2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Vacant Parking Space Detection based on Task Consistency and Reinforcement Learning

Manh-Hung Nguyen
Faculty of Electrical Electronic Engineering, HCM-UTE
hungnm@hcm.ute.edu.vn

Tzu-Yin Chao
Department of Computer Science
National Chiao Tung University
chaoziyin@gmail.com

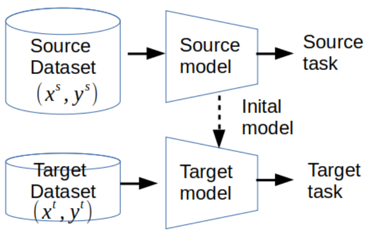
Ching-Chun Huang
Department of Computer Science
National Chiao Tung University
chingchun@cs.nctu.edu.tw

***Abstract*—In this paper, we proposed a novel task-consistency learning method that allows training a vacant space detection network (target task) based on the logic consistency with the semantic outcomes from a flow-based motion behavior classifier (source task) in a parking lot. By well designing the reward mechanism upon semantic consistency, we show the possibility to train the target network in a reinforcement learning setting. Compared with conventional supervised detection methods, this work's main contribution is to learn a vacant space detector via semantic consistency rather than supervised labels. The dynamic learning property may make the proposed detector been deployed and updated in different lots easily without heavy human loads. The experiments show that based on the task consistency rewards from the motion behavior classifier, the vacant space detector can be trained successfully.**

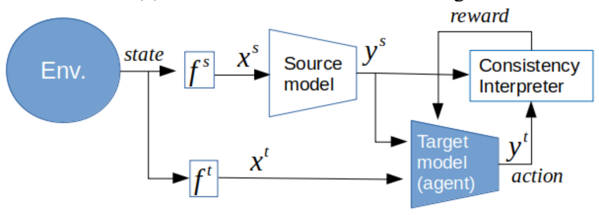
I. Introduction

Vacant space detection [1], [2] is one of the fundamental functions for a modern vision-based parking lot management system. However, training a universal vacant-space detector suitable for different lots is still a challenging task. For instance, in [1], the authors collect a huge amount of labeled samples to train a state-of-the-art network for vacant space detection. Although the method can achieve 99.74% detection accuracy, the trained model cannot be easily applied in the other lots without massive human efforts due to the variation of parking-lot layouts, mutual occlusion patterns, and camera poses. In general, the cost of model customization for each parking lot and heavy data labeling would prevent the supervised method to be commercialized.

To relieve the problem mentioned above, transfer learning [3] via a pre-trained model is a popular solution. It has been shown that a deep network trained on a labeled large-scale dataset can extract generalized semantic features for other relevant tasks. If we use the pre-trained model for feature extraction, a new target network can be trained by using fewer labeled samples for model refinement. However, to ensure the success of transfer learning, two critical assumptions must be satisfied. First, the target task and the source task are required to be highly relevant. The related work [4] has pointed out that not any task can successfully leverage transfer learning to inherit knowledge. Usually, a similar source domain is better. In addition, it still requires enough new labeled samples to fine-tune the target network, given a powerful pre-trained model. Hence, it raises a research topic to learn from the source task without transfer learning.



(a) Conventional transfer learning



(b) Task-consistency learning via reinforcement learning (RL).

Fig. 1: A comparison between "transfer learning" and "task-consistency learning". (x^s, x^t) are the source and target inputs. (y^s, y^t) are the outputs of the source and target tasks. In our RL-based task-consistency learning, (y^s, x^t) is treated as the environment state (observation); and (y^t) is regarded as the decision/action determined by the target agent. If the target decision is consistent with the source state y^s, the reward for the decision should be higher .

For a parking lot surveillance system, multiple vision tasks should be implemented to monitor the same zone to provide a comprehensive service. These tasks may run independently, but the inference outcomes should be consistent to follow some high-level semantic logic. In our application, we aim to build a vacant space classifier (target task) to efficiently identify free spaces. On the other hand, a motion pattern classifier (source task) was implemented to analyze the vehicle motion behavior in a parking space. Here, we ask the question if we can train a vacant space classifier given the inference output of the motion pattern classifier. If the answer is positive, we may have two compelling application benefits. First, the vacant space classifier can be dynamically updated without extra labels. Second, the self-learning property enables us to adapt the vacant space classifier for diverse parking lots with less human labor.

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9412152

To answer the question, we proposed a task-consistency learning (TCL) method that allows training a target network for vacant space detection based on the logic consistency with the semantic outcomes from the (source) motion pattern classifier. From the motion pattern classifier, we may know the motion status of a parking space, such as the “car moving in” state, “car moving out” state, and “no motion” state. If the target and source tasks are synchronized, the parking status and the motion status of the same space should coincide. The observation raises a “learning from multi-task consistency” idea that we may dynamically train the target network through satisfying the outcome consistency given a well-trained source-task model.

We may use Fig. 1 to explain the difference between ”transfer learning” and ”task consistency learning.” For transfer learning, shown in Fig. 1(a), a source model is used to initialize the target model; next, the target model is fine-tuned based on the target-task dataset. Instead, the ”task-consistency learning” uses the inference outcome from the source-task as supervision; the target task is learned based on the consistency with the source decision. Compared with transfer learning, our ”task-consistency learning” framework brings two differences. First, the target task is trained conditionally on the source decision under high-level logic; therefore, the learning framework does not require the source and target tasks coming from similar or relevant domains. This mechanism allows all semantic information extracted from different tasks, but the same environment can be utilized for training. Second, the framework enables training a task network indirectly via logistic consistency with other tasks; extra labeled samples may not be necessary.

