Shared Multi-Task Imitation Learning for Indoor Self-Navigation

Junhong Xu, Qiwei Liu, Hanqing Guo, Aaron Kageza, Saeed AlQarni, Shaoen Wu
Computer Science
Ball State University
Muncie, IN USA
{jxu7, qliu4, hguo, agkageza, saalqarni, swu}@bsu.edu

***Abstract*—Deep imitation learning enables robots to learn from expert demonstrations to perform tasks such as lane following or obstacle avoidance. However, in the traditional imitation learning framework, one model only learns one task, and thus it lacks of the capability to support a robot to perform various different navigation tasks with one model in indoor environments. This paper proposes a new framework, *Shared Multi-headed Imitation Learning*(*SMIL*), that allows a robot to perform *multiple* tasks with *one* model without switching among different models. We model each task as a sub-policy and design a multi-headed policy to learn the shared information among related tasks by summing up activations from all sub-policies. Compared to single or non-shared multi-headed policies, this framework is able to leverage correlated information among tasks to increase performance. We have implemented this framework using a robot based on NVIDIA TX2 and performed extensive experiments in indoor environments with different baseline solutions. The results demonstrate that *SMIL* has doubled the performance over non-shared multi-headed policy.**

I. Introduction

One of the main challenges in robotics is to enable robots to interact with a dynamically changing environment and to perform different tasks with minimal prior knowledge. It requires the robot to perceive the environment, understand the context of the tasks, and make decisions accordingly. Traditional methods rely on accurate manual modeling for each task, such as visual SLAM systems for navigation [17]. In contrast, deep learning based methods simplify the need of manual modeling. They directly learn a policy that maps a sensor input to a corresponding control command. Because of the advance of deep learning models, especially convolutional neural networks (CNNs), learning control commands directly from environmental images becomes feasible [13].

However, literature imitation learning solutions can only learn one task per model. This prevents robots from executing complex actions. For example, if the robot is asked to fetch a cup in the kitchen, it needs to decompose the task into a few sub-tasks such as *goto*, *traverse*, and *fetch*. In this work, we propose a framework that solves the two navigation problems with only one model, *goto* and *traverse*. The framework is called ***S**hared **M**ulti-task **I**mitation **L**earning (SMIL)*, which gives robots the ability to perform different navigation tasks in indoor environments. The framework is based on the multi-task learning (MTL) [3] and aims at solving task agnostic problem of imitation learning. Although MTL has been researched for a long time, multi-task imitation learning has rarely been researched until recently [5], [8].

The proposed framework uses a shared CNN to learn an environment model that extracts environmental features. Different sub-task policies are represented by a multi-headed fully connected network whose inputs are from the last layer of the shared CNN. While current literature solutions such as the work [4] do not consider the relevance between the sub-tasks, our proposed framework rather makes use of the relevant information among sub-tasks. In order to solve poor generalization and distribution mismatch problems, we apply off-policy imitation learning [12], data augmentation, and dropout [21] during training. During testing, the framework switches between sub-policies based on human navigation commands.

Our contributions are as below:

- We propose a new network architecture that leverages task relationships by summing up activations from sub-policies.
- Off-policy learning procedure is used to train our framework.
- Dropout and image augmentation are used to improve generalization.

In the rest of this paper, Section II reviews the related work of the imitation learning and multi-task learning. Our solution *SMIL* framework is described in Section III including the network architecture and detailed training procedure. Next, Section IV presents extensive performance evaluations of *SMIL* in real indoor environments. The conclusion and future work are presented in Section V.

II. Related Work

Learning based algorithms have been applied to a variety of robotics control problems. These algorithms can learn an end-to-end controller directly from data. For example, in [24], reinforcement learning is used to train a siamese neural network to navigate to a target position. Similarly, the work [16] uses auxiliary losses to train a reinforcement learning agent to navigate through complex maze environments. In contrast to reinforcement learning, imitation learning has been applied to many real-world applications such as robotic grasping [11], UAV flight control [7], self-driving cars [2], and rope manipulation [18].

The above mentioned algorithms only consider completing one task at a time, but this is not enough for many robotic tasks. Therefore, we target on the idea of multi-task learning (MTL) [19]. Researchers have proposed a learning architecture that uses one single model to jointly learn image classification, speech recognition, and translation problems and yielded encouraging results [10]. Long et. al. place a matrix prior to fully connected layers in a CNN to learn the relations of multiple tasks [15]. Multi-task imitation learning has drawn attentions in recent years including learning multiple tasks together [4], [8] or one-shot learning [5], [6], [20]. Among these works, two works [4], [8] are most similar to ours. Hausman et. al. propose a multi-model imitation learning framework that separates video segments into different skill trajectories and imitate the demonstrated skills jointly [8]. In [4], the authors propose a framework that learns sub-policies using a multi-headed network in the autonomous driving setting. However, these works have not considered the relevant information among tasks. Our proposed framework learns the relationships across tasks by combining learned features across sub-policies. By learning these relationships, the model is able to yield a more general representation of various navigation tasks.

III. Shared Multi-task Imitation Learning

In this section, we first formally define the problem of imitation learning and multi-task imitation learning. Next, we present our deep learning network architecture. Finally, our training procedure is described in detail.

A. Problem Formulation

To formally define the problem, let $\mathcal{X}$ and $\mathcal{Y}$ denote recorded observations and corresponding expert control commands respectively. A pair of ($\mathcal{X}^k$, $\mathcal{Y}^k$) is defined as the k-th *demonstration* of a robot. To simplify the notation, we assume the environment is Markovian; namely the current observation includes all history information of the environment.

1) Traditional Imitation Learning: In the traditional imitation learning setting, the demonstrations are associated with a single task. Thus, $\mathcal{X} = \{\mathcal{X}^1, \mathcal{X}^2, ..., \mathcal{X}^n\}$ and $\mathcal{Y} = \{\mathcal{Y}^1, \mathcal{Y}^2, ..., \mathcal{Y}^n\}$ represent n demonstrations of the task. The imitation learning aims to learn a policy that maps an observation to a probability distribution of control command $\pi : \mathcal{X} \rightarrow \pi(\mathcal{Y}; \theta)$, where θ denotes the parameters of a weight vector, e.g. neural network. The parameters θ can be found by solving a maximum-likelihood estimation (MLE) problem: $\theta^* = arg\,\underset{\theta}{max} \sum_{n=1}^{N} \pi(\mathcal{Y}^n|\mathcal{X}^n; \theta)$. If the policy π is Gaussian and it is parameterized by a weight vector θ, the MLE objective can be transformed into a $l2$ regression problem: $\theta^* = arg\,\underset{\theta}{min} \frac{1}{N} \sum_{n=1}^{N} ||\pi(\mathcal{X}^n; \theta) - \mathcal{Y}^n||_2^2$. In our system, we assume the policy is Gaussian thus we use the $l2$ objective to optimize the parameters θ.

2) Multi-Task Imitation Learning: In the multi-task imitation learning setting, observations $\mathcal{X}$ and the corresponding control commands $\mathcal{Y}$ are representing more than a single task in demonstrations. Instead, they consist of multiple demonstrated tasks and are denoted as $\mathcal{X} = \{X_1, X_2, ...X_i\}$ and $\mathcal{Y} = \{Y_1, Y_2, ..., Y_i\}$, where $X_i = \{X_i^1, X_i^2, ..., X_i^n\}$ and $Y_i = \{Y_i^1, Y_i^2, ..., Y_i^n\}$ represent the observations and control commands of task i. It should be noted that although here we use the same number of demonstrations across tasks to simplify the notations, it is not required to be the same. In addition, we introduce the concept of task embedding denoted as $T = \{T_1, T_2, ...T_i\}$. Similar to word embedding [14], the task embedding is a feature vector for each task that embeds the high-level representations into a low-dimensional space.Therefore, the policy π maps observations $\mathcal{X}$ and task embedding T to a distribution of control command $\mathcal{Y}$ and is formulated as $\pi : \{\mathcal{X}, T\} \rightarrow \pi(\mathcal{Y}; \theta)$. Instead of minimizing the objective for one task, multi-task imitation learning aims to find a set of parameters that are to be optimized over multiple tasks:

$$\theta^* = arg\,\underset{\theta}{min} \frac{1}{I} \frac{1}{N} \sum_{i=1}^{I} \sum_{n=1}^{N} ||\pi(\mathcal{X}_i^n, T_i; \theta) - \mathcal{Y}_i^n||_2^2, \quad (1)$$

where I is the number of tasks.

