via Self-Supervision

Bibek Poudel*
University of Memphis
Memphis, TN 38111
bpoudel@memphis.edu

Taylor Michael Villarreal*
University of Memphis
Memphis, TN 38111
tmvllrrl@memphis.edu

Ryan Wickman*
University of Memphis
Memphis, TN 38111
rwickman@memphis.edu

Abstract

Robustness gains from self-supervision have only been recently discovered with promising results. However, the work has been limited to classification tasks. This project explores the use of self-supervision as a pretraining step in a target supervised learning task of steering angle prediction (regression). We aim to make model output invariant to changes brought by common input corruptions (naturally occurring) and adversarial examples (generated by an attacker). In addition, the task dataset poses unique challenges like high class imbalance and high correlation among image sequences.

1 Introduction

Due to the inherent assumption of independent and identically distributed (i.i.d) data in machine learning (including computer vision), it is implied that the data seen by models at training and test times are drawn from the same distribution. However, that rarely occurs when models are deployed in the wild; co-variate shift [19] may occur under natural [11, 17] or adversarial [20, 9] conditions causing state-of-the-art deep learning models to perform poorly even when perturbations that are imperceptible to humans are added to the input. However, in many applications, model robustness under co-variate shift is equally desired as model performance in "regular" conditions [8, 21]. This desire is pronounced, especially in areas where the consequences of incorrect decisions taken by deep learning models can be fatal, e.g., autonomous driving [6] and healthcare [7].

In the past, robustness against naturally occurring (common) input corruptions has been achieved at the cost of sample complexity (a large number of data augmentations) [16, 13, 23] whereas robustness against adversarial corruptions incurs significant computational overhead (gradient-based, iterative) [15]. Even then, there have been mixed results on their success; some works report accuracy trade-off when robustness is increased [] whereas others report that increase in robustness to one type of corruptions simultaneously leads to a decrease in robustness to the other []. More recently, however, self-supervised pre-training has effectively established itself as an alternative to "robustify" a model under, among other conditions, common input corruptions and adversarial examples [12]. In addition to robustness, it has also been shown to address the previously mentioned issues, i.e., achieve faster convergence and better accuracy [4].

In this work, we experiment with the idea of using self-supervised pre-training to ① evaluate robustness against common and adversarial input corruptions, and ② denoise corrupted inputs in steering angle prediction task. The supervised dataset consists of a sequence of (image, steering angle) pairs obtained from real-world driving sequences and the corruption scheme is defined as various benchmarking datasets by [18]. We make use of the simCLR framework [5] to pre-train

*equal contribution
COMP 8118 Data Mining, Fall 2021. (LaTeX template borrowed from NeurIPS 2020.)

(during which we don't use angle labels) and fine-tune a ResNet50 model [10]. We compare the robustness results with various models such as a standard ResNet50 (with no pre-training) and a state-of-the-art gradient-free approach [18]. Further, we use the pre-trained ResNet50 as an encoder in an Autoencoder architecture to denoise the corrupted images.

2 Related Work

Hendrycks et al. [12] found that self-supervision can benefit several aspects of model robustness. On the CIFAR-10-C dataset [11], they obtained a accuracy of 95.5% (higher than 94.7% via standard training) on clean data and average accuracy of 76.9% (higher than 72.3% via standard training) on data corrupted by common corruptions . Similarly, they outperformed adversarial training with PGD (state-of-the-art adversarial defense technique) by 5.6% on ImageNet using rotation angle prediction as a self-supervised task. More recently, Chen et al. [4] validated robustness to adversarial examples from self-supervised pretraining in CIFAR-10 dataset. To make robust training computationally efficient, the pretraining step included a combination of self-supervision and adversarial training. Further, the simCLR framework has also been used in pretraining. Jiang et al. [14] use the SimCLR framework to introduce *Adversarial Contrastive Learning (ACL)* that learns representations that are consistent under both data augmentations and adversarial perturbations.

