

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 9, 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Raghav Kapoor
Casey Nguyen

ENTITLED

Neural Network Interpretability for Autonomous Driving Neural Networks

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE COMPUTER SCIENCE AND ENGINEERING

David C. Anastasiu
David C. Anastasiu (Jun 9, 2022 15:38 PDT)

Thesis Advisor

N. Ling
N. Ling (Jun 9, 2022 16:32 PDT)

Department Chair

Neural Network Interpretability for Autonomous Driving Neural Networks

by

Raghav Kapoor
Casey Nguyen

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 9, 2022

Neural Network Interpretability for Autonomous Driving Neural Networks

Raghav Kapoor
Casey Nguyen

Department of Computer Science and Engineering
Santa Clara University
June 9, 2022

ABSTRACT

In the field of neural networks, there has been a long-standing problem that needs to be addressed: gaining insight into how neural networks make decisions. Neural Networks are still considered black boxes and are often difficult to understand. This lack of understanding becomes an ethical dilemma especially in the domain of self-driving cars. Given the limited number of works geared towards unravelling neural network logic for autonomous driving vehicles, our team seeks to create a novel neural network interpretability method to influence the neural network during its training process.

Deep Neural Networks have demonstrated impressive performance in complex tasks, such as image classification and speech recognition. However, due to their multi-layer structure combined with non-linear decision boundaries, it is hard to understand what makes them arrive at a particular classification or recognition decision given new data. Recently, several approaches have been proposed to understand and interpret the reasoning in a deep neural network. In some state-of-the-art solutions, researchers try to actively improve the network during the training process through the use of penalty functions that added into the chosen layers of the model. However, in many other state of the art solutions, interpretability is done after the model is trained. This means that no modifications are made to the network until after it is fully trained. This results from this method of interpretability often take form of decision trees, extracting logical rules, or highlighting images. However, improvement will have to be done through an additional training process, meaning that more time will be taken. This is often the case with vision-based models such as those used in self-driving neural networks.

Based on previous works done in the field of neural network interpretability, we propose adding a penalty function in the feature extraction layers of an autonomous driving neural network model rather than taking the common passive interpretability approach. Our method is intended to prevent the model from developing complex data representations that are not human-understandable. Our group specifically chose L1 regularization as our penalty function for the purpose of creating sparse feature maps that can ignore noise. Through the use of an autonomous driving simulator and feature extraction methods, we proved that our regularized model was more effective and interpretable than an baseline version of it. To determine model effectiveness, we observed the autonomy of both models. To determine interpretability, we measured the compressibility and randomness of the features learned by both models. For feature compressibility, the Principle Component Analysis (PCA) algorithm was applied from the features extracted from the model. Furthermore, the randomness of the features were calculated using entropy. Our results show that our regularized model was more autonomous and learned more compressible and less random features than the original baseline.

ACKNOWLEDGMENTS

We would like to thank our advisor, Dr. David Anastasiu, for being the greatest advisor in the world. The resources and guidance provided by Dr. Anastasiu have been a tremendous help for this project and our future research pursuits.

We would also like to thank Dr. Nam Ling, the Chair of the Computer Science and Engineering Department, for assisting us in navigating the requirements of the degree that is about to be conferred upon us.

Last, but not least, we would like to thank our families. Without their support, we would have never been able to complete this thesis or pursue our dreams.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Challenges of Implementing Neural Network Interpretability on Self-Driving Vehicles	1
1.3	Solution	2
2	Literature Survey	3
2.1	Current Methods	3
2.1.1	Passive Interpretability	3
2.1.2	Active Interpretability	4
2.1.3	Interpretability Hybrid	6
2.2	Benchmarks	6
3	Methods	8
3.1	Dataset	8
3.2	Self-Driving Model	8
3.3	Regularization	10
3.3.1	Test Environment	11
3.3.2	Data Collection	12
4	Evaluation	13
4.1	Experimental Design	13
4.2	Model Training	13
4.2.1	Evaluation Metrics	14
4.3	Experimental Results	15
4.3.1	Trained Models	15
4.3.2	Autonomy	15
4.3.3	Interpretability	16
5	Future Work	18
5.1	Improving the Dataset	18
5.2	Improving the Self-Driving Model	18
5.2.1	Cost Function	18
5.2.2	Model Input	19
5.2.3	Model Output	19
5.3	Additional Regularization	20
5.3.1	Mutual Information	20
5.3.2	Tree Regularization	20
6	Societal Issues	21
6.1	Ethical	21
6.2	Trust	22
6.3	Safety	22
6.4	Lifelong Learning	23

7	Conclusion	24
7.1	What We Have Learned	24
7.2	Why it is Important	25
A	Additional Model Training Plots	26
A.1	Baseline Model Training Plots	26
A.2	InterpNet Model Training Plots	27

List of Figures

3.1	The graphical representation of the Rectified Linear Unit activation function is shown above.	10
3.2	The graphical representations of two regularization functions are shown above. The left-hand figure (a) represents L1 Norm Regularization and the right-hand figure (b) represents L0 Norm Regularization	11
4.1	Training plots of the optimal models with training and validation Loss are shown above. The left-hand figure (a) depicts the training plot of the optimal baseline model. The right-hand figure (b) depicts the training plot of the optimal InterpNet model.	15
4.2	Compressibility of the feature maps for the optimal models is shown above. The left-hand figure (a) depicts the number of PCA components required to compress the feature maps extracted from the optimal baseline model. The right-hand figure (b) depicts the number of PCA components required to compress feature maps extracted from the optimal InterpNet model.	16
4.3	Entropy/Randomness of the feature maps for the optimal models is shown above. The left-hand figure (a) depicts the entropy of the feature maps extracted from the optimal baseline model. The right-hand figure (b) depicts the entropy of the feature maps extracted from the optimal InterpNet model.	17
A.1	Four additional baseline training plots are shown above for learning rates of $1e-2$ and $5e-3$ on batch sizes 32 and 64.	26
A.2	Six additional InterpNet model training plots are shown above for l1 regularization strengths of $1e-3$ and $5e-2$, learning rates of $5e-4$ and $1e-4$, and batch sizes of 32 and 64.	27

Chapter 1

Introduction

1.1 Motivation

The problem with neural networks in modern-day technology is that they are considered black boxes. When the performance of the model is substandard or unexpected, machine learning engineers have a difficult time figuring out why the neural network produced that prediction. This can be especially problematic in the domain of autonomous driving, since the lack of understanding of a neural network's incorrect decision can present a significant negative impact on the safety of the passengers in those vehicles.

To improve the technology of autonomous cars and make them more widely trusted, we need to be able to understand their decision-making logic and debug the misinterpretations of algorithms. In 2018, a Tesla Model X crashed into a cement wall on the freeway, and the company could not understand the reasoning behind the car's movements. With an interpretability method, a company could trace the path that the data takes in providing a driving decision. This tracing could take place prior to actual deployment of the self-driving system to reduce the probability of such accidents happening in the real-world.

1.2 Challenges of Implementing Neural Network Interpretability on Self-Driving Vehicles

There are two main reasons why autonomous self-driving systems are not inherently interpretable as indicated by Zablocki et al. [1]:

1. Deep learning models may potentially face the limitations that datasets contain numerous biases, are too general, and are not precisely curated. This leads the system to learn from spurious correlation and overfit to certain situations.
2. Self-driving systems have to solve incredibly complex problems. For humans, it may be simple, but it is very difficult for a system to solve related tasks with different environments. The model that led to the prediction is

very chaotic, and we have to be able to navigate it to make it interpretable for humans.

Autonomous driving is a high-stake safety-critical application. From a societal point of view, performance guarantees should be mandatory. However, there are many scenarios where self-driving models are not testable due to the impossibility of listing and evaluating every scenario that a model can encounter. A solution to this issue is to be able to explain a model's decision making process in making certain decisions in a given scenario and pinpoint the area of error.

