Optimizing the Deep Learning Framework for Robotic Autonomous Navigation

An Honors Thesis (HONR 499)

by

Devon Current

Thesis Advisor

Dr. Shaoen Wu

Ball State University

Muncie, Indiana

April 2020

Date of Graduation

May 2019

Abstract

Autonomous systems, including autonomous cars, have become a prominent research topic in recent years due to their potentials for improving safety for users and the potential commercial success for automotive industries. The increase in studying autonomous vehicles is recently motivated by the advancements in machine learning. One of the most challenging tasks of autonomous vehicles is self-navigation, which dynamically makes decisions for movement based on the learned surrounding environment. Although many efforts have been denoted to improving the self-navigation, most of the efforts focus on solely using deep neural networks (DNN) to analyze the environmental images captured by the video sensors on the vehicles. This, along with best practices of gathering data and labelling the data for training, have led to high prediction accuracy. There, however, exists many undiscovered ways to optimize performance of nontrivial tasks that must navigate dynamic environments. In this paper, a hybrid deep learning network model composed of both convolutional layers and Long Short-Term Memory (LSTM) has been proposed to learn an environment in images by exploiting the time-series imaging data and accordingly make navigation predictions. This leads to a significant improvement in accuracy in comparison to models that do not exploit temporal information in the environmental imaging data.

Acknowledgments

I want to start by thanking my wife and family. They motivate me to do my best and have always been there to support me throughout my academic career.

I would also like to thank Dr. Shaoen Wu for being a great professor and mentor. He provided me an amazing opportunity to begin working in the field of machine learning. I know I will continue to pursue opportunities in this field, which may have not happened if I had not met him.

Process Analysis Statement

After discussing my research opportunities with Dr. Wu, we decided that I should explore robotic autonomous navigation. This is one of the fastest growing areas of machine learning in recent years, with companies such as Uber and Tesla promoting their development of self-driving cars. A previous Master's student named Junhong Xu wrote a paper over *Avoidance of Manual Labeling in Robotic Autonomous Navigation Through Multi-Sensory Semi-Supervised Learning on Big Data* (Xu, et al 2017). This paper discusses the first known case of implementing imitation learning (that is, the robot imitates the actions of an expert) with multi-sensory design. This framework was further improved in his work *Shared Multi-Task Imitation Learning for Indoor Self-Navigation* (Xu, et al 2018). Dr. Wu and I discussed the potential for improving this framework, and this proposed model is discussed in Section IV.

I began my research with no knowledge of machine learning, advanced statistics, or robotic frameworks. I spent nearly half of my research time studying these topics, so that I would have a better understanding of how to improve the existing framework for autonomous navigation. After learning enough to begin working, I began the arduous process of integrating the existing framework over to the new robot that I was to use. The previous robot was a ROOMBA with a TX1 board and a ZED camera. The robot that I used for my research was a racecar robot with a TX2 board, attached with a ZED camera, an odometer, and VESC motors. Junhong had made this robot partially work, but some of the libraries and software were outdated when I began working. I decided it would be easier to recreate environments and update everything myself before beginning my work.

I then could recreate the results that Junhong gathered using his framework from his papers. I needed to recreate these on the racecar robot because I could not compare his results of the old ROOMBA with the newer racecar robot. This is because the newer robot is faster and behaves differently, so I needed to train the model Junhong created again. I began by collecting a large dataset of images by manually driving the robot through classrooms and hallways. I then trained these images on the neural network that Junhong created, in order to make the robot navigate through environments autonomously. Afterwards, I created my own model and trained it using the same dataset that was used for Junhong's model.

An alternative topic that I covered in my paper is how to collect training data. Junhong used dataset aggregation (DAgger) to collect data. This involves using imitation learning to manually drive around the robot for the first iteration. The following four iterations have the robot drive itself, while labelling what the robot should have been doing. This leads to the robot crashing into obstacles while labelling that it should have avoided them. In real applications this could be difficult to replicate since it could place the driver or the vehicle in harm's way. I also use a method I call safe imitation learning that allows the robot to remain in control until it is about to crash. Then, before it crashes, the master takes back control of what the robot should do.

1. Introduction

1.1 Problem Statement

The subfield of machine learning known as deep learning has sparked an interest in researching autonomous driving over the past decade. With the advancements in computation power in the last decade, it has become feasible to use artificial neural networks as opposed to Visual SLAM to solve autonomous navigation problems (Milz, et al, 2018). Moreover, deep learning could potentially lead to more efficient results in comparison to SLAM. Most of the deep learning based autonomous navigation solutions use deep convolutional neural networks (CNNs) for dynamic problems such as navigating unpredictable surroundings, but there lacks the capability to exploit the temporal information that is embedded alongside the movement of vehicles.

CNNs are typically applied to imagery for computer vision. These neural networks consist of convolutional layers that allow for breaking down images into feature maps. This is applicable for tasks such as image classification, facial recognition, and optical character recognition. In contrast to CNNs, recurrent neural networks (RNNs) form a directed graph and are useful for processing temporal sequences. Their applications include sentiment analysis, speech recognition, and text classification. RNNs are capable of learning from previous inputs when making predictions of current actions. One problem that can arise from RNNs is that gradients can vanish if unable to backpropagate far enough in a sequence. Alternatively, exploding gradients could occur if weights earlier in the sequence are affected too much. Long short-term memory (LSTM) is a specific type of RNN that avoids this vanishing gradient problem

using a forget gate that allows for retaining information from previous layers (Hochreiter & Schmidhuber, 1997).

In this thesis, we propose a LSTM-CNN model with the architecture synergizing the advantages of both CNN and LSTM models. Autonomous navigation requires the use of temporally sequential images, and so both types of neural networks are required to achieve higher results. In the proposed model, the input of images are fed into CNNs, which output the feature maps of an image. The feature maps are then fed into an LSTM so that previous features can be remembered while making decisions. The output of this LSTM-CNN architecture is then further processed for multiple navigation tasks.

