Reconfigurable Network for Efficient Inferencing in Autonomous Vehicles

Shihong Fang* and Anna Choromanska*

Abstract— **We propose a reconfigurable network for efficient inference dedicated to autonomous platforms equipped with multiple perception sensors. The size of the network for steering autonomous platforms grows proportionally to the number of installed sensors eventually preventing the usage of multiple sensors in real-time applications due to an inefficient inference. Our approach hinges on the observation that multiple sensors provide a large stream of data, where only a fraction of the data is relevant for the performed task at any given moment in time. The architecture of the reconfigurable network that we propose contains separate feature extractors, called experts, for each sensor. The decisive block of our model is the gating network, which online decides which sensor provides the data that is most relevant for driving. It then reconfigures the network by activating only the relevant expert corresponding to that sensor and deactivating the remaining ones. As a consequence, the model never extracts features from data that are irrelevant for driving. The gating network takes the data from all inputs and thus to avoid explosion of computation time and memory space it has to be realized as a small and shallow network. We verify our model on the unmanned ground vehicle (UGV) comprising of the 1/6 scale remote control truck equipped with three cameras. We demonstrate that the reconfigurable network correctly chooses experts in real-time allowing the reduction of computations cost for the whole model without deteriorating its performance.**

I. INTRODUCTION

A plethora of real-life problems are currently addressed with convolutional neural networks (CNNs), including image recognition [1], [2] and segmentation [3], speech recognition [4], and natural language processing [5]. In some learning tasks the performance of deep networks exceeds that of a human [2]. Due to the rapid advances in hardware technologies leading to powerful and efficient GPUs and mobile supercomputers, deep learning techniques were more recently successfully used in the complex intelligent autonomous systems such as self-driving cars [6], [7], [8]. Deep learning techniques enable automatic extraction of data features and consequently allow scaling-up learning systems to large data settings. Due to automatic feature extraction, employing multiple sensors in platforms based on deep learning became much easier than in case of rule-based models. Using multiple sensors improve the safety of autonomous vehicles by increasing its perception abilities and became an industrial standard used in Advanced Driver Assistance Systems (ADAS) and autonomous car driving platforms of Google, NVIDIA, UBER, or Intel.

*The authors are with the Machine Learning Lab, Department of Electrical and Computer Engineering, New York University, 5 MetroTech Center, USA. {sf2584, ac5455}@nyu.edu

The bottleneck for using multiple sensors on the autonomous platform is the size of the network that needs to process the data registered by all the sensors in order to output the steering command. The size of network needs to expand proportionally to accommodate the incoming data which hurts the inference time and affects the real-time operation of the resulting model. Our work addresses this problem. We design a reconfigurable network, which contains feature extractors, that we call experts, each processing the data coming from a different sensor, where at any given point in time only one expert is active and the remaining ones are deactivated. Thus the network reconfigures itself online guided by the gating network, which each time chooses the most relevant sensor. To empirically verify our approach we built a UGV equipped with three front-facing cameras covering left, center, and right field of view. We demonstrate that our model can steer the UGV in real time and choose correct sensors for navigation. Furthermore, we demonstrate that the reconfigurable network exhibits similar performance to the standard network without the gating mechanism and requires significantly less computations.

This paper is organized as follows: Section II reviews relevant literature, Section III discusses the reconfigurable network (architecture and training), Section IV shows empirical evaluation, and Section V concludes the paper with a brief summary of findings.

II. RELATED WORK

A. End-to-end learning for autonomous vehicles

An end-to-end learning system for steering autonomous vehicles typically learns the mapping from the raw readings from the sensors to the corresponding steering commands via supervised learning. The training data are acquired from human drivers, where inputs to the learning system obtained from the sensors are captured together with driver actions. A system known as Autonomous Land Vehicle in a Neural Network (ALVINN) [9] was the first end-to-end learning system for autonomous driving and was based on fully-connected network. Later introduced convolutional neural networks (CNNs) for data feature extraction [10] were applied in DARPA Autonomous Vehicle (DAVE) project [11]. The DAVE robot was able to drive autonomously using left and right cameras. Recently, CNNs were used to train a real car to drive autonomously with good performance [7]. Authors used three cameras in data collection phase and a single one for actual driving. Their later work [12] explains why CNNs are very effective in self-driving tasks. Besides cameras, other sensors like LiDARs were used to provide multi-modal input to the networks [13], [14] and make the system more robust to the variability of the real world.