1.3 Solution

Our team created a solution to improve the interpretability of neural networks on autonomous cars by creating a mechanism that helps interpret the neural network. This occurred as the model was learning, such that the learning process does not compromise prediction accuracy. This mechanism consisted of adding the L1 norm regularization function to our model so that it penalized the neural network during training. The penalty prevented the neural network from developing some chaotic data representation that may deliver high accuracy but diminishes interpretability. In order to verify this, we executed a method of visualizing what features the model has learned based on how input data is transformed by the neural network.

Chapter 2

Literature Survey

In this chapter, we discuss prior works done towards interpreting the decision-making logic used by neural networks towards accomplishing a certain task. Within the domain of neural network interpretability, various benchmarks and interpretability methods have been defined and are organized using the taxonomy defined by Zhang et al. [2]. Within this taxonomy, interpretability techniques can be decomposed into three dimensions.

The first of these three dimensions specifies the type of engagements. This is further broken down into passive and active. Passive engagement introduces interpretability after model training. Active engagement introduces interpretability behavior during model training and actively influencing network or training process

The second dimension is type of explanation. This is decomposed into the following: examples, attribution, hidden semantics, or rules.

The third dimension is the focus of the interpretability method. This is categorized as local, semi-local, or global. Local focus seeks to explain network's prediction based on individual sample. Semi-local focus attempts to explain a group of similar inputs. Lastly, global focus strives to explain the network as a whole.

2.1 Current Methods

2.1.1 Passive Interpretability

As mentioned in the prior section, prior neural network interpretability methods consist of three different attributes. Within the subdomain of passive interpretability, Woh and Liang [3] used a technique from statistics known as influence functions, which are capable of indicating how a change in one data instance can impact an overall estimator. Within this particular work, Woh and Liang applied influence functions to trace the path taken by a training data instance from the model's prediction through the model and back to the original training instance. In order to determine the influence of a particular data instance on a model, approximations of the original influence function were employed due to the expensive nature of performing exact influence calculations on models with millions of parameters across all training points. In the empirical evaluation of this approach on a Support Vector Machine with a Radial Basis Kernel

and the state-of-the-art Inception v3 convolutional neural network, Woh and Liang demonstrated how they were able to understand model behavior, assess model vulnerability to adversarial training examples, debug models, and detect dataset errors. However, since this model mainly focuses on the impact of individual training points towards a model's prediction output under the assumption of minimal model change, it does not possess the capability of explaining global changes made to a network.

Bojarski et al. [4] produced a new method in the field of neural network interpretability of autonomous driving called VisualBackProp. VisualBackProp visualises and determines which set of pixels of the input image made the most contributions to the prediction made by the convoluted neural network(CNN). Due to the fact that this technique only highlights regions of interest and does not change the network during training, we can categorize this as a passive method. Its form of explanations is through providing highlighting visuals and works on a semi-local scale as it operates on a set of pixels rather than individuals.

To test their method, the authors used NVIDIA's neural network system for autonomous driving called PilotNet to determine which elements of the road played the largest factor in influencing its steering direction highlighting the object. Bojarski et al. were able to identify significant sets of pixels that are crucial in steering direction, highlighting features such as cars, lane markers, parked cars, and the edge of roads. It was even able to discern that steering direction doesn't change with a crosswalk and did not highlight it. VisualBackProp successfully provides valuable insights into the decision-making process of the end-to-end learning systems. When it comes to computation speed for calculating masks, VisualBackProp took 2.0ms while another method, LRP, took 24.6ms. Bojarski et al's method was shown to be computationally competent as well as able to provide efficient and accurate visualizations. Although this method is very fast and applicable to real time application, it does not provide any practical information for debugging the original autonomous driving neural network.

2.1.2 Active Interpretability

Within the domain of active interpretability methods, many works have been done to introduce an element or inductive bias or prior knowledge into neural network models. Wu et al. [5] created a new tree regularization method in order to make deep models that can be closely modeled by decision trees with a few nodes, so that the deep models can be human-simulatable. This tree regularization method takes form of a true-average-path-length penalty function, such that the deep model would still maintain its accuracy and avoid complexity, which is defined as long average path length. After evaluating this regularization method on a toy dataset, hospitalized septic patients dataset, HIV therapy dataset, and English stop phonemes dataset, the authors discovered that tree regularized models were able to outperform conventional regularization methods such L1 norm and L2 norm in accuracy, while maintaining interpretability through the use of decision tree proxies. Even with these exemplary results though, this tree regularization technique limited only to input features that are interpretable and would not work on other types of input features, such as pixel

data from images.

Wu et al. [6] investigated activation function regularization using three regularization schemes, KL divergence, Cos system, and smooth system, in order to influence a model's activation functions to behave in accordance with a certain target pattern. They evaluated each of these regularization techniques on Gaussian Mixture Models, Deep Neural Networks with ReLU, tanh, and sigmoid activation functions, and other proposed models across three speech tasks: the Wall Street Journal continuous speech recognition, eight IARPA Babel conversational telephone speech tasks, and the U.S English broadcast news task. Through the use of activation function visualization and word error rate as the metrics for comparison, the authors demonstrated that the proposed regularization schemes continually yielded similar or better performance than the baseline models. Furthermore, they also showed that the KL divergence regularization on the sigmoid activation function consistently yielded the lowest word error rate. While being able to provide visualizations for convolutional neural networks used in speech recognition tasks, these target-pattern-enforcing regularization techniques have not been verified on other neural network architectures such as recurrent neural networks.

Du et al. [7] developed a framework called CREX, which influences deep neural networks to align their predictions with evidence provided to it. To accomplish this, CREX regularizes the local explanations produced by a neural network model to align with the domain-specific evidence provided that mentions the relevant features used in making accurate predictions. In their evaluation of this framework, the authors created a two-dimensional convolutional neural network, a unidirectional LSTM model, and a bidirectional LSTM model and measured the credibility and accuracy of these deep neural networks on the test sets of a movie review dataset and product review dataset. To measure the credibility of the local explanations produced by these models, the Du et al. used symmetric KL divergence between the model's local explanations and the ground-truth rationale. From their evaluation, the authors demonstrated that the CREX framework allows neural network to generalize better to data instances beyond the test set, but does not guarantee improved prediction accuracy unless the domain-specific knowledge supplied is of high-quality. However, even without the domain-specific rationales, the CREX framework is still capable of behaving similarly to the L1 norm regularization method, by influencing a model's local explanations to be sparse.

Alvarez-Melis and Jaakkola [8] also designed an interpretability method within the subdomain of active interpretability by creating a bottom-up interpretable model that maintain desirable characteristics of simple linear models in terms of features and coefficients without limiting performance. They made a self-explaining neural network model (SENN) that progressively generalizes linear classifiers to complex architecturally explicit models. Instead of looking at single pixels on predictions their model aims to examine the higher level features because individual pixels tend to be hard to analyze and often lead to chaotic explanations. Their results were judged on the criteria of explicitness/intelligibility, faithfulness, and stability.

To assess the relevancy of their model, Alvarez-Melis and Jaakkola observed the effects of removing features on

their model’s prediction. To test their method, they used the MNIST digit dataset, UCI datasets, and propublica’s COMPAS datasets. The results showed that they were able to create complex models with robust explanations. To measure the stability of their explanations, they used an explanation general model. Since the model is an end-to-end differential with respects to concepts SENN was directly able to be calculated in comparison to other methods.

2.1.3 Interpretability Hybrid

While most works on neural network interpretability fall into either the category of passive or active, Plumb et al. [9] designed a hybrid approach known as EXPO. Within this method, a domain-independent regularization technique is applied to black-box models that provides greater control over the quality of post-hoc explanations generated to elucidate the logic used by the model. In order to assess this hybrid approach, the authors evaluated their EXPO-regularized model on seven regression problems from the UCI collection, the MSD dataset, and the Support2 dataset against the following metrics: model accuracy, point-fidelity, neighborhood-fidelity, and stability. In addition, a user study was conducted by the authors to further qualitatively validate the quality of the post-hoc explanations produced by EXPO. The results of the application of EXPO regularization to increase fidelity and stability show that this method slightly improves model accuracy and greatly improves the quality of interpretability across each of the datasets. However, this method does not provide a resolution to the issue of vulnerability of local explanations to adversarial training examples and does not provide results when trained on data without semantic features.