1.2 Potential Research Problems

Robotic navigation is essential to autonomous driving. In recent years, the efficiency of self-localization and path planning has improved. There are many techniques ranging from using reinforcement learning (Faust, et al, 2018), utilizing gateway beacon emitters (Ozick, et al, 2015), or biologically inspired methods using rats' strategies for navigation (Rama, et al, 2018). Some implementations perform better than others, depending on the applications we are using robotic navigation for. In this paper, our goal is to design a method that is efficient for generalized navigation across different indoor environments, regardless of dynamic or static obstacles.

1.3 Motivation for Deep Learning

To solve robotic navigation, we use deep learning. This allows for a flexible model that can adapt to different environments that could have dynamic obstacles. Deep learning has the

ability to learn multiple layers of representation, where each layer corresponds to its own features it abstracts from the network's input. The higher-level features build upon lower-level features, which allows for abstraction that can make sense of data (Deng & Yu, 2014). It is for this reason that deep learning is effective at making decisions that can adapt to different environments, and thus it is useful for robotic navigation.

In section 2 of this thesis, we will cover the backgrounds of CNNs and RNNs, and compare both networks' architectures. We will also discuss a related work known as the Shared Multi-Task Imitation Learning (SMIL) model, and what attributes are incorporated into our model. Similar deep learning architectures applied in sequence prediction problems are also discussed. In section 3, we present an analytical approach to viewing the LSTM-CNN architecture proposed in this paper. We also discuss alternative dataset collection methodologies that will be proposed in this paper, in an attempt to discover improvements to existing practices. Next, in section 4, we present the experiment and environment settings used in order to replicate the results of this paper. The overall performance between the Shared Multi-Task Imitation Learning (SMIL) architecture is also compared to the LSTM-CNN architecture. Finally, the conclusion and future work are discussed in Section 5.

2. Background & Related Works

2.1 Background

An image can pass through a convolutional neural network (CNN) to map the features that an image contains (Alom, et al, 2018). Features can be anything ranging from the size and shapes of objects, to locations of pixel colors in an image. The images that pass through a CNN

are stored as a tensor comprised of the images' pixels and color channels. The CNN uses kernel convolution to shift filters over a N-dimensional matrix of the images' input values. This produces a N-dimensional matrix of features. In the example presented in figure 2.1, we use a 2D matrix for input with a kernel size of 2x2. The input layer may be more than 2-dimensions if we use multiple color channels, or if the output of a CNN is fed into another CNN layer.

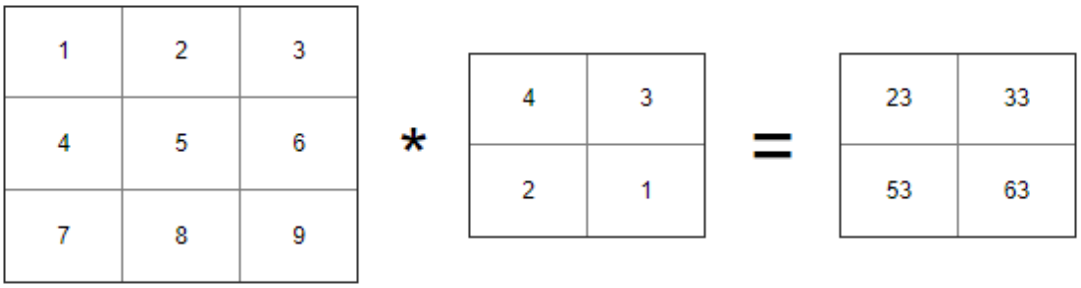


*Figure 2.1: A stride of 1 is used as the filter passes over the input values. The value of 23 is found by calculating 1*4 + 2*3 + 4*2 + 5*1. The filter then shifts to the right until it reaches the end, and then moves down to the next row.*

CNNs are a type of feedforward neural network that do not form a loop, so backpropagation is used to train the model. After passing through the CNN, the gradient of the loss function is computed with respect to each weight by the chain rule. This is performed backwards iteratively, one layer at a time, which is called backpropagation. Backpropagation can be performed by taking the derivatives of the parameters and update the weights using an optimization algorithm, such as gradient descent, to minimize the cost function.

Recurrent neural networks (RNN) are not feedforward networks. They are composed of consecutive layers, where each unit in a layer is connected to every unit in subsequent layers. This allows nodes to be connected to both upstream and downstream nodes. We can then backpropagate and adjust the weights for each neuron. However, one weakness in an RNN is that it is difficult to backpropagate in long sequences of recurrent networks. Alternatively, it

can also lead to the opposite problem with exploding gradients. One solution to this problem is a special type of RNN known as long short-term memory (LSTM).

In addition to input and output gates, LSTMs also contain a forget gate. Figure 2.2 illustrates the architecture of LSTM units unfolded so that the output Y_{t-1} of a unit feeds into the next unit along with the input X_t. The memory cell C is updated as it passes through each unit. By adjusting the forget gate and input gate, the LSTM can decide the weights of previous units' outputs while calculating the output of the current unit.