B. *Gating mechanism*

The first gating network proposed in the literature [15] divides learning task into subtasks and uses separate expert for each subtask. Each expert takes the same input and the gating network decides which experts to use at any given moment. It can be viewed as an ensemble method that uses a combination of selected experts to improve prediction. This idea was extended [16] to a tree-structured architecture called Hierarchical Mixture of Experts that solves nonlinear supervised learning problems by dividing the input space into a nested set of regions and fitting simple surfaces to the data that fall in these regions. The gating mechanism was also applied in mixture of SVMs [17], where the authors demonstrated the feasibility of training SVMs on large datasets. A stacked Deep Mixture of Experts model [18] with multiple sets of gating and experts was later proposed utilizing the concept of conditional computation. Conditional computation policies were also explored in the context of reinforcement learning [19]. Another notable work [20] presents an approach that can adaptively choose parts of the whole neural network to be evaluated for efficient inference purposes. Similarly, this can be achieved by introducing the sparsely-gated mixture-of-experts layer [21] that allows to pick the best performing subset of k experts, which constitute the feed-forward sub-networks. A trainable gating network determines a sparse combination of these experts to use for each example. The authors demonstrate the plausibility of their approach in language modeling problems and machine translation, where they show improvements in the computational costs compared to standard approaches.

Gating mechanism was also applied to sensor fusion in order to boost model's performance. Multi-sensor fusion allows capturing more diverse data representation (i.e. multiple sensors can capture various modalities) and benefit from wider space coverage (i.e. multiple sensors can have large combined field of view). These factors are crucial to increase reliability and accuracy of the system. For example, in object detection problems, the network performing fusion of LiDAR and camera data has shown to be successful in practical settings [22]. Combining such approach with gated multi-modal method was found to outperform other conventional fusion techniques [23].

Our work builds on the intuition that in the autonomous driving applications, due to heavy redundancies in the data, one can use a subset of the inputs at any given moment to steer the car, i.e. when driving forward, the sensors situated on vehicle's side and back most often do not provide relevant information. Thus, we propose to utilize the idea of gating network and sparse gating as a practical way to handle multi-sensor input and at the same time avoid the explosion of the computational cost. Our approach differs from other works existing in the literature in that it allows switching the experts on/off (in contrary to [13]) and it allows the experts to use different input data (in contrary to [21]).

III. Reconfigurable Network

The reconfigurable network proposed in this paper is captured in Fig. 1. The network receives n input signals from various sensors, i.e. the input to the network can be written as $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, where x_i is the signal from i^{th} sensor. Reconfigurable network consists of n expert networks $E_1, E_2, \ldots, E_n$, the gating network G, and the fully connected layers. The expert networks extract features from the inputs and the gating network decides which input signals x_i should be processed for a given input $\mathbf{x}$. Thus, only the experts corresponding to the selected input signals need to perform calculations. Finally, the fully connected layers form the final prediction based on combined feature vector from all experts that processed the data. Thus, the final prediction $\hat{y}$ can be written as:

$$\hat{y} = F\left(\sum_{i=1}^{n} G_i(\mathbf{x}) E_i(x_i)\right), \tag{1}$$

where $G_i(\mathbf{x})$ is the weight of expert i ($\in [0,1]$, where 0 denotes the situation when the expert is not used) for input $\mathbf{x}$ assigned by the gating mechanism, $E_i(x_i)$ is the feature vector outputted by expert i for input x_i, and F denotes the action of fully connected layers.

The gating network in the proposed architecture selects the subset of experts which will process the data. Thus, the gating network needs to properly estimate the relevance of each expert for the given learning task, not just the input x_i itself, i.e. even if x_i contain relevant information for driving the autonomous vehicle, expert E_i should not be activated if it is unable to extract high-quality features from x_i. It makes the reconfigurable network difficult to train in an end-to-end fashion as there is no explicit correlation between experts and the feature extractors used in the gating network. In order to overcome the mentioned problem we propose the dedicated training procedure.

