MultiNet: Multi-Modal Multi-Task Learning for Autonomous Driving

Sauhaarda Chowdhuri[1] Tushar Pankaj[2] Karl Zipser[3]

***Abstract*— Several deep learning approaches have been applied to the autonomous driving task, many employing end-to-end deep neural networks. Autonomous driving is complex, utilizing multiple behavioral modalities ranging from lane changing to turning and stopping. However, most existing approaches do not factor in the different behavioral modalities of the driving task into the training strategy. This paper describes a technique for using Multi-Modal Multi-Task Learning, which we denote as MultiNet which considers multiple behavioral modalities as distinct modes of operation for an end-to-end autonomous deep neural network utilizing the insertion of modal information as secondary input data. Using labeled data from hours of driving our fleet of 1/10th scale model cars, we trained different neural networks to imitate the steering angle and driving speed of human control of a car. We show that in each case, MultiNet models outperform networks trained on individual tasks, while using a fraction of the number of parameters.**

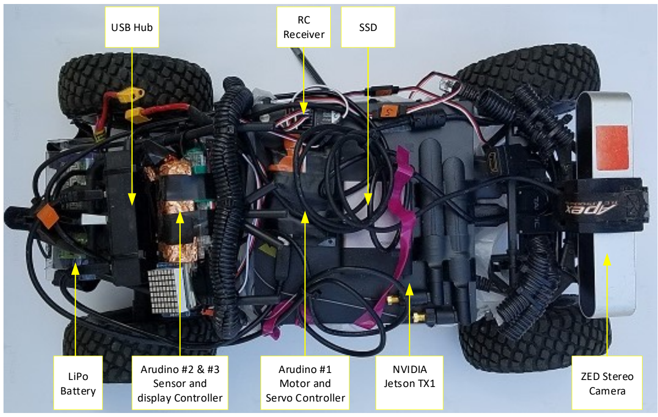

USB Hub
RC Receiver
SSD
LiPo Battery
Arudino #2 & #3 Sensor and display Controller
Arudino #1 Motor and Servo Controller
NVIDIA Jetson TX1
ZED Stereo Camera

Fig. 1: Car Diagram

I. INTRODUCTION

Most current research on driving with DNNs has focused on a single driving modality, e.g. lane following or obstacle avoidance [1], [2], [3], [4]. We consider these approaches as *Single Task Learning* (STL), as they focus on training to perform an individual task.

Multi-task learning (MTL) research has shown that training on side tasks related to the main operation of a deep neural network can enhance its learning capabilities [5], [6]. These side tasks, such as finding the position of sidewalks in the image in addition to driving with lane following, may allow networks to break down image processing into stages and develop specific filters for individual steps in a processing pipeline. In MTL, these side tasks are not used when evaluating networks in inference mode and instead improve performance on the primary task; e.g. steering angle prediction.

Additional research is being conducted on multi-modal learning, a method in which networks are trained on several distinct modes of operation, all of which are used during inference. For example, a network which has the single task of transcribing audio to text may be given a side task of sentiment analysis to improve performance in the transcribing task [7]. This is a multi-task learning network as the side-task of sentiment analysis is not needed during inference and is merely used to improve performance on a different task. If instead the network is given the task of transcribing text in two modes: one for audio, and the other for video recordings [8]. This is an example of multi-modal learning as there are multiple modes of running the network either of which can be used during evaluation.

Work on multi-modal learning has predominantly been focused in fields other than robotics or locomotion; e.g. speech recognition with audio and video [8], [7]. Within these works, it is common for DNNs to be given input that could correspond to any or multiple modes of operation.

In the context of performing multiple tasks multi-modal networks have significantly fewer parameters when compared to systems with multiple networks, as multiple related tasks can be completed by a single network rather than multiple networks. Smaller network sizes are desirable as they allow for fast over the air model updates in self driving cars, and deployment on Field Programmable Gate Arrays (FPGA) [9]. In this paper, we propose a new method for combining multi-modal learning with MTL for autonomous driving. In this method, the MTL side tasks consist of additional motor and steering values inferred by the network, which are not used for actuation on the vehicles at time of evaluation. These side-tasks are akin to sentiment analysis in the prior example. Additionally, we introduce multiple distinct driving behaviors, or *behavioral modes*, in which the model car can operate. These behaviors constitute the modes of evaluation for multi-modal learning, akin to audio and video in the previous example. The behavioral modes are given to the network as a privileged secondary input, allowing for separate driving behaviors to form within a single network.