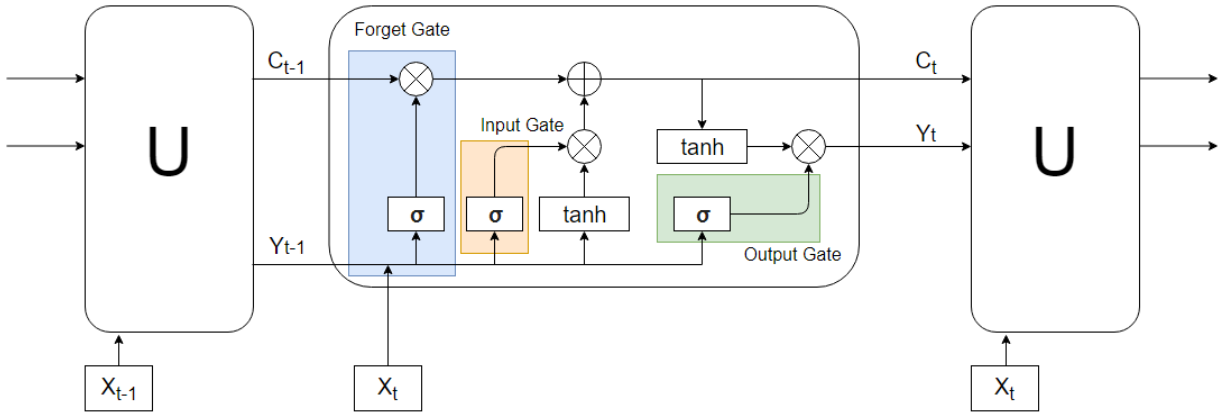


Figure 2.2: Each unit of an LSTM contains a forget gate, input gate, and output gate. The sigmoid function, σ, and hyperbolic tangent functions are used to update each gate.

2.2 Related Works

Robotic navigation is a sequential prediction problem, where imitation learning is often used to collect data. The methodology behind dataset collection is important in determining the outcome of neural networks. The datasets that are used help to dictate the performance of the policies that are trained. Dataset aggregation (DAgger) is popularly used for imitation learning, as it is well-known for training a deterministic policy with high performance and

allowing the policies to collect training data in off-course situations (Ross, et al 2010; Zhang & Cho 2016).

In Dagger, the first iteration uses expert decisions to collect data. The following iterations use the newest policy to collect data that will train the next policy. This means that the policy is making actions (that control the car in our example). The labelling of the data, however, is performed by the expert. DAgger is further illustrated in figure 2.3 using a flowchart to compare how each iteration collects data for dataset D.

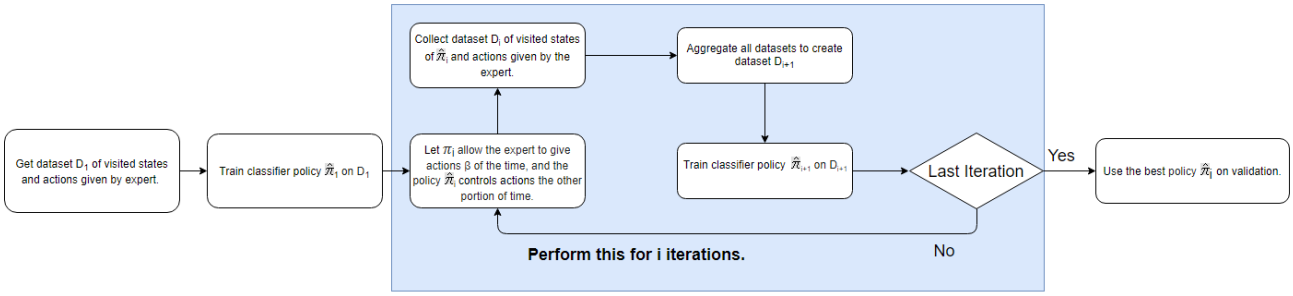


Figure 2.3: DAgger first trains a policy on expert decisions. Each successive policy is trained based on states and actions from both the current policy and the expert.

An alternative to DAgger is safe imitation learning. This is a method to train imitation learning safely in the intended, non-simulated environment, which is to be discussed in Section 3.

Self-navigation in a dynamic environment is very challenging for machine learning. Complex tasks that robots must perform cannot be defined as a single task. Tasks must be decomposed into multiple components, and the robot must become proficient in each sub-task. This is discussed in *Shared Multi-Task Imitation Learning for Indoor Self-Navigation* (Xu, et al 2018), where a remote-controlled car must drive through different environments without

collisions. Each environment demands different behaviors from the car, but all tasks are similar in that they must identify their surroundings and predict which direction to move. The multiple tasks can therefore be shared and learn from each other as they are trained. This idea of shared multi-task learning has been researched before (Caruana 1998) but multi-task imitation learning has not been researched until recently (Codevilla, et al 2017; Duan, et al 2017). Imitation learning allows for imitating a master that does not make mistakes.

In addition to using the SMIL framework, the model proposed in this paper contains Long Short-Term Memory (LSTM) units to help predict future actions from previous images. In general, LSTM-CNN architectures are utilized in application that have sequences of images (Hochreiter & Schmidhuber 1997). This is demonstrated in video action recognition where a CNN breaks down features in images and an LSTM can predict what is being done in a video based on the sequences of images (Wang, et al). An example of LSTM-CNN in robotic navigation recently is (Yang, et al 2018) where multi-modal multi-task learning is used. Our architecture is unique in that it uses shared multi-task imitation learning (SMIL) to hopefully reach higher performance with less training data required.

3. SMIL-LSTM

In this thesis, we propose a model called SMIL-LSTM, which exploits both the advantages of CNN and RNN. In the paper *Shared Multi-Task Imitation Learning for Indoor Self-Navigation* (Xu, et al 2018), robotic navigation was effective when using the SMIL model. A higher level of performance was observed compared to similar models that did not take advantage of multi-task settings or shared learning. However, there is room for improvement in the architecture of the network. Robotic navigation is a sequential problem, and so using

recurrent neural networks such as an LSTM permits the network to view previous images while calculating an action. This can be inserted into the architecture, after the CNNs break down images into features in the network.

3.1 Architecture Overview

The SMIL-LSTM model consists of three portions of the network that can be viewed as sub-networks (figure 3.1). Input will enter the model through the environmental interpretation network, which uses a ResNet-18 to break down images into features. This is passed along to the temporal network that is composed of an LSTM layer for sequential data. The last portion of the model is the task sharing network, which permits the use of shared multi-task learning for the model to distinguish tasks and train quickly.