Dong et al. [10] built a novel neural network architecture specifically within the domain of autonomous driving known as a global soft attention model. This model consists of a feature extractor, a Transformer, and decision-making logic generator. This novel model was trained on the Berkeley Deep Drive Object Induced Actions dataset to include both the original video frames with driving actions and explanations. The global soft attention model was then compared to other baseline models including the regional hard-attention model, the regional soft-attention model, and the global no-attention model. In order to evaluate this architecture, the authors chose to use the overall F1 score and training curve as the metrics. The results produced by the authors support the accuracy improvement brought by the global soft attention model compared to the baselines due to its focus on global features, fusion of individual pieces of information, and ability to capture long-range correlations in the input data. However, even with the introduction of some form of set of explanations for the logic used by this architecture, the authors did not evaluate the quality of these explanations.

2.2 Benchmarks

In current state-of-the-art neural networks, a commonly used framework to validate the reliability of these interpretability methods is to detect the drop of accuracy as important features are removed from the input as shown by Samek et al. [11]. Although this method is inexpensive, it comes with a significant drawback: when a set of features is removed

it changes the distribution of data creating a set of features with different probability distributions. This violates a fundamental rule of machine learning: the training and data must come from the same distribution.

Hooker et al. [12] developed a way to evaluate the approximate accuracy of interpretability methods that estimate feature importance in deep neural networks using a method they named "ROAR". Also known as Remove and Retrain, ROAR removes the fraction of input features deemed to be the most important according to each estimator and measures the change to the model accuracy upon retraining. The most accurate estimator will identify important inputs based on whose removal causes the most damage to model performance relative to all other estimators. While re-training data is a computationally expensive aspect of ROAR, it is a crucial aspect because, without re-training, we don't know whether the degradation in performance is caused by the introductions of objects outside of the original training data or because the information was actually removed.

In their results, they found that the commonly-used base estimators, Gradients/Sensitivity Heatmaps, Integrated Gradients, and Guided BackProp were worse or on par with a random assignment of importance. SmoothGrad was more computationally intensive but did not improve upon a single estimate and were worse in some cases. However, on a positive note, VarGrad and SmoothGrad-Squared improved the quality of these methods and greatly outperformed a random guess.

Chapter 3

Methods

3.1 Dataset

To train the end-to-end self-driving model, the Udacity Self Driving Car Dataset 3-1: El Camino is used. This dataset consists of 3 hours of driving data from the Udacity office in Mountain View to San Francisco and from San Francisco back to the Mountain View Udacity office. The data was collected on a 2016 Lincoln MKZ with the following sensors: 2 Velodyne VLP-16 LiDARs, 1 Delphi radar, 3 Point Grey Blackfly cameras, an Xsens IMU, and an Electronic Control Unit (ECU) sensor. With these sensors, the following information was collected: left camera view, center camera view, right camera view, Velodyne VLP-16 LIDAR packets, steering angle, steering wheel torque, vehicle speed, and vehicle position (i.e latitude, longitude, altitude).

This dataset is then preprocessed to only contain the center camera view with the corresponding steering wheel angle. Furthermore, the dataset images in which the car is stationary either in the beginning or the end of dataset are removed, resulting in a final dataset size of 210,000 data points. The first 190,000 of these points are used for training the model and the last 20,000 are used as a validation set, to verify that the model is not overfitting to/memorizing the training data labels. Further preprocessing is the applied to the center camera images, by resizing the input images from 480x640 (height x width) to 125x349, converting the dataset image representation from 8-bit unsigned integer to Tensorflow's 32-bit floating-pointing number, and converting the images from RGB to grayscale. These images are then fed into the self-driving model.

3.2 Self-Driving Model

The end-to-end driving model used in this work was developed by Bojarski et al. [4] called NetHVF. This model was initially designed as a means of validating the VisualBackProp approach of extracting feature maps that accurately represent the concepts learned by a vision-based neural network. However, in this project, the application of this model is extended beyond its original scope and evaluated for its driving capabilities. The composition of this model and additional details are provided on the following page.

Table 3.1: Architecture of NetHVF

NetHVF			
Layers	Layer Output Size	Filter Size	Stride Size
conv	32 x 123 x 349	3 x 3	1 x 1
conv	32 x 61 x 173	3 x 3	2 x 2
conv	48 x 59 x 171	3 x 3	1 x 1
conv	48 x 29 x 85	3 x 3	2 x 2
conv	64 x 27 x 83	3 x 3	1 x 1
conv	64 x 13 x 41	3 x 3	2 x 2
conv	96 x 11 x 39	3 x 3	1 x 1
conv	96 x 5 x 19	3 x 3	2 x 2
conv	128 x 3 x 17	3 x 3	1 x 1
conv	128 x 1 x 8	3 x 3	2 x 2
FC	1024	-	-
FC	512	-	-
FC	1	-	-

Note the following details about the design of the NetHVF model. Each layer except for the last fc layer in Table 3.1 is followed by ReLU activation. Each conv layer is preceded by a batch normalization layer. Let n be the number of feature maps, h be the height and w be the width. For conv layers, layer output size is $n \times h \times w$. Filter size and stride are given as $h \times w$.

For further clarification, the conv layers specified are an abbreviation for convolutional layers, FC layers are an abbreviation for fully-connected/feedforward layers, and ReLU is an abbreviation for rectified linear unit. Additional details on the meanings of each of these terms as well as others will be provided in the following short paragraphs.

Convolutional layers employ convolution operations or strided dot products on their provided input, usually images. The use of convolutional layers in NetHVF is intended to extract features from images provided during training that aid model in making an accurate decision. Specifically, the convolutional layers at the top of the diagrams shown above learn low-level features such as edges, and the last few convolutional layers learn more high-level features, such as lane markers.

Fully-connected layers work by passing the outputs of each of the artificial neurons in that layer to every neuron in the next layer.

Rectified Linear Unit (ReLU) is a mathematical function that is expressed in the following form.

$$A(x) = \max(0, x) \quad (3.1)$$

Additionally, the graphical representation of the ReLU is shown on the following page.

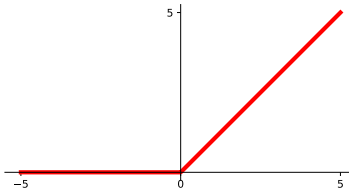


Figure 3.1: The graphical representation of the Rectified Linear Unit activation function is shown above.

This function is commonly used to add non-linearity to each of the layers in a neural network to aid it in learning very complex decision boundaries.

Batch Normalization layers are normally placed between two adjacent hidden layers, or neural network layers between the input and output layers, in order to reduce variance in the prior layer’s output. This layer does so by taking the output of the previous neural network layer and normalizing it, such that the following layer sees the normalizing output. This aids the model in reaching an optimal solution.

When combined together as shown in Table 3.1, the mechanics of the model can be described in the following manner. A preprocessed image is passed an input into the model, in which it is first normalized before going through convolutional layer for feature extraction. This process is repeated multiple times to extract higher-level features from the input image while also making sure that the model can reach an optimal solution. Once input has passed through the batch normalization and convolutional layers, the output of the last convolutional layer is flattened out as single-dimensional vector before it is passed through the following fully-connected layers. This is where the model takes in the learned feature and learns to make a steering-wheel angle decision based on the features it extracted from the image. After going through these fully-connected layers, the steering wheel angle output is produced.

3.3 Regularization

In order to make this model more interpretable, our proposed method involves applying active model interpretability in the form of the penalty function on the model’s optimization function, also known as regularization. The specific form of regularization applied to this model is L1 Norm Regularization. For additional clarification, from a theoretical point of view, the application of L0 Norm Regularization would be more ideal over L1 Norm Regularization, since the former can guarantee sparsity. As shown by the graphical depictions of L0 and L1 Norm Regularization in Figure 3.2, L0 Norm is axis-aligned, which means that the application of this regularization will constrain optimal solutions of the model to be axis-aligned as well. However, since solving an optimization problem with L0 Norm Regularization is NP hard, L1 Norm Regularization is used instead where optimal sparse solutions are more likely than non-sparse solutions at the vertices of its graphical representation.