Multi-task consistency does not directly provide target labels. The conventional supervised way thus is not suitable for consistency learning. To train the target network, we proposed to maximize the consistency between tasks in a reinforcement learning [5] style. As shown in Fig. 1(b), we convert task consistency as a reward and treat the target model as a policy decision agent. However, not all the task consistency observations are equally important. Some observations may even deliver the wrong messages. Moreover, the outcomes from the source network may be wrong due to false inference. It would directly lead to corrupted rewards. Finally, according to [6], reinforcement learning may become hard to converge if the reward signals are sparse. In this work, though truly challenging, we attempt to face these fundamental technique issues to make task-consistency learning possible from corrupted and sparse rewards. To summarize, the contributions of this work are as follows:

- We introduce a learning method that uses task consistency to train a vacant-space detection network via reinforcement learning. The reward mechanism, policy decision network, and training method have been discussed to learn from corrupt and sparse rewards.
- Task consistency learning allows training of our vacant-space detection network without labeled samples. By dynamically collecting rewards from the motion pattern classification task, the detection agent can be updated continuously and work better day by day.
- Task consistency is verified in the semantic and logistic level. Therefore, the method allows tasks from different domains to help training.

II. Related Works

Transfer learning: Transfer learning [3] is a well-known method that allows a source task to help a target task. Based on large-scale data sets, some deep networks such as VGG16 [7], ResNet50 [8], and Inception [9] had been trained and achieved promising classification results. These networks can extract generalized semantic features for many inference tasks. Therefore, researchers treated the networks as the source models to initialize the target network. While many successful works had been reported, this transfer learning method requires a source task and the target task to be relevant. Besides, extra labeled data is needed for fine-tuning. Also, the training process should be repeated for each new target task manually.

Reinforcement learning: Reinforcement learning (RL) [5], fundamentally suitable for dynamically learning, tries to maximize the accumulation reward. The method has been successfully applied to tackle the sequential decision-making tasks such as Atari Game [10]. Recently, some authors tried to expand the possible applications for RL. Tang [11] applied RL to solve a skeleton action recognition problem; Pirinen [12] used RL for object region proposal; Guo [13] realized an object tracking function by using an RL framework. While applications can be diverse, most of them used supervised labels to define RL rewards. Next, the rewards are used to train the task agents. Hence, label data is a critical factor in making successful learning. Compared with these RL based methods, our method aims to use a logical consistency to form a reward rather than labels for RL learning. Notably, the rewards might be noisy and corrupted.

Vacant space detection:To detect vacant spaces in a parking lot, conventional methods [14] used hand-crafted features and shallow classifiers for detection. Huang [2] introduced 3D geometric models to overcome the inter-object occlusion and the perspective distortion. Recently, a deep learning approach [1] used a spatial transform network and introduced a novel contrastive loss to overcome the vehicle size and vehicle displacement problems. The success of these supervised methods comes from lots of label samples. In contrast, our approach used task consistency to train a vacant-space detection agent.

III. The Proposed Method

In this section, we introduce the concept of task-consistency learning in section III-A. Later, we explain the design details of training a vacant space detection network based on task-consistency learning in section III-B. To be focused, we assume an ”imperfect” source network for motion pattern classification has been deployed in a parking lot. The source network can identify three possible motion states for a parking slot over a time segment that are denoted as ”car moving in”,

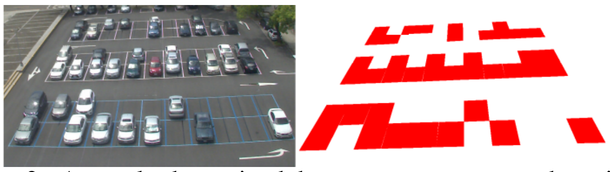


Fig. 2: A result determined by our vacant space detection method. The left-hand side is the input image; the right-hand side is the detection result.

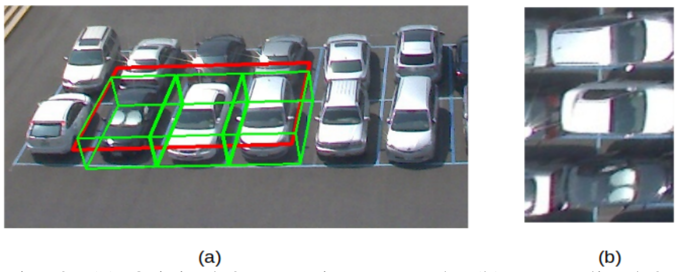


Fig. 3: (a) Original 3-space image patch. (b) Normalized 3-space image patch

"car moving out", and "no car motion". Founded on the task-consistency criterion, we aimed to train an inference agent that can detect vacant spaces in the parking lot. In Fig. 2, we show a detection result based on the vacant space detection network trained by task-consistency learning.