B. Network Architecture

With the multi-task imitation learning problem formulated as above, we have designed a ***S**hared **M**ulti-task **I**mitation **L**earning* (SMIL) framework that learns to perform four tasks based on human commands. This framework is shown in Fig. 1, which consists of two modules: image feature extractor and shared multi-headed policy, which will be explained in details in the following.

1) Image Feature Extractor: The image feature extractor is a fine-tuned and pre-trained ResNet-18 by excluding the classifier layer in the original ResNet-18, but preserving the average pooling layer. It is used to project raw image inputs to a low-dimensional feature space and is denoted as $f = I(x, \theta_I)$, where $f \in \mathbb{R}^{1\times1\times512}$ is the extracted feature vector, x is the input image, and θ_I is the parameter set of ResNet-18. The idea of using pre-trained model has been extensively researched by the literature and it is proven to have faster convergence than training from scratch [9]. The feature vector f is then flattened and passed to the shared multi-headed policy to generate control commands. In addition, to learn a more general representation of indoor environments, image feature extractor also predicts which indoor environment the robot is currently in: $p = E(f)$, where E represents a linear classifier. In our case, it predicts two class labels: hallway and classroom. Environment prediction is jointly trained with control commands.

2) Shared Multi-headed Policy: The shared multi-headed policy learns shared knowledge across tasks as well as task specific knowledge. As shown in Figure , the shared multi-headed policy consists of three parts: switch operation, addition operation, and sub-policies represented by fully connected layers. In our case, we define four tasks to be learned: *traverse hallway, traverse classroom, to hallway,* and *to classroom*. We denote the shared multi-headed policy as $a = \pi(f, T; \theta_\pi)$, where $\theta_\pi = \{\mathcal{W}^1, ..., \mathcal{W}^4\}$ is the parameter of the entire policy and $\mathcal{W}^i$ is the parameter set of the ith sub-policy. It