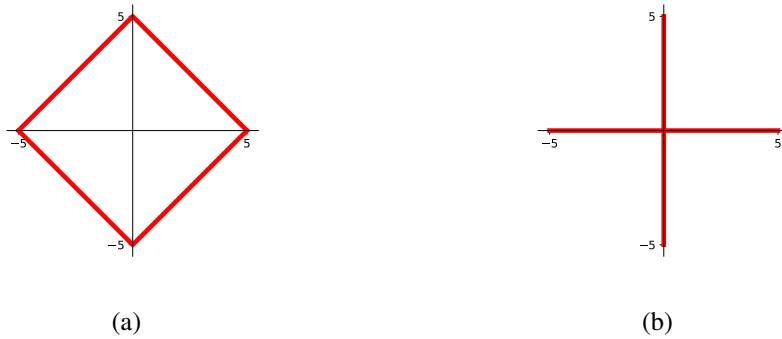


Figure 3.2: The graphical representations of two regularization functions are shown above. The left-hand figure (a) represents L1 Norm Regularization and the right-hand figure (b) represents L0 Norm Regularization

From a mathematical point of view, L1 Norm Regularization is shown as the following:

$$\|\theta_i\|_1 = \lambda \sum_{i=1}^n |\theta_i| \quad (3.2)$$

This operation of this regularization term is as follows. For all values of θ_i , which represent the parameters that the model learns during training, take the summation of the absolute values of those terms. Semantically, this equates to inducing sparsity in the parameters learned by a model, which means that some of these parameters become or approach zero. Additionally, the impact of L1 Norm Regularization on the model can be varied via the lambda parameter denoted in the equation above, in which higher values of lambda result in greater sparsity of the model's parameters and lower values of lambda result in less sparsity of the model's parameters.

Within this particular project, L1 Norm Regularization was applied mainly to the convolutional layers of the NetHVF model. The purpose of this is to induce sparsity in the features learned by the network, such that the reduction in parameters would allow the model to be more selective in the features that it learns during training. Through this process of selecting the features learned in each of the convolutional layers, the model may be able to filter out some unnecessary noise that may be irrelevant to autonomous driving and learn identifiable patterns. This in turn may make the features learned by the model more interpretable for humans.

3.3.1 Test Environment

To evaluate our model, we decided to use the open-source driving simulator CARLA. This simulator has a diverse set of driving environments with a flexible API for greater ease in adding different vehicle sensors. In addition, CARLA has a comprehensive driving benchmark called Corl2017. This benchmark test evaluates the autonomous driving model on two towns, Town01 and Town02, with 24 different experiments run on both towns. Within these experiments, the autonomous driving model is evaluated for being able to drive straight, make a single turn, go to an arbitrary position, and go to an arbitrary position with random moving objects in the environment. For each of these

tasks, the weather conditions of the simulation environment are changed to the following: clear noon, heavy rain noon, clear sunset, after rain noon, cloudy after rain, and soft rain sunset. In total, this equates to the autonomous driving model being evaluated on 600 different simulations for a total of 39 hours. The first 24 hours are evaluated on Town01 and the next 15 hours are evaluated on Town02. These times may vary based on the performance of the model in the benchmark.

For this project, only the Town02 simulations are run, since Town01 is used in the training dataset that CARLA provides and not utilized to train the baseline and regularized NetHVF models.

3.3.2 Data Collection

During the benchmark test, the number of human interventions, autonomous driving time, and images viewed by the models is collected. For the number of the human interventions, this quantity is incremented based on observations made when the simulated car with the loaded model drifted out of a lane or crashed into a pole or building during each of the simulation episodes. For the autonomous driving time, this is measured by taking the total time during which the simulated car with the loaded model drives continuously until the car is no longer able to navigate after crashing or for the duration of the simulation. Lastly, for the images viewed by the models, these images are collected after every 50 frames.

Chapter 4

Evaluation

4.1 Experimental Design

4.2 Model Training

After collecting the required data and building the self-driving model using the open-source Keras Python Deep Learning API with Tensorflow as the API's backend framework, various model configurations were trained. One set of model configurations consisted of the baseline model without regularization and the second set used L1 Norm regularization. The following hyperparameters were modified for both of these types of models: batch size and model learning rate. Furthermore, the strength of L1 Norm regularization applied to our InterpNet model was also varied. The following list illustrates the specific parameters that were evaluated.

- Batch Size: 32, 64
- Learning Rate: 1e-2, 5e-3, 1e-3, 5e-4, 1e-4
- L1 Norm Strength: 1e-3, 5e-3, 1e-2

All the model configurations were trained to minimize the mean squared error between their predicted and ground-truth steering wheel angles using mini-batch stochastic gradient descent. This optimization algorithm takes batches of data with the specified batch size and randomizes either all the data points in all the batches or only the batch order. For this project, only the order of the batches were randomized, since autonomous driving relies on concepts and ideas from reinforcement learning. Specifically, since reinforcement learning does not make the assumption that the data points in the relevant data distribution are independent, this means that there is some correlation between some of the data points. Thus, in order to ensure that each model configuration learn this correlation, the order of the data points in each of the batches of training data was fixed.

In terms of hardware, the training of all the models was divided between Santa Clara University's WAVE High Performance Computer (HPC) and Data Mining Lab12 system (DM12). On the WAVE HPC, model training was done on an Nvidia Tesla V100 GPU with 32 GB of RAM, whereas, on DM12 system, training was done on an Nvidia

TITAN XP GPU with 12 GB of RAM. Our choices of batch size in this project were restricted by the RAM capacity of the GPU hardware used.

4.2.1 Evaluation Metrics

To evaluate the model, the effectiveness and interpretability of the baseline and InterpNet model were examined.

For model effectiveness, the autonomy of the models was measured using the number of human interventions that was recorded during the 10 hour benchmark test. The precise formulation of this metric, as stated by Bjarski et al. [13], is provided below.

$$autonomy = 1 - \left(\frac{\text{number of interventions} \times 6 \text{ seconds}}{\text{elapsed time [seconds]}} \right) \quad (4.1)$$

For further clarification, the autonomy of the model was calculated indirectly by measuring how non-autonomous the model is. This non-autonomy portion of the metric was calculated by taking the number of human interventions multiplied by 6 seconds over the total elapsed driving time. The number of human interventions was multiplied by 6 seconds because it takes 6 seconds on average for a person to correct a driving error.

For model interpretability, the compressibility and randomness of the features learned by both models were used as indicated by Samek et al. [11]. To extract these learned features, the VisualBackProp approach [4] was used to extract the learned features of both models on the image collected during the benchmark test. After completing this extraction process, the interpretability measurements were performed.

First, the compressibility of the features were calculated using the Principal Component Analysis (PCA) machine learning algorithms. This algorithm is a commonly used dimensionality reduction technique to linearly transform a given set of features into another feature space of reduced size to maximize the variance captured from the original set of features. In other words, PCA takes an input and tries to reduce the size of this input without losing too much of the information contained within the original features. Within this project, the scikit-learn implementation of PCA was used and configured such that the algorithm would find the reduced size of the transformed features that capture 90% of the variance within the original features.

Second, the randomness of the features was calculated using a concept from information theory called entropy, which measures how much disorder or randomness a provided input has. The range of randomness for entropy ranges between 0 and 1 inclusive, in which 0 indicates no randomness and 1 indicates complete randomness. For this project, the entropy function provided by the scikit-image library was used.

4.3 Experimental Results

4.3.1 Trained Models

As prior described, various model configurations were tested for the baseline and InterpNet models. For the baseline model, a total of 10 model configurations were evaluated. For the InterpNet model, a total of 30 model configurations were evaluated. After training these models until convergence, the optimal solution for each configuration was determined based on the epoch at which minimal validation loss was achieved. After comparing the optimal solutions for all the baseline models, it was determined that the optimal baseline model was that trained for 52 epochs with batch size of 32 and learning rate of $5e-4$. This model achieved a validation loss of 0.00255298. After comparing the optimal solutions for all the InterpNet models, it was determined that the optimal InterpNet model was that trained for 260 epochs with a batch size of 32, learning rate of $1e-4$, and L1 Norm regularization strength of $1e-3$. This model achieved a validation loss of 0.00200723. The training plots for optimal baseline and InterpNet model are shown in the following figures.

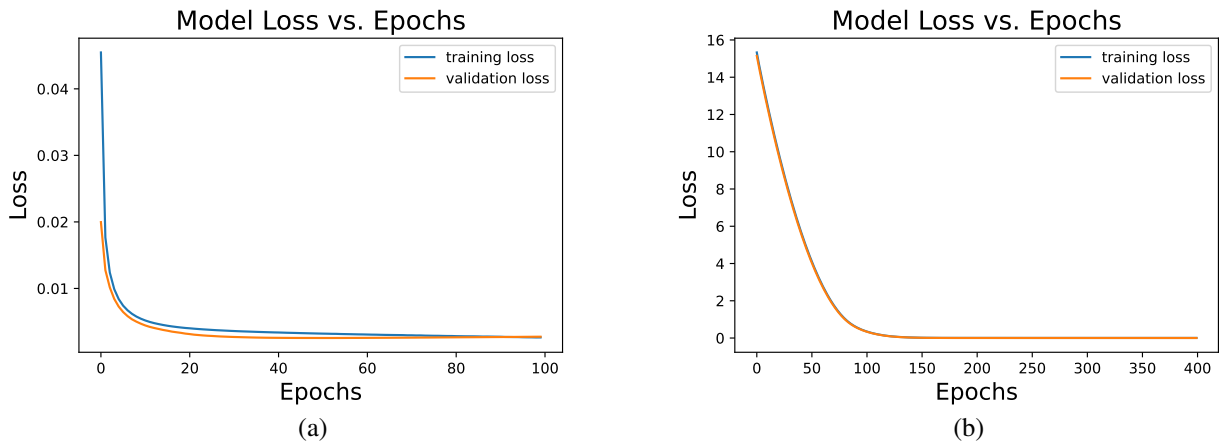


Figure 4.1: Training plots of the optimal models with training and validation Loss are shown above. The left-hand figure (a) depicts the training plot of the optimal baseline model. The right-hand figure (b) depicts the training plot of the optimal InterpNet model.

4.3.2 Autonomy

Based on the data collected on the number of human interventions from the Corl2017 benchmark tests for the baseline model and InterpNet, the autonomy results are shown in Table 4.1 on the following page.

Table 4.1: CARLA Autonomy Measurement

	Baseline	InterpNet
Number of Interventions	618	598
Autonomous Driving Time (seconds)	2870	3090
Autonomy	43%	46%

As shown in Table 4.1, the baseline model drove autonomously for a smaller amount of time compared to the InterpNet model and required more human interventions as well. In total, this equated to the InterpNet model being more autonomous than the baseline model by 3%.

4.3.3 Interpretability

Based on the data collected on the images viewed by the baseline and InterpNet models, histograms were created to describe the compressibility and randomness of the features learned by both models were extracted.

Feature Compressibility

The histograms for measuring the compressibility of the features learned by the baseline model and InterpNet are shown below in Figure 4.2. For clarification, each of the bins in these two histograms represents the number of images requiring a certain number of Principal Components in order to hold 90% of the information from the original features. As the number of Principal Components required increases, this indicates that the features are less compressible.

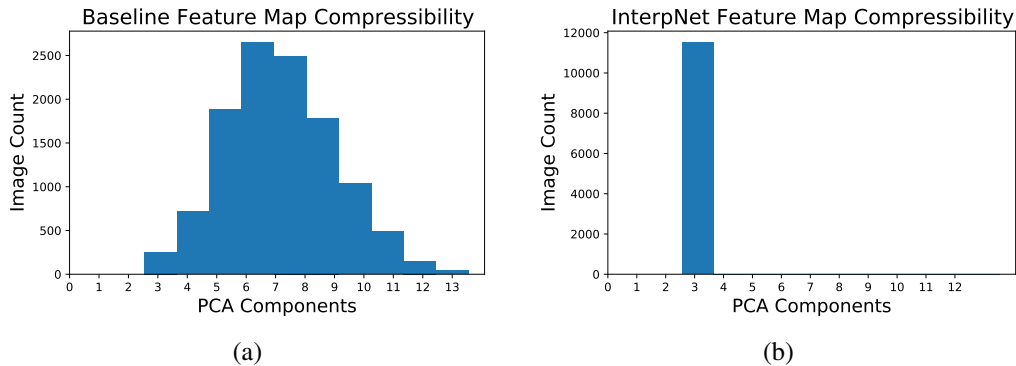


Figure 4.2: Compressibility of the feature maps for the optimal models is shown above. The left-hand figure (a) depicts the number of PCA components required to compress the feature maps extracted from the optimal baseline model. The right-hand figure (b) depicts the number of PCA components required to compress feature maps extracted from the optimal InterpNet model.

From these histograms 4.2, it is observed that the distribution of the compressibility of the features learned by the baseline model closely resembles a Gaussian distribution with a greater amount of variation compared to the distribution of the compressibility of the features learned by InterpNet. Additionally, the peak compressibility of the learned features of the baseline model requires 7 Principal Components compared to the 3 Principal Components

always for learned features of InterpNet. This indicates that the features learned by InterpNet are generally more compressible than those learned by the baseline model.

Feature Randomness

The histograms for measuring the randomness of the features learned by the baseline model and InterpNet are shown below in Figure 4.3. For clarification, each of the bins in these two histograms contains images whose entropy values fall within the an entropy range specified by the tick marks to the left and right side of the bin. For example, the rightmost bin in the feature randomness graph for the baseline mode contains images whose entropy values are between 0.9 and 1.0. As the entropy range values, this indicates that the features are more random.

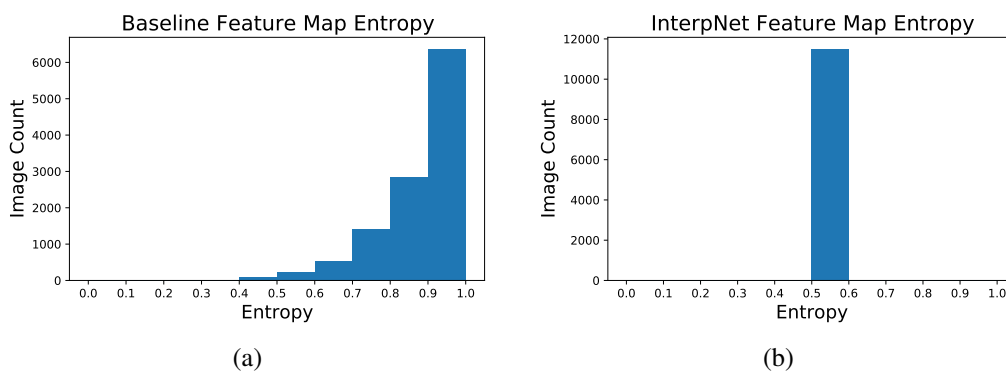


Figure 4.3: Entropy/Randomness of the feature maps for the optimal models is shown above. The left-hand figure (a) depicts the entropy of the feature maps extracted from the optimal baseline model. The right-hand figure (b) depicts the entropy of the feature maps extracted from the optimal InterpNet model.

From these histograms 4.3, a similar trend for the histograms is observed as from the histograms of the features learned by both model. Specifically, the peak randomness of the learned features of the baseline model falls within the range of 0.9 and 1.0 compared to 0.5 and 0.6 range for learned features of InterpNet. This indicates that the features learned by InterpNet are generally less random than those learned by the baseline model. However, the distribution for the randomness of the feature learned by the baseline model are skewed to the right/maximum entropy instead of closely resembling a Gaussian distribution. Nonetheless, this still indicates that the features learned by InterpNet are generally less random than those learned by the baseline model.