A. *Task-consistency learning*

To be clear, we use Fig.1(b) to explain the proposed RL-based task-consistency learning. In the same working environment, we use an information selector $f^s(.)$ to collect the input data x^s for the source task and the other selector $f^t(.)$ to collect the synchronized input data x^t for the target task. Here, we assume the source task network has been trained by using a source data set (x^s, y^s). At the same time, we collect target input data x^t while the source network is functioning. Next, to complete the RL setting at time moment t, we model our target network as the policy decision agent $\pi_\theta(a_t \mid s_t)$. The output of the target network y^t_t is treated as the action a_t. As for the environment state s_t, it is defined as (y^s_t, x^t_t), the combination of the output of the source network y^s_t and the input of the target network x^t_t. Finally, we use a consistency evaluation function $C(.)$, defined in equation (1), to calculate the reward $r(a_t)$ of doing action a_t at time t given by the previous and next environment states $(y^s_i, x^t_i, a_i)_{i=t-1\sim t+1}$. By considering the information around the current time, we can use temporal consistency and task consistency to evaluate the reward of the action a_t.

$$r(a_t) = C(a_t \mid (x^t_i, y^s_i, a_i)_{i=t-1\sim t+1}) \tag{1}$$

To train our target agent $\pi_\theta(a_t \mid s_t)$, a standard policy-based RL method [5] is applied. It tries to find the optimal agent parameters θ by maximizing the expectation of the accumulated reward over a decision trajectory τ, which is defined as $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[R(\tau)] = \int \pi_\theta(\tau)R(\tau)d\tau$. Here, $R(\tau)$ is the accumulation reward over τ. The optimal solution can be found by using gradient ascent $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$, where the policy gradient $\nabla_\theta J(\theta)$ can be determined by equation (2). Sometimes, a baseline b can be used as a reference to reduce a model variance during training process. Therefore, the gradient could be modified as equation (3) by using the baseline-removed reward, $R(\tau) - b(\tau)$. Finally, for each iteration of parameter updating, several quadruples $e_t = (s_t, a_t, r_t, s_{t+1})$ from an experience-replay buffer [15] are used to estimate the policy gradient.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta log\pi_\theta(\tau)R(\tau)\right] \tag{2}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta log\pi_\theta(\tau)(R(\tau) - b(\tau))] \tag{3}$$

In our parking lot management system, the training trajectory $\tau = \{s_1, a_1, s_2, a_2, ..., s_T, a_T\}$ for reinforcement learning, is composed of T consecutive environment states $\{s_t\}_{t=1\sim T}$ and T status decisions $\{a_t\}_{t=1\sim T}$. T is the number of decisions in one training trajectory. We would clearly explain the way to collect training trajectories below. Given the environment state s_t=(y^s_t, x^t_t), the reward for the decision a_t at time t is denoted as $r(a_t, x^t_t, y^s_t)$. Thus, we may rewrite the policy gradient as equation (4). By calculating the policy gradient over many training video segments, any off-policy method [16] can be used to train the task model.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)} \sum_{t=1}^{T}[\nabla_\theta log\pi_\theta(a_t \mid x^t_t, y^s_t) \\ (r(a_t, x^t_t, y^s_t) - b(a_t, x^t_t, y^s_t))] \tag{4}$$

B. *Task-consistency learning for vacant space detection*

Problem statement: In this section, we applied task-consistency learning to implement vacant space detection. Instead of using supervised status labels of many spaces for training [2], our detector (or the policy agent) is trained based on the task-consistency rewards collected from the motion pattern of parking slots over many video segments. The input of the detector is a normalized 3-space image patch, as shown in Fig. 3; the output (or the agent action), denoted as $a \in \{vacant, occupied\}$, is the space status of the middle slot. To set up our task-consistency-based reinforcement learning, we define the environment states (s) as $s_t = (x^t_t, y^s_t)$ at time t where (x^t_t) is a normalized 3-space image patch (x^t_t) [2] and y^s_t is the motion state of the central space of the patch determined by the source network.

Next, we automatically collected many useful training trajectories form the parking lot scene and dynamically train our vacant space detection agent using equation (4). For training trajectory collection, we relied on our source network, a motion pattern classifier. Given a sequence of images and optical flow maps, the source network detects the motion events for each parking space based on the magnitude of optical flow. For each motion event that happened at one space, a video segment is cropped whose middle-time frames have high flow magnitudes around the space, and other time frames are stable with limited motion. An example can be found in our supplementary. Every frame of the video segment is a normalized 3-space image patch centered at the slot, which is (x^t_t) in our setting. Next, the whole video segment is classified

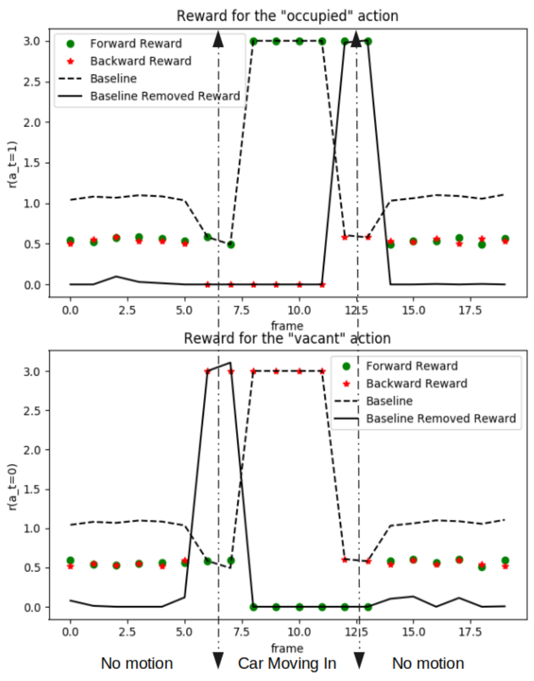


Fig. 4: Intuition of learning mechanism

as "car moving in" behavior or "car moving out" behavior. Accordingly, every frame in the video segment can be labeled as "car moving in," "car moving out," or "no car motion"; this provides the motion state y^s_t. In our system, these 3-space image patches over T frames, coupled with their motion states, are treated as the environment states s of one training trajectory $\tau = \{s_1, a_1, s_2, a_2, ..., s_T, a_T\}$. To be clear, we summarized the proposed method as Algorithm 1.