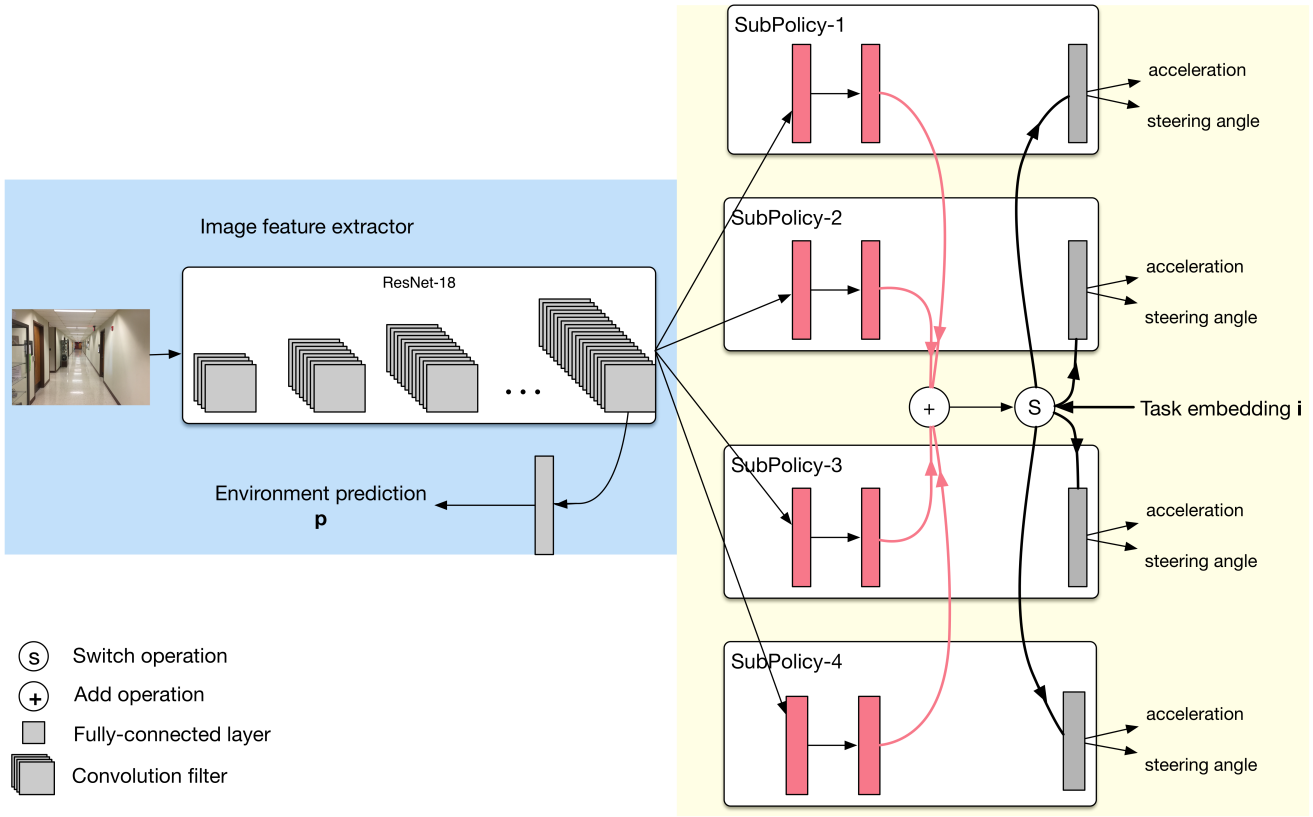


Fig. 1: An illustration of SMIL framework. An image observation is first passed through a ResNet-18 to extract features. The features are then passed through a linear classifier as well as a shared multi-headed policy to predict environment labels and control commands.

takes two inputs, extracted features f from the image feature extractor module and task embedding T, and gives the output of the control commands a corresponding to a specific task. In our case, since we have four tasks, we denote the task embedding T as a 4-dimensional one-hot vector [4]. It is used to determine the sub-policy to be activated. Note that T can also be learned in an unsupervised way [8]. The action space is two-dimensional: acceleration and steering angle.

Each sub-policy is a three-layer fully connected neural network. Because the tasks are highly correlated, it is useful to learn the task relationship through which the activated sub-policy can exploit useful information from other sub-policies. For example, the hallway navigation sub-policy can leverage obstacle avoidance knowledge learned by the classroom navigation sub-policy because classroom is a more complicated environment with different types of obstacles. Formally, denote $\mathcal{W}_l^i$ the set of parameters at layer l in i-th sub-policy and $h_l^i = g_l(\mathcal{W}_l^{i^T} x_l^i)$ the corresponding output, where function $g_l(\cdot)$ denotes a non-linear function and x_l^i is the input of that layer. Our network uses ReLU for the first two layers' non-linear functions and an identity function for the last layer. The input to first layers of the sub-policy networks is the extracted features f from the image feature extractor module. The information across sub-policies is shared by using an addition operation that combines all the outputs from the second layers of each sub-policy network:

$$j = \sum_{i=1}^{L} h_2^i, \tag{2}$$

where L is the total number of tasks. Although the literature has shown that higher layers learn more task specific features that are difficult to transfer [23] , we choose to share the information from the second layers, because the tasks are highly correlated and the learned features are easier to transfer over sub-policies. After the addition operation, the task embedding T selects a sub-policy to use via a switch operation. The switch operation routes the output from the addition operation to the final layer of the selected sub-policy. The final output will be:

$$a = h_3^t = g_3^t(W_3^t j). \tag{3}$$

This design enables each sub-policy to learn the task specific controls in the final layer as well as to share knowledge through the addition operation across different sub-policies.

C. Training Procedure

It is important to train the multi-task imitation learning framework for high performance. We employ three different training techniques to train a robust SMIL framework: dropout, data augmentation, and noise injection. Because *SMIL* predicts steering angle and acceleration, which are real-valued numbers, we use mean squared error (MSE) as the loss function during the training.

1) Dropout and Data Augmentation: As opposed to [22] that is designed for single task frameworks, our goal is to train a robust SMIL framework that is able to perform in different environments from real-world experience, instead of learning from images generated by a hand-engineered simulator. It is necessary to collect images from diverse indoor environments to prevent a deep learning model from overfitting, but collecting data is time consuming. Hence, we use data augmentation and dropout [21] to train a robust model. For data augmentation, we randomly apply contrast change, Gaussian noise, pixel dropout, random cropping, and horizontal flip (steering angle is also flipped). Different augmentation strategies are shown in Fig. 2. Dropout is used to prevent model overfitting by randomly zeroing out an neuron's activation. In addition, dropout can also stabilize the performance of the robot. Because of the addition operation, the norm of activation input to the final layer is possible to be very large and yields the outputs of very different control commands. Dropout is only added to the first and second layers of sub-policies.