This work uses a denoising autoencoder. Autoencoders [2] are a type of neural network architecture, comprised of an encoder, a bottleneck, and a decoder, that allows for the neural network to automatically encode data and then reconstruct it back. The reconstruction is compared to the original input, and the difference/error is backpropagated through the network. The bottleneck has less dimensions than the original input in order to force the neural network to generalize to the data being fed forward, rather than just copy the data through the encoder to the decoder. Due to the fact that no labels are being used, just the data itself, autoencoders are an unsupervised learning model. In mathematical notation, the encoder is a function $g_\theta(.)$, the bottleneck is z, and the decoder is a function $f_\phi(.)$ where θ parameterizes $g(.)$ and ϕ parameterizes $f(.)$. If there is some input x, then $z = g_\theta(x)$ and the reconstruction x' equals to $f_\phi(g_\theta(x))$ In 2008, Vincent et al. [22] developed the denoising autoencoder. A denoising autoencoder works the same way as a regular autoencoder except for the fact that the original input to the network is first augmented in some way. Then, the augmented image is passed through the autoencoder. The reconstruction is then compared to the original input without augmentations in order to train the autoencoder to denoise the data. Vincent et al.'s work also found that training an autoencoder to denoise images results in the trained autoencoder being more robust to corruptions.

3 Preliminaries

In this section, we introduce the self-supervised learning algorithm used in the pretraining step along with algorithms used to generate adversarial examples in the robustness task. In addition, we define curriculum learning, which is used while training the Autoencoders for denoising task.

3.1 SimCLR Framework

The SimCLR framework was designed to make it simple to do effective self-supervised contrastive learning. Because it is self-supervised, the data does not need to have any labels attached to it as they are unnecessary for training. It being contrastive learning means that SimCLR will find what data is similar to other data and what data is dissimilar to other data. That is the main idea behind contrastive learning: maximizing the similarity between data that are indeed similar to each other and minimizing the similarity between data that are not similar to each other. The pipeline of the SimCLR framework starts with taking a batch of images, let's say of size N. Then, two random transformations are applied to each image, resulting in pairs of images, in the batch to get a new batch size of $2N$. Each pair of images is passed through an encoder. The original SimCLR paper uses a ResNet50 model [10] for the encoder, which is what this project uses as well. This results in representations h_i and h_j for each of the images in the pair. The pair are then passed through a projection head that is made up of several non-linear layers. This results in another pair of representations z_i and z_j from the original image. The similarity between z_i and z_j are calculated using cosine similarity. The cosine similarities are calculated for all possible pairs within the batch (that is the batch with size $2N$), which are then

used in the loss function. The loss function is called NT-Xent loss. Describing NT-Xent begins with the Noise Contrastive Estimation (NCE) loss, which is equation (1). This is where the calculation of the similarities for all possible pairs can be seen as this occurs in the denominator.

$$l(i,j) = -log\frac{exp(s_{i,j})}{\sum_{k=1}^{2N} l_{[k!=i]} exp(s_{i,k})} \tag{1}$$

The $s_{i,j}$ term represents the cosine similarity between the representations z_i and z_j while the $s_{i,k}$ term represents the cosine similarity between z_i and all other representations z_k where i cannot equal k. NCE loss is used in NT-Xent loss, which is equation (2).

$$\mathcal{L} = \frac{1}{2N}\sum_{k=1}^{N}[l(2k-1,2k) + l(2k,2k-1)] \tag{2}$$

Equation (2) calculates the loss over all pairs in the batch and takes an average over them. The main steps of the algorithm that have just been described are summarized in Algorithm 1, which comes from the SimCLR paper. This loss function is what is used for backpropagation in order to update the network's parameters.

After training the SimCLR model (which is the encoder and the projection head), the projection head can be removed and just the encoder is used for downstream tasks or transfer learning or pre-training. This is the process that was used in this project in order to get the pre-trained SimCLR ResNet50 encoder that is used in other architectures and neural networks.

Algorithm 1 SimCLR's Main Learning Algorithm

input: batch of size N, encoder function f, project function g
for sampled mini-batch **do**
  for all $k \in$ 1,...,N **do**
    Select 2 random augmentations functions t and t'
    z_i = $g(f(t(x_i)))$
    z_j = $g(f(t(x_j)))$
  end for
  for all i and $j \in$ 1,...,$2N$ **do**
    Calculate cosine similarity
  end for
  Calculate $\mathcal{L}$ using Equations 1 and 2
  Update f and g parameters to minimize $\mathcal{L}$
end for
return f for downstream tasks

3.2 Adversarial Examples

An adversarial example $\mathbf{x}^*$ for a deep learning model f is crafted by imposing minimal perturbation $\delta_\mathbf{x}$ to an original signal $\mathbf{x}$, where $\mathbf{x}^*$ is be obtained by solving the following optimization problem [20]:

$$\begin{aligned} minimize \quad & D(\mathbf{x}, \mathbf{x}+\delta_x) \\ s.t. \quad & f(\mathbf{x}) \neq f(\mathbf{x}^*), \end{aligned} \tag{3}$$

where D is a distance metric such as L_0, L_2 or L_∞. Multiple methods can be used to solve Eq. 3. Two of them are examined in this work; the Fast Gradient Sign Method (FGSM) [9] and Projected Gradient Descent (PGD) [15]. FGSM is designed to produce an adversarial signal in a fast, non-iterative manner. It computes the gradient of the cost function J w.r.t input $\mathbf{x}$ and scales the sign of the gradient by a L_∞ constraint for generating δ:

$$\mathbf{x}^* = \mathbf{x} + \varepsilon \cdot sign\left(\nabla_x J\left(\theta, \mathbf{x}, y\right)\right), \tag{4}$$

where θ represents the parameters of f; y is the target label; and ε is the L_∞ constraint parameter controlling the attack magnitude. Increasing the value of ε will increase the attack effectiveness but

also make the adversarial example more distinguishable (i.e., having large $\delta_{\mathbf{x}}$). PGD is a stronger attack based on iteratively performing FGSM on smaller steps. Formally, the iterative procedure follows:

$$\mathbf{x}^*_{l+1} = \prod_{\mathbf{x}+S} \mathbf{x}^*_l + \alpha \cdot \nabla_x J(\theta, \mathbf{x}, y) \tag{5}$$

where Π denotes the projection operator, which clips the input at the positions around the predefined perturbation range. means a gradient step size, and $\mathbf{x} + S$ represents the perturbation set. Usually, L_∞ and L_2 norms are currently used as the constraint that bound the perturbation to be small.

3.3 Curriculum Learning

Curriculum learning is an idea based on how humans typically learn things in school. In school, humans first learn easy topics when they first start school and then gradually increase the difficulty of topics that they learn as the progress through school. The researchers in [3] found that this same concept can be applied to machine learning. There is no specific way to apply curriculum learning as it is merely a concept, but the idea is train a model by starting on easy inputs for the model to learn. Then, gradually increase the difficulty of the inputs that the model sees as the training progresses. When to increase the difficulty can be based on as the model's loss decreases or based on the number of epochs that have passed and so on. It was found that curriculum learning can significantly improve generalization performance and lead to faster convergence.

4 Methodology

Robust Regressor: All three models used in robustness task have a ResNet-50 [10] as the backbone model. The first is a ResNet-50 with no modifications to the original architecture trained using Mean Squared Error (MSE) loss function, referred to as the "Standard" model. The second, also a ResNet-50 architecture, however, trained using the robustness approach proposed in [18], referred to as the "Robust". Finally, the third Resnet-50 model is pre-trained using simCLR learning algorithm (Algorithm 1) and NT-Xnet loss (Equation 2) followed by fine tuning similarly to the Standard model. The third model is referred to as "simCLR" in the rest of the text. The experiment results on Standard and Robust models are averaged over two variations trained with different randomization of initial weights. Although all the models were trained for equal number of total epochs (500), simCLR model only used labels for the 100 epochs of fine tuning.

Denoising Autoencoder. The methodology behind the denoising autoencoder is to train the autoencoder to denoise images by feeding it batches of augmented images so that it can encode them and decode them. This results in a batch of reconstructions. These reconstructions are compared against the images that do not have any augmentations applied to them (so clean images) and then, that error is backpropagated through the autoencoder. The way that this was done for this project will now be described. During training, a group of clean images would be taken and a total of 15 augmentations would be applied to each image. For example, if the group of clean images contained four clean images, then there would be a total of 60 augmented images. Those 60 augmented images form a batch of inputs that get passed through the autoencoder. This results in 60 reconstructions that then get compared to the original four clean images (to make the sizes of the batches match, the four images would be copied 15 times each to create a batch of 60 clean images). The goal of the autoencoder is to minimize the Mean Squared Error (MSE) loss over a series of 50 epochs.