Agent Network: Our agent has two network inputs, x^t and y^s. The 3-space patch (x^t) is fed to a VGG network for visual feature extraction. In this work, we used the output of the seventh layer of VGG16 [7] as our features. On the other hand, the motion state (y^s) of the corresponding space determined by the source network is also considered. Again, the motion status of the middle space of the 3-space patch can be "car moving in," "car moving out," or "no car motion" state. To fuse two feature modalities, x^t and y^s, we proposed to use Adaptive Instance Normalization (AdaIN) [17]. After feature extraction and fusion, a CNN-based classifier is designed to predict the parking status of the input space. The network design details can be found in our supplementary manuscipt.

TABLE I: Forward-Backward Reward. The first, second, third rows are corresponding rewards conditional on "car moving in" (CI),"no moving" (NM), and "car moving out" (CO) states in y^s space. Here, $P^O_{t-1} = P(a_{t-1} = 1) = \pi_\theta(a_{t-1} = 1 \mid x^t_{t-1}, y^s_{t-1})$ and $P^V_{t-1} = P(a_{t-1} = 0) = \pi_\theta(a_{t-1} = 0 \mid x^t_{t-1}, y^s_{t-1})$ are the probabilities that an action is "occupy" or "vacant" respectively given by input x^t_{t-1} and source prediction y^s_{t-1} in the previous frame. The probabilities could be estimated by the vacant space detector. λ is a hyper-parameter that controls the importance of confident rewards.

	Forward Reward $r_F(a_t \mid a_{t-1}, y^s_{t-1})$		Backward Reward $r_B(a_t \mid a_{t+1}, y^s_{t+1})$		
y^s_{t-1}	occupy ($a_t = 1$)	vacant ($a_t = 0$)	occupy ($a_t = 1$)	vacant ($a_t = 0$)	y^s_{t+1}
CI_{t-1}	$1 * \lambda$	0	0	$1 * \lambda$	CI_{t+1}
NM_{t-1}	P^O_{t-1}	P^V_{t-1}	P^O_{t+1}	P^V_{t+1}	NM_{t+1}
CO_{t-1}	0	$1 * \lambda$	$1 * \lambda$	0	CO_{t+1}

Algorithm 1 Task Consistency learning (TCL)

```
1: procedure TCL( Require: $M^s_\phi(x^s_i)$, Environment, $f^s(s_t), f^t(s_t)$; Output: target network $\theta$)
2:     $D_{online} = \varnothing$
3:     Random initialize $\theta$
4:     while collect training trajectories do          ▷ Data collection
5:         Query $s_t$ from the environment
6:         $x^s_t = f^s(s_t)$, $x^t_t = f^t(s_t)$          ▷ Select source and target input
7:         $\hat{y}^s_t = M^s_\phi(x^s_t)$          ▷ Predict source decision
8:         if $P(y^s_t) > \delta$ then
9:             Extract $x^t_i = f_t(s_i) \mid i = 1 \sim T$ from the segment trajectory around the frame $t^{th}$.
10:        end if
11:        Add the tuple $(x^s_i, \hat{y}^s_i, x^t_i)_{i=1:T}$ into $D_{online}$
12:    end while
13:    while Training do
14:        for each training trajectory do
15:            Run the target / policy model $\pi_\theta(a_t \mid x_t)$ to get $a_t$ or $y_t$
16:            Estimate $r(a_t)$ by equation (5)
17:            Estimate baseline value by equation (6)
18:            Estimate the gradient in equation (4)
19:            Update the policy $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
20:        end for
21:    end while
22:    Repeat the data collection process in the 4th line
23:    Repeat the training process in the 13th line
24: end procedure
```

Reward design: To train our policy agent, a reward function is needed. Here, we based on the logical consistency between the decision of motion recognition y^s and the inference of vacant space detection a^t to define our reward. We consider two temporal passes. For the forward pass, if y^s_{t-1} is "moving in" at the previous time, we would expect the slot is occupied at time t. If y^s_{t-1} is "moving out", we prefer the slot is vacant at time t. Next, if the action a_t matches the status expectation, we assign λ as the reward. Otherwise, the reward is set to zero. For the case y^s_{t-1}="no motion", the expected slot status is ambiguous. To set the reward, we proposed temporal consistency to encourage same action decisions of a_t and a_{t-1}. Thus, we use the action probability $P(a_{t-1})$ as the reward. To be clear, we summary our reward function $r_F(a_t \mid a_{t-1}, y^s_{t-1})$ of the forward pass In Tab. I. Following the similar concept but for the backward pass, we also summarize the other reward function $r_B(a_t \mid a_{t+1}, y^s_{t+1})$ in Tab. I. By considering both the forward and backward passes for training, the reward function is estimated by equation (5). Finally, we proposed a reward baseline, which is defined in equation (6). The baseline helps to adjust the summation reward in equation (5) dynamically and force the agent to focus on the critical observations when the reward difference between two decisions is large.