In the first step of the proposed training procedure, we train the sensor fusion network proposed before in the literature [13] and illustrated in Fig. 2. In this step, the gating network uses experts as feature extractors, which solves the problem mentioned before. The outputs of the experts are scaled by the outputs of the gating network and then concatenated to form the combined feature vector. The obtained feature vector is then passed to the fully-connected

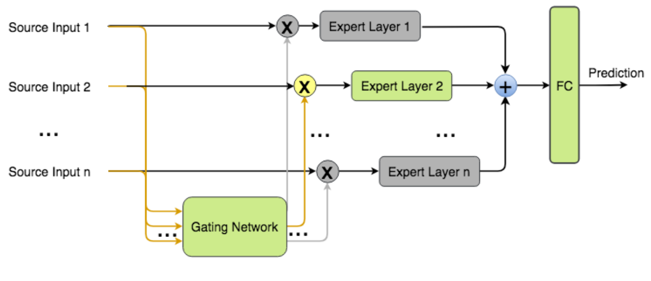


Fig. 1. Reconfigurable network with the gating mechanism. In this example, the second expert is activated based on the output of the gating network and the other experts are disabled (colored in gray).

layers (FC), which forms the final prediction. To ensure that feature extractors capture necessary representation, we first pre-train each expert individually to make the final prediction. We then borrow weights from our pre-trained models for initialization. Finally, we train this network end-to-end to obtain the properly trained experts and a functional gating network.

In the second step of the proposed training procedure, we construct the reconfigurable network that contains a very specific gating component as shown in Fig. 3 (we ask the reader to look either at the top or bottom figure as both of them contain the same gating network shown in the shaded green box). In this step we only train the gating network. The newly constructed gating network has it own feature extractors, which are significantly smaller than experts in the main part of the network. The new gating network is trained in a supervised setting to mimic the behavior of the reference gating network obtained from the first step. At this stage we are enforcing sparsity by modifying training labels for the new gating network. In particular, we convert label vector $\mathbf{v} = [v_1, ..., v_n]$ obtained from reference gating network, and convert it into one-hot vector, where max value from $\mathbf{v}$ is converted to 1, and rest to 0. Additionally, after training the new gating network, we add the hard thresholding on the output, so the output is always one-hot vector. This will force the reconfigurable network to use only one input, or equivalently expert, at a time. The extension to constrained number of inputs will be explained later in this section. At the end of this stage we obtain the compact version of gating network with desired behavior.

In the third step, we fine tune the experts and the fully-connected layers (keeping the gating network fixed) by training them all together on the same data as in the first step. This step allows the reconfigurable network to adjust to the modified behavior of the gating network. In the experimental section the resulting network is called *"Reconf_Concat"* (see top of Fig. 3).

So far, we have used concatenation for combining feature vectors obtained from the experts. This results in a large combined feature vector which leads to significant amount of computations. Instead we propose to use point-wise summation instead of concatenation since we want to use one selected input at a time. In this case, the combined feature vector is equal to the feature vector corresponding to the selected input. As the encoding of input by expert may be different for each input and expert, we pass the gating network output together with a combined feature vector to the fully connected layers. This way, the fully connected layer has the information about which input is used at the moment and can process it properly. Finally, the expert networks are fine-tuned and the the fully-connected layers are trained from scratch, while keeping the gating network fixed. In the experimental section the resulting network is called *"Reconf_Select"* (see bottom of Fig. 3). .

In each step, we train the network until convergence. The first step of training is critical. As the gating network is trained in the first step and used as a reference in the following steps, shortening the first step too much may significantly degrade the performance.

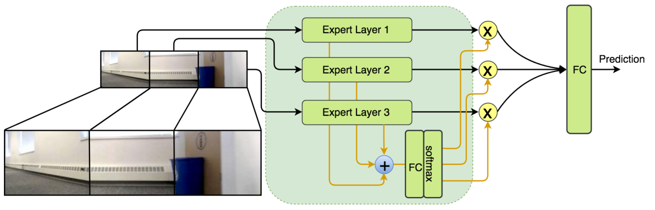
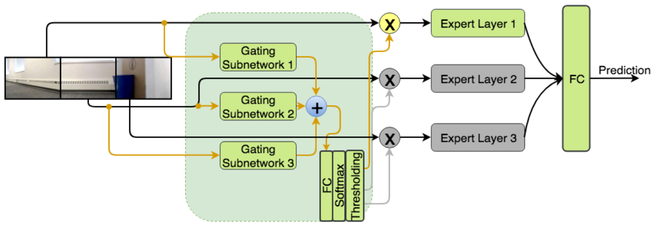


Fig. 2. The architecture of the network used in the first step of the proposed training procedure. The part of the network inside the green shaded box performs the task of the gating network. Thus, the gating network uses experts as feature extractors.