The 15 augmentations that were used to train the denoising autoencoder are: R, G, B, H, S, V altering (these involved altering these channels to increase or decrease the values so there is actually 12 augmentations instead of just the six listed), distortion of the image, blurring the image, and adding Gaussian noise to the image. Each augmentation has five associated levels attached to it such that one can make the augmentation more or less intense. For example, Gaussian noise at level one does not add as much Gaussian noise at level 5 to an image. Curriculum learning was employed throughout the training of the autoencoder such that at the start of training, the autoencoder was only learning how to denoise images with less intense augmentations, but as the autoencoder improved, more intense augmentations were given to the autoencoder.

After training the autoencoder, the autoencoder would then be tested on 3 test datasets that are based on the Honda Research Institute driving dataset. The goal is to evaluate how well the reconstructions

are (that is, how well denoised the images are); however, in order to do this, a different metric is used. This metric is the Mean Accuracy metric that was used in the previous sets of experiments from above. The pipeline for experiments is: the augmented images from the clean, combined, and unseen test sets are passed through the autoencoder being tested on, then the reconstructions are passed through the Standard model, which outputs a steering angle prediction. The steering angle predictions are then compared across the different autoencoder models being tested to see which autoencoder was able to denoise the best.

The reasoning behind feeding the reconstructions through the Standard model is because each autoencoder is trained to reconstruct the image such that the image is closer to the original (or clean) image. The Standard model was trained on only clean images so for the experiments involving the autoencoders, it is essentially being used as a black box. The Robust model was not selected to be used for the pipeline due to the fact that it was trained with more than just clean images. While this could have improved the overall results of the autoencoders, the goal was to evaluate the robustness of just the autoencoders. Since the Robust model is already robust, it would have thrown off the evaluation of the autoencoders. Also, all of the autoencoder models used Mean Squared Error (MSE) loss, which results in reconstructions that are blurry. This means that some of the difference in Mean Acurracy results could be from the blurriness from the reconstructions; however, all of the autoencoders (while they may not have the same level of blurriness) return blurry reconstructions.

For this project, three autoencoder models were experimented/tested on. They are referred to as the Simple AE, Resnet50 AE, and the SimCLR AE. More detail is given about them in the Experiments section.

5 Experiments

This section details the various experiments performed in both the tasks (robustness and denoising). Three models each (Standard, Robust, and simCLR) on robustness task and (Simple AE, ResNet50 AE and SimCLR AE) denoising task are evaluated across different validation sets. For all results, benchmark validation sets and the evaluation metric (Mean Accuracy) are obtained from [18].

Setup: Two standard ResNet-50 models were trained on a Intel core i9 $-$ 11900K with 32G RAM and a Nvidia-RTX 3070 GPU for 500 epochs whereas, two robust ResNet-50 models were robust pre-trained for 400 epochs and fine-tuned for 100 epochs on a Intel core i7 $-$ 11700K with 32G RAM and a Nvidia-RTX 3080 GPU. The simCLR ResNet-50 model was pre-trained for 200 epochs in Lambdalabs [1] cloud server with a Nvidia-RTX A6000 GPU and 100G RAM, it was fine-tuned for 100 epochs in a RTX 3080 for both the robustness and denoising tasks. Both the Simple AE and the ResNet-50 AE were trained on Intel core i7 $-$ 11700K with 32G RAM and a Nvidia-RTX 3080 GPU. The SimCLR AE was trained using Google Colab with a Nvidia-Tesla P100 GPU with 16G RAM. All AEs were trained for 50 epochs.

Model	Mean Accuracy
Standard	**95.01**%
Robust	93.91%
SimCLR (ours)	90.72%

Table 1: Robustness task; mean accuracy of various models on clean validation set. The standard trained model has the highest mean accuracy on clean validation set whereas the simCLR trained model has the lowest.

Results on robustness: From Table 1, we observe that the Standard model (95.01%MA) outperforms Robust model (93.91% MA) and SimCLR model(90.72%) on clean validation set (no input corruptions present). From Table 2 however, the Standard model performs the worst among the three models in all possible combinations (six combinations of common input corruptions applied at various levels), i.e., robustness achieved by Robust and simCLR models in combined perturbation set comes at the cost of clean accuracy. The simCLR model outperforms others in combinations two, three, and four, whereas the Robust model outperforms others in combinations one, five, and six. In combinations one and six, the Robust model severely outperforms ($> 30\%$) the simCLR model whereas in combination five, it marginally outperforms by 5%.