$$r(a_t) = r_F(a_t \mid a_{t-1}, y^s_{t-1}) + r_B(a_t \mid a_{t+1}, y^s_{t+1}) \tag{5}$$

$$b(a_t) = min(r(a_t = 1), r(a_t = 0)) \tag{6}$$

The intuition of learning mechanism: Next, we explain the intuition of the learning mechanism behind the reward design and task-consistency learning. In Fig. 4, we show a slot training trajectory composed of a "no motion (stable)" phase, a "car moving in (transient)" phase, and followed the other "no motion (stable)" phase. During the RL training phase, we ultimately aim to maximize the accumulation reward so that the vacant space detection agent makes accurate and stable status inference for the "stable" stages and ignore the "transient" phase. At the beginning of the training, the decision of the agent is almost equal to a random guess. That is $P(a_t = 1) = P(a_t = 0) = 0.5$. According to our reward design, the rewards for different agent decisions (occupied and vacant) over the training trajectory are drawn in Fig. 4. To maximize the baseline-removed rewards at this moment, our agent is required to output the "vacant" state at the time (6,7), choose the "occupied" state at the time (12,13), and do not care at the other moments. After this training iteration, the agent would make $P(a_6 = 1) = 1$ and $P(a_{13} = 0) = 1$. In the next training iterations, owing to the temporal consistency in the "no motion" state, the reward would propagate backward from time 6 to time 1 and propagate forward from time 13 to 19. Thus, the agent would be trained to output the "vacant" state from time 1 to 7 and output the "occupied" state from time 13 to 19. Note that due to baseline reward removal, the importance of the frames within the "transient" phase is highly reduced. Hence, we can say that the (6,7,12,13) are keyframes that have high confident information to guide others.

Particularly, the reward baseline plays an essential role in stabilizing the training. We may find the summation rewards within the "transient" stage are both high, no matter the occupied decision or vacant decision. That means the "transient" phase provides few useful supervised messages for the training agent since the reward difference is small. However, the RL would focus on gathering rewards within the "transient" stage if we do not apply the baseline-removed reward. Moreover, during the "transient" phase, we may collect many 3-space patches with moving cars. Intuitively, these samples are not suitable for training.

IV. Experimental Results

To evaluate the proposed method, we collect training data and validation/testing data. The training dataset includes videos in a parking lot, and the testing dataset includes images in the same scenario. The frame rate for video collection is one frame per second. One video includes 500 sequential frames. In total, 120 videos have been randomly collected within three weeks for training. From these videos, the process explained in section III-B had been applied to acquire 1530 slot training trajectories accompanied by the motion state for each frame. The length of every trajectory is 60. Besides the training trajectories, testing images are needed for quantitative evaluation. Three hundred nine images collected within 15 days have been labeled. Each frame has 64 parking-slots; hence, the evaluation dataset has 19776 testing samples composed of 11944 vacant samples and 7832 occupied slots. In the testing phase, to evaluate the trained vacant space detection network, we set the motion state to "no car moving" and only use an RGB 3-space image patch as input.

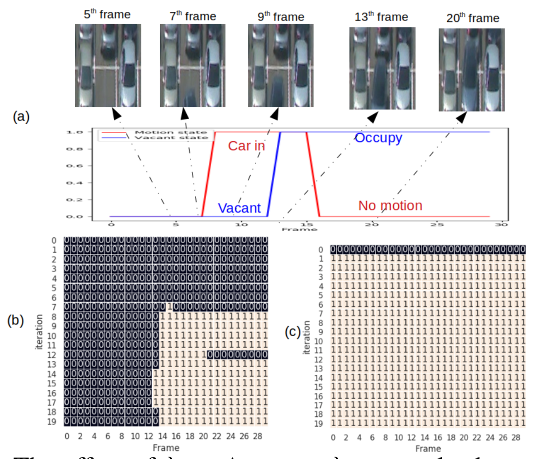


Fig. 5: The effect of $\lambda =$. A proper $\lambda =$ may lead to a better learning model. (a): A training trajectory with its true motion states (red color) and expected parking states (blue color). The car moves in from the 7^{th} frame to the 16^{th} frame. (b) and (c) are the parking status inference of each frame over the training procedure under $\lambda = 30$ and $\lambda = 5$. "1" means occupied state; "0" means vacant state. X-axis: the frame ID; Y-axis: the index of training iteration. $\lambda = 30$ gives a better result.

Moreover, to compare "task-consistency learning" and "fully-supervised learning," the other training dataset collected over 30 days has been labeled for supervised learning. The dataset includes 14667 vacant slots and 20021 occupied slots. By using the same agent network but trained in a supervised learning way, we achieve 99.81% accuracy (ACC) of detection in the testing dataset. The true-positive rate (TPR) and false-positive rate (FPR) are 99.78% and 0.14% correspondingly. In comparison, our RL-based method has 98.89% ACC, 99.36% TPR, and 1.92% FPR.

The explanation of our experiments is organized as follows. First, we discuss the effect of the parameter λ on the learning process in section IV-A. Second, an ablation study is discussed to show the effectiveness of the proposed reward design in section IV-B. Later, section IV-C evaluates the quantitative performance of the vacant space detector supervised by a weak motion classifier. Last but not least, section IV-D introduces an online learning on a new parking lot to prove the benefit of our work when deploying a new model.

