Intelligence & Robotics

Research Article

Open Access

Reinforcement learning methods for network-based transfer parameter selection

Yue Guo[1], **Yu Wang**[2], **I-Hsuan Yang**[2], **Katia Sycara**[1]

[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA.
[2]PlusAI, Inc., Santa Clara, CA 95054, USA.

Correspondence to: Yue Guo, School of Computer Science, Carnegie Mellon University, 1602B Newell-Simon Hall 5000 Forbes Ave, Pittsburgh, PA 15213, USA. E-mail: yueguo@cs.cmu.edu

How to cite this article: Guo Y, Wang Y, Yang IH, Sycara K. Reinforcement learning methods for network-based transfer parameter selection. *Intell Robot* 2023;3(3):402-19. http://dx.doi.org/10.20517/ir.2023.23

Received: 20 Jun 2023 **First Decision:** 18 Jul 2023 **Revised:** 14 Aug 2023 **Accepted:** 23 Aug 2023 **Published:** 31 Aug 2023

Academic Editor: Jianjun Ni, Simon X. Yang **Copy Editor:** Yanbin Bai **Production Editor:** Yanbin Bai

Abstract

A significant challenge in self-driving technology involves the domain-specific training of prediction models on intentions of other surrounding vehicles. Separately processing domain-specific models requires substantial human resources, time, and equipment for data collection and training. For instance, substantial difficulties arise when directly applying a prediction model developed with data from China to the United States market due to complex factors such as differing driving behaviors and traffic rules. The emergence of transfer learning seems to offer solutions, enabling the reuse of models and data to enhance prediction efficiency across international markets. However, many transfer learning methods require a comparison between source and target data domains to determine what can be transferred, a process that can often be legally restricted. A specialized area of transfer learning, known as network-based transfer, could potentially provide a solution. This approach involves pre-training and fine-tuning "student" models using selected parameters from a "teacher" model. However, as networks typically have a large number of parameters, it raises questions about the most efficient methods for parameter selection to optimize transfer learning. An automatic parameter selector through reinforcement learning has been developed in this paper, named "Automatic Transfer Selector via Reinforcement Learning". This technique enhances the efficiency of parameter selection for transfer prediction between international self-driving markets, in contrast to manual methods. With this innovative approach, technicians are relieved from the labor-intensive task of testing each parameter combination, or enduring lengthy training periods to evaluate the impact of prediction transfer. Experiments have been conducted using a temporal convolutional neural network fully trained with the data from the Chinese market and one month's US data, focusing on improving the training efficiency of specific driving scenarios in the US. Results show that the proposed approach significantly improves the prediction transfer process.

Keywords: reinforcement learning, transfer learning, deep learning, self-driving, network-based transfer

1. INTRODUCTION

The prediction module in autonomous vehicles (AVs) plays a pivotal role, as it enables these AVs to anticipate the intentions of surrounding vehicles. Data gathered from perception devices are processed by this module, which typically uses a neural network (NN) to generate a label or trajectory that represents these intentions. The accuracy of this prediction is undoubtedly critical to the ability of AVs to understand the environment and serves as a prerequisite for making autonomous and correct decