Department of Computer Science

Submitted in part fulfilment for the degree of BEng

Data Augmentation in Deep Learning Systems

Pratham Bhat

03 May 2023

Supervisor: Simos Gerasimou

ACKNOWLEDGEMENTS

Deepa Prabhu for being a better mother than I could ever ask for

Dr. Simos Gerasimou for impeccable advice and guidance

My friends for letting me use their computers every night to train and test my models

STATEMENT OF ETHICS

Click or tap here to enter text.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Executive summary

Machine Learning (ML), or the application of Artificial Intelligence (AI) that provides systems the ability to automatically learn and improve from experience has had an explosive impact on across all industries internationally. One such industry that has organisations investing a great deal into ML is the automotive industry. Applications range from driver assistance and driverless automobiles, to supply chain optimisation and predictive maintenance. However, with the increasing reliance on ML, there is a greater demand for more robust and predictable solutions for safety critical systems. Although it has come a long way [1], the application of ML-based solutions for self-driving and/or driverless cars has been a challenge [2].

The NHTSA Crash Reporting for Level 2 Advanced Driver Assistance Systems (US Dept. of Transportation, June 2022) [3] gives us an idea as to how the application of ML-based solutions in safety critical systems is still in its infancy. Serious injuries of a fatality occurred in 11 of the 98 crashes where crash severity was reported. Naturally, the best way to reduce this risk is by thoroughly training and testing systems to ensure correctness and robustness. This is easier said than done and covers a wide range of practices. The objective of this paper is to explore a small part of the process of training a neural network, that is the data augmentation done during training. By augmenting the data used for testing, we can discover edge case behaviours to help us better understand the decisions made by the model. By augmenting the data used to training, not only can we work with limited data, but also influence the training such that the model learns relevant features to make better decisions for edge and adversarial cases.

A total of 9 datasets will be generated from a small portion of the Udacity dataset [4], with varying levels of data augmentation to observe the effects of domain randomisation, and to find a “sweet-spot” where the level of augmentation maximises training efficiency, by increasing correctness and robustness of the model to adversarial and edge cases. These results will then be analysed to see if the techniques can be used to improve the safety of ML-based driverless solutions.

Due to all of the top performing models in the Udacity Self-Driving Car Challenge being LSTM-based, the objective of this project narrows itself down to the exploration of the effects of data

augmentation when training and testing, specifically on an LSTM-based steering model. It was found that not only does data augmentation reduce the error from testing on augmented and unaugmented data, but it also significantly reduces zero-shot testing error.

1 Introduction

Data augmentation is a set of techniques to artificially increase the amount of data by generating new data points from existing data. This includes making small changes to data or using deep learning models to generate new data points. For this project, we will be focusing on domain randomisation to create new data points from existing ones in an attempt to test models for edge cases, as well as to retrain them to improve performance. The objective is to increase both correctness and robustness against adversarial/edge case data, and to find the saturation point of augmentation at which it starts to be detrimental to the correctness of the model.

Note that the augmentations applied do not always resemble real-life edge cases, and they are instead abstracted to create a similar effect. These augmentations are justified with real-life examples as precedent.

We will be analysing the effect of data augmentation on the Udacity Self-Driving Car dataset [4], for the models developed for Challenge #2 (steering angle prediction). The dataset consists of 2 phases, driving data from El Camino Real (small curves, mostly straight driving), and the second phase is driving data from San Mateo to Half Moon Bay (curvy and highway driving). The objective of the challenge was for the model to predict the real steering input made during the recording.

The Komanda model (which placed 1st) [4], will be used to train and test the data, as it used negligible data augmentation with no complex domain randomisation during training and testing.

In Section 2 (Background Literature), we will cover the basics of machine learning, the relevant research done in the area of data augmentation, as well as the dataset that will be used in this project.

In Sections 3.1, 3.2, and 3.3 (Methodology), we will cover the changes made to the model and the domain randomisation techniques experimented with, along with the justifications for the choices. In Sections 3.4 and 3.5, the training and testing processes will be covered respectively, and the results which lead to the analyses made in Section 4 (Results & Analysis).

Section 5 (Conclusion) will provide a brief summary of the Results and Analysis, along with the potential for future work in the field.

2 Background Literature

2.1) **What is Deep Learning?**

Machine Learning (ML) is a subset of Artificial Intelligence (AI) which pertains to the development of algorithms that use data from training to build a model which can make decisions pertaining to fresh, unseen data. As such, ML models can perform tasks without explicitly being programmed to do so. Deep Learning (DL) is a further subset of ML which is based on artificial neural networks, modelled after mammalian brains. The architecture of a DL model would consist of several layers of neurons, each layer using the output of the previous layer as input. This ‘depth’ of the models enables them to learn highly specific features or semantics of the input. However, this depth comes at the cost of requiring a large amount of training data.

By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via DL, given sufficiently large models and sufficiently large datasets of labelled training examples [5]. The applications of DL consist of classification, regression, clustering, association, and dimensionality reduction, among others.

More information on the fundamentals of neural networks can be found on the 3Blue1Brown YouTube channel [6].

2.2) **How are models developed and evaluated?**

Depending on the problem that needs to be solved, training usually falls into the categories of supervised or unsupervised learning. For regression and classification problems, supervised learning is used, where using labelled input and output data, the model can measure its accuracy and learn over time. For clustering, association, and dimensionality reduction, unsupervised learning is used in analysing and cluster unlabelled data sets. These algorithms discover hidden patterns in data without the need for human intervention. The evaluation of a ML model, however, is not as clear cut. Given that the

focus of this project is supervised learning, we will be reviewing literature relevant to that.

The evaluation metric(s) of models trained with supervised learning can vary based on the problem the model is designed to solve. These metrics can be roughly grouped into 7 main categories, as shown in the figure below.

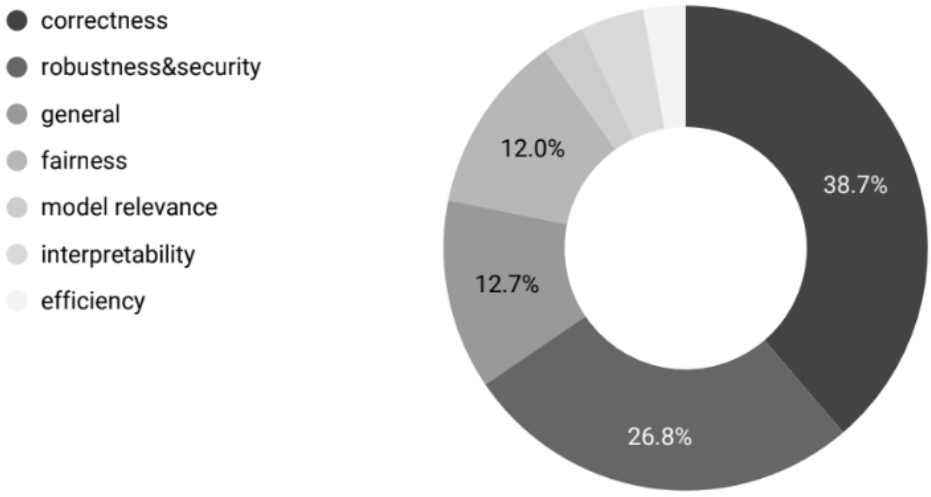

Figure 1: Research distribution among different testing properties [7]

In figure 1, we also see the distribution of research for each of the properties, with correctness and robustness & security being the majority. Correctness refers to the accuracy of the model during testing, where the model is tasked with behaving in an expected way when presented with never-before-seen information, requiring a degree of extrapolation on the model's behalf. Robustness pertains to the ability of a model to retain an acceptable level of correctness when presented with data that has been perturbed (data with added noise). The more resistant a model is to the presence of noise, the more robust it is. In turn, it is also more secure. Although the other properties (shown in figure 1) of a model are also important, this project will primarily focus on correctness and robustness.

2.3) **What is data augmentation?**

Due to the aforementioned requirement, the applications for DL-based solutions would be limited to scenarios where a large amount of training data is readily available. A small dataset can easily lead to overfitting, where the model simply memorises the training data and is therefore unable to generalise for the unseen data. However, with

data augmentation, this constraint can greatly be eased. Data augmentation is the process of augmenting the data (e.g., rotation, mirroring, etc.) to create training data that will be interpreted as unique by the model. This is done under the assumption that more information can be extracted from the original dataset through augmentations. These augmentations artificially inflate the training dataset size by either data warping or oversampling.

Data warping augmentations transform existing images such that their label is preserved. This encompasses augmentations such as geometric and colour transformations, random erasing, adversarial training, and neural style transfer. Oversampling augmentations create synthetic instances and add them to the training set. This includes mixing images, feature space augmentations, and Generative Adversarial Networks (GANs). Oversampling and Data Warping augmentations do not form a mutually exclusive dichotomy [8]. A small amount of data augmentation is potentially enough to make the use of a DL-based solution viable.

2.4) **What is domain randomisation?**

One of the approaches to make a small amount of data viable is to create synthetic data from the existing dataset, which if done by a computer can be relatively inexpensive. However, with this comes the issue of the ‘reality gap’, or the inability for it to fully generate to the real-world data, for numerous reasons including textures, lighting, and domain distributions. In an attempt to narrow this reality gap, domain randomisation is introduced to simulate a sufficiently large number of variations (e.g., view angles, textures, shapes, shaders, camera effects, scaling, etc.) such that real world data is viewed as simply another domain variation [9]. Essentially, domain randomisation involves randomising non-essential aspects of the training distribution in order to better generalise to a difficult-to-model test distribution [10].

Tobin et al. [9] tested the following hypothesis: if the variability in simulation is significant enough, models trained in simulation will generalise to the real world with no additional training. It was found that the number of images per class and the number of unique textures used in the images contributed to correctness the most. Occlusion and camera positioning were also significant, while the addition of random noise was not.

The popular papers on domain randomisation were by Josh Tobin et al. [9] [10], in which the technique was applied to train and test a robotic arm to grasp certain objects placed on a table. Domain randomisation as an idea, however, has great potential for other problems as well, such as object viewpoint estimation. Movshovitz-Attias et al. [11] explored this using photorealistic synthetic images, where they showed that that generalising from synthetic data is not more difficult than the domain adaptation required between two real-image datasets.

2.5) **Domain randomisation for testing – DeepXplore**

Kexin Pei et al. demonstrate the power of domain randomisation for white box testing with their tool DeepXplore [12]. In their paper, they simulate different types of realistic lighting and occlusion on the Udacity Self-Driving Car Challenge dataset (among other datasets) to show erroneous behaviours in a tested model. It was also shown that the test inputs generated by DeepXplore can also be used to retrain the corresponding model to improve the model's accuracy.

2.6) **What is surprise adequacy?**

The correctness of a model can be evaluated by testing it on unseen data. However, this leads the issue of the correctness score being reliant on the unseen data as much as the model itself. Hence, it is vital that appropriate test data is used such that most, if not all of the edge cases are tested. Therefore, surprise adequacy can be used to determine if the test input is satisfactory. Kim et al. define the surprise adequacy criterion as, "A good test input should be sufficiently but not overtly surprising compared to training data", where the surprise of an input is the difference in the model's behaviour between the input and the training data [13]. In their paper, Kim et al. show that surprise coverage is sensitive to adversarial samples and retraining on such samples yields better improvements in accuracy.

Surprise adequacy is done by comparing the surprise input's activation trace to a benchmark set of activation traces determined by a test input set. Fundamentally, for a test input set T, the ordered set of activation traces $A_N(T)$ is computed, where N is the ordered set of the neurons in the network. These activation traces are then compared to the activation trace $A_N(x)$ of an input x. The quantitative similarity measure between these is called the surprise adequacy of input x. There are 2 common ways to measure the surprise

adequacy of an input x: Likelihood-based (LSA), and distance-based (DSA), both of which have been further optimised by Weiss et al. [14] to be less computationally expensive. SA also can be highly sensitive to the non-determinism associated with the DNN training procedure. This section has been included as SA is referenced in the Conclusion of this paper, for potential future work.

2.7) **The Udacity Self-Driving Car Challenge dataset**

The Udacity Self-Driving Car Challenges consisted of around 51 teams with 312 participants internationally, where the objective was to open-sourced driving data (which was provided in the ROSBAG format) to develop a model that can predict the steering angle at each frame of a video feed of a car. The total training data across all challenges consisted of several hours or driving footage. Given that data augmentation is most useful when there is limited data available, this project will only use a portion of the training data from Challenge #2 for training and validation, namely the data from the HMB1 rosbag. The footage is from 3 onboard cameras mounted on the left, center, and right of what appears to be the bonnet/hood of the car. The video is broken down into 20 frames per second of 640x480 RGB pixels each, with the weather in the recorded footage being described as “daytime with shadows”. In total, this gives us a dataset with 13,203 images to work with. Along with this, the dataset also contains the timestamp, steering angle, torque, and speed of the vehicle at each timestep.

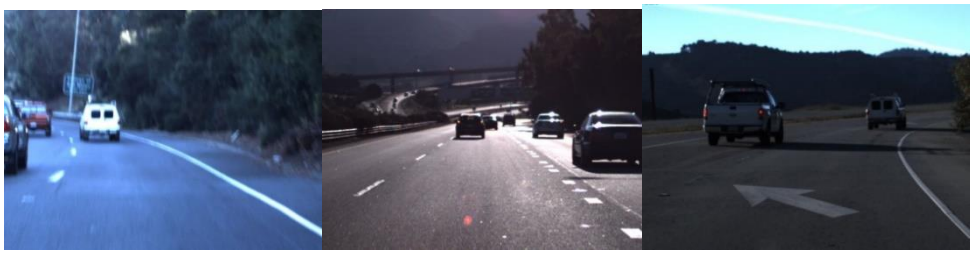

Figure 2: Some sample images from the HMB1 rosbag

2.8) **Why do the Udacity challenge models use LSTMs?**

LSTM (Long Short-Term Memory) networks are a type of RNN (Recurrent Neural Network) that can process sequential data, where chronological ordering matters. LSTMs are improved versions of RNNs and can interpret longer sequences of data. Each timestep is created with knowledge of the previous timestep, creating a new output by applying the same function to the previous output.

However, RNNs suffer from the diminishing and exploding gradient problems. An LSTM-based RNN may struggle to “remember” causes and effects that took place several timesteps ago and any error in predictions will compound, causing the model to get into a bad feedback loop. This makes the architecture a suitable candidate for steering models (as shown in the Udacity challenge), as steering a vehicle is effectively making local decisions without considering what happened several minutes ago, or what may happen several minutes later. With appropriate gating, the model can “forget” older events and give precedence to more recent ones. Naturally, this adjustment is key to ensuring robustness of the model to sudden changes in input, such as weather and lighting.

2.9) **The Komanda Model**

The Komanda model [15] was developed by Ilya Edrenkin for competing in the Udacity Self-Driving Car Challenge, which placed 1st overall. The model consists of 2 LSTM layers and a simple RNN, for which the predicted angle, torque, and speed serve as the input to the next timestep. The model uses both the ground truth and its previous output (autoregressive) during training, and only its previous output during testing. According to Team Komanda, “Convolution was performed not only in the image plane, but also in time; it allowed the model to build motion detectors and account for the driving dynamics. The recurrent cell tracked the state of the vehicle, which was supported by an auxiliary cost function — the model tried to predict not only the steering angle, but also the steering wheel torque and the speed of the vehicle. Well-established methods like residual connections, layer normalization and an aggressive regularization via dropout were necessary to obtain the best results.” [16]. Given that the problem being explored in this project is the scarcity of training data (an aspect not challenged in the competition), the size of the neural network has been significantly reduced.

3 Methodology

3.1) **Model Alterations**

In order to run the code, significant clean-up was required due to its age, however every effort was made to ensure that the updated code is functionally identical to the original code written by Ilya Edrenkin. The code is open source and so no permission was required for use in this project.

Since the original Komanda model was designed to be trained on a much larger dataset, it was too complex for the dataset used in this project. And so, in the interest of speed and efficiency, the number of model parameters were cut down significantly. This was done by reducing the number of units and the output dimensionality for the projection matrices by half (from 32 to 16) in the LSTM cell. In order to capitalise on the LSTM functionality (even in its reduced state), the batch size, the number of left context frames, and the sequence length were left untouched. Furthermore, the number of channels in the convolutional and fully connected layers were reduced to a quarter of their original size (e.g., from 128 to 32). Not only did this allow the model to now run on a commodity laptop, it also mitigated any risk of overfitting.

A link to the code used in this project can be found here, under the folder PRBX_komanda.

3.2) **The Dataset**

The data is open and so no permission was required for use in this project. The original dataset used in the competition consisted of a total of 101,396 images. To explore a variety of data augmentation approaches, only the first 13,203 frames were used as the foundation. For ease of understanding this unaugmented data will be hereafter referred to as the foundation data. Since the dataset consists of frames from 3 onboard cameras (left, center, right) recording simultaneously, it was decided to create subsets of 3 images per timestep, where each image in a subset has the same augmentation applied to it. This creates consistency, enabling the model to learn to correlate between the 3 viewpoints.

3.3) **Data Augmentation**

It was decided to undertake 3 approaches to the task, each being an increment in complexity compared to the previous. For the sake of legibility, they have been called "levels" 1 to 3. For each level, 3

duplicates of the dataset were created, where each duplicate dataset was augmented corresponding to the level. This resulted in a total of 10 different datasets (1 original dataset and 3 for each level). The model underwent a fresh initialisation (no transfer of weights) before training with each dataset, enabling us to directly compare the levels. Images were not scaled down in any phase of this project.

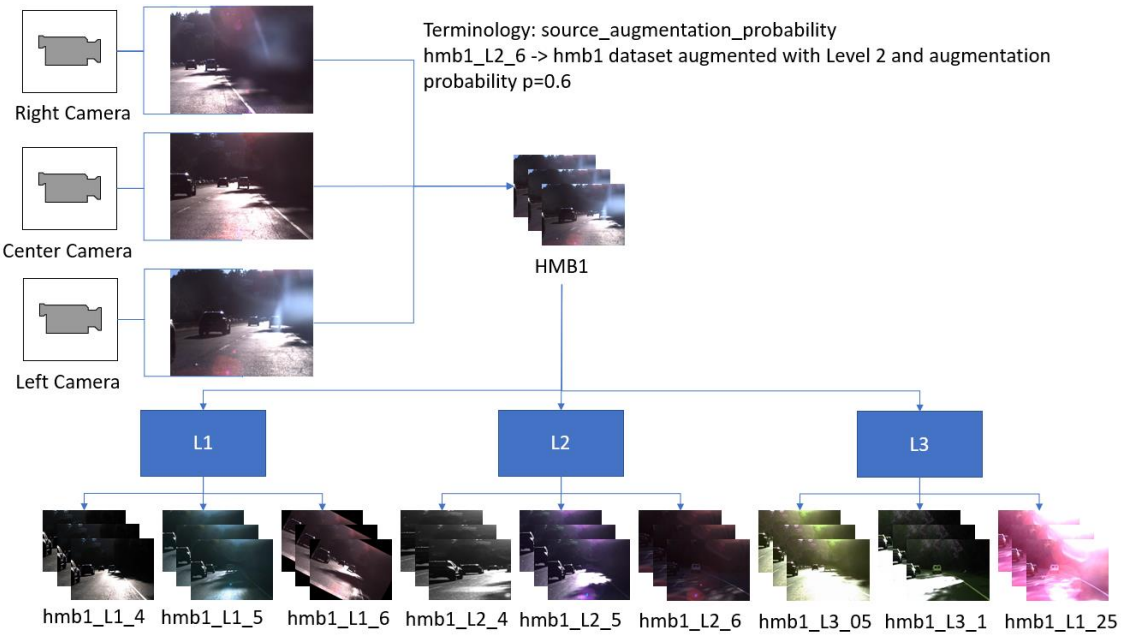

Figure 3: Dataset generation pipeline

In each level, the fundamental augmentations applied are the same. The levels only differ in the way said augmentations are applied. The fundamental augmentations are as follows.

1) Rotation - can be up to 45 degrees, clockwise or counter-clockwise. It was decided to be capped at 45 degrees as that would be the limit of adhesion for most road vehicles. The same angle is applied for all 3 images in a timestep, as when driving on a banked road, we can expect all 3 cameras to show the same angle. It was decided to leave the new space from rotation as simply black to make the augmentation less difficult, given the size of the network.

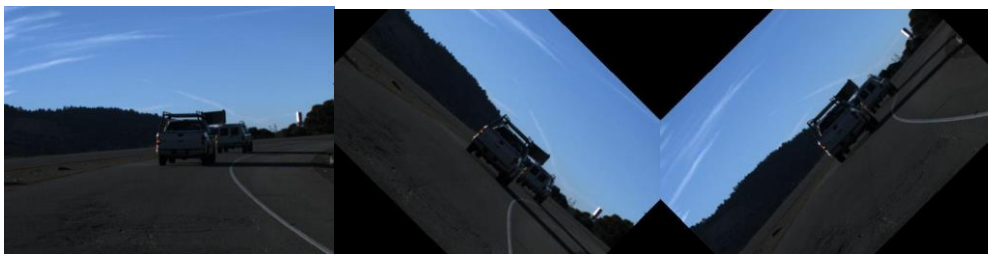

Figure 4: raw image, maximum angles of rotation.

2) Shear - This acts as a noise augmentation, designed to disrupt a clean video feed. Makes objects in the image appear wider. The grey scaling also adds a slight difficulty. The effect on one image does not correlate with the other two.

Figure 5: raw image, minimum and maximum shear levels.

3) Zoom - Similar purpose to shearing, but even more disruptive. Distorts the image and zooms in or out at a random place within the image, making objects in the image appear larger or smaller. Can go as far as creating an image with only horizontal lines and no other information. The grey scaling helps to slightly increase the difficulty. Similar to shear, this effect does not correlate. Therefore, in order to decide what steering angle to output in such situations, the model will have to rely on the other cameras. If all 3 cameras provide noisy frames, the model will have to rely on the previous frames via LSTM functionality, and essentially predict the trajectory of the vehicle.

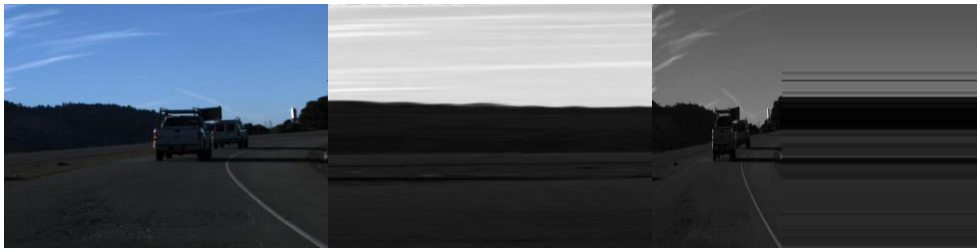

Figure 6: raw image, extremes of zoom intensity.

4) Occlude - Draws a solid black rectangle of random shape, size, and location on the image. This effect does not correlate across the 3 cameras, as the intention was to simulate random objects getting stuck on the camera when driving. These objects could be dirt, leaves, insects, etc., but when they are so close to the camera lens it becomes difficult to distinguish them. Hence, in the interest of the simplicity, all of these objects were abstracted as black rectangles.

Figure 7: raw image, sample occlusions.

5) Hue - Randomly changes the hue of the frames. This effect correlates across the 3 cameras, and acts as a proxy for a real-life situation when a vehicle passes through a lit tunnel for example, when there would be a sudden and dramatic change in colour of the lighting.

Figure 8: raw image, sample hue changes.

6) Brightness - Can reach the extremes of providing simply a near black or a near white image, depending on the base brightness of the raw image. Similar to hue, this also correlates across the 3 cameras. The intuition behind this is simple, as one can imagine a vehicle driving in the shade and then suddenly being exposed to direct sunlight, or vice versa.

Figure 9: raw image, sample brightness changes.

Level 1 Data Augmentation

The first level consisted of the most straightforward implementation of the aforementioned functions. The expected outcome is an improvement in generalisation by the model, and therefore a marginal increase in correctness and robustness. Although relatively

simple, this level is still non-trivial, as for example a real-life scenario of when an autonomous vehicle might encounter such changes would be if it were driving on rough terrain and the input images would experience sudden changes in orientation.

Technically, this was done by implementing a function that takes 3 images as input and with probability p, applies the same augmentation to all 3 frames. i.e., if p=0.5, each subset of 3 images in the dataset has a 50% chance of being augmented. The augmentation probability p is independent and the same for each subset. For n possible augmentations to choose from, each one has a probability of 1/n of being chosen. Therefore, any given image from the dataset can have at most 1 augmentation applied to it.

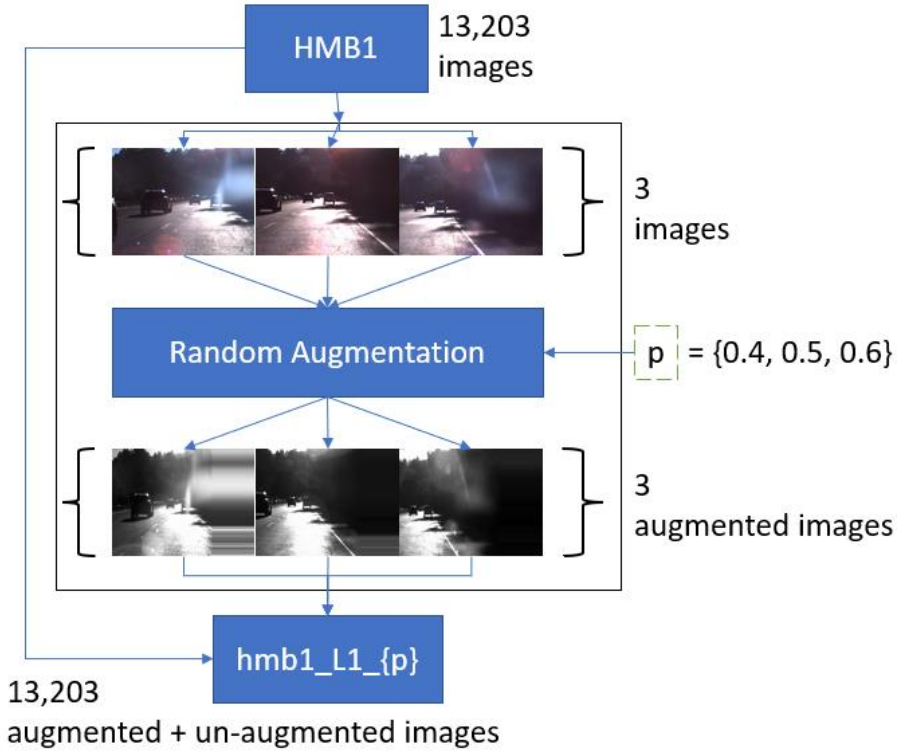

Figure 10: Level 1 data augmentation pipeline

Figure 11: Before and after images of Level 1 data augmentation (occlusion and shearing)

Three different datasets were generated in this manner, with their augmentation probability p differing (0.4, 0.5, and 0.6).

Level 2 Data Augmentation

The second level consisted of the same data augmentations, but now with smoother transitions (e.g., instead of adding a sudden 45-degree rotation to just 1 timestep, it would be a gradual increase to 45 degrees over several timesteps, and then a gradual decrease back to 0 degrees over several timesteps after the peak). This enables use of more extreme data augmentation without proving too disruptive for the LSTM.

This was implemented in a similar fashion to Level 1, but now with an added counter that kept track of the gradual increase and decrease. When an augmentation was chosen and its parameter(s) were randomly initialised, these parameters would be kept in memory. At each timestep, the parameter, multiplied by a scale factor, would be passed to the augmentation function. This scale factor starts at 0.1 and increments by 0.1 at each timestep until it reaches 1.0 (the peak of the augmentation), after which it is decremented by 0.1 at each timestep until it is finally 0.0 again. As a result, the augmentations at the start and end timesteps are only a tenth of the intensity of the augmentation at the peak. Hence, we get a smooth transition in the augmentation, that happens over several timesteps. During this period of 19 timesteps (peak, plus 9 on each side of the peak), no other augmentations are allowed to take place.

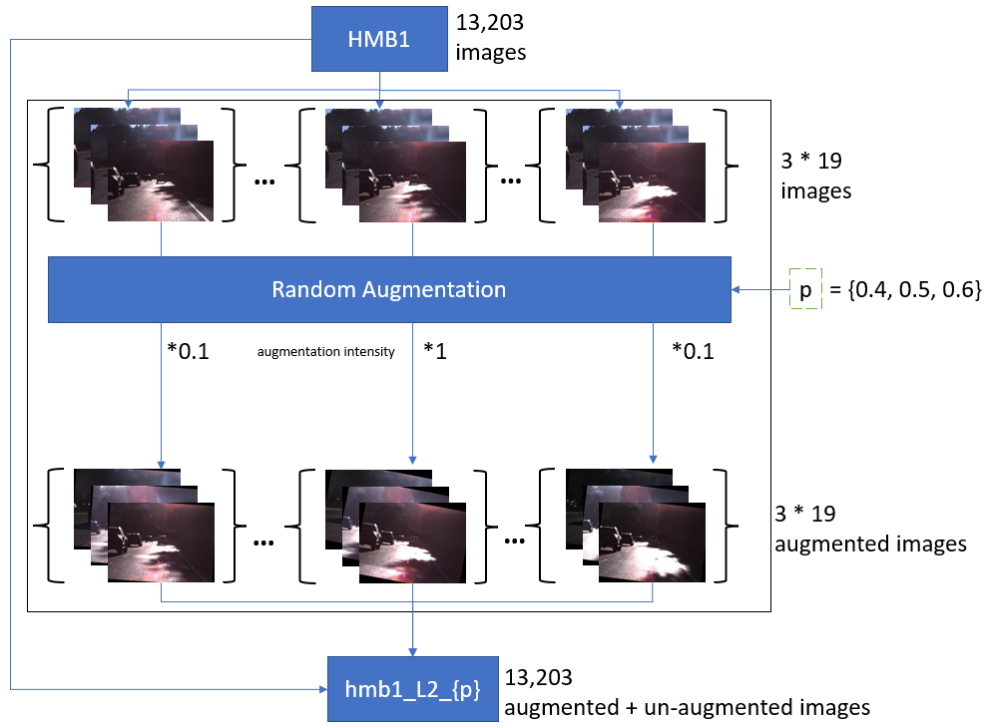

Figure 12: Level 2 data augmentation pipeline

Figure 13: Gradual intensity in Level 2 data augmentation (brightness)

Similar to Level 1, 3 different datasets were generated, with their augmentation probability p differing (0.4, 0.5, 0.6).

Level 3 Data Augmentation

The third level is an increase in complexity from the second level, where there can now be multiple smooth transitioned augmentations taking place at the same time, However, 2 augmentations do not peak at the same timestep.

This was implemented in a similar fashion to Level 2, but now with an added data structure to keep track of which augmentations to apply (along with their corresponding parameters and scale factors) at any given timestep. Furthermore, to prevent unreasonable magnification of augmentations, an augmentation cannot be applied twice to any image. As such, with n different types of augmentations available to choose from, a frame can have at most n unique augmentations applied to it, at varying intensities.

Figure 14: Sample augmented images from Level 3 data augmentation (zoom + shear, rotation + occlusion, rotation + shear)

During implementation it was found that augmentation probability p values greater than 0.25 to be too disruptive to be useful, in some cases being undecipherable even by a human. Thus, augmentation probability p values of 0.05, 0.1, and 0.25 were used instead. Although weather effects were not added to the set of augmentations, the noise, hue and brightness changes, and the occlusion together act as an abstract substitute. Furthermore, although it was initially planned to add these explicitly, increasing the number of possible augmentations meant the datasets at Level 3 would be even more difficult than they are now.

3.4) Training

Each dataset contains exactly 13,203 images, of which the first 10,000 are used for training, and the next 3,000 are used for validation after each training epoch. The remaining 203 images are not used in this phase. Only the 10,000 training images are augmented. When the model improves on its last best validation score, the weights are saved. Thus, only the best performing weights from the training phase are stored.

There was initial consideration to concatenate the augmented data to the foundation data, effectively doubling its size. This was not included in the final version of this project, since it deviated from one of the objectives of this paper, which is augmenting data to increase training efficiency. Efficiency in this case being the improvement of the model without increasing the training time. Furthermore, it would introduce entirely new permutations, causing the number of possible

datasets to increase from 10 to 1024 (2^{10}). Although this would provide a finer level of granularity to find the saturation point, initial experiments with concatenated datasets did not yield results satisfactory enough to dedicate time and resources to exploring it further.

On the other hand, the clear advantage of not concatenating the data was the ability to train models on augmented data at the same speed as on the foundation data. The only extra time required was to generate the augmented data, and so more models could be trained and tested in the allotted time.

The training phase was limited to 40 epochs per dataset. After training the model on a dataset the best fit weights were saved, and the model was reinitialised so as to create a level playing field for all the levels of augmentation. This was done once per dataset, and twice with the foundation data (HMB1). In total, this resulted in 11 different sets of weights for the model, corresponding to the 10 datasets it was trained on. The foundation data was trained on twice to mitigate the stochasticity for the baseline.

3.5) Testing

In the testing phase, fresh augmented datasets were generated, where all 13,203 images were subject to augmentation (recall that during training, only the first 10,000 were subject to augmentation). Each of the set of weights were loaded onto the model, after which it was tested with the entirety of each dataset, recording the Root Mean Squared Error (RMSE). For reference, a "bare minimum" has also been provided in the table below. This is the RMSE for the dataset if the model were to predict 0 steering angle every time. Furthermore, the weights were also tested on the entire competition dataset (hmb_concat) of 101,396 frames, to observe zero-shot behaviour.

The table below shows the average RMSEs for each category of dataset, tested by each category of weights. For example, the category of hmb1_L1 consists of the datasets hmb1_L1_4, hmb1_L1_5, and hmb1_L1_6. **A zoomed in version of this table is available in Appendix A**.

		Test Data (hmb1_L1 implies the datasets were generated from the HMB1 ROSBAG, using Level 1 data augmentations)				
	RMSE averages	hmb1	hmb1_L1	hmb1_L2	hmb1_L3	hmb_concat
	hmb1	0.0778672857518702	0.0897715457517658	0.0912454481588686	0.0955781689308091	0.2935333931598220
Training	hmb1_L1	0.0732842101114488	0.0749747804824965	0.0838170608268003	0.0877664739973350	0.2686865179184620
data	hmb1_L2	0.0671915591022779	0.0727459578531029	0.0758796695840244	0.0832973027183012	0.2604351712862080
	hmb1_L3	0.1128326231593200	0.0931759298844949	0.0811496935995853	0.0759601337047995	0.2694778639158270
	bare_min	0.0843095192934775				0.2716917077978770

Table 1: The average RMSEs when testing on the datasets.

We see in table 1 above that the model performs best on the foundation data (HMB1) when trained on a dataset augmented with Level 2 augmentations, and worst when trained with Level 3 augmentations. We see that all the weights show a gradual decrease in fitness as augmentations become more severe, except for the weights trained with Level 3 augmentations, which show the opposite behaviour. hmb_concat is a concatenated dataset consisting of the entire competition dataset. The model performs the worst on hmb_concat (even worse that the aforementioned bare minimum) when trained exclusively on the foundation data. However, it must be noted that the hmb_L3 category weights (weights set by training on a Level 3 augmented dataset) do not perform significantly worse than those of the hmb_L1 category.

In table 1, the results in the hmb1 column (and its constituent columns in table 2) are used to compare the correctness of the sets of weights trained. The results from the hmb1_L1, hmb1_L2, and hmb1_L3 columns (and their respective constituent columns in table 2) are used to observe the behaviour of the sets of weights on increasingly difficult adversarial cases.

4 Results and Analysis

Below is a more detailed table, containing the individual RMSEs for each test epoch by each set of weights. This was the table used to compute the values in table 1. **A zoomed in version (up to 16 decimal places) is available in Appendix B**.

	RMSE values	hmb1	hmb1_L1_4	hmb1_L1_	hmb1_L1_6	hmb1_L2_4	hmb1_L2_5	hmb1_L2_6	hmb1_L3_05	hmb1_L3_1	hmb1_L3_25
	hmb1	0.0762	0.0911	0.0919	0.0913	0.0916	0.0922	0.0924	0.0943	0.0968	0.0980
	hmb1	0.0795	0.0871	0.0887	0.0884	0.0894	0.0901	0.0919	0.0933	0.0947	0.0963
	hmb1_L1_4	0.0743	0.0772	0.0796	0.0761	0.0817	0.0846	0.0878	0.0865	0.0895	0.0906
Training	hmb1_L1_5	0.0731	0.0753	0.0742	0.0761	0.0809	0.0885	0.0940	0.0871	0.0859	0.0894
data	hmb1_L1_6	0.0724	0.0703	0.0729	0.0730	0.0763	0.0783	0.0823	0.0863	0.0831	0.0916
	hmb1_L2_4	0.0615	0.0659	0.0715	0.0735	0.0736	0.0753	0.0799	0.0810	0.0816	0.0885
	hmb1_L2_5	0.0685	0.0692	0.0714	0.0747	0.0744	0.0766	0.0816	0.0809	0.0814	0.0879
	hmb1_L2_6	0.0716	0.0745	0.0755	0.0785	0.0743	0.0716	0.0755	0.0802	0.0808	0.0875
	hmb1_L3_05	0.0981	0.0933	0.0920	0.0863	0.0834	0.0800	0.0709	0.0767	0.0802	0.0872
	hmb1_L3_1	0.0985	0.0888	0.0926	0.0917	0.0892	0.0878	0.0781	0.0714	0.0794	0.0769
	hmb1_L3_25	0.1419	0.0999	0.0962	0.0978	0.0819	0.0799	0.0790	0.0708	0.0709	0.0702

Table 2: RMSEs when testing on the datasets.

To understand table 2 at a glance the cells have been colour-coded, with green signifying a low RMSE, and red signifying a high RMSE. As expected, we see relatively low RMSEs in the identity, where the test data is most similar to the training data. However, this trend is not strictly followed, as when tested on HMB1 we see the weights set by training on Level 1 and 2 augmentations outperform both sets of weights obtained by training exclusively on the foundation data (HMB1).

	RMSE values	hmb1
	hmb1	0.0761931842110232
	hmb1	0.0795413872927171
	hmb1_L1_4	0.0743209520450165
Training	hmb1_L1_5	0.0730975095276850
data	hmb1_L1_6	0.0724341687616448
	hmb1_L2_4	0.0614668501564052
	hmb1_L2_5	0.0685210657553548
	hmb1_L2_6	0.0715867613950736
	hmb1_L3_05	0.0981275422611263
	hmb1_L3_1	0.0985119827274685
	hmb1_L3_25	0.1418583444893640
	bare_min	0.0843095192934775

Table 3: RMSE values when testing on foundation data (HMB1) (excerpt from table 2)

We also see that the weights set by training on hmb1_L2_4 perform the best by a significant margin. Even if we were to consider this to be an anomaly, we can look at the other results around it and say with confidence that the ideal amount of augmentation lies somewhere between hmb1_L1_6 and hmb1_L2_5. We also see that only the weights trained on Level 3 augmentations perform worse than the bare minimum. Hence, we can say that this is the saturation point, where the level of augmentation proves to be detrimental to the correctness of the model. This is because the Level 3 augmentations are so disruptive that the dataset no longer resembles the foundation dataset. As such, the model ends up learning features that may be relevant in a Level 3 augmented dataset, but irrelevant in the HMB1 dataset. This becomes more apparent when looking at the RMSE values when testing on the Level 3 datasets (table 4).

	hmb1_L3_05	hmb1_L3_1	hmb1_L3_25
hmb1	0.0943022407363760	0.0967812292871089	0.0979890976104743
hmb1	0.0933336954260189	0.0947194834179436	0.0963432671069328
hmb1_L1_4	0.0864549978541254	0.0895187804624201	0.0905678748754100
hmb1_L1_5	0.0870872042204023	0.0858878051586019	0.0894057665450590
hmb1_L1_6	0.0863133744887382	0.0830836677164991	0.0915787946547589
hmb1_L2_4	0.0809614508043111	0.0815961091604534	0.0884511499773771
hmb1_L2_5	0.0808712556221933	0.0814007973029243	0.0878853501237209
hmb1_L2_6	0.0801826032379780	0.0808082429775732	0.0875187652581799
hmb1_L3_05	0.0766680276059165	0.0801907191184995	0.0872195693798668
hmb1_L3_1	0.0714275768214177	0.0793673717063523	0.0768773935519967
hmb1_L3_25	0.0708469033926005	0.0708706683410822	0.0701729734254629
bare_min	0.0843095192934775		

Table 4: RMSE values when testing on Level 3 augmentations (excerpt from table 2)

As shown in table 4, we see that only the weights set by training on Level 3 augmentations perform well. Furthermore, only hmb1_L3_25 increases in correctness as augmentations become more severe. Thus, the domain has been randomised to an extent that it no longer corresponds to the source domain. However, it must also be noted that the weights trained on Level 3 augmentations still manage to learn useful features to outperform the baseline on the zero-shot dataset (hmb_concat), as shown in table 5 below.

		hmb_concat
	hmb1	0.3105041523810350
	hmb1	0.2765626339386080
	hmb1_L1_4	0.2685974767829800
Training	hmb1_L1_5	0.2685134487527190
data	hmb1_L1_6	0.2689486282196860
	hmb1_L2_4	0.2666457784728790
	hmb1_L2_5	0.2657325805188970
	hmb1_L2_6	0.2489271548668470
	hmb1_L3_05	0.2671862820797450
	hmb1_L3_1	0.2703285283422510
	hmb1_L3_25	0.2709187813254840
	bare_min	0.2716917077978770

Table 5: RMSE values when testing on the competition dataset.

In the above table, we see that all the weights perform poorly, which is to be expected given the reduced size of the network and the training data. However, we can still discern some useful points from these observations, such as how weights set by training on Level 2 augmentations perform best (coloured in shades of blue), and the weights trained on the foundation data perform the worst (coloured in shades of red). This, paired with their poor performance when being tested with augmented data is evidence for the lack of variety in the foundation dataset, causing even a significantly cut down neural network to overfit and be unable to generalise. They are also the only weights that failed to achieve RMSEs below the bare minimum.

We also see in table 5 that there is only a minor difference in RMSEs between the weights trained on Level 1 augmentations and the ones trained on Level 3 augmentations, implying that some level of generalisation was achieved by the Level 3 augmentation weights, even though they performed poorly on the foundation data.

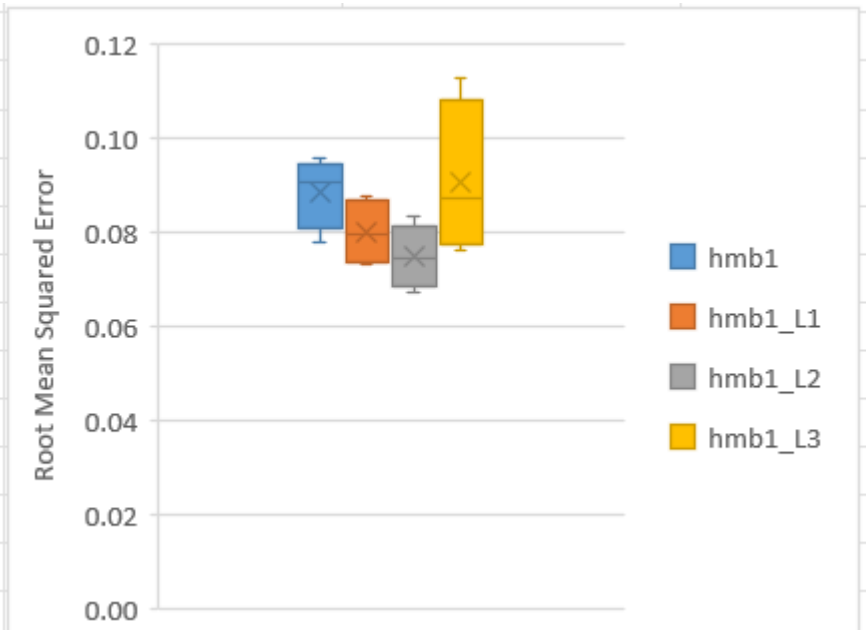

Figure 15: Box plots of RMSE averages when testing on HMB1 and augmented data.

Figure 15 gives us an idea of the consistency in correctness. Although some sets of weights trained on Level 3 augmentations (hmb1_L3, in yellow) outperform the baseline (hmb1, in blue), their overall performance is observed to be too erratic to be considered as a viable solution. Weights trained on Level 1 augmentations on the other hand (hmb1_L1, in orange), are observed to be even more consistent than those of Level 2 augmentations (hmb1_L2, in grey). A smaller range means the model is more predictable, and it is therefore safer. Figure 15 does not include the RMSEs when testing on hmb_concat, as those results should be viewed together with the other RMSEs, but should not be mixed with them in calculations. This decision was made due to the much larger size of the hmb_concat dataset and its RMSE values.

Referencing back to table 1, we observe that the average RMSEs of the baselines increase suddenly when testing on augmented data. This is evidence of the susceptibility of the baseline to adversarial and edge cases. Even if its correctness is acceptable when trained on unaugmented data, its lack of robustness to augmented data shows us how brittle the model is, where even Level 1 augmentations at probability=0.4 can cause a 15% increase in root mean squared error, to a level well below the bare minimum. Therefore, this shows us the importance of domain randomisation in this situation. In theory, these results should be scalable, and so if

the competitors of the Udacity Self-Driving Car Challenge had implemented a significant (but not excessive) level of domain randomisation during the training phase, it would have increased not only the correctness, but also the robustness of their models. The significance on the competition was to find community driven solutions to autonomous road vehicles. Understandably so, the competitors did not bother with data augmentation due to the large amount data available to them for training. However, in this project we see that data augmentation does not necessarily only have to be used to increase the amount of data available, but can also be used to increase the efficiency of the training phase by coercing the model to learn features that are more consistent. However, we have also seen that data augmentation must be done with caution, as beyond a certain point we experience a domain shift, causing the augmented data to be detrimental to the training process. From table 2, we see that sudden, single timestep augmentations (Level 1) increase the model's resilience to adversarial cases, and prolonged augmentations that take place over several timesteps (Level 2) further improves its correctness and robustness. However, the combination of several augmentations transitioning simultaneously (Level 3) proves to be the point of the aforementioned domain shift (as shown with the yellow line in figure 16 below).

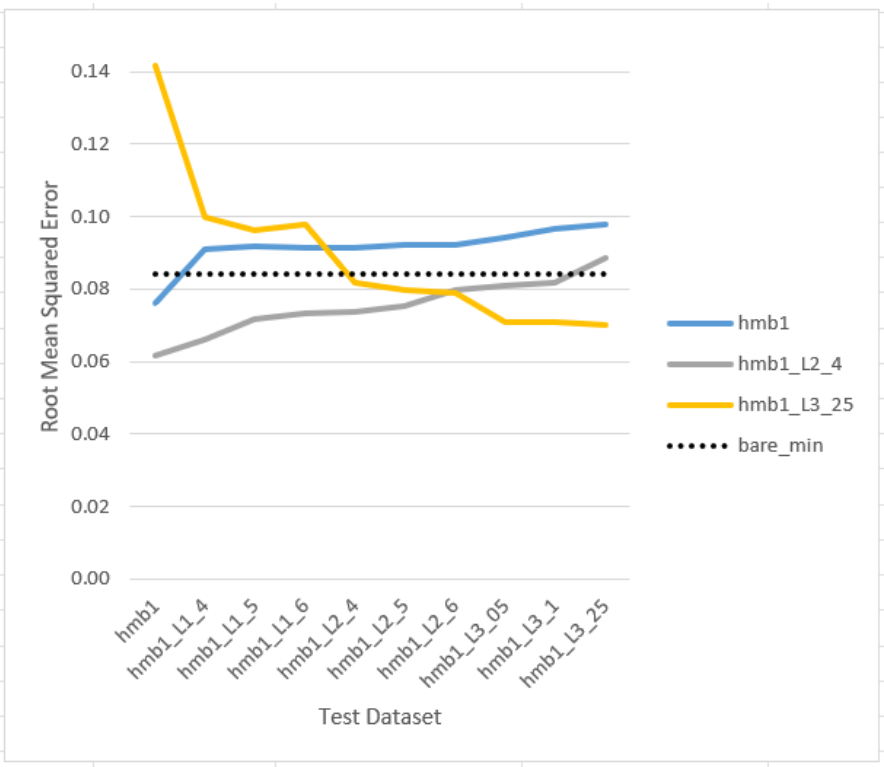

Figure 16: best and worst performing weights, compared to baseline.

Figure 16 is a representation of the stark contrast between a “good” and an “excessive” level of augmentation, with the baseline and bare minimum for comparison. As mentioned before, this graph makes it more apparent how the baseline’s RMSE (hmb1, blue) goes over the bare minimum as soon as a few adversarial cases are introduced in the test dataset. On the other hand, we see that the weights set by training on hmb1_L2_4 (grey) is much more robust, and only goes over the bare minimum RMSE at the tail-end of extreme data augmentation.

Since the model, dataset size, and training time are all kept constant, we can make a direct comparison between the datasets used. The intuition is that augmenting the data has caused the model to learn different features, ones that are less mutable. Although extracting and analysing the exact features learned from each training phase is not possible due to time constraints, we can make an educated guess by looking at the augmentations applied. For example, 2 out of the 6 augmentations turn the image into grey scale (shear and zoom), and a further 2 directly affect the colours in the scene (hue and brightness). Therefore, we can assume that the baseline, trained on unaugmented data, may have been using the colours of objects in the scene to make its decisions. While colours are not inherently a bad feature to learn, we see that the overreliance on it can lead to a lack of robustness. An example scenario would be an autonomous vehicle observing a car on the road with iridescent paint, causing its colour to rapidly change depending on the viewing angle and environmental lighting. Although simply using grey scale data to train the model would remove this reliance on colours, it gives 1 less dimension for the model to work with, and so I believe is not the optimal solution. Using RGB data, but augmenting it enables the model to use colours in its decision making, but discourages overreliance on them. Similarly, we can use this method of reasoning to guess that the absolute position of the road in the scene may have been a feature the baseline relied on, whereas the weights trained on augmented datasets may have relied on its relative position. That is, the baseline may have learned that the road is always going to be at the bottom of the image, and so had difficulty predicting the steering angle when the road was to the bottom left or bottom right (rotation). The evidence for this is the lack of banked roads in the foundation data, but with banking as steep as 45 degrees in the augmented data. Therefore, in order to not be thrown off by rotation, we can hypothesise that the model when trained on augmented data had to learn the relative positioning of the road to objects in the scene, such as vehicles, or the Armco/guardrails on the sides of the road.

As for the specific levels of augmentations, we see that Level 2 augmentation on average performs significantly better than the

second best, Level 1. The intuition is that the smooth transitions enable the model to learn the features at a variety of intensities, allowing it to potentially only learn features that are constant regardless of the intensity. For example, since the footage was recorded at 20 frames per second, a gradual increase/decrease in brightness over the course of 10 frames allows the model to observe what the scene would look like at 10 different intensities of the same augmentation. On the other hand, when the model is trained on Level 1 augmentations, it only views the scene once with the augmentation, that too with a random intensity. As a result, the Level 1 datasets cover a smaller spectrum of the intensities of each augmentation. The evidence for this is the increase in both correctness and robustness when the model is trained with Level 1 augmentations with a higher augmentation probability p (such as 0.6). With more images in the dataset being augmented at random intensities, the model can observe a greater spectrum of intensities, similar to if it were trained with Level 2 augmentations.

This is evident in figure 17 below, where we see the weights trained on hmb1_L1_6 (orange) does not perform that much worse on adversarial/augmented data when compared to weights trained on hmb1_L2_4 (grey).

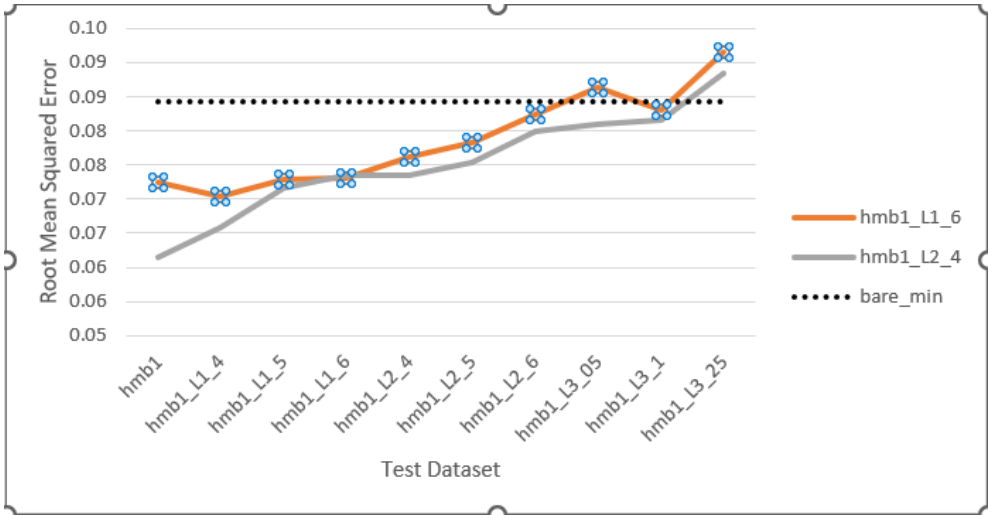

Figure 17: Level 1 and 2 augmentations compared to baseline.

Although similar in results, figure 17 also shows us Level 2 augmentation is the clear winner between the two.

Finally, we will investigate the differences in the Level 2 augmentations.

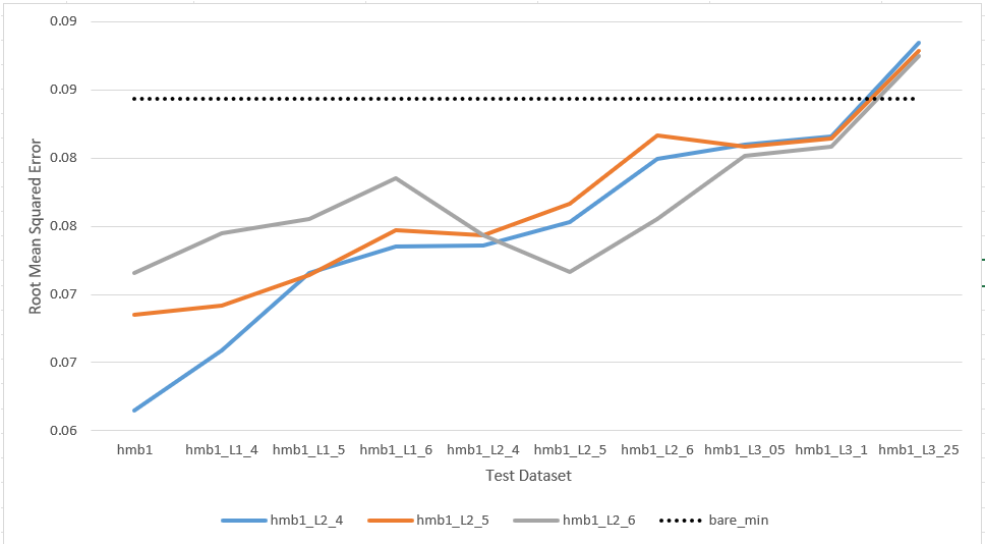

Figure 18: Comparing the Level 3 augmentations.

Figure 18 shows us how similar the weights trained on hmb1_L2_5 (orange) is to the weights trained on hmb1_L2_4 (blue) in terms of robustness, similar to figure 17. The weights trained on hmb1_L2_6 (grey) however, has a significantly different line. We see an increase in RMSE throughout the Level 1 tests, after which there is a sudden drop during the Level 2 tests. The understanding behind this is that the model may have relied on the transitions itself to make its decisions, which is why we see smaller RMSEs when testing on Level 2 than Level 1 datasets. We also see the grey line to be the lowest of the 3 throughout the tail-end of the graph, when the testing was on Level 3 augmentations. Therefore, the increase in augmentations resulted in the model essentially expecting the augmentations to be observed as a smooth transition, therefore struggling with the sudden augmentations in the Level 1 datasets, and the complete lack of them in the foundation data. As expected though, all 3 perform poorly when there are multiple augmentations combined, a phenomenon they did not observe during training.

To summarise the analyses made in this section, the experiments conducted in this project show that a moderate level of data augmentation can greatly increase both the correctness and robustness of an LSTM-based steering model, without having to increase the size of the dataset or the duration of training.

5 Conclusion

The objective was to experiment with, observe, and analyse the effects of data augmentation on the correctness and robustness of an LSTM-based steering model, without having to increase the dataset or the time it takes to train on the dataset. A total of 11 sets of weights were generated for the Komanda model by training on 10 datasets with increasing levels of augmentation. These 11 sets of weights were loaded and tested on the 10 datasets, and their root mean squared errors were recorded. With these values, we have observed the patterns/trends in these results and have hypothesised some plausible reasons. Overall, it was observed that even a small amount of augmented data in the training set can result in significant improvements to both correctness and the robustness to adversarial/edge cases. Furthermore, it was also observed that data augmentation can also increase correctness in zero-shot testing, where even the weights trained on a dataset beyond the saturation point (hmb1_L3_25) was able to outperform the baseline by a substantial margin. Therefore, data augmentation does not necessarily only have to be considered when data is scarce but must also be used to increase the overall efficiency of the training, even when the data available is vast.

A piece of future work that could be explored would be the scalability of the techniques presented in this paper. Due to hardware and time constraints, the training and augmentations had to be done on a small portion of the full competition dataset, and with a reduced neural network. These exact same techniques could potentially be applied on a larger scale to the whole dataset and trained/tested not just with the full size Komanda model, but with the other community models as well, such as Rambo and Chauffeur. Furthermore, the exact point of saturation for a model/dataset combination could be found using Surprise Adequacy, where instead of the technique being used to incrementally make the test data more difficult [13], it could be used to incrementally increase the level of augmentation in the training data. Additionally, more complex augmentations could also be implemented, as the ones used in this project are abstractions of some real-world phenomena, but not all.

Appendix A

Zoomed in version of table 1.

		Test Data	
	RMSE averages	hmb1	hmb1_L1
	hmb1	0.0778672857518702	0.0897715457517658
Training	hmb1_L1	0.0732842101114488	0.0749747804824965
data	hmb1_L2	0.0671915591022779	0.0727459578531029
	hmb1_L3	0.1128326231593200	0.0931759298844949
		hmb1_L2	hmb1_L3
	hmb1	0.0912454481588686	0.0955781689308091
	hmb1_L1	0.0838170608268003	0.0877664739973350
	hmb1_L2	0.0758796695840244	0.0832973027183012
	hmb1_L3	0.0811496935995853	0.0759601337047995
		hmb_concat	(hmb1_L1 implies the
	hmb1	0.2935333931598220	datasets were
	hmb1_L1	0.2686865179184620	generated from the
	hmb1_L2	0.2604351712862080	HMB1 ROSBAG, using
	hmb1_L3	0.2694778639158270	Level 1 augmentations)

Appendix B

Zoomed in version of table 2.

RMSE values	hmb1	hmb1_L1_4	hmb1_L1_5	hmb1_L1_6
hmb1	0.0761931842110232	0.0910864885681520	0.0919491507009386	0.0913341271178608
hmb1	0.0795413872927171	0.0871018330678838	0.0887086447852588	0.0884490302705009
hmb1_L1_4	0.0743209520450165	0.0771724042320959	0.0795637402203551	0.0761463623892724
hmb1_L1_5	0.0730975095276850	0.0753219088806858	0.0742339065300268	0.0761058334178029
hmb1_L1_6	0.0724341687616448	0.0703316212707988	0.0728699972742927	0.0730272501271378
hmb1_L2_4	0.0614668501564052	0.0658946748886285	0.0715478598541687	0.0735006561318872
hmb1_L2_5	0.0685210657553548	0.0692090792347732	0.0713860479559081	0.0747177440373197
hmb1_L2_6	0.0715867613950736	0.0744687651123135	0.0755054684388775	0.0784833250240497
hmb1_L3_05	0.0981275422611263	0.0932600040885736	0.0919918567836642	0.0862997358760284
hmb1_L3_1	0.0985119827274685	0.0888314227856979	0.0925704910473571	0.0917349947087293
hmb1_L3_25	0.1418583444893640	0.0998912188353096	0.0961653659751253	0.0978382788599688

	hmb1_L2_4	hmb1_L2_5	hmb1_L2_6
hmb1	0.0915769339529500	0.0921501758471909	0.0923611370239986
hmb1	0.0893808173570375	0.0901417288537122	0.0918618959183227
hmb1_L1_4	0.0816977831325178	0.0845794556001296	0.0878254619481234
hmb1_L1_5	0.0808905881295086	0.0884716589567735	0.0940040893957614
hmb1_L1_6	0.0762552995076312	0.0782882381924730	0.0823409725782843
hmb1_L2_4	0.0735798723187453	0.0753354800212131	0.0799179256219755
hmb1_L2_5	0.0743576461468576	0.0766196087109020	0.0816434868708331
hmb1_L2_6	0.0742987279817900	0.0716451852662899	0.0755190933176126
hmb1_L3_05	0.0834129316319488	0.0800408253723741	0.0709401183552612
hmb1_L3_1	0.0891925870974485	0.0877754357272834	0.0781104723601812
hmb1_L3_25	0.0819356942654990	0.0799227005054316	0.0790164770808402

	hmb1_L3_05	hmb1_L3_1	hmb1_L3_25
hmb1	0.0943022407363760	0.0967812292871089	0.0979890976104743
hmb1	0.0933336954260189	0.0947194834179436	0.0963432671069328
hmb1_L1_4	0.0864549978541254	0.0895187804624201	0.0905678748754100
hmb1_L1_5	0.0870872042204023	0.0858878051586019	0.0894057665450590
hmb1_L1_6	0.0863133744887382	0.0830836677164991	0.0915787946547589
hmb1_L2_4	0.0809614508043111	0.0815961091604534	0.0884511499773771
hmb1_L2_5	0.0808712556221933	0.0814007973029243	0.0878853501237209
hmb1_L2_6	0.0801826032379780	0.0808082429775732	0.0875187652581799
hmb1_L3_05	0.0766680276059165	0.0801907191184995	0.0872195693798668
hmb1_L3_1	0.0714275768214177	0.0793673717063523	0.0768773935519967
hmb1_L3_25	0.0708469033926005	0.0708706683410822	0.0701729734254629

6 Bibliography

[1] M. Bojarski and others, “End to End Learning for Self-Driving Cars,” NVIDIA Corporation, 2016. [Online]. Available: https://arxiv.org/pdf/1604.07316.pdf.

[2] Q. Rao and J. Frtunikj, “Deep Learning for Self-Driving Cars: Chances and Challenges,” in *ACM/IEEE 1st International Workshop on Software Engineering for AI in Autonomous Systems*, 2018.

[3] National Highway Traffic Safety Administration, “nhtsa.gov,” June 2022. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADAS-L2-SGO-Report-June-2022.pdf.

[4] Udacity, “Udacity Self-Driving Car Challenge,” 2016. [Online]. Available: https://github.com/udacity/self-driving-car.

[5] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[6] 3Blue1Brown, “But what is a neural network?,” [Online]. Available: https://www.youtube.com/watch?v=aircAruvnKk.

[7] J. M. Zhang, M. Harman, L. Ma and Y. Liu, “Machine Learning Testing: Survey, Landscapes and Horizons,” 2019. [Online]. Available: https://arxiv.org/pdf/1906.10742v2.pdf.

[8] Shorten and Khoshgoftaar, “A survey on Image Data Augmentation,” Journal of Big Data, 2019.

[9] J. Tobin and others, “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[10] J. Tobin and others, “Domain Randomization and Generative Models for Robotic Grasping,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[11] Movshovitz-Attias and others, “How Useful Is Photo-Realistic Rendering for Visual Learning?,” *Springer International Publishing,* 2016.

[12] K. Pei, Y. Cao, J. Yang and S. Jana, “Deep Xplore: Automated Whitebox Testing of Deep Learning Systems,” in *ACM Symposium on Operating Systems Principles (SOSP ’17)*, New York, NY, USA, 2017.

[13] J. Kim, R. Feldt and S. Yoo, “Guiding Deep Learning System Testing Using Surprise Adequacy,” in *IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019.

[14] M. Weiss, R. Chakraborty and P. Tonella, “A Review and Refinement of Surprise Adequacy,” in *IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)*, 2021.

[15] Komanda, “GitHub,” [Online]. Available: https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/komanda.

[16] Medium.com, [Online]. Available: https://medium.com/udacity/teaching-a-machine-to-steer-a-car-d73217f2492c.