Fig. 2: An illustration of randomly chosen augmentation strategies. From left to right are original, random cropping and pixel dropout, additive Gaussian noise and contrast changing, and more aggressive random cropping.

2) Noise Injection: Distribution mismatch between the supervised training and the robot learning is an essential problem in imitation learning: because the human operator is proficient in demonstrating tasks, there are no demonstrations of recovering from dangerous or erroneous states. As a result, the robot does not know how to correct itself in experiencing these abnormal states. We explore the off-policy training for the SMIL framework to address this issue, where the robot learns from another policy. We employ the noise injection approach [12], which injects an optimized Gaussian noise to an expert policy to maximize the probability of a human demonstrator making the same mistakes as the robot.

Given a robot policy π_θ, an expert policy π^*, DART algorithm [12] aims to find a covariance matrix of Gaussian noise that maximizes the probability of expert taking the robot policy:

$$\Sigma^* = \underset{\Sigma}{arg\,min}\ E_{p(\xi|\pi^*,\Sigma)} - \sum_{t=0}^{T-1} log[\pi * (\pi_\theta|x_t, \Sigma)] \quad (4)$$

where ξ means trajectories encountered by executing the expert policy with noise injected and Σ is the covariance matrix. A shrinkage estimation is then utilized to scale the covariance matrix and derive a closed form solution:

$$\Sigma^\alpha = \frac{\alpha}{Ttr(\Sigma^*)}\Sigma^*;$$
$$\Sigma^* = \frac{1}{T}E_{p(\xi|\pi^*,\Sigma)} \sum_{t=0}^{T-1} (\pi_\theta(x_t) - \pi^*(x_t))(\pi_\theta(x_t) - \pi^*(x_t))^T \quad (5)$$

where α is the prior knowledge of the final error of the robot policy on the training dataset. This algorithm is best used in an iterative approach, so we collect expert demonstrations for k iterations and update the covariance matrix at the start of every iteration except for the first iteration with the covariance matrix as 0. In our case, we found $\alpha = 2$ gives the best results, which is reasonable because we normalize value of steering angle and acceleration between -1 and 1, and the largest MSE should not exceed 4.

IV. Performance Evaluation

We have implemented our framework and elevated in real world environments. Our experiments have been designed to answer the following questions:

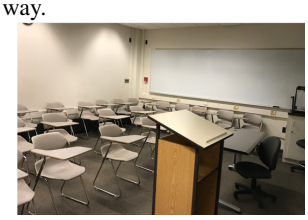
1) Is the shared task representation (the addition operation) necessary to the multi-task imitation learning when the tasks are highly correlated?
2) What is the performance difference of the multi-headed sub-policy framework compared to a single-headed policy?
3) Does the environment prediction task improve the performance?
4) Are data augmentation and dropout important to the model generalization and robustness?

A. Testbed, Experiment Environment

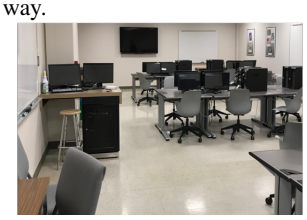
Our framework is implemented into an iRobot Create2 robot. The linear speed of this robot is in the range of -0.5 m/s to 0.5 m/s and the angular velocity is in the range of -4.5 rad/s to 4.5 rad/s, where a negative linear speed represents moving backward and a negative angular speed represents going right. The only sensory data we used are RGB images from a ZED Stereo camera. The valid depth estimation is between 0.5m and 20m. We use a NVIDIA Jetson TX2 as the main computation resource to do inference.

(a) Robert Bell first floor hallway.

(b) Robert Bell third floor hallway.



(c) Robert Bell first floor classroom.



(d) Robert Bell third floor classroom.

Fig. 3: Training environments (b) and (d) have different geometric and color appearances as testing environments (a) and (c).