Chapter 5

Future Work

5.1 Improving the Dataset

One of the first key points regarding improvements made to this project revolves around finding a more suitable dataset. Due to complexity of the autonomous driving task, the dataset fed into the model should reflect similar complexity in terms of environment and actions taken. Within the dataset used for this project, the Udacity Self-Driving Dataset 3-1: El Camino, it was observed that this dataset was biased towards highway driving on a sunny day during the afternoon. Additionally, the center camera provided by the dataset was not properly centered, and the driving capabilities of the NetHVF model on the Udacity dataset was not demonstrated in the original paper [4] In total, these issues with the dataset would influence the model's ability to generalize to new environments and accurately respond to input data that is not properly represented within the training dataset distribution. To resolve this issue, we recommend using a different dataset, such as that described by Hecker et al. [14], which consists of more driving data in diverse driving environments. The use of this dataset for this project may prove more beneficial towards having a more competent baseline model that can properly demonstrate the impact our regularization on the model.

5.2 Improving the Self-Driving Model

Generally speaking, some straightforward modifications made towards improving the self-driving model would focus on more testing. With the current parameters specified in the methods section, further testing in this regards would include testing more values for L1 Norm Regularization strength, model learning rate, and batch size in order to cover a greater scope of potential optimal solutions for the baseline model and Interpnet. More involved improvements to InterpNet are specified in the following subsections.

5.2.1 Cost Function

In terms of improving the self-driving model, one approach to solving this issue would be from the point of view of objective/cost functions. Specifically, instead of giving the NetHVF model the task of minimizing the mean squared

error between the predicted and ground-truth steering wheel angle output, a better cost function could be used. From the literature on self-driving models by Tampuu et al. [15], it has been stated by that giving self-driving models to the task of minimizing the mean absolute error between the predicted and ground-truth steering wheel angle results in better driving capability of the model.

5.2.2 Model Input

From the point of view of the model's input, an improvement could be made by adding in additional camera input instead of just the center camera image. As pointed out by Hecker et al. [14], the development of a model that can take inputs from cameras positioned at multiple angles to take in a 360 degree view of the vehicle results in better model performance. Additionally, LIDAR sensors could be added as potential model input; however, the main hesitancy with the addition of LIDAR is the significant increase in cost for autonomous vehicles. Therefore, it would be prudent to include combinations of inputs to a self-driving model that avoid using LIDAR. very chaotic, and we have to be able to navigate it to make it interpretable for humans. Autonomous driving is a high-stake safety-critical application. From a societal point of view, performance guarantees should be mandatory. However, there are many scenarios where self-driving models are not testable due to the im- possibility of listing and evaluating every scenario that a model can encounter. A solution to this issue is to be able to explain a model's decision making process in making certain decisions in a given scenario and pinpoint the area of error. 1.3 Solution Our team created a solution to improve the interpretability of neural networks on autonomous cars by creating a mechanism that helps interpret the neural network. This occurred as the model was learning, such that the learning process does not compromise prediction accuracy. This mechanism consisted of adding the L1 norm regularization function to our model so that it penalized the neural network during training. The penalty prevented the neural network from developing some chaotic data representation that may deliver high accuracy but diminishes interpretability. In order to verify this, we executed a method of visualizing what features the model has learned based on how input data is transformed by the neural network. 2

5.2.3 Model Output

With regards to the output produced by the model, an improvement could be made in terms of transitioning from steering wheel angle to waypoint prediction. In other words, the model would be plotting its proposed location for a series of consecutive time steps, such as for the next 10 image frames. This would allow for the transformation of the model's output from low-level to high-level that would be easier to interpret and encode more information. For example, as described in by Tampuu et al. [15], work done towards waypoint prediction for autonomous driving contains information such as steering wheel angle and speed, which can be extracted using a controller module. While the inclusion of speed output to the original model may seem simpler, the inclusion of an additional output would

place constraints on the regularization functions we seek to apply, which will be described further in the next section. Therefore, in order to preserve a single output from the model, it would be beneficial to outputting vehicle waypoints.

5.3 Additional Regularization

While the use of L1 Norm Regularization improved the performance of the NetHVF model, the use of additional regularization terms may result in additional improvements in model effectiveness and interpretability.

5.3.1 Mutual Information

One such improvement would include the use of regularizing for mutual information along with L1 Norm in the convolutional/feature extraction layers of the NetHVF model. For clarification, the mutual information measure indicates how much information is shared between two variables. For a self-driving model, the penalty/regularization function applied would constrain the model's optimal parameters, such that each of the features learned during training are sparse and independent between and across layers. Conceptually, this would encourage the model the learned features that are distinct and may be interpretable given the sparsity constraint added.

5.3.2 Tree Regularization

Although our main constraints to the model have targeted the feature extraction layers, regularization can also be applied to the dense/decision-making layers as well to promote interpretability. This would be done using the tree regularization term described by Wu et. al. [5]. This regularization terms was specifically designed to allow developers and consumers to trace through the decision-making process of time-series models and multi-layer perceptrons using the decision tree machine learning model. Since this type of machine learning model is inherently interpretable, as described by Molnar [16], the use of tree regularization and distillation of the model's dense layers into a decision tree may allow for consistent or improved accuracy and greater interpretability.

Chapter 6

Societal Issues

6.1 Ethical

Today, cars are able to perform complex tasks such as braking, steering, and object detection, without the assistance of the driver. Big tech companies are aiming to deploy fully autonomous vehicles in the United States in the next two to three years. The discussion of self driving vehicles can be tackled from many perspectives. On one hand, one can make the argument that self driving car is gradually improving and function better than human drivers. On the other hand, many bring up the concern of unsolvable decision making problems like the trolley problem. As software is now playing a key role in modern vehicles, software engineers must be proactive in ensuring that their solutions and products address ethical and social concerns that are prevalent in the domain of modern day vehicles. We asked ourselves:

- If needed to choose between two bad outcomes, how do we choose what the model values more?
- Should there be a governing entity that tells us what to prioritize?
- Who is to blame if something goes wrong? The driver, company, or engineer?

This dilemma is similar to the trolley problem. The solution to the trolley problem is extremely limited and all answers can be perceived as ethically bad or wrong. A typical approach in answering this problem is by analyzing it through various ethical theories such as utilitarianism or deontological ethics. For example, Utilitarianism stressed the greater good meaning that the ethically correct answer according to that idea of thinking would be to choose the least amount of casualties hence maximising utility. Depending on the ethical framework, different theories can be used to justify a decision. These conclusions are only drawn by humans, but how do these cars come to their conclusion? Should we be implementing moral principles into our algorithms? Who chooses what is morally correct?

6.2 Trust

While self autonomous vehicles promise to provide numerous benefits, one key barrier is the lack of trust that the public has with this technology. The complex models of Deep Neural Networks make it hard to understand and reason the predictions. Due to the complexity, there is a lack of trust with consumers using self-driving technology. Consumers may feel more at ease with a well-understood model. Especially in the domain of vehicles, understanding what the model is interpreting as significant could help engineers understand what modifications they may need to make on the model to improve its capabilities.

To give consumers more trust in their self-driving technology, companies can opt to be transparent about their process (i.e data collection, model decision, and training results) as well as having transparency in the code they release. Datasets fundamentally influence a model's behavior. Making sure that the dataset is diverse, unbiased, and relevant will ensure good results [17]. If companies are more transparent about their sourcing and training process, consumers will feel more inclined to use their technology since it is publicly available and easily verifiable for anyone who wishes to verify the legitimacy of the model as the public easily has access to attempt to recreate it and voice their concerns for improvement.

A note of concern that in doing this will most likely not be feasible in a predominately capitalist society as it will discourage innovation in public companies when they forced to reveal their company secrets. Another way to ensure that a model works is to create an interpretable model, a model that is easier for humans to understand. Some interpretable methods are to use regularizers, decision trees, or feature maps. Regularizers on existing models zeroes out some parameters inducing sparsity. This in turn makes the model more generalizable. A decision tree allows for humans to easily follow the logical parameters or rules that the model sets. It also shows how the model will reach its decision. Feature maps highlight the area that the model deems as significance. Gaining inspectable internal representation on a model plays a crucial role in letting companies and the public trust the model more.