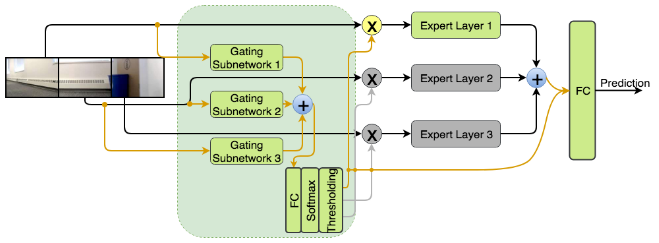


Fig. 3. The architecture of *"Reconf_Concat"* (**top**) and *"Reconf_Select"* (**botttom**). The part of the network inside the green shaded box performs the task of the gating network and is trained in the second step of the training process. In this case the gating network has its own feature extractors, which are significantly smaller then experts in the main part of the network. For each data example, only the experts selected by the gating network are fine-tuned (on the picture above only the first expert is activated since the outputs of the gating network for the other two experts are zero for the given input image and thus are colored in gray). **Top**: The concatenated feature vectors are fed to the fully connected layers of the main network. **Bottom**: The selected feature vector is fed to the fully connected layers of the main network together with the output of the gating network.

At the end of the training process we obtain the network, which has similar computational cost at inference to the network processing only a single input. The proposed network selects one most relevant sensor at any given moment. Extending to the case where a subset of sensors is used can be done in the following way. After the first step of training, the gating network outputs a vector with continuous weights corresponding to the experts. In order to keep k experts, in the second step of training the labels used for training the gating network should be modified appropriately, e.g. if one wants to use 2 out of 3 sensors and use weights for each input, one should only zero-out the smallest weight.

IV. Experiments

To validate our approach, we implemented the proposed reconfigurable network to steer the autonomous platform equipped with three cameras.

A. Hardware Overview

The block diagram of the our autonomous platform used for the experiments is shown in Fig. 4. We used a Traxxas X-Maxx remote control truck as a base for the autonomous platform and NVIDIA Jetson TX1 for computations. We installed three Logitech HD Pro C920 cameras on the platform. The center camera is oriented straight. The side cameras are mounted at angles to allow capturing front side views. Cameras capture non-overlapping views. A PCI Express (PCIe) USB 3.0 Card was connected to NVIDIA Jetson TX1 to provide the bandwidth for three cameras enabling them to operate at the same time. For controlling actuators of the autonomous platform we used Micro Maestro 6-Channel USB Servo Controller. We use the same platform for autonomous driving and training data collection.

In the training data collection mode we use the wireless gamepad to control the car. The corresponding steering commands and images captured by the cameras are saved on the local storage of the NVIDIA Jetson TX1.

In the autonomous testing mode, we run our network on NVIDIA Jetson TX1 in real time. The predicted commands are used to steer the platform while the speed is controlled by the operator.

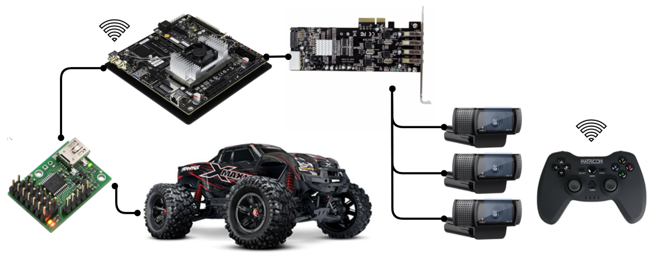


Fig. 4. Block diagram of the autonomous platform used for experiments.

B. Data Preprocessing

Before training the network using the collected data, we pre-process the data in order to improve training efficiency. We perform two preprocessing steps, data balancing and data augmentation. We describe these steps below. The data balancing and augmentation are used to achieve a good driving performance. This step is the same for the reconfigurable network and all other methods we experimented with. Data balancing is crucial to prevent the car from overfitting to driving straight and augmentation ensures the same number of turns to the left and right.