A. *The effect of λ on the learning process*

In this section, we discuss the effect of λ parameter. First, we use one training trajectory to update our vacant space detector. The trajectory includes 30 frames, as shown in Fig. 5(a), where a car moves into the slot from the 7^{th} frame to the 15^{th} frame. Also, the vacant state is "occupied" from the 13^{th} frame. Fig. 5(b) and Fig. 5(c) show the inference of the parking status of each frame during the training procedure under $\lambda = 30$ and $\lambda = 5$, respectively.

From Fig. 5, we see the value of λ plays the role of controlling the quality of our reward function. We may explain it by following the intuition illustrated in Fig. 4. In the early training phase, we expect the agent mainly focuses on making correct inferences for the anchor frames. Later, the agent may leverage the temporal consistency to correct the inference of the neighboring frames. However, the expectation may not happen if λ is small, in which the agent may try to collect the temporal consistency rewards from many "no motion" frames and ignore the few and critical anchor frames. When this happens, the agent may be stuck in a locally optimal solution and lead the inference of all frames to be "vacant" or "occupied." The phenomena can be found in Fig. 5(c). To overcome the problem, the rewards, λ, for the anchor frames must be large enough . In the early training phase, we find setting λ equal to the number of frames in a training trajectory can always achieve a better learning result.

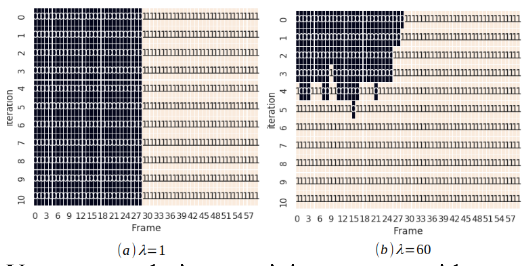


Fig. 6: Vacant state during a training process with a corrupted trajectory.(a): $\lambda = 1$, (b): $\lambda = 60$

In our experience, a fixed and large λ can work well if the supervision provided by the anchor frames is always correct. However, the imperfect motion pattern classification raises another practice challenge, where the labels of a few training trajectories may be corrupted. We would discuss the issue of noisy labels deeper in subsection IV-C. To solve the problem, we borrowed the concept from simulated annealing and proposed to cool down λ dynamically. In the initial phase, we set a high λ and trust the supervision from the motion pattern classifier to train the vacant space detector. Although the result is not perfect, our detector can still learn the discriminative features for vacant space detection given the correct supervision is major. Later, we continuously reduce the value of λ. The idea is to improve the detection model by incrementally leveraging the unsupervised image feature similarity through temporal rewards and decreasing the importance of motion supervision. Therefore, the effect of corrupted supervision can be significantly alleviated. An example can be found in Fig. 6. in which the detection model has been trained for a while. Next, a "car moving in" trajectory is misclassified as "car moving out" and is used to update the detector. If we set $\lambda = 60$ to trust the supervision, we can find the detector is guided in the wrong direction after several training iterations even though the previously-trained model works correctly in the initial iterations. In contrast, if $\lambda = 1$, the effect of corrupted supervision is overwhelmed by the temporal rewards and the previously-trained model.

B. *Ablation Study*

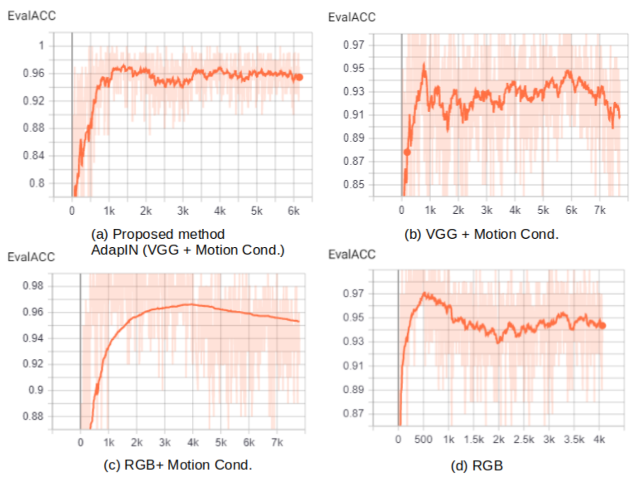


Fig. 7: Accuracy on evaluation set over the training iteration.

In this section, we do an ablation study to understand the function of the proposed agent network. We focus on three parts: the AdaIN module, the VGG feature, and the motion state input. The accuracy distributions of the evaluation/testing dataset under different settings are shown in Fig 7. Note that we show the testing performance over the training iteration. The x-axis represents the training epoch index, and the y-axis presents accuracy (ACC_t). The narrow line shows the accuracy estimated by using randomly-selected 100 testing samples; the wide line is a smoothed accuracy ($SACC_t$), which is calculated by $SACC_t = 0.9{*}SACC_{t-1}{+}0.1{*}ACC_t$.

AdaIN contribution: To better understand the contribution of the AdaIN [17] module, we directly concatenate the image features and motion features for feature fusion to replace AdaIN. Next, the feature is fed into our classifier for vacant detection. The accuracy curve of the setting is in Fig.7(b). Compared with our proposed method in Fig.7(a), the setting without AdaIN tends to degrade the accuracy. In Fig.7(c) and Fig.7(d) where no AdaIN is used, we may also observe a similar phenomenon. These observations show that the AdaIN operation is a better way for feature fusion.