We denote our multi-modal MTL networks as *MultiNets* in contrast to the MTL networks trained in a single behavioral

[1] Sauhaarda Chowdhuri is with Berkeley DeepDrive Center, Berkeley, CA 94720, USA sauhaarda@berkeley.edu

[2] Tushar Pankaj is with Berkeley DeepDrive Center, Berkeley, CA 94720, USA tpankaj@berkeley.edu

[3] Karl Zipser is with Berkeley DeepDrive Center and Redwood Center for Theoretical Neuroscience, University of California, Berkeley, CA 94720, USA karlzipser@berkeley.edu

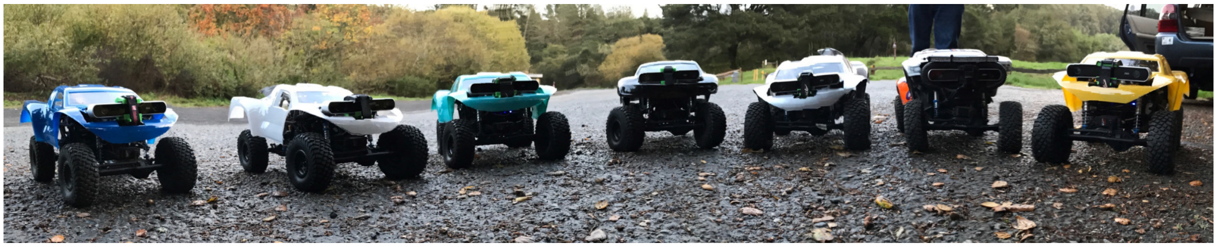
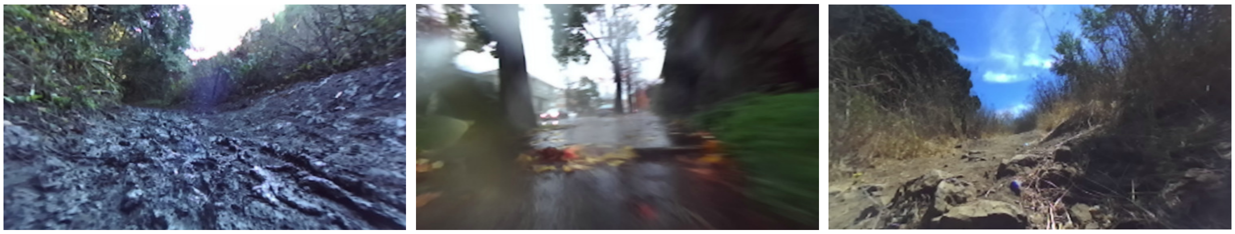


Fig. 2: Fleet of Model Cars

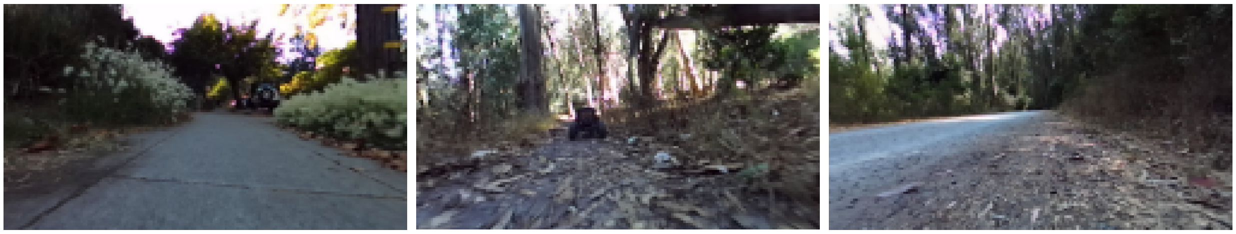


(a) Muddy Area

(b) Rainy Area

(c) Bumpy Area

Fig. 3: Diverse Conditions in Dataset



(a) Direct Mode

(b) Follow Mode

(c) Furtive Mode

Fig. 4: Behavioral Mode Sample Data from Car's Point of View

mode. We show that in addition to having a size advantage over a simple MTL approach, MultiNets exceed the performance of multiple MTL networks on the same tasks in evaluation on a validation dataset as well as in on-the-road experiments.

The concurrent work of [10] investigates a multi-modal approach with multiple sub-networks for each mode and provides a mathematical justification for the insertion of privileged modal data. Our approach differs in that a single general and scalable network is used to infer an arbitrary number of behavioral modalities using a novel logical modal switch in the processing stream of the network.

This paper is organized as follows. Section II covers the methods of collection of our dataset, as well as detailing the robotic cars used in the work. Section III describes the specific innovations of MultiNets, as well as introducing our own deep convolutional neural network, *Z2Color*, used for training and running experiments. Section IV covers the experiments conducted through evaluation of network validation loss for multiple and individual behavioral modes, as well as evaluation in on-the-road tests. Finally, Section V summarizes the major contributions of this paper and suggests areas for future work.

II. DATASET

A. *Fleet of Cars*

The dataset was collected using a fleet of 1/10th scale RC model cars (Figure 2) similar to that of [10], [11], [12] for recording data in unstructured off-road environments as well as sidewalks. Figure 1 shows the main sensor and control components of the car. The small size of the model car provides the flexibility to experiment in diverse driving terrains and lighting conditions. Our dataset contains hundreds of hours of data from environments including city sidewalks, parks, forests and snowy environments. Data were collected in different weather conditions and at different times of the day (Figure 3). Additionally, the small size of the cars allow for experiments with atypical driving behaviors and the collection of data involving the vehicle making and recovering from mistakes.

There are four computing nodes in the car – one NVIDIA Jetson TX1 [1] and three Arduino Uno[2] micro-controllers.

The nodes communicate with one another using Robot Operating System (ROS) Kinetic [13]. Arduino #1 performs

[1] TX1 Developer Kit at https://developer.nvidia.com/embedded/buy/jetson-tx1-devkit

[2] Arduino Unos available at `https://www.arduino.cc/`

pulse width modulation for the steering servo motor and power control of the DC drive motor. It also connects to an RC receiver, through which user steering and drive power commands from the RC transmitter are received. Arduino #1 provides the RC controller data to the TX1 and receives servo position and motor power information in return. The main sensor for the car is the ZED RGB stereo camera, developed by StereoLabs[3], connected to the TX1. There are optional auxiliary sensors, such as the gyroscope and accelerometer, which are controlled by Arduino #2, although no auxiliary sensors were used for the experiments described in this paper. Arduino #3 is dedicated to real-time debug message display using an 8x8 LED panel.

During the data collection process, the car is controlled using an RC Transmitter. Every 33ms, the left and right RGB images from the stereo camera, along with steering position and motor power level, are saved to on-board SSD storage. The datasets are labelled according to the behavioral mode and operation mode used.

B. Behavioral Modes

The dataset also contains annotated modes of behavior for the model car to constitute the behavioral modes used during inference for the MultiNets. We use three distinct behavioral driving modes:

1) **Direct Mode** consists of data with the car driving with few obstructions or obstacles, usually on a winding sidewalk or forest path (Figure 4a).
2) **Follow Mode** consists of data with the car following a lead car in front of it. In this mode, speed modulation occurs as maintaining an uniform distance from the lead car is attempted during driving (Figure 4b).
3) **Furtive Mode** consists of data where the car attempts to drive slowly in close proximity to perceived boundaries e.g. shrubbery or bushes on either side of a path. If no such boundaries are identified, the car speeds up along the path until one is found (Figure 4c).

C. Operation Modes

During data collection runs, the car operates in one of three operational modes: *autonomous*, *expert*, and *correctional*. The autonomous mode is used for evaluating trained networks by allowing a trained network to infer the speed and steering of the model car from the input camera data. During the autonomous mode, the RC receiver remains active, allowing the user to manually override the network's predicted steering values when needed. If a human expert monitoring the car adjusts the speed or steering on the RC transmitter, the car will automatically move into correctional mode allowing for human correction and recovery to avoid the car flipping over or hitting an obstacle. Finally, in the expert mode the expert has full control of the vehicle and records data for future imitation learning.

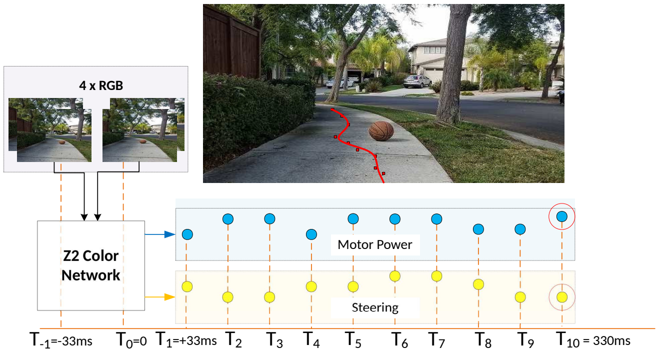


Fig. 5: Data Moment

D. Dataset Aggregation

Our system utilizes imitation learning. Imitation learning has a basic problem, known as the data mismatch problem, which occurs when a trained network encounters new situations which aren't represented in the dataset of expert driving. In this situation, error compounds quadratically over time to bring the network farther away from expert trajectory [14].

To solve this problem we implemented a novel approach to the DAgger algorithm [15] which traditionally requires manual labeling of expert trajectories after data are collected from the running network. The small size of the car allowed us to safely record data of the RC vehicle making and recovering from mistakes. New data are then merged with the active dataset for future training in the next iteration. Due to the live corrections, we are able to streamline the data collection process by solving the data mismatch problem while eliminating the need for expert labeling after the data are collected. Our dataset consisted of 19.24% correctional data and 80.76% expert data at the time of training and evaluation of the models presented in this paper.

E. Data Moments

The ROS system gathers time stamps for recorded motor, steer, and camera data. After this data are collected it is processed, interpolated, and synchronized into into packets we call *data moments*. We define a data moment as a set of four RGB input images and an associated collection of ten drive speed and steering angle values. Our networks are trained and evaluated on series of data moments. A data moment associates the input camera images to motor power and steering angles which when actuated create a spatial trajectory for the car to follow (Figure 5).

For perception of depth, we use left and right images from the stereo camera. To allow the network to perceive motion we use image pairs from two time steps – one image pair gives the current position of the car, and the other is from 33ms in the past. This way each data moment contains four RGB images.

Motor, steering, and image data are collected and stored from the car every 33ms. The latency between the network's prediction and actuation on the vehicle is 330ms to mimic

[3] ZED Stereo Camera from `https://www.stereolabs.com/zed/specs/`

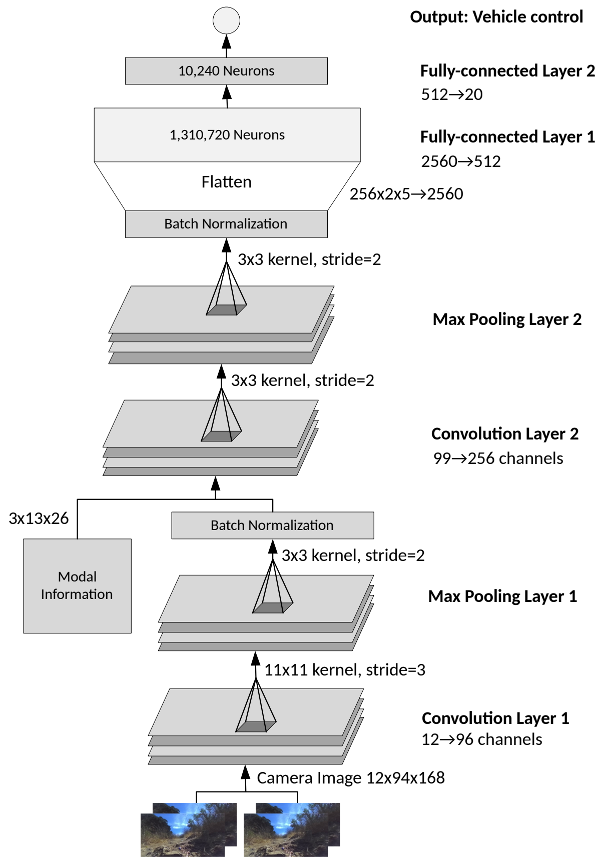


Fig. 6: MultiNet Z2Color Network Architecture with Modal Insertion

human reaction time. Thus the network predicts 330 ms into the future to account for this delay.

Rather than only training the network to predict a single steering and drive speed value 330ms into the future, we instead utilize multi-task learning to improve the network's performance. To accomplish this, we train the networks to predict 10 future time steps, each 33ms apart. In this case, only the 10th value is used for actuation and inference, while the other values serve as side-tasks to improve the cars understanding of the scene and performance on the actuated values.

While it is well known that the addition of such side-tasks benefits learning [5], we confirmed these improvements through on-the-road experiments. In these experiments, networks predicting only final actuation values were compared with MTL networks. It was observed that the MTL networks required far less manual correction, suggesting that the side tasks provide the network improved spatial awareness and driving capability.

III. METHODOLOGY

A. Network Architecture

For inference, we employ an NVIDIA Jetson TX1 system, and run a custom network we call Z2Color at a 20 Hz frequency. The network consists of two convolutional layers, followed by two fully connected layers shown in (Figure 6).

Max pooling and batch normalization were done after each convolutional layer. Max pooling allowed us to efficiently reduce dimensionality and batch normalization prevented internal covariate shift [16]. The stride and kernel sizes were found empirically through numerous cycles of training and on-road evaluation. The convolutional layers were designed to act as feature extraction layers whereas the final fully connected layers act as a steering controller. However, the network was trained in an end-to-end manner so we did not isolate different forms of processing to specific sections of the network.

The network is compact, with no more than 1.7 million parameters, taking approximately 6.5 MB for each model, allowing for Field Programmable Gate Array (FPGA) or Application-Specific Integrated Circut (ASIC) deployment. Iandola states ”FPGAs often have less than 10MB of onchip memory and no off-chip memory or storage. For inference, a sufficiently small model could be stored directly on the FPGA instead of being bottlenecked by memory bandwidth.” [9]

While a single MultiNet Z2Color network could fit on such a platform, multiple MTL Z2Color networks trained for just two behavioral modalities would would result in a 13 MB model incapable of running directly on a 10 MB FPGA. For these reasons, as the number of modalities increases multiple distinct networks become increasingly impractical for deployment whereas a MultiNet Z2Color network allows for deployment on faster platforms with smaller network sizes.

B. Modal Information

When collecting data from the cars, along with motor, steering, and image data, we also store the behavioral mode in which the car is being operated. We have trained networks with and without the insertion of the behavioral information and when added, networks more distinctly exhibit individual modal behaviors.

A network without this modal information could potentially learn multiple behavioral modes distinctly, but it would take a great amount of careful training for the filters to separate for each behavioral modality. By adding the logical modal switch in the processing stream, it becomes easier for the network to create independent filters for each behavioral mode.

The behavioral information is inserted as a three channel binary tensor, where each channel represents a behavioral modality. The behavioral mode information insertion point in the network was chosen to be after the first convolutional layer in Z2Color (Figure 6), allowing for the earlier convolutional layer to generalize basic image processing of the input data without considering behaviors of individual modalities. This replicates the visual insertion in the macaque monkey visual cortex's processing of visual data where the early visual cortex gets contextual information from the frontal cortex's feedback connections from a higher visual cortex. [17]. Additionally, similar mode agnostic convolution

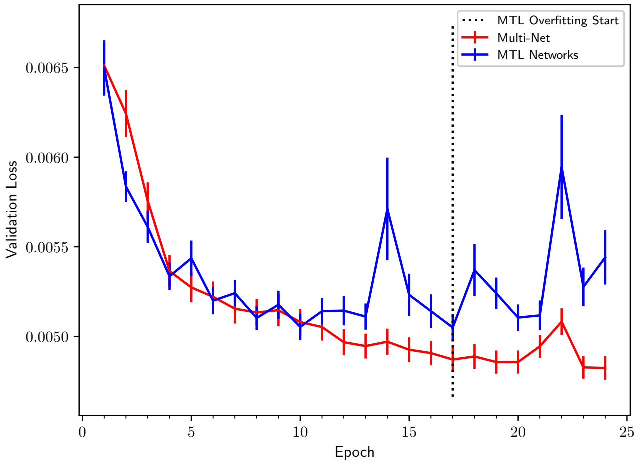


Fig. 7: Multi-Modal Validation of MultiNet and MTL Networks with 95% Confidence Intervals