1) Data Balancing: First, we normalize the values of the captured steering commands to the range -1 to 1, where 0 means driving straight. Next, we organize all the training examples into 7 categories based on their steering command as shown in Table I. We collected 68245 scenes for training, 9896 scenes in category 1, 1671 in category 2, 2505 in category 3, 39926 in category 4, 4882 in category 5, 2383 in category 6, and 6982 in category 7. Therefore, collected data are heavily imbalanced in terms of captured steering commands. We address this problem with data balancing procedure. For every training epoch, we sample uniformly at random from each of these seven categories (70000 in total) in order to create our actual training data set.

2) Data augmentation: In order to increase the size of the training data set we perform data augmentation. We use horizontal flipping of the images. As we use three cameras mounted symmetrically on the autonomous platform, we also have to swap left and right camera images. In order to adjust the steering command accordingly we multiply it by -1 when we perform the flipping. We apply the augmentation with probability 0.5.

Finally, note that we use the same amount of data for our network as well as other methods, including the ones without the gating mechanism, thus introducing the gating mechanism does not require an increased amount of training data.

C. Experimental Results

In the experiments we use the network which takes three camera images on its input and outputs a steering command for the autonomous platform. We construct and train the network as described in Section III. Each expert has an architecture provided in Table II. The fully connected layers block consists of two fully connected layers, for both, the reconfigurable network and its component, the gating network.

We use separate recordings for training and testing. The training and test data were recorded in different driving environments, i.e. different parts of the building, thus the training data are not replicated in the test set. The training data consisted of 70000 scenes, as explained before, and the test data had 5738 scenes.

The results obtained for the reconfigurable network as given in Fig. 2 obtained after the first step of the training process are shown in Fig. 5. The results show that the network learned to correctly predict the steering command and also properly choose the most relevant input, i.e. a center camera when driving straight and side cameras in tight turns.

In the second step of the training procedure we trained multiple gating networks of various architectures given in Table III. We modify the training labels as described before, to predict single most relevant input. After the training we are adding hard thresholding on the gating network output to form a one-hot output vector. The comparison of the performance of the trained gating networks is shown in Fig. 6. The performance is measured with the classification accuracy. Based on our experiments we found that the

TABLE I

SEVEN CATEGORIES FOR DATA BALANCING

Category number	Steering Command
1	[-1,-0.67)
2	[-0.67,-0.33)
3	[-0.33,0)
4	0
5	(0,0.33]
6	(0.33,0.67]
7	(0.67,1]

TABLE II

EXPERT LAYER ARCHITECTURE.

layer name	output size	parameters
conv1	$16 \times 58 \times 78$	5×5, stride=2, BN, ReLU
conv2	$32 \times 27 \times 37$	5×5, stride=2, BN, ReLU
conv3	$64 \times 12 \times 17$	5×5, stride=2, BN, ReLU
conv4	$96 \times 4 \times 7$	5×5, stride=2, BN, ReLU
conv5	$128 \times 2 \times 5$	3×3, stride=1, BN, ReLU
conv7	$128 \times 1 \times 4$	2×2, stride=1, BN, ReLU
vectorize	512	

gating architecture that performs well can be obtained by significantly scaling down the input image, which is done by introducing large stride in the first layer of the gating network, and using shallow CNN architecture, which helps to minimize required computations for the gating network. For further experiments we use the best performing gating network which uses feature extractor with an architecture given in case 6 in Table III.

Next, we train two versions of the reconfigurable network described in the third step of the training procedure in Section III. The obtained results are shown in Fig. 7. The performance of both networks is very similar while the *Reconf_Select* has significantly smaller fully-connected layer.

For comparison, we also trained a single-input network, a three-input network without any gating mechanism, and a soft-attention mechanism (first three rows in Table IV). Both networks use experts that have the same architecture as other considered networks. The single-input network uses only the center camera. We compare the performance of all considered networks in terms of the test loss and the amount of computations needed for the forward pass through the network. The test loss is measured as the mean squared error between predicted and reference steering command and the amount of computation is measured in the number of floating-point operations (FLOPs). We summarize the results of the comparison in Table IV.