Contribution of VGG features : While some promising results can be observed in Fig.7(a) and Fig.7(b), we may not understand the performance majorly coming from the pre-trained VGG features or the designed reward. Therefore, we train the whole network from scratch without using the pre-trained VGG features. The performance of the setting is presented in Fig7c. After 7000 training epoch, the accuracy with and without the VGG features are 95.50% and 95.30%.

It shows that our reward design and the training method can learn to infer the slot status even from scratch. However, by using the VGG features, the network tends to convert faster. Also, the VGG features help to reduce the over-fitting effect.

Motion condition as an input: To understand the function of using motion condition as one of the inputs, we compared with the other trained networks without motion condition. The result of this setting is illustrated in Fig.7(d). Although the network can be trained, we find the learning curve is more sensitive and unstable. The accuracy also degrades compared with the proposed method, shown in Fig.7(a). The result points out that our reward design and RL-based learning can help to train a vacant detector without the motion condition as an input. However, if the current motion state is available, we may have more information for robust learning.

C. Learn from the imperfect motion pattern classification

In this section, we aim to evaluate the task consistency learning given the imperfect motion pattern classification. Note that the semantic output of the motion pattern classification is the primary supervision for task consistency learning. However, no classifier is perfect. In our experiments, we designed a baseline motion pattern classifier and used it to verify the function of task consistency learning. Due to the limited space, we detail our based line motion classifier in the supplementary manuscript.

According to the motion pattern classification, all the frames within the motion segment would be labeled by the same motion state. These motion frames, coupled with extended stable frames, compose a training trajectory. However, to ensure the robustness of task consistency learning, we hope to only use the confident training trajectories to train the vacant space detector. Here, a probability threshold δ is applied to the output of our motion pattern classification network to select the training trajectories. A higher δ is usually preferred to make sure the training data quality.

Next, we used 1530 training trajectories to evaluate the task consistency learning given the imperfect motion pattern classification. To filter out less confident trajectories, three thresholds, $\delta = 0.8$, $\delta = 0.7$, and $\delta = 0.6$, have been applied.We repeated the training process 10 times and used a fixed and high λ for this experience. The accuracy distribution is visualized in Fig. 8(a). When $\delta = 0.8$, only 426 training trajectories are finally selected to train the vacant space detector; among them, 33 trajectories are corrupted. In this setting, the average accuracy of the vacant space detector is 89.87%. When $\delta = 0.7$, the number of qualified training trajectories increases to 748; however, the number of corrupted trajectories increases slightly to 82. In this setting, the average accuracy increases to 92.76%. If we keep reducing the threshold to $\delta = 0.6$, we have more training trajectories but also more corrupted ones. In this case, the average detection accuracy reduces significantly to 90.14%.Obviously, the performance is affected by the corrupted supervision. As discussed in section IV-A, to overcome the problem, we linearly cool down λ by following the mechanism in Fig. 8(b). Eventually, the average accuracy increases significantly to 96.06%.

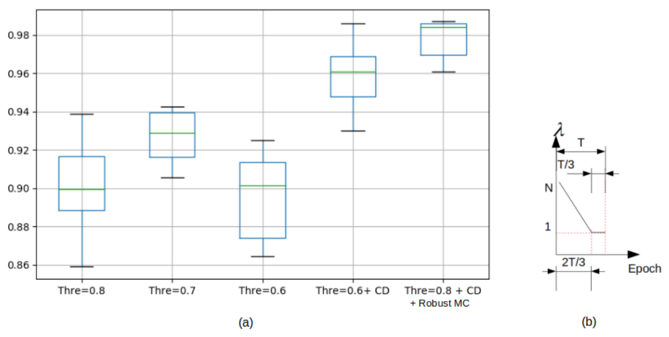


Fig. 8: The performance under corrupted training trajectories. (a) Accuracy distributions over 10 times of training. (b) A cooling down (CD) mechanism to reduce λ during a training process (T=6).

In this experiment, we recognize that the vacant space detector's performance is affected by two factors. The first factor is the number of qualified training trajectories. A higher threshold usually implies that fewer training trajectories can be collected. In this case, our system may spend more time to collect enough training trajectories to update the agent. The second factor is the corrupted reward. The result in Fig. 8 points out that only a few false motion labels can significantly degrade our system performance. Therefore, it is worthwhile to discuss a better mechanism to filter out the corrupted trajectories during the learning process. We find a more robust motion pattern classifier can help to reduce the noisy trajectories. Thus, we design another stronger motion classifier to replace the baseline one. By using the robust motion classifier and the cooling down technique, the accuracy increases to 98.89% as shown in Fig. 8. To not obscure the focus of this work, we provide the details of the robust motion classifier and report its performance in the supplementary document.

D. Online learning in a new parking-lot

In this section, we prepared a transfer learning experiment to compare the difference in setting up a new parking lot based on SL and TCL for fine-tunning. The original lot ("90^o view") and the new lot are shown in Fig. 9. For performance evaluation in the new environment, 12257 vacant samples and 8623 occupied slots were collected for testing. Besides, we use 34688 samples to train the vacant-space detector in the "90^o view" lot and use it as the initial model for transfer learning. Note that the model achieves 99.81% detection accuracy in the "90^o view". However, the accuracy is only 70.87% if simply applying the model in the "45^o view" without fine-tunning. The domain gap significantly degrades accuracy.