We have extensively evaluated our solution in a real environment: the Robert Bell Hall building at Ball State University. To test the generalization, we have trained the robot in the third floor but tested the robot on the first floor of Robert Bell building. The geometric and color appearances are very different in these two environments. The testing and training scenes are presented in Fig. 3.

B. Task Description

We have evaluated our framework on four correlated tasks that are described in Table I. Since classroom indoor environments contain fewer free space and are far more complex compared to hallway environments, we set less constraints on classroom related task, i.e. traverse classroom and to hallway. The goal of these four tasks is to simulate the robot in a multi-task decision making environment, where it is required to go to different indoor locations based on human command.

C. Experiment Configurations

A variety of baselines have been designed to evaluate the performance. We compare the $SMIL$ full architecture with five baselines:

- **Multi-headed network**: this model uses the same architecture without the shared learning representation (with the addition operation excluded).
- **Plain network**: this is the traditional imitation learning approach, where there is one single network that maps the input to output control commands. It is not aware of the multi-task setting.
- $SMIL$ **w/o data augmentation**: this baseline excludes data augmentation in training.
- $SMIL$ **w/o dropout**: this baseline removes dropout at the first and second layers of sub-policies.
- $SMIL$ **w/o environment prediction**: this model does not predict environment class labels.

We use the same hyper-parameters to train each baseline network across all the tasks. We use stochastic gradient descent (SGD) optimizer with a learning rate of 0.01 and a decay factor of 10 for every 5 epochs. We train each network for 30 epochs with a batch size of 256. For the $SMIL$ framework and multi-headed network, we split training examples of each task evenly in each batch. We train the plain network on all the training data. The plain network is task-agnostic, so we do not inform it which task to perform explicitly. We set the weight decay to 0.0005. For the networks that use dropout, we set the dropout rate as 0.2 implying that with the probability of 20%, a neuron's activation will be set to 0.

We have conducted 10 experiments for each task and recorded the success rate and the averaged time duration with standard deviation for each experiment as shown in Table II and Fig. 4.

D. Overall Comparisons

As we can observe from Table II and Fig. 4, in terms of the success rate, it is clear that our model, $SMIL$, achieves the best performance across all the tasks. In terms of time duration, our model is able to maintain the longest averaged travels among all other models in traverse tasks. It needs slightly more time to complete to classroom task. This is because some baseline models complete all easy runs that take less time, e.g. the classrooms that are in the robot's field of view, but failed difficult runs that take longer time.

E. Comparisons on Model Architectures

The comparisons address the first three questions at the beginning of this section. We can draw three conclusions: **First**, the first floor has more obstacles than the third floor, which is used for training. Thus, it is necessary to reuse the knowledge learned from traverse classroom task while performing traverse hallway task. By comparing $SMIL$ and multi-headed network on traverse hallway task, we observe that $SMIL$ is able to reuse obstacle avoidance knowledge from traverse classroom sub-policy. **Second**, from the result, we observe that it is necessary to add the environment prediction auxiliary task to provide additional training single to the image feature extractor, which allows the image feature extractor to learn a more robust representation. **Third**, although the plain network is trained on all the data that contains 80,000 images, it still fails tremendously especially on the tasks of *to classroom/hallway*. This is because the mode averaging [1] and inaccurate labeling cause bias to the network. The tasks of *to classroom/hallway* inherit a different mode from that in the *traverse classroom/hallway* tasks and they are more difficult because the robot needs to avoid obstacles and find the targeted place. Since the plain network is trained on all tasks, it is possible that the plain network ignores differences across each task and thus yield a poor policy. In addition, it is task-agnostic, so it can not respond to human command.

TABLE I: Task Description

Task	Task Description	Time Limit	Failure Condition
Traverse Hallway	The robot is initialized in hallway at a fixed position. It is asked to traverse hallway without collision within the time limit.	1 min	If the robot collides into obstacles, we count it as a failure. If it goes into classroom, we count it as a failure.
Traverse classroom	The robot is initialized in classroom at the door position. It is asked to traverse classroom without collision within the time limit	30 sec	Because classroom is highly complex, we give the robot one more chance in this task. If the robot collides into obstacles, we reroute it back to free space. If the robot collides again, we count it as a failure. In addition, if it goes outside of the classroom, we count it as a failure.
To classroom	The robot is initialized in hallway at a fixed position that is 15 meters away from a classroom. It needs to go from the classroom to the hallway.	2min	If the robot can not complete the task within time limit or collide into obstacles, we count it as a failure. In addition, if the robot passes two nearest classrooms from its initial position, it is a failure.
To hallway	The robot is initialized in a classroom at the furthest corner away from the door. It needs to go through the classroom to the hallway.	1 min	If the robot can not complete the task within time limit or collide into obstacles, we count it as a failure. Same as traverse classroom, we give the robot a second chance.