The results clearly show that both versions of the proposed reconfigurable network achieve the performance that is close to network with three inputs and they use almost the same amount of computations as the single-input network. The results also show that shallow gating network is sufficient for selecting a relevant input at any given moment. We also found that roughly the same number of errors come from choosing the wrong camera and insufficiency of one sensor in making a prediction.

Finally, we tested the second version of the reconfigurable network on our autonomous platform, where the network steers the car in the indoor environment. We recorded the videos captured by the three cameras as well as the gating network output. We used the recorded gating network output to highlight the images corresponding to the sensors that are activated by the gating mechanism at any given moment. The resulting video is attached as the Supplementary material. The example images captured by the three cameras during autonomous driving are shown in Fig. 8.

TABLE III

DIFFERENT ARCHITECTURES OF THE FEATURE EXTRACTOR USED IN THE GATING NETWORK. **FROM TOP TO BOTTOM**: CASES 1 - 9.

Case 1:

layer name	output size	parameters
conv1	$3 \times 60 \times 80$	1×1, stride=2, BN, ReLU
conv2	$16 \times 28 \times 38$	5×5, stride=2, BN, ReLU
conv3	$32 \times 12 \times 17$	5×5, stride=2, BN, ReLU
conv4	$48 \times 4 \times 7$	5×5, stride=2, BN, ReLU
conv5	$64 \times 2 \times 5$	3×3, stride=1, BN, ReLU
conv6	$96 \times 1 \times 4$	2×2, stride=1, BN, ReLU
vectorize	384	

Case 2:

layer name	output size	parameters
conv1	$3 \times 30 \times 40$	1×1, stride=4, BN, ReLU
conv2	$16 \times 13 \times 18$	5×5, stride=2, BN, ReLU
conv3	$32 \times 5 \times 7$	5×5, stride=2, BN, ReLU
conv4	$48 \times 3 \times 5$	3×3, stride=1, BN, ReLU
conv5	$64 \times 1 \times 3$	3×3, stride=1, BN, ReLU
vectorize	192	

Case 3:

layer name	output size	parameters
conv1	$3 \times 24 \times 32$	1×1, stride=5, BN, ReLU
conv2	$16 \times 10 \times 14$	5×5, stride=2, BN, ReLU
conv3	$32 \times 3 \times 5$	5×5, stride=2, BN, ReLU
conv4	$48 \times 1 \times 3$	3×3, stride=1, BN, ReLU
vectorize	144	

Case 4:

layer name	output size	parameters
conv1	$3 \times 15 \times 20$	1×1, stride=8, BN, ReLU, Maxpool
conv2	$16 \times 6 \times 9$	3×3, stride=1, BN, ReLU, Maxpool
conv3	$32 \times 2 \times 3$	3×3, stride=1, BN, ReLU, Maxpool
conv4	$48 \times 1 \times 2$	2×2, stride=1, BN, ReLU, Maxpool
vectorize	96	

Case 5:

layer name	output size	parameters
conv1	$3 \times 12 \times 16$	1×1, stride=10, BN, ReLU, Maxpool
conv2	$16 \times 5 \times 7$	3×3, stride=1, BN, ReLU, Maxpool
conv3	$32 \times 2 \times 3$	2×2, stride=1, BN, ReLU, Maxpool
conv4	$48 \times 1 \times 2$	2×2, stride=1, BN, ReLU, Maxpool
vectorize	96	

Case 6:

layer name	output size	parameters
conv1	$3 \times 12 \times 16$	1×1, stride=10, BN, ReLU
conv2	$16 \times 6 \times 8$	3×3, stride=1, padding=1 BN, ReLU, Maxpool
conv3	$32 \times 2 \times 3$	3×3, stride=2, BN, ReLU, Maxpool
vectorize	192	

Case 7:

layer name	output size	parameters
conv1	$3 \times 12 \times 16$	1×1, stride=10, BN, ReLU
conv2	$12 \times 6 \times 8$	3×3, stride=1, padding=1 BN, ReLU, Maxpool
conv3	$24 \times 2 \times 3$	3×3, stride=2, BN, ReLU, Maxpool
vectorize	144	

Case 8:

layer name	output size	parameters
conv1	$3 \times 12 \times 16$	1×1, stride=10, BN, ReLU
conv2	$8 \times 6 \times 8$	3×3, stride=1, padding=1 BN, ReLU, Maxpool
conv3	$16 \times 2 \times 3$	3×3, stride=2, BN, ReLU, Maxpool
vectorize	96	