Model	Comb.1	Comb.2	Comb.3	Comb.4	Comb.5	Comb.6
Standard	86.84%	88.70%	90.82%	89.20%	88.76%	80.65%
Robust	**94.91**%	95.12%	90.65%	88.05%	**94.86**%	**94.52**%
SimCLR (ours)	58.17%	**95.67**%	**91.64**%	**93.86**%	89.82%	47.21%

Table 2: Robustness task; mean accuracy of various models on data with combined (seen) perturbations in six different levels. SimCLR trained model performs the best in combinations 2,3, and 4 whereas, the Robust trained model performs best in all other combinations.

From Table 3, on ImageNet-C based corruptions (seven naturally occurring corruptions applied at five levels), simCLR model does not outperform Standard and Robust models in any combination of corruptions. Except for Pixelate and JPEG Compression compression at levels one, two, and three under all corruption schemes, the Robust model outperforms others. This result implies that the

Sn.	Perturbation	Level 1	Level 2	Level 3	Level 4	Level 5
Standard	Motion Blur	92.85%	91.22%	89.04%	86.6%	85.12%
	Zoom Blur	90.58%	89.35%	88.29%	87.7%	87.3%
	Pixelate	**94.22**%	**94.22**%	**93.84**%	93.50%	93.22%
	JPEG Compression	**94.20**%	**93.74**%	**93.57**%	92.80%	92.12%
	Snow	78.48%	63.75%	62.67%	59.94%	67.08%
	Frost	72.97%	76.55%	79.64%	79.54%	81.28%
	Fog	75.35%	74.27%	73.47%	73.39%	73.19%
Robust	Motion Blur	**92.91**%	**91.63**%	**89.72**%	**87.56**%	**86.36**%
	Zoom Blur	**92.28**%	**91.61**%	**91.07**%	**90.68**%	**90.41**%
	Pixelate	93.86%	93.91%	93.76%	**93.87**%	**93.91**%
	JPEG Compression	93.74%	93.58%	93.55%	**93.27**%	**92.94**%
	Snow	**88.95**%	**88.99**%	**89.90**%	**91.18**%	**90.24**%
	Frost	**88.14**%	**89.71**%	**90.12**%	**90.64**%	**90.87**%
	Fog	**86.88**%	**86.97**%	**86.64**%	**86.13**%	**85.78**%
SimCLR (ours)	Motion Blur	89.97%	89.08%	87.94%	86.23%	85.28%
	Zoom Blur	88.53%	87.65%	87.01%	86.81%	86.66%
	Pixelate	90.49%	90.49%	90.36%	90.30%	90.19%
	JPEG Compression	90.58%	90.38%	90.42%	90.46%	90.33%
	Snow	80.15%	82.72%	80.05%	76.40%	77.64%
	Frost	81.64%	80.17%	78.70%	77.75%	75.87%
	Fog	62.67%	56.49%	53.96%	55.22%	54.17%

Table 3: Robustness task; mean accuracy of various models on unseen perturbations at five levels. Out of the 35 total tests, Robust trained models perform the best in 29 and Standard trained models perform the best in 6.

Method	Model	$\epsilon = 0.01$	$\epsilon = 0.025$	$\epsilon = 0.05$	$\epsilon = 0.075$	$\epsilon = 0.1$
FGSM	Standard	**93.57**%	91.11%	88.92%	85.58%	81.42%
	Robust	92.39%	**92.67**%	**92.78**%	**92.84**%	**92.82**%
	SimCLR (ours)	90.65%	90.26%	89.57%	88.35%	87.48%
PGD	Standard	**93.55**%	91.05%	88.89%	85.96%	80.59%
	Robust	92.52%	**92.83**%	**93.07**%	**93.09**%	**93.28**%
	SimCLR (ours)	90.59%	90.26%	89.49%	88.36%	87.53%

Table 4: Robustness task; mean accuracy of various models on validation set with Adversarial Examples from FGSM and PGD in five levels. On both algorithms, Robust model performs the best at $\epsilon = 0.025, 0.05, 0.075, 0.1$ whereas, Standard model performs best at $\epsilon = 0.01$.

simCLR pre-training approach did not significantly affect out-of-distribution performance at test time. In addition, the simCLR model performed worse than the standard model at all levels in cases under Fog.