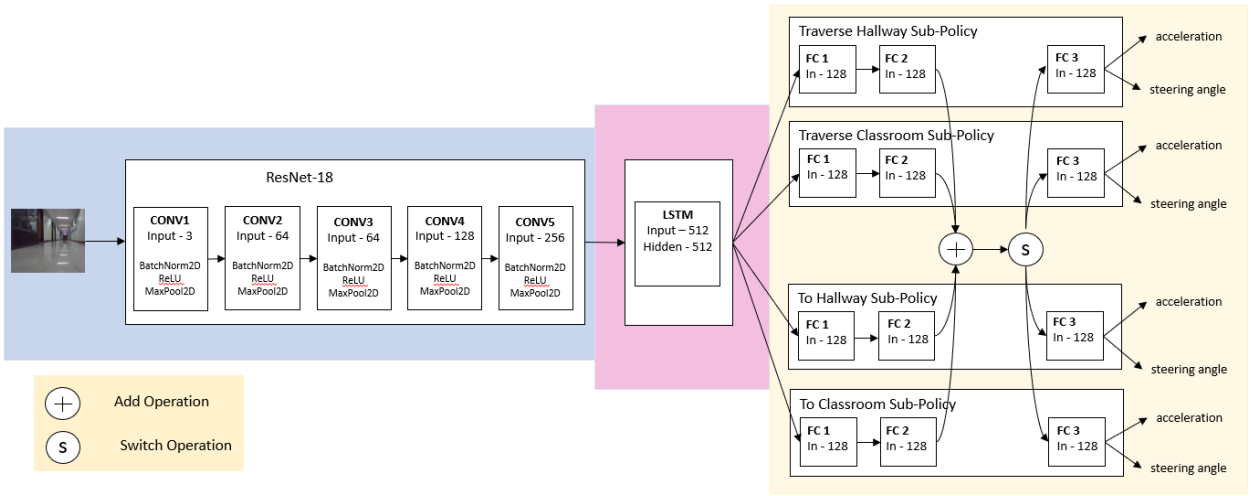


Figure 3.1: The SMIL-LSTM architecture contains 4 separate sub-policies to allow for the different tasks. These all benefit from shared learning during back-propagation.

3.2 Environmental Interpretation Network

The architecture of the LSTM-CNN model begins by passing an image into a Resnet-18 (figure 3.2). Deep networks are useful for image recognition; however, they can lead to degradation of the model. This is caused by the increasing network complexity and the diminishing gradient if the networks are too deep. Residual networks (ResNets) allow for the use of deep CNNs without impeding performance (Kaiming He, et al, 2015) . ResNets have residual blocks that skip over layers. This allows for the identity function to easily be learnt, but to also possibly learn more from each layer. Each layer in this Resnet consists of a convolutional layer, batch norm 2D, ReLU, and a maxpool 2D.

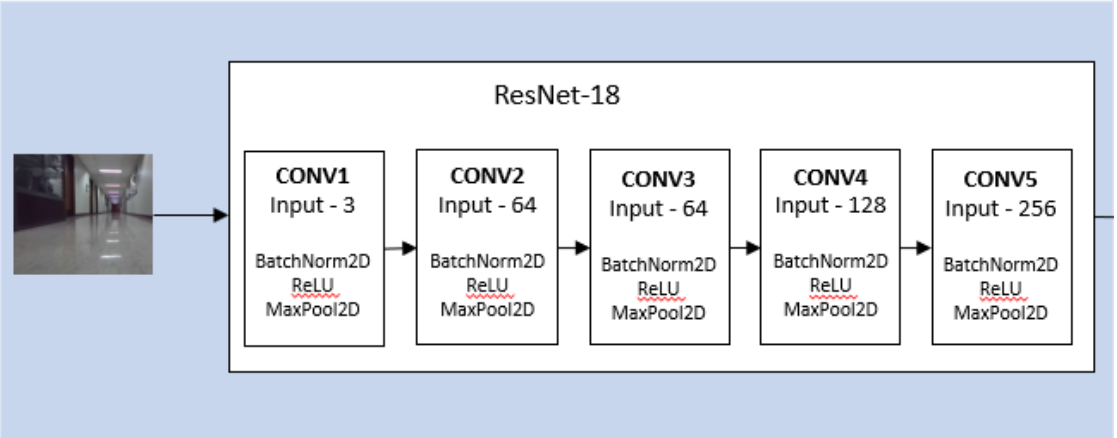


Figure 3.2: A ResNet-18 is initially used in the network for convolutional layers for visual recognition through the robot's camera.

3.3 Temporal Network

The output of the Resnet-18 is passed to a unidirectional LSTM layer (figure 3.3). This is unidirectional because the network cannot view future images in real-time while the robot is navigating an environment. This layer contains a hidden dimension of 512, meaning that there are 512 features that are created by the LSTM. The graph is retained in order to maintain the

hidden states for backpropagation. The hidden states will reset when a new video begins (in order to reset the LSTM), or if memory runs out in the system.

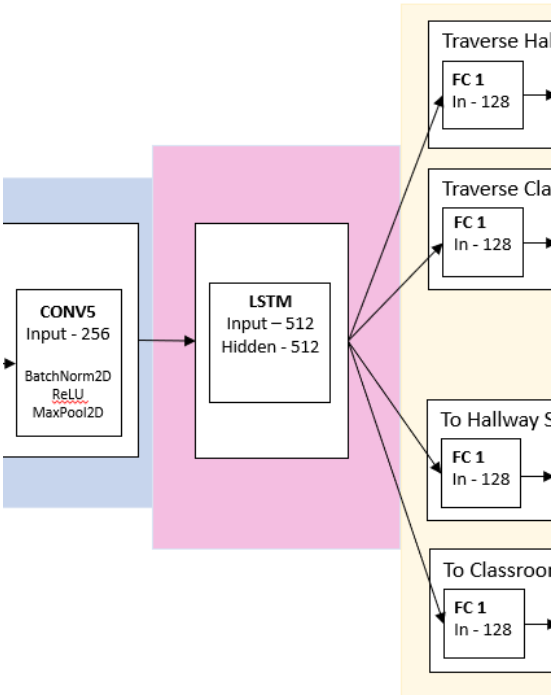


Figure 3.3: The output of the ResNet is fed into an LSTM to allow for future decisions to be influenced by previous images.