Case 9:

layer name	output size	parameters
conv1	$3 \times 6 \times 8$	1×1, stride=20, BN, ReLU, Maxpool
conv2	$16 \times 2 \times 3$	3×3, stride=1, BN, ReLU, Maxpool
conv3	$32 \times 1 \times 2$	2×3, stride=1, BN, ReLU, Maxpool
vectorize	64	

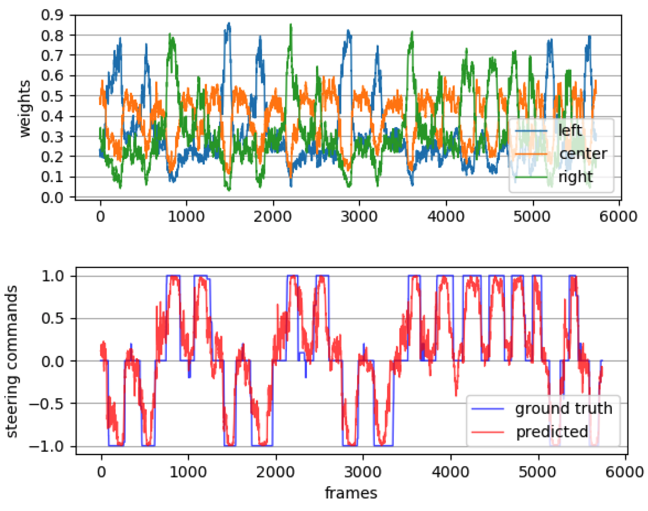


Fig. 5. **Top:** The output of the gating component of the reconfigurable network indicating the relevance of the inputs. **Bottom:** The comparisons between the actual steering command and predicted steering angles produced by network after the first step of the training procedure.

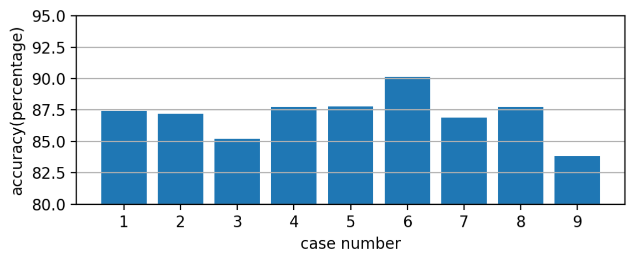


Fig. 6. Performance comparisons of 9 various gating network architectures.

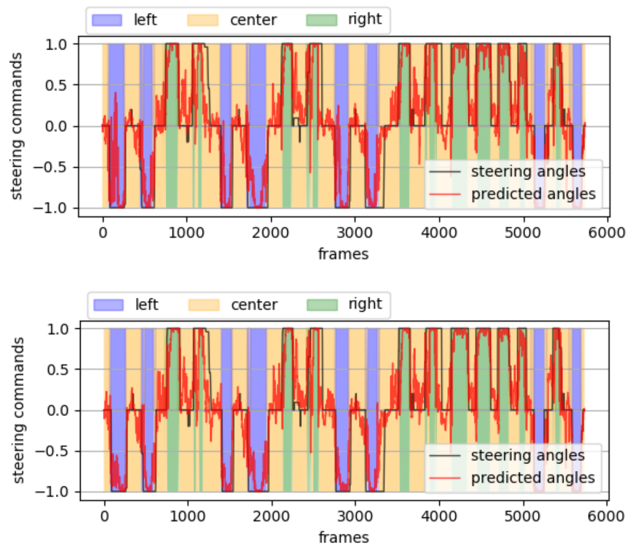


Fig. 7. The comparisons between the actual steering command and predicted steering angles produced by the first (**top**) and second (**bottom**) version of reconfigurable network after the third step of the training procedure. The output of the gating network is marked with shaded areas in different colors for each selected input.

TABLE IV

COMPARISON OF DIFFERENT NETWORKS IN TERMS OF THE TEST LOSS AND REQUIRED AMOUNT OF COMPUTATIONS.