Table 4 shows the mean accuracy when adversarial examples are crafted to the input images (from the clean set) using the FGSM and PGD algorithms at five levels. Attack strength increases as we move from the left column to the right; under both attack algorithms, the Standard model performs the best under least attack strength ($\epsilon = 0.01$), whereas, for all other attack strengths Robust model consistently maintains the highest performance within $\pm 1\%$. Further, the Standard model suffers from the highest performance drop under a strong attack (accuracy drops to 81.42% under FGSM and drops to 80.59% under PGD). In contrast, for the simCLR model, the performance drops to 87.48% under FGSM and 87.53% under PGD at the same attack strength.

Results on denoising: For some context for the proceeding results, the Simple AE model is an autoencoder that is comprised of a total of 10 layers: five layers for the encoder and five layers for the decoder. It is labeled as Simple AE because of this simple architecture. It uses ReLU activation for every layer except for the last layer where Sigmoid activation is used in order to get the values of the output to be in the range [0,1]. The ResNet50 AE model is comprised of a ResNet50 encoder and then a five layer decoder (the same five layers from the Simple AE). The SimCLR AE model is composed of a pre-trained SimCLR ResNet50 encoder model and then the same five layer decoder from the previous autoencoder models. The reasoning behind just switching the encoder models is because SimCLR specifically results in a pre-trained ResNet50 encoder model. We wanted to evaluate the impact of using this encoder model when it comes to the task of denoising, so in order to make sure the experiments were fair, we just changed the encoder model being used and used the same decoder for every autoencoder model.

The three autoencoder models were first evaluated on their ability to reconstruct the clean test dataset, which can be seen in Table 5. This was to evaluate their ability to denoise and reconstruct images that did not have any noise or augmentations to begin with. The Simple AE is able to denoise the clean images the best. The SimCLR AE was able to do better than the ResNet50 AE meaning that the pre-training could have helped with the autoencoder's ability to denoise the image.

Model	Mean Accuracy
Simple AE (ours)	**88.89**%
ResNet50 AE (ours)	86.32%
SimCLR AE (ours)	86.87%

Table 5: Denoising task; mean accuracy of various autoencoder models on clean validation set. The Simple AE results in the best reconstructions of the images.

Model	Comb.1	Comb.2	Comb.3	Comb.4	Comb.5	Comb.6
Simple AE (ours)	**88.89**%	**77.30**%	**88.31**%	**86.29**%	80.23%	**84.61**%
ResNet50 AE (ours)	83.16%	76.75%	84.38%	84.35%	**83.92**%	82.67%
SimCLR AE (ours)	84.40%	72.93%	86.41%	84.25%	70.53%	76.35%

Table 6: Denoising task; mean accuracy of various autoencoder models on data with combined (seen) perturbations in six different levels. The Simple AE performs the best on five of the combined datasets with the ResNet50 AE performing the best on the remaining one combined dataset. The SimCLR AE is never the best on any of them, but stays close to the other two models on a couple of the combined sets (for example, Comb.4 and Comb.3).

On the combined datasets, the Simple AE is, once again, the overall best performing model. The SimCLR AE model is able to perform better than the ResNet50 model on a couple of the datasets (Comb.1 and Comb.3), but is never better than the Simple AE model. So overall, the SimCLR AE model is not able to do as well as originally thought.

Finally, when it comes to the unseen datasets in Table 7, the Simple AE performed the best overall and was able to outperform the other two autoencoder models on 25 of the unseen datasets. The