6.3 Safety

According to the World Health Organization, approximately 1.3 million people die each year as a result of road traffic crashes. Self driving vehicles have the potential to reduce a number of those deaths by eliminating human errors. Improving safety is primary objective of autonomous vehicles. In the domain of self driving cars, there are pre-market safety processes that involve measuring certain kinds of safety violations, such as near misses, pedestrian detection, correct traffic light and sign recognition, etc. Post market reporting as well as audits and updates can help engineers improve on their model and make sure its performance stays up to par. It also allows for companies to detect a shift in errors which may suggest that the data is no longer like the training data. When taking proper safety protocols, self driving vehicles can shift from being an alien software that is difficult to understand and trust to a more safe and

welcoming vehicles with statistics to back up the claim.

With interpretability one can figure out why a prediction was made by a model. In some cases, knowing the why is not necessary but in a high risk field such as self-autonomous vehicles, the why can help one know more about the problem, the data, and the reason why the model might fail.

6.4 Lifelong Learning

As the field of Deep Learning becomes more integrated into the daily life of society, the need for engineers to understand how a model works and being able to provide an explanation for its decision process becomes increasingly important. In the process of this project, we were able to educate ourselves on the state-of-the-art technology of self driving models and interpretability methods. In addition, we learned how to develop an end-to-end deep learning pipeline and experience the research of process of thinking about and improving upon shortcomings in current research work. Especially since self driving technology and other fields of Deep Learning are still currently being developed, there is still ample amount of opportunity to explore and consider fundamental aspects of these models. As this technology continues to develop, our goal as researchers is to not only keep up with new developments, but also play a significant role in this innovation. And playing a significant role in innovating deep learning and self-driving technology requires that we continue to apply the skills we have learned, including finding research problems, developing a comprehensive understanding of past work, deeply thinking about the connection between these works and the problem, and designing an experiment to evaluate novel solutions that we generate.

Chapter 7

Conclusion

7.1 What We Have Learned

In conclusion, our evaluation demonstrated the potential of Interpnet as an improvement upon the original baseline model given our results for the model's autonomy and feature randomness and compressibility. However, another important observation that we made was the relatively static nature of the feature maps we extracted from Interpnet, which indicates that our model may be generalizing a little too much or overfitting to the validation data. When looking back at the training curves for the InterpNet model, we did notice a higher training loss compared to that of the baseline model. From this, we learned that simply looking for the smallest validation loss may not be the best evaluation metric for optimal models. Instead, it may be better to also observe how far apart the training loss and validation losses are. Additionally, we learned the importance of making sure the validation data used for choosing the optimal model is well representative of the training data in spite of what the literature may specify. In particular, while the authors of the NetHVF model specified that the last 20,000 images of the Udacity dataset be used for validation, this may not have been an adequate representation of human driving for the InterpNet model to perform equal well at in conjunction with the training dataset.

Aside from the technical aspects related to the impact of dataset splits on the model, we learned much about the theoretical aspects involved in implementing a self-driving model. Especially since autonomous driving is a reinforcement learning problem that uses deep learning, we could not rely on conventional deep learning knowledge. In particular, this meant not assuming that our each driving image in our dataset was independent, since each segment of driving data has correlations that need to be learned by the model. Additionally, we could not use a test set of images and corresponding steering wheel angles to evaluate the effectiveness of the baseline and InterpNet models, since this method would not properly demonstrate driving capability. This warranted the need for a simulator. However, we also learned that the use of any simulator with non-realistic image frames was not appropriate. As we observed, a simulator whose environment is not represented in the model's training data provides an additional barrier for the model. This occurred, at least in our case, since the baseline and InterpNet models learned how to driving the real-world and were

therefore not well equipped to drive in a simulated/virtual world.

Along with the theoretical lessons learned from this project, we also learned important lessons related to developing an input pipeline for loading data into the model and configuring the CARLA simulator to load the optimal baseline and InterpNet model. With regards to model input, we learned how to develop an input pipeline for a model from scratch using low-level Tensorflow functions, since the high-level data-loading functions in Tensorflow were not particular adapted for our dataset. In terms of the CARLA simulator, we learned how to setup a virtual environment with the right versions of various Python modules in order to run our code.

From a more general point of view of conducting research, we learned how important it is to choose problems whose computational resources are available. Especially since self-driving requires a large amount of data to properly train on, the lack of dedicated supercomputer for this project hindered investigation into whether our proposed modifications would truly improve interpretability and effectiveness for state-of-the-art autonomous driving models.

7.2 Why it is Important

Our project has value as it tackles the long-standing issue of gaining insight in how neural networks think in mission-critical applications, such as autonomous driving. Generally speaking, as the autonomous driving feature becomes more prominent in more and more vehicles, it is important for engineers to be able to debug erroneous models quickly in order to prevent further loss of life.

Within the scope of this project, the impact of applying sparsity towards the features a model learns is important not only for interpretability but also potentially for effectiveness. Therefore, our work may suggest another area of interest for the autonomous driving research community to consider the theoretical aspects involved in robust reinforcement learning rather than primarily focusing on collecting larger and larger datasets to improve model generalizability.

Additionally, our work promotes the combination of different interpretability approaches for complete insight into image-based deep learning models in general. Based on the analyzed literature, much work on interpreting image-based deep learning models focuses solely on showing the feature maps learned by the feature extraction layers of a model. However, as expressed in future works chapter 5, tree regularization also has the potential to be applied to the NetHVF model to understand the decision-making layers of the model as well, given how InterpNet was able to produce interpretable feature maps.

Appendix A

Additional Model Training Plots

The following appendix contains additional plots of the training and validation losses of the other model configurations tested in pursuit of the optimal baseline and InterpNet Models.

A.1 Baseline Model Training Plots

This section contains the training plots for the baseline models. Below each graph, the learning rate and batch size hyperparameters that were used to train the corresponding model are specified.

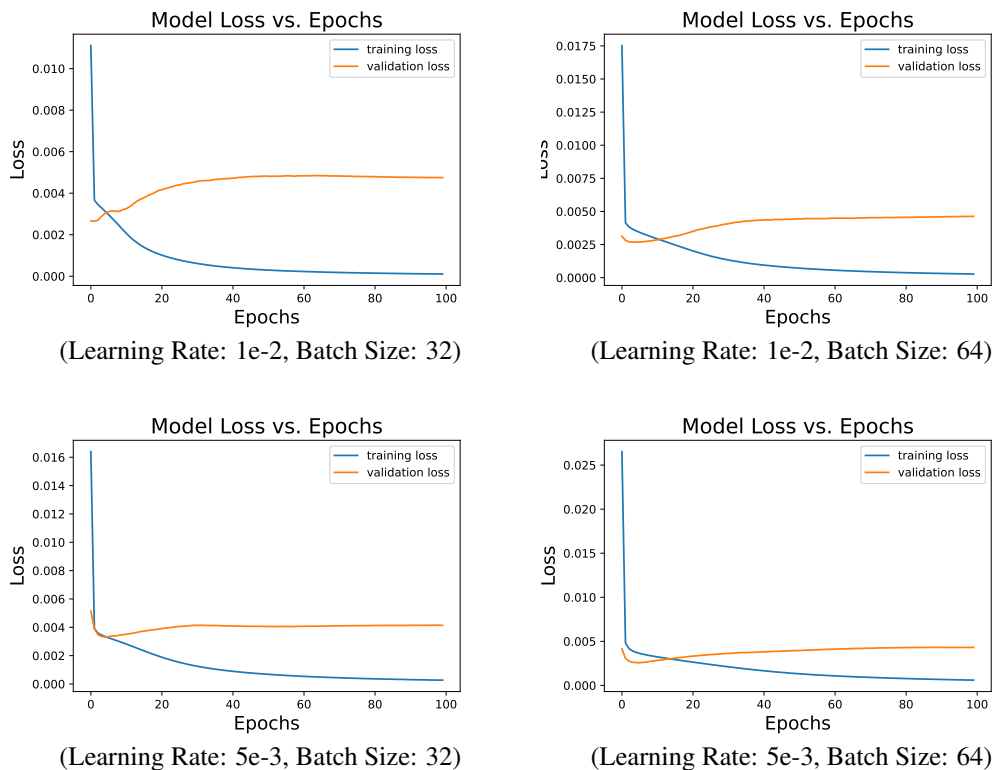
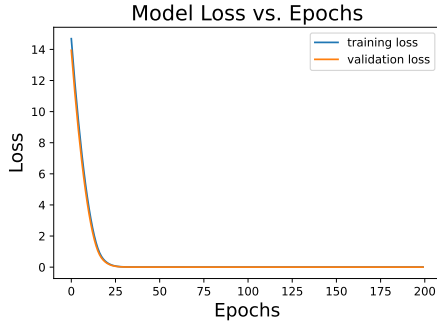