TABLE II: Success Rate

	Traverse Hallway	Traverse Classroom	To Classroom	To Hallway
SMIL	**80%**	**60%**	**70%**	**60%**
Multi-headed	50%	30%	50%	40%
Plain	50%	30%	20%	0%
$SMIL$ w/o augmentation	40%	30%	40%	30%
$SMIL$ w/o dropout	30%	30%	50%	40%
$SMIL$ w/o environment	70%	50%	50%	50%

F. Comparison on generalization and robustness

This comparison answers the fourth question: both dropout and data augmentation are necessary to train a robust model. By removing any of them, the robot has the similar performance. In terms of the success rate, after adding both methods, the robot's performance is almost doubled. It shows that these two methods are complementary to each other. Dropout prevents the robot from aggressive turning caused by large activations from the addition operation. By augmenting the training dataset, the trained model learns to ignore geometric difference and different lighting effects.

V. Conclusion

In this paper, we propose a deep multi-task shared imitation learning framework, *SMIL*, that can learn to work on multiple tasks with multiple sub-policies by learning the relations shared among these policies/tasks. Compared to the plain neural network, this framework allows the robot to follow human instructions. In addition, by leveraging the task relations, this framework is highly robust to new environments and produces the best results over all baselines. We have evaluated our framework in a real environment that is different from the training environment. The results show its robustness and great generalization to new environments.

References

[1] C. M. Bishop. Mixture density networks. 1994.

[2] 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.

[3] R. Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[4] 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.

[5] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[6] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

[7] 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*, 1(2):661–667, 2016.

[8] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using gen-

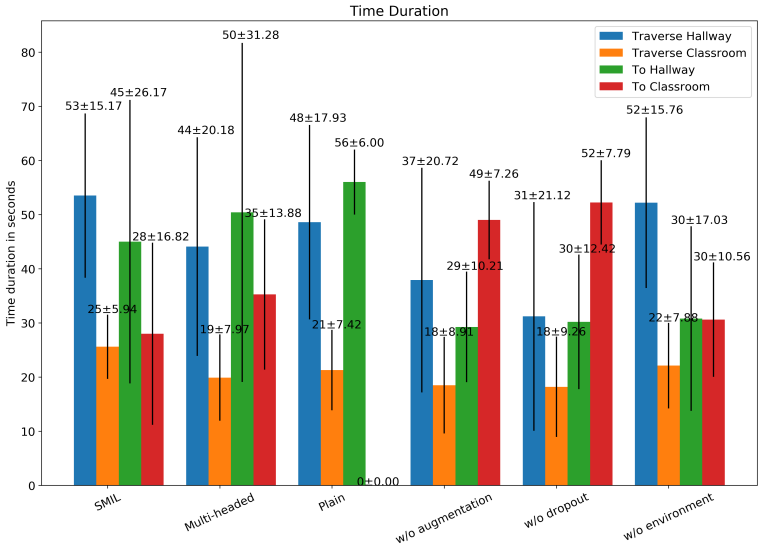


Fig. 4: Averaged time duration and standard deviation for each task. For traverse hallway/classroom task, The higher the averaged time duration means the the robot gets less collision. For to hallway/classroom tasks, the lower the averaged time duration means the faster the robot completes the task.

erative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 1235–1245, 2017.

[9] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[10] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[11] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg. Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations. In *IEEE International Conference on Automation Science and Engineering*, pages 827–834, 2016.

[12] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, pages 143–156, 2017.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[14] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[15] M. Long and J. Wang. Learning multiple tasks with deep relationship networks. *arXiv preprint arXiv:1506.02117*, 2015.

[16] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[18] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2146–2153. IEEE, 2017.

[19] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[20] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[22] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[24] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. 2016.