Next, we used TCL to update the model in an unsupervised way dynamically. To automatically collect supervision from task consistency, we apply the robust motion classifier mentioned in section IV-C in the new lot. A threshold $\delta = 0.9$ used in the motion classifier is selected to remove the unconfident

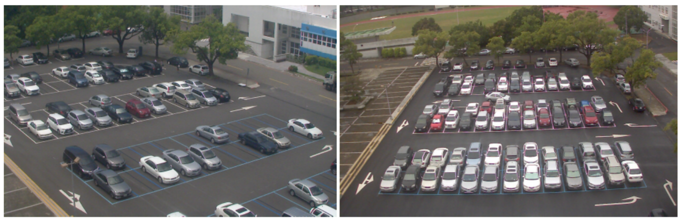

45-degree view(as new parking lot scenario)
90-degree view (as original parking lot scenario)

Fig. 9: An example of "45^o view" and "90^o view".

trajectories for TCL training. As shown in Table II, as the collected trajectories increase (N goes from 300 to 1000), TCL improves the pre-trained model and achieves a 99.69% accuracy. For comparison, we fine-tune the pre-trained model using SL. The number of labeled samples for fine-tuning is M = (600, 1000, 1400, 1700, 2000). The results in Table II show that TCL performs slightly better than SL in this experiment. The improvement may come from the way for supervision collection and the learning mechanism. For TCL, the sequential frames of a trajectory are quite similar but only different in the target space. The systematically-collected supervision forces the network to focus on learning the features that present the main difference. In contrast, the labeled samples for SL are selected randomly and may be redundant.

Also, we discuss the cost of building up a new learning system. The time cost includes training (1) data collection, (2) data labeling time, and (3) network training time. Among them, the time for data collection and selection is critical. It takes much more time and human labor for supervised learning (SL). Instead, task consistency learning (TCL) collects the representative samples automatically. This property may make TCL a more convenient way for system setup. Moreover, the environment may dynamically change over time (seasons). Compared with SL, the design of TCL is more suitable for dynamic learning. For data labeling, we need 2 seconds on average to label one sample for SL. Hence, the labeling time efforts of M samples for supervised SL is $N/30$ minutes. In contrast, TCL does not need time for data labeling. The comparison of computational time is summarized in Table III.

TABLE II: Accuracy of the "45^o view" lot via transferring the "90^o view" pre-trained model. M: the supervised sample number. N: the trajectory number for TCL. (Unit is %)

SL	97.93 (M=600)	98.18 (M=1000)	98.57 (M=1400)	98.76 (M=1700)	99.21 (M=2000)
TCL	98.84 (N=300)	99.15 (N=500)	99.37 (N=700)	99.57 (N=850)	99.69 (N=1000)

TABLE III: A time cost comparison between TCL and SL

Task consistency learning			
N trajectory	700	850	1000
Iteration	3500	4250	5000
Learning time (minutes)	29.75	36.125	42.5

Supervised learning			
N sample	1400	1700	2000
Iteration	280	340	400
Labeling time (minutes)	46.67	56.67	66.67
Learning time (minutes)	1.26	1.53	1.8
Total time (minutes)	46.67	58.19	68.46

We can see that SL can learn faster than TCL. However, SL spends extra time on data labeling.

V. Conclusions

In this paper, we proposed a task-consistency learning method to train a vacant space detector via a naive motion classifier in a reinforcement way. Instead of using labeled data to train a target network, we use the inference consistency between the source and target tasks to define a reward system for agent training. The learning framework has two benefits. First, even the source task and target task have domain differences, task-consistency learning is still possible. Second, it reduces human labor for data labeling. Experimental results prove that the reward design can help to transfer knowledge from the motion classifier to the vacant space detector. However, false detection from the motion classifier would introduce corrupted rewards and degrade vacant detection performance. Our future work will aim to eliminate the negative message transfer.

References

[1] H. T. Vu and C. C. Huang, "Parking space status inference upon a deep cnn and multi-task contrastive network with spatial transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 4, pp. 1194–1208, Apr. 2019.

[2] C. C. Huang and H. T. Vu, "Vacant parking space detection based on a multilayer inference framework," *IEEE Trans. Circuits Syst. Video Technol.*, 2017.

[3] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, Dec. 2016.

[4] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[5] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances Neural Inf. Process. Syst.*, pp. 1057–1063, 1999.

[6] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," *Int. Conf. Learning Representations*, Feb 2018.

[7] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks For Large-Scale Image Recognition," *Proc. IEEE Conf. Learn. Repr.*, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2016.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *NIPS Deep Learning Workshop*, 2013.

[11] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou, "Deep progressive reinforcement learning for skeleton-based action recognition," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[12] A. Pirinen and C. Sminchisescu, "Deep reinforcement learning of region proposal networks for object detection," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[13] M. Guo, J. Lu, and J. Zhou, "Dual-agent deep reinforcement learning for deformable face tracking," *Proc. Eur. Conf. Comput. Vis.*, Sept. 2018.

[14] L. Wang and D. Jiang, "A method of parking space detection based on image segmentation and LBP," *Int. Conf. Multimedia and Secur.*, 2012.

[15] D. Zhao, H. Wang, K. Shao, and Y. Zhu, "Deep reinforcement learning with experience replay based on SARSA," *IEEE Symp. Ser. Comput. Intell.*, 2017.

[16] D. Lyu, B. Liu, M. Geist, W. Dong, S. Biaz, and Q. Wang, "Stable and efficient policy evaluation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 6, pp. 1831–1840, 2019.

[17] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," *Proc. IEEE Int. Conf. Comput. Vis.*, 2017.