Sn.	Perturbation	Level 1	Level 2	Level 3	Level 4	Level 5
Simple AE (ours)	Motion Blur	**88.53**%	**88.05**%	**87.18**%	**85.99**%	**85.10**%
	Zoom Blur	**88.62**%	**88.30**%	**88.06**%	**87.86**%	**87.53**%
	Pixelate	86.82%	86.83%	86.76%	86.81%	86.78%
	JPEG Compression	**88.82**%	**88.81**%	**88.80**%	**88.91**%	**88.86**%
	Snow	**86.53**%	**85.44**%	**85.46**%	**85.51**%	**85.54**%
	Frost	**84.97**%	**84.70**%	**84.72**%	**84.45**%	**84.28**%
	Fog	79.76%	77.99%	76.84%	76.31%	76.13%
ResNet50 AE (ours)	Motion Blur	86.05%	85.78%	85.43%	84.79%	84.22%
	Zoom Blur	86.25%	86.17%	86.05%	85.90%	85.78%
	Pixelate	86.30%	86.28%	86.20%	86.30%	86.28%
	JPEG Compression	86.29%	86.22%	86.30%	86.44%	86.38%
	Snow	84.47%	81.50%	82.44%	80.01%	77.68%
	Frost	81.98%	79.55%	78.33%	78.66%	77.50%
	Fog	**85.14**%	**85.61**%	**83.06**%	**80.76**%	**78.88**%
SimCLR AE (ours)	Motion Blur	86.52%	86.23%	85.82%	85.26%	84.69%
	Zoom Blur	86.70%	86.62%	86.49%	86.39%	86.27%
	Pixelate	**88.82**%	**88.80**%	**88.80**%	**88.80**%	**88.77**%
	JPEG Compression	86.85%	86.83%	86.86%	86.91%	86.88%
	Snow	82.61%	78.33%	78.75%	77.78%	77.84%
	Frost	79.53%	82.59%	83.26%	83.91%	84.13%
	Fog	82.83%	82.11%	80.27%	78.72%	77.47%

Table 7: Denoising task; mean accuracy of various autoencoder models on unseen perturbations at five levels. Out of the 35 total tests, the Simple AE model performed the best in 25, the ResNet50 AE model performed the best in five, and the SimCLR AE model performed the best in five.

experiments on unseen augmentations is different from testing on the clean or combined datasets because the autoencoder has never seen the augmentations being looked at. This means that the autoencoders are having to denoise augmentations that it does not know anything about. The Simple AE performed the best meaning that it was more robust to unseen augmentations compared to the other two autoencoder models. When comparing the ResNet50 AE to the SimCLR AE, the SimCLR AE performs better than the standard ResNet50 AE. It is able to outperform the ResNet50 AE in 25 datasets out of 35 datasets showing that the SimCLR pre-training can help improve performance; however, that is only in the context of comparing a pre-trained SimCLR ResNet50 encoder to a ResNet50 encoder with random initialization of weights. The fact that the SimCLR AE (and the ResNet50 AE for that matter) were not able to overall outperform the Simple AE is disappointing. There may be a need for further testing such as pre-training the Simple AE's encoder using SimCLR and comparing those results.

6 Conclusion

Following the promising results seen in other domains, in this project, we investigated the use of self-supervised learning using the simCLR framework as a pre-training step on steering angle prediction. Specifically, we examined whether pre-training would make both a Regressor and a denoising Autoencoder (consists of Regressor as encoder) models robust against common and adversarial corruptions. Unfortunately, the approach did not work as expected in this project; except for three combined perturbation sets on the robustness task, we did not see any benefits of using the simCLR pre-training approach. However, these results should only be considered preliminary because we experimented with a small set of possible learning strategies and parameter settings. In the future, we would like to broaden our investigations by implementing learning strategies such as longer pre-training, using task-specific data augmentations, and widening the model architecture.

References

[1] lambdalabs.com, gpu cloud, workstations, servers, laptops for deep learning.

[2] Dana H Ballard. Modular learning in neural networks. In *AAAI*, volume 647, pages 279–284, 1987.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[4] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 699–708, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[6] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[7] Samuel G Finlayson, Hyung Won Chung, Isaac S Kohane, and Andrew L Beam. Adversarial attacks against medical deep learning systems. *arXiv preprint arXiv:1804.05296*, 2018.

[8] Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. Maxup: A simple way to improve generalization of neural network training. *arXiv preprint arXiv:2002.09024*, 2020.

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[12] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *arXiv preprint arXiv:1906.12340*, 2019.

[13] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[14] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. In *NeurIPS*, 2020.

[15] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[16] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] Evgenia Rusak, Lukas Schott, Roland S Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. A simple way to make neural networks robust against diverse image corruptions. In *European Conference on Computer Vision*, pages 53–69. Springer, 2020.