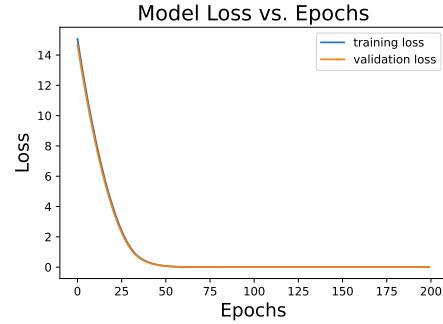
Figure A.1: Four additional baseline training plots are shown above for learning rates of 1e-2 and 5e-3 on batch sizes 32 and 64.

A.2 InterpNet Model Training Plots

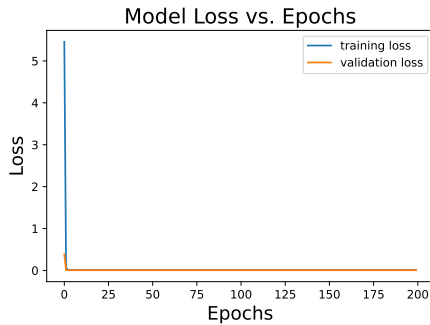
This section contains the training plots for the InterpNet models. Below each graph, the L1 regularization strength (abbreviated L1 Reg), learning rate and batch size hyperparameters that were used to train the corresponding model are specified.



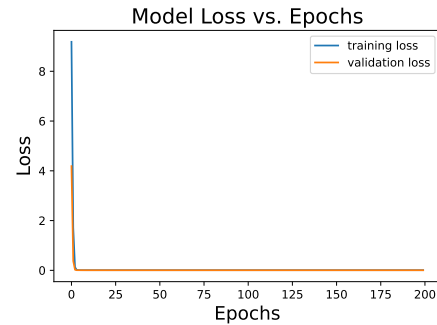
(L1 Reg: 1e-3, Learning Rate: 5e-4, Batch Size: 32)



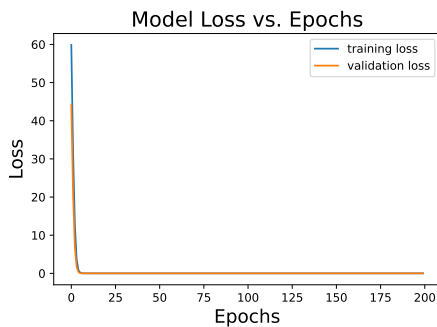
(L1 Reg: 1e-3, Learning Rate: 5e-4, Batch Size: 64)



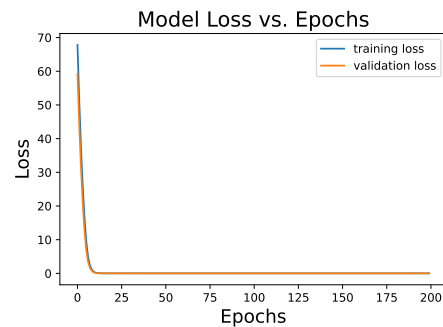
(L1 Reg: 1e-3, Learning Rate: 1e-2, Batch Size: 32)



(L1 Reg: 1e-3, Learning Rate: 1e-2, Batch Size: 64)



(L1 Reg: 5e-3, Learning Rate: 5e-4, Batch Size: 32)



(L1 Reg: 5e-3, Learning Rate: 5e-4, Batch Size: 64)

Figure A.2: Six additional InterpNet model training plots are shown above for 11 regularization strengths of 1e-3 and 5e-2, learning rates of 5e-4 and 1e-4, and batch sizes of 32 and 64.

Bibliography

- [1] É. Zablocki, H. Ben-younes, P. Pérez, and M. Cord, “Explainability of vision-based autonomous driving systems: Review and challenges,” *ArXiv*, vol. abs/2101.05307, 2021.
- [2] Y. Zhang, P. Tiño, A. Leonardis, and K. Tang, “A survey on neural network interpretability,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, pp. 726–742, 2021.
- [3] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894, PMLR, 06–11 Aug 2017.
- [4] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. J. Ackel, U. Muller, P. Yeres, and K. Zieba, “Visualbackprop: Efficient visualization of cnns for autonomous driving,” pp. 4701–4708, 2018.
- [5] M. Wu, M. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez, “Beyond sparsity: Tree regularization of deep models for interpretability,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, Apr. 2018.
- [6] C. Wu, M. J. F. Gales, A. Ragni, P. Karanasou, and K. C. Sim, “Improving interpretability and regularization in deep learning,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 256–265, 2018.
- [7] M. Du, N. Liu, F. Yang, and X. Hu, “Learning credible deep neural networks with rationale regularization,” in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 150–159, 2019.
- [8] D. Alvarez Melis and T. Jaakkola, “Towards robust interpretability with self-explaining neural networks,” in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.
- [9] G. Plumb, M. Al-Shedivat, A. A. Cabrera, A. Perer, E. Xing, and A. Talwalkar, “Regularizing black-box models for improved interpretability,” in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 10526–10536, Curran Associates, Inc., 2020.
- [10] J. Dong, S. Chen, S. Zong, T. Chen, and S. Labi, “Image transformer for explainable autonomous driving system,” in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2732–2737, 2021.
- [11] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, “Evaluating the visualization of what a deep neural network has learned,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2660–2673, 11 2017.
- [12] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim, “A benchmark for interpretability methods in deep neural networks,” pp. 9737–9748, 2019.
- [13] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016.
- [14] S. Hecker, D. Dai, and L. V. Gool, “End-to-end learning of driving models with surround-view cameras and route planners,” 2018.

- [15] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, “A survey of end-to-end driving: Architectures and training methods,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 1364–1384, apr 2022.
- [16] C. Molnar, *Interpretable Machine Learning*. 2 ed., 2022.
- [17] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé, and K. Crawford, “Datasheets for datasets,” 2018.









Neural_Network_Interpretability_Autonomous_Driving_Neural_Networks_Publication

Final Audit Report

2022-06-09

Created:	2022-06-09
By:	Darcy Yaley (dyaley@scu.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAAfWI1pQz_V68LsQMPkUbYvoJU8o6fMFj

"Neural_Network_Interpretability_Autonomous_Driving_Neural_Networks_Publication" History

-  Document created by Darcy Yaley (dyaley@scu.edu)
2022-06-09 - 10:36:56 PM GMT
-  Document emailed to David C. Anastasiu (danastasiu@scu.edu) for signature
2022-06-09 - 10:37:47 PM GMT
-  Email viewed by David C. Anastasiu (danastasiu@scu.edu)
2022-06-09 - 10:37:52 PM GMT
-  Document e-signed by David C. Anastasiu (danastasiu@scu.edu)
Signature Date: 2022-06-09 - 10:38:14 PM GMT - Time Source: server
-  Document emailed to N. Ling (nling@scu.edu) for signature
2022-06-09 - 10:38:15 PM GMT
-  Email viewed by N. Ling (nling@scu.edu)
2022-06-09 - 11:31:59 PM GMT
-  Document e-signed by N. Ling (nling@scu.edu)
Signature Date: 2022-06-09 - 11:32:17 PM GMT - Time Source: server
-  Agreement completed.
2022-06-09 - 11:32:17 PM GMT