Model	Test loss	FLOPs
Network using only center camera	0.20	36.36M
No gating mechanism; images from all three cameras are used	0.09	109.08M
Network with gating mechanism and no thresholding (soft attention mechanism)	0.09	109.15M
Reconf_Concat	0.12	36.54M
Reconf_Select	0.11	36.41M

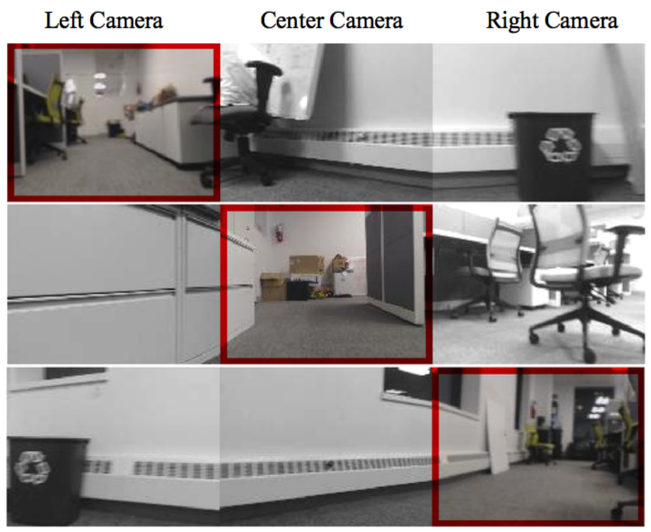


Fig. 8. The exemplary images captured by the three cameras during autonomous driving when turning left (**top row**), driving straight (**middle row**), and turning right (**bottom row**). The color images in red frames are the inputs selected by the gating network.

V. CONCLUSION

In this work, we propose the reconfigurable network for handling multiple sensors on autonomous platforms. We discuss its architecture and the training procedure. We show that the proposed reconfigurable network effectively selects the most relevant input at any given moment and by far (achieving almost two times smaller test loss) outperforms single-sensor network, while using the same amount of computations. We also demonstrate that the compact gating network is sufficient for selecting the relevant input at any given moment. Thus, the proposed concept can be easily scaled up to large number of sensors while keeping computations at a reasonable level, especially in applications were only a subset of sensors are expected to suffice to perform the learning task at any given time.

ACKNOWLEDGMENT

We would like to thank Dr Mariusz Bojarski from NVIDIA for hardware assistance and helpful discussions.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in NIPS, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” arXiv:1606.00915, 2016.

[4] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, “Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition,” in ICASSP, 2012.

[5] J. Weston, S. Chopra, and K. Adams, “#tagspace: Semantic embeddings from hashtags,” in EMNLP, 2014.

[6] 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:1504.01716, 2015.

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” CoRR, vol. abs/1604.07316, 2016.

[8] C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving.” in ICCV, 2015.

[9] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” in NIPS, 1989.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, “Off-road obstacle avoidance through end-to-end learning,” in NIPS, 2006.

[12] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, “Explaining how a deep neural network trained with end-to-end learning steers a car,” arXiv:1704.07911, 2017.

[13] N. Patel, A. Choromanska, P. Krishnamurthy, and F. Khorrami, “Sensor modality fusion with cnns for ugv autonomous driving in indoor environments,” in IROS, 2017.

[14] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” in CVPR, 2017.

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” Neural computation, vol. 3, no. 1, pp. 79–87, 1991.

[16] M. I. Jordan and R. A. Jacobs, “Hierarchical mixtures of experts and the em algorithm,” Neural computation, vol. 6, no. 2, pp. 181–214, 1994.

[17] R. Collobert, S. Bengio, and Y. Bengio, “A parallel mixture of svms for very large scale problems,” in NIPS, 2002.

[18] D. Eigen, M. Ranzato, and I. Sutskever, “Learning factored representations in a deep mixture of experts,” arXiv preprint arXiv:1312.4314, 2013.

[19] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, “Conditional computation in neural networks for faster models,” arXiv:1511.06297, 2015.

[20] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, “Adaptive neural networks for efficient inference,” in ICML, 2017.

[21] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” arXiv:1701.06538, 2017.

[22] J. Schlosser, C. K. Chow, and Z. Kira, “Fusing lidar and images for pedestrian detection using convolutional neural networks,” in ICRA, 2016.

[23] O. Mees, A. Eitel, and W. Burgard, “Choosing smartly: Adaptive multimodal fusion for object detection in changing environments,” in IROS, 2016.