[18] Yu Shen, Laura Zheng, Manli Shu, Weizi Li, Tom Goldstein, and Ming Lin. Gradient-free adversarial training against image corruption for learning-based steering. *Advances in Neural Information Processing Systems*, 34, 2021.

[19] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[20] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[21] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[22] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[23] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

7 Appendix

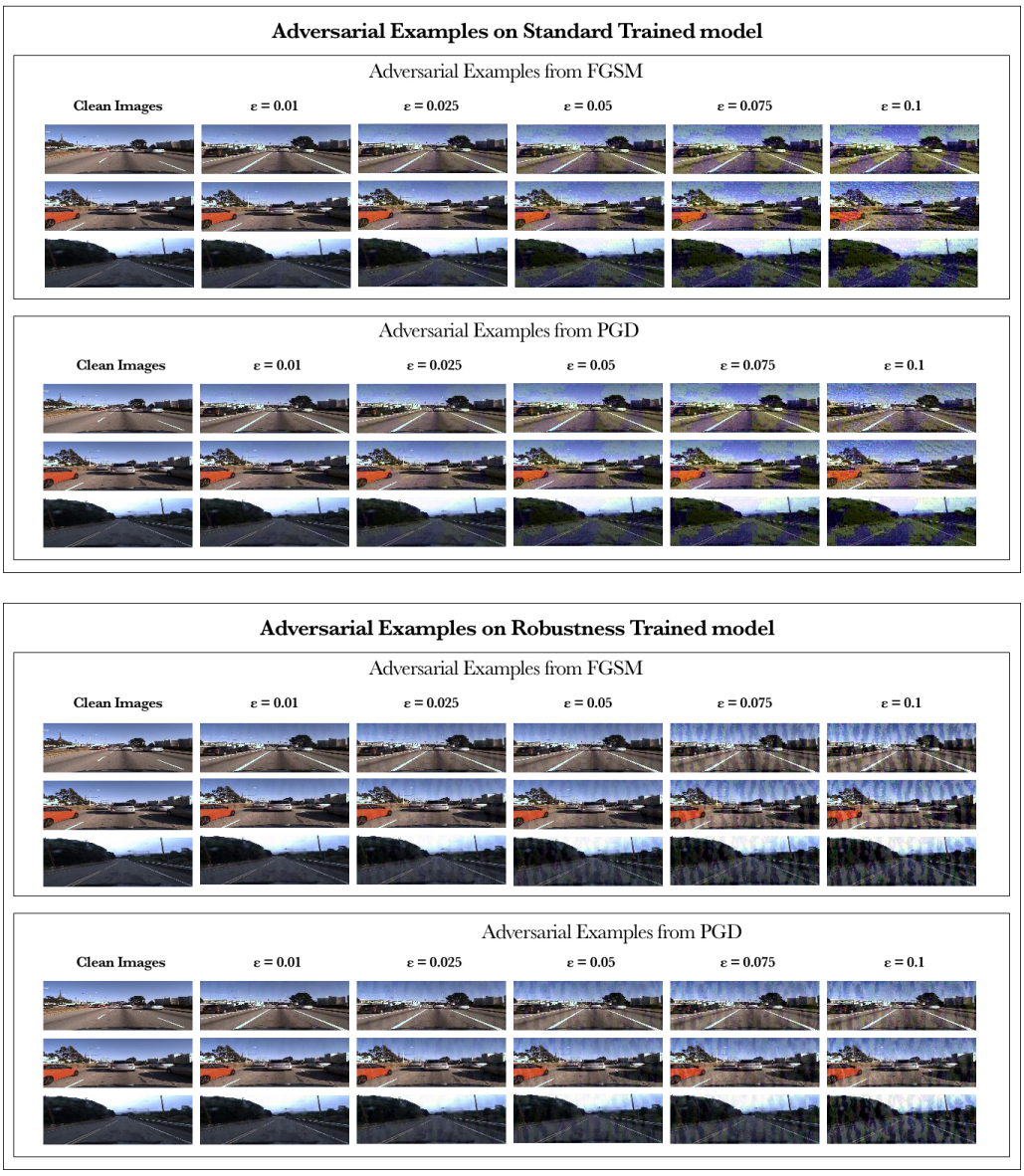

Figure 1: Clean images and their adversarial counterparts at various levels from FGSM and PGD.