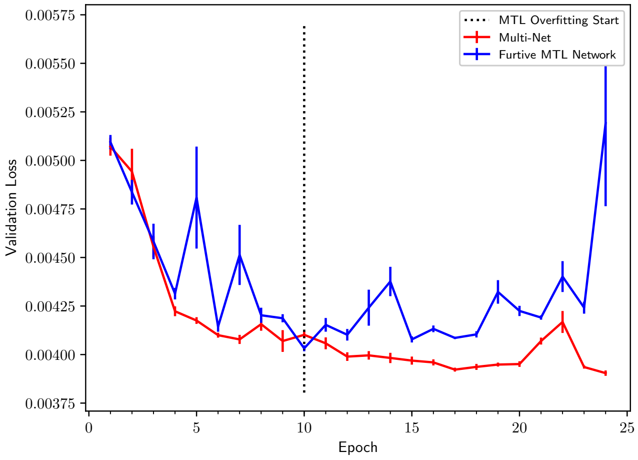


Fig. 8: Furtive Mode Validation of MultiNet and MTL Networks with 95% Confidence Intervals

or processing in initial layers of a network have been shown to be effective with multiple modalities [18].

IV. EXPERIMENTS

A. Training

To train our networks, we used the PyTorch[4] deep learning framework. All networks were evaluated with Mean Squared Error (MSE) Loss and used the Adadelta Optimizer [19]. The same set of training data were used for each of the networks, as well as the same unseen validation set. All experiments were replicated eight times with randomly initialized networks and shuffled datasets. The results here depict the mean across these trials, with error bars representing 95% confidence intervals.

Our dataset contains approximately 1.93 million usable data moments for training and validation on the networks. 10% of the collected data were kept for use in an unseen validation dataset for the evaluation of the networks. All data were equally distributed for each modality in both the training and validation sets.

B. Multi Modal Comparison

In our initial experiment a MultiNet Z2Color network trained in a multi modal dataset of direct, follow, and furtive was compared to three MTL Z2Color networks trained on direct, follow, and furtive modes separately. Both networks were trained to output additional intermediate motor and steering values not used during inference. The three MTL networks were only fed data that corresponded to their individual behavioral modes. The MultiNet network was fed data containing all three behavioral modes. The results are summarized in Figure 7 where the losses of the three MTL networks are averaged for direct comparison to the MultiNets.

Initially, from epochs 1 to 4, the MultiNets have similar but slightly poorer performance compared to the MTL networks. This is due to the wide variety of data the MultiNets receive requiring greater generalization initially, while the MTL networks can immediately specialize to specific modes.

From epochs 4 to 10, the MultiNets begin to surpass the MTL networks while remaining close in performance. During this period the MTL networks begin to differentiate between individual driving modalities by using the provided modal information data.

From epochs 10 to 17, the MultiNets drastically outperform the MTL networks, which flatten off in their loss curve here. The MTL loss curve begins to move erratically by getting caught in various local minima. However it doesn't yet begin overfitting, which we characterize as consistently having a loss value above the absolute minimum. The MultiNets steadily improve through the use of the additional modal data. From epochs 17 to 24, the MTL networks begin to overfit dramatically, while the MultiNets continue to decline in loss despite a small bump at epochs 21 and 22. This suggests MTL networks are more susceptible to overfitting and local minima than their MultiNet counterpart.

C. Performance in Individual Modes

To further investigate the network's performance in individual behavioral modes, we decided to compare MultiNet to a single MTL network in a single mode: furtive. This mode is interesting as it involves more complex behaviors than traditional driving tasks like lane following. In these experiments (Figure 8), the MultiNets were trained on Direct, Follow, and Furtive modes but were only validated in Furtive mode, while the MTL net was trained and validated specifically on Furtive mode data.