3.4 Task Sharing Network

The LSTM passes its output to a sub-policy that contains three fully connected layers (figure 3.4). The sub-policy n is chosen based on which environment the robot detects. The output of layer l_2 for each sub-policy is placed through an add operation, and then passed to l_3. This layer then predicts the acceleration and steering angle in response to the image that was passed through the network. Backpropagation allows for shared learning between all sub-policies since the gradients will move through the add operation and through l_2 of each sub-policy.

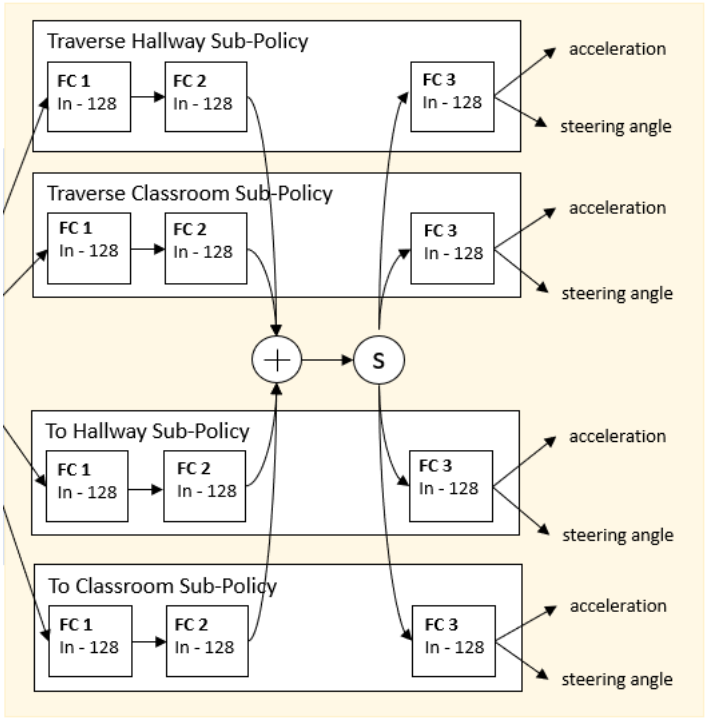


Figure 3.4: The output of the LSTM is fed into sub-task linear layers, depending on which environment the image was recognized to be trained in. Backpropogation during training causes shared learning between all sub-tasks.

3.5 Safe Imitation Learning

This research also presents safe imitation learning - an alternative to DAgger. It may not always be possible to collect off-course data, due to endangering researchers or the cost of damaging the vehicle. In these situations, it would be best to allow the policy to run but take control before any collision. DAgger is shown to be a no-regret method for online machine learning (Ross, et al 2010; Zhang & Cho 2016). This means that DAgger is able to achieve efficient results in hindsight when working on sequential data that cannot be in batch form. In this paper, we will compare safe imitation learning to DAgger and determine if it is as efficient as DAgger, which would result in it being a no-regret method.

Figure 3.5 illustrates the differences between safe imitation learning and DAgger. The first iteration for safe imitation learning is the same as DAgger – it collects images based on the

expert's decisions. The following iterations collects data based on the newest policy to train the next policy. In other words, actions and images are labelled by the policy's decisions with no expert intervention. The only instances that the expert will intervene is when the car is going to collide with an obstacle, in which case the expert takes over labelling and controlling the robot. This methodology would allow for the robot to decide what actions are best for pursuing during the training iterations, and the expert only intervenes when the robot begins to go off-track. Minimizing expert intervention allows for an alternative so that the robot can learn more naturally what actions to take.

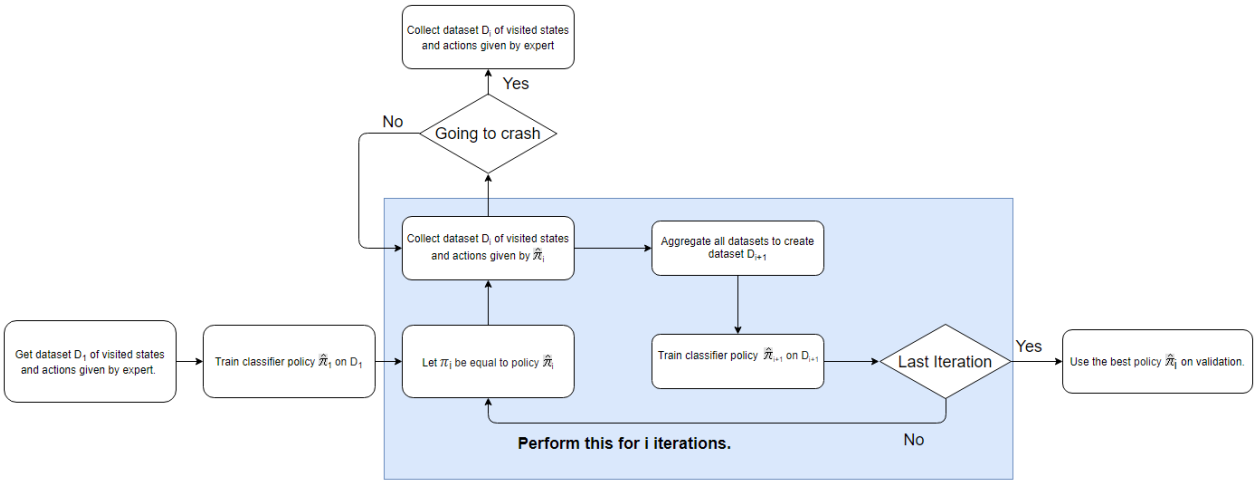


Figure 3.5: Safe imitation learning trains the first policy based on expert decisions. Unlike DAgger, each successive policy's dataset is collected by only having actions and states from the expert if the policy goes off-course.

4. Implementation and Performance Evaluation

4.1 Experiment Settings

4.1.1 Device and Equipment

The robotic vehicle that was driven in this research used a NVIDIA TX2 board, attached with a front-facing ZED camera, an odometer, and VESC motors (figure 4.1). Spring suspension

systems were used to cushion the robotic car from collisions. Robot Operating System (ROS) allowed for recording data and communicating between processes. The model was trained and tested in PyTorch. This allowed for quick, iterative implementations using Python, while also using a robust framework that provides optimized deep learning algorithms.

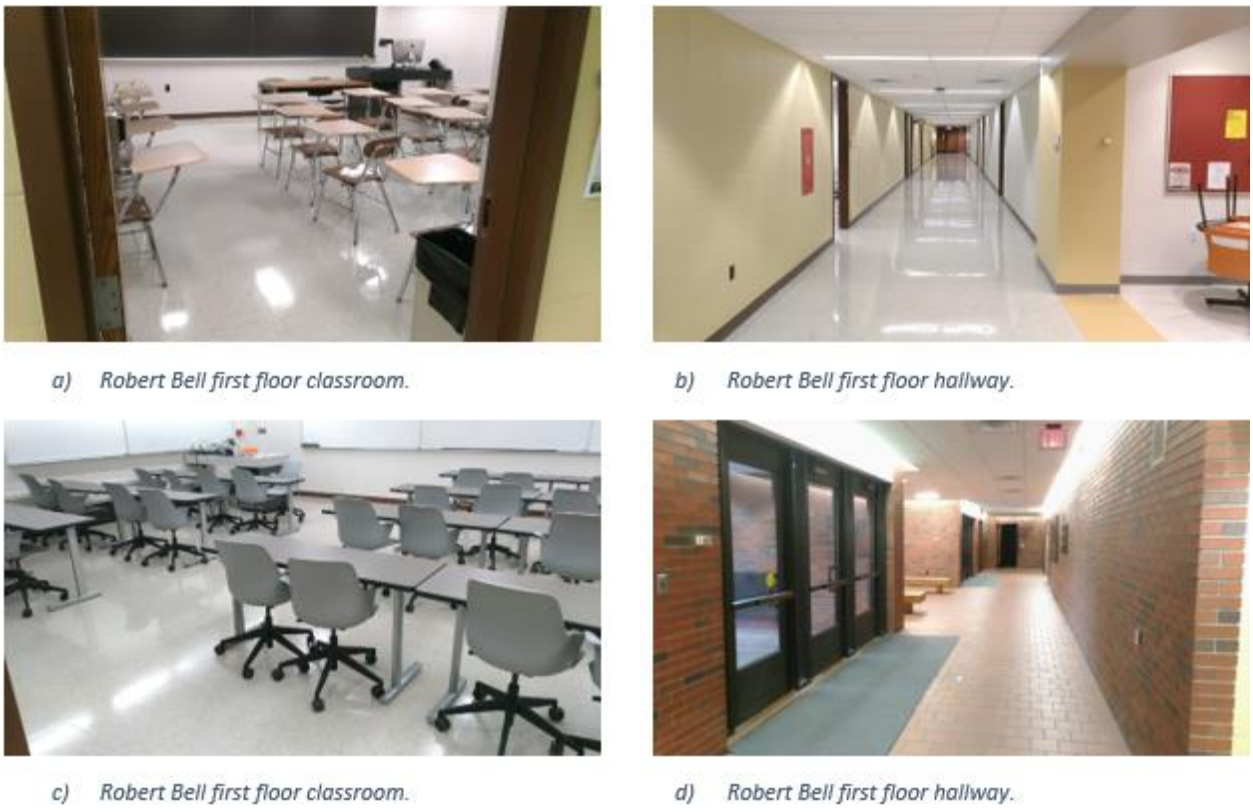
Figure 4.1: The car is built for maneuverability and durability as it navigates through classrooms and hallways.

The hyperparameters used in training were the same for all models across all tasks. Adam optimizer is used with a learning rate of $5e^{-4}$ and a weight decay of $1e^{-5}$. Although Adam often fails to converge to optimal solutions compared to SGD, it is preferred in this problem statement as it allows for quicker converging (Kingma & Ba, 2017). We train each model's policy for 30 epochs with a batch size of 256 for the SMIL models and a batch size of 64 for the LSTM-SMIL models. The LSTM-SMIL models need a smaller batch size because of limitations imposed on the memory. Dropout is set to 0.3, meaning the probability of a neuron's activation is 30% that it will be set to 0.

4.1.2 Environments and Methodologies

Both the SMIL model and LSTM-CNN model are trained iteratively 5 times each using the proposed method of safe imitation learning. Each iteration of both models collects 8,000 images for each of the four sub-tasks, which are described in the next section, for a total of 32,000 images per iteration. After collecting the new dataset, the newest policy is loaded up and trained further. Separately, two other models are trained for the SMIL and LSTM-CNN architectures using dataset aggregation (DAgger) instead of safe imitation learning.

Multiple training environments were used to help mitigate overfitting of the model (figure 2). Varying levels of brightness, color, and spatial restrictions prevented the model from learning secondary patterns that might not be relevant in practice. All models were trained in the same locations. This ensured that there were no bias to the performance of these models caused by dataset collection. Each policy's dataset consisted of ~16,000 images at Ball State University on the third floor of Robert Bell, and ~16,000 images on the first floor of Robert Bell. The testing environments to compare results of each model were held on the first floor of Robert Bell, in separate locations from where training data was collected.



a) Robert Bell first floor classroom.

b) Robert Bell first floor hallway.

c) Robert Bell first floor classroom.

d) Robert Bell first floor hallway.

Figure 4.2: Training environments (a) and (b) are different spatially and have different obstacles compared to the testing environments (c) and (d).

4.1.3 Tasks

Each model was tested with ten separate attempts to perform each task. Table 4.1 defines the descriptions and conditions for passing a task. The conditions were created in such a way that no task would perform perfectly. This allows for knowing the full extent of each model's ability, and the accuracy for each models' tasks. Time limits are imposed on each task during testing to measure if the model can discover the correct path. These were determined based on the distance that the robot would have to travel in order to reach its target location. The traverse-classroom and to-hallway tasks were also considered more challenging, with many obstacles that are difficult to detect, such as the metal legs of chairs. To make up for this, we allow for a second chance at traversal if there is a collision during an attempt.

Table 4.1: The four sub-tasks are described and failing conditions during testing are analyzed. Time limits are imposed in case the model is unable to reach its intended goal within a reasonable time.

Task	Task Description	Failing Condition	Time Limit
Traverse Hallway	The robot is initialized at one end of a hallway. It must traverse approximately 40 meters through hallways without collision.	If the robot collides into objects or exceeds the time limit, we count it as a failure. If the robot passes through a classroom, we count as a failure.	1 min 30 sec
To Classroom	The robot is initalized randomly 10 meters away from the classroom. It must pass through the entrance and reach 5 meters inside of the classroom without collision.	If the robot collides into an object or exceeds the time limit without passing through a classroom, we count it as a failure.	30 sec
Traverse Classroom	The robot is initialized in one corner of the classroom. It must traverse to the other side of the classroom without collision.	If the robot collides with an object, we reroute it back on track. If it collides a second time or exceeds the time limit without traversing the entire room, we count it as a failure.	1 min
To Hallway	The robot is initialized 10 meters away from the classroom exit. It must go through the doorway and reach 5 meters inside the hallway without collision.	If the robot collides with an object, we reroute it back on track. If it collides a second time or exceeds the time limit without exiting the classroom, we count it as a failure.	30 sec

4.2 Results

The validation loss of the SMIL-LSTM model suggests that the LSTM provides higher accuracy for the model over the simple SMIL model (figure 4.3). The first and last training iterations of the policies are compared when DAgger was used for dataset collection. The Means Square Error (MSE) was used to calculate the loss for each epoch. The comparisons of the final training runs show the validation loss of SMIL-LSTM to be nearly half that of the SMIL model.

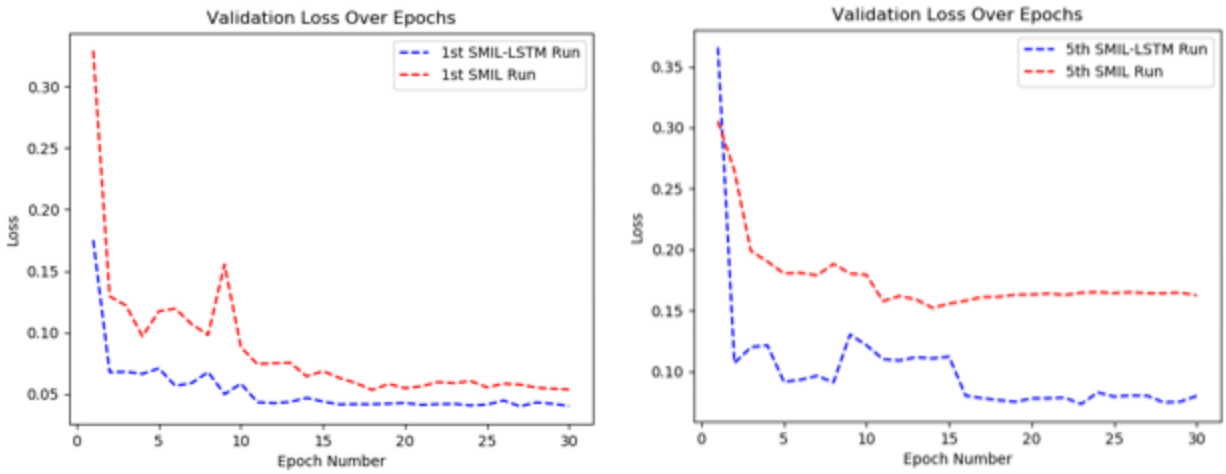


Figure 4.3: The SMIL-LSTM validation loss is less than the SMIL loss for both the first and last policies.

The SMIL-LSTM model was proven to be more accurate than the SMIL model, especially in traversing through spaces (table 4.2). When navigating all environments, the LSTM model was observed to take advantage of wider berths around edges of objects, so as not to make unnecessary collisions. In comparison, the SMIL model would cut corners and often collide when it could have avoided errors. Both models had difficulties, however, when faced with going into classrooms and hallways. When viewed from the side, all models would often avoid turning toward the doorframe. This could be caused on how the models were trained but could also have been improved with the use of multiple cameras to detect images of the doorway from the side of the car.

Table 4.2: Architecture Performances

	traverse hallway	traverse classroom	to classroom	to hallway
SMIL-LSTM DAgger	**90%**	**80%**	**60%**	**60%**
SMIL DAgger	80%	60%	70%	60%
SMIL-LSTM safe imitation	30%	40%	20%	30%
SMIL safe imitation	30%	20%	20%	20%

DAgger data collection resulted in much higher accuracy than the proposed safe imitation learning data collection, with performance more than doubled (table 4.2). The poor performance of the safe imitation learning was most likely caused by confusing directions during training. Therefore, safe imitation learning is not a no-regret method that can be used for semi-supervised learning. The autonomous policy directed the model into objects, whereas the human (expert) directed the model away. These conflicting directions most likely led to inconsistently training the model.

5. Conclusions

The addition of a long short-term memory cell to the existing SMIL model resulted in higher accuracy and more cautious behavior in the model. While attempts were made to optimize this model, different developments might also lead to better results. In contrast to the architecture used in this work, reinforcement learning might be an alternative that allows for less tedious data collection (Talpaert, et al 2019). AlexNet has also been shown to be effective when used for single color videoframes for autonomous vehicles (Teti, et al 2018), and so future work may replace the ResNet with an AlexNet. Hardware improvements also would improve performance remarkably. Cameras allowing multi-directional vision would allow the racecar to recognize objects that would otherwise be unidentified. A laser rangefinder may also provide better multi-sensory data to the model to make more informed decisions.

Dataset collection using the proposed method of safe imitation learning was inefficient compared to DAgger. This is most likely caused by conflicting commands when directing the robot. When this data is collected and trained, the model is incapable of minimizing the loss

function due to these contradictory commands. In the future, there may be further ways to improve on the quality and quantity of data. Adaptive swarm intelligence, for example, would lead to collaborative “swarm” computing continuously feeding data into the model (Zedadra, et al 2018; Vega & Pradip 2019). Both the model’s architecture and methodologies will lead to further improvements in autonomous navigation.

Bibliography

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., … Asari, V. K. (2018, September 12). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. Retrieved April 20, 2020, from https://arxiv.org/abs/1803.01164

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp,P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316

R. Caruana. (1998). Multitask learning. InLearning to learn, pages 95–133.Springer.

F. Codevilla, M. Muller, A. Dosovitskiy, A. L´opez, and V. Koltun. (2017). End-to-end driving via conditional imitation learning. arXiv preprintarXiv:1710.02410

Li Deng and Dong Yu (2014), "Deep Learning: Methods and Applications", Foundations and Trends® in Signal Processing: Vol. 7: No. 3–4, pp 197-387. http://dx.doi.org/10.1561/2000000039. Retrieve from https://www.nowpublishers.com/article/Details/SIG-039

Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. (2017). One-shot imitation learning. InAdvances in neural information processing systems, pages 1087–1098.

A. Faust *et al*., "PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, 2018, pp. 5113-5120. doi: 10.1109/ICRA.2018.8461096. Retrieved from https://ieeexplore.ieee.org/abstract/document/8461096.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2016.90

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation 9(8): 1735-1780*. Retrieved from https://www.bioinf.jku.at/publications/older/2604.pdf.

Kim, D. K., & Chen, T. (2015). Deep Neural Network for Real-Time Autonomous Indoor Navigation. Retrieved from https://arxiv.org/abs/1511.04668

Kim, J., & Canny, J. (2017). Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention. *2017 IEEE International Conference on Computer Vision (ICCV)*. doi: 10.1109/iccv.2017.320

Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." ArXiv:1412.6980 [Cs], Jan. 2017. arXiv.org, http://arxiv.org/abs/1412.6980.

Stefan Milz, Georg Arbeiter, Christian Witt, Bassam Abdallah, Senthil Yogamani (2018). Visual SLAM for Automated Driving: Exploring the Applications of Deep Learning The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 247-257.

Endri Rama, Genci Capi, Yusuke Fujimura, Norifumi Tanaka, Shigenori Kawahara, Mitsuru Jindai. Novel Biological Based Method for Robot Navigation and Localization. Journal of Electronic Science and Technology, 2018, 16(1): 16-23. Retrieved from http://www.xml-data.org/DZKJDXXBYWB/html/20180103.htm

Ross, Stephane & Gordon, Geoffrey & Bagnell, J. (2010). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. Journal of Machine Learning Research - Proceedings Track. 15.

Talpaert, Victor & Sobh, Ibrahim & Kiran, Bangalore & Mannion, Patrick & Yogamani, Senthil & Sallab, Ahmad & Perez, Patrick. (2019). Exploring applications of deep reinforcement learning for real-world autonomous driving systems.

Teti, Michael & Barenholtz, Elan & Martin, Shawn & Hahn, William. (2018). A Systematic Comparison of Deep Learning Architectures in an Autonomous Vehicle.

Vega, A. & Pradip, B. (2019). Cognitive IoT Systems via Adaptive Swarm Intelligence. *IBM T. J. Watson Research Center.* Retrieved from https://apps.dtic.mil/docs/citations/AD1076211

Wang, X., Gao, L., Song, J., & Shen, H. (2017). Beyond Frame-level CNN: Saliency-Aware 3-D CNN With LSTM for Video Action Recognition. *IEEE Signal Processing Letters,24*(4), 510-514. doi:10.1109/lsp.2016.2611485

Xu, J., Liu, Q., Guo, H., Kageza, A., Alqarni, S., & Wu, S. (2018). Shared Multi-Task Imitation Learning for Indoor Self-Navigation. *2018 IEEE Global Communications Conference (GLOBECOM)*. doi:10.1109/glocom.2018.8647614

Xu, J., Zhu, S., Guo, H., & Wu, S. (2017). Automated Labeling for Robotic Autonomous Navigation Through Multi-Sensory Semi-Supervised Learning on Big Data. *IEEE Transactions on Big Data*. doi: 10.1109/tbdata.2019.2892462

Yang, Z., Zhang, Y., Yu, J., Cai, J., & Luo, J. (2018). End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perceptions. *2018 24th International Conference on Pattern Recognition (ICPR)*. doi:10.1109/icpr.2018.8546189

Zedadra, Ouarda & Guerrieri, Antonio & Jouandeau, Nicolas & Spezzano, Giandomenico & Seridi, Hamid & Fortino, Giancarlo. (2018). Swarm intelligence-based algorithms within IoT-based systems: A review. Journal of Parallel and Distributed Computing. 122. 10.1016/j.jpdc.2018.08.007.

Zhang, J., & Cho, K. (2016). Query-Efficient Imitation Learning for End-to-End Autonomous Driving. Retrieved from https://arxiv.org/abs/1605.06450