Initially, from epochs 1 to 4 the MultiNets have similar performance to the furtive networks. From epochs 4 to 10, the MultiNets fall steadily in loss, while the MTL networks oscillate erratically in local minima. From epochs 10 to 24, the MTL networks overfit while the MultiNets continue learning. This demonstrates that for any given mode, a

[4] `https://github.com/pytorch/pytorch`

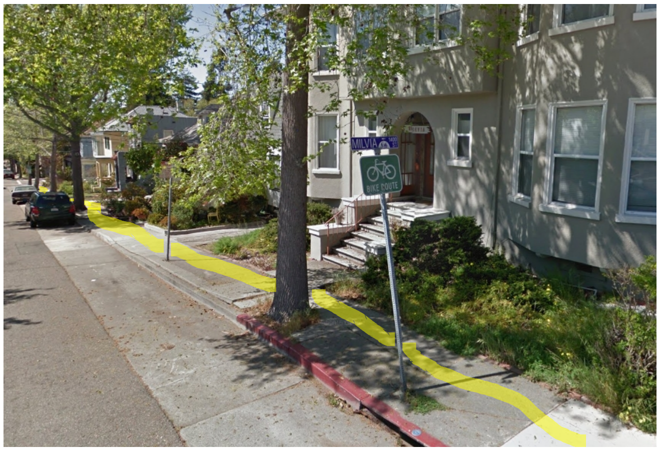


Fig. 9: Segment of Evaluation Circuit

MultiNet network can outperform an MTL network trained for the specific mode.

D. Evaluation on Model Cars

To test the proficiency of the cars in real world driving situations, we measure the percentage autonomy metric [1] measured as

$$autonomy = (1 - \frac{correction\ time}{elapsed\ time}) \cdot 100 \qquad (1)$$

MTL and MultiNets were evaluated on a winding 200 m loop of sidewalk (Figure 9) with sufficient obstacles within a one hour interval. Follow mode was excluded as driving of the leader car could affect performance of the following car. The networks were chosen at point of absolute minimum average validation loss across trials, which occured at 23 epochs on a specific trial.

The results are summarized as follows: The MultiNet in direct, and furtive mode scored 92.68% and 88.23% autonomy respectively. The MTL networks scored 84.27% and 87.55% in direct and furtive modes. Comparatively, in direct mode the MultiNet was 8.31 % more autonomous than the network trained only on direct mode data. In furtive mode the MultiNet was 0.68 % more autonomous than the MTL net, matching the results from validation on furtive mode and across modes (Figures 7 and 8).

V. CONCLUSION & FUTURE WORKS

This paper proposes a methodology for training DNNs to perform several distinct behavioral modalities simultaneously, through the insertion of modal information. This MultiNet approach is shown to exceed the performance of multiple individual networks trained separately, while using fewer parameters. These results are then verified with real world evaluation of the networks in sidewalk driving situations using 1/10th scale model cars. Future work could include work on adapting the approach to full size vehicles and making modal information available from higher-level networks trained to select behavioral modes, thereby granting the system a qualitatively higher level of autonomy.

ACKNOWLEDGMENTS

The authors gratefully acknowledge NVIDIA for the donation of the NVIDIA TX1 Developer Kits, as well as Berkeley DeepDrive and it's sponsors for their support. We thank Sascha Hornauer and Eric Hou for their review, as well as Bhaskar Chowdhuri for assistance with figures.

REFERENCES

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

[2] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, 2006, pp. 739–746.

[3] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[4] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[5] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

[6] Y. Zhang and D. Yeung, "A convex formulation for learning task relationships in multi-task learning," *CoRR*, vol. abs/1203.3536, 2012.

[7] M. L. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6965–6969.

[8] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[9] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016.

[10] F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun, "End-to-end driving via conditional imitation learning," *arXiv preprint arXiv:1710.02410*, 2017.

[11] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1714–1721.

[12] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[14] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 661–668.

[15] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[17] K. Zipser, V. A. Lamme, and P. H. Schiller, "Contextual modulation in primary visual cortex," *Journal of Neuroscience*, vol. 16, no. 22, pp. 7376–7389, 1996.

[18] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 656–664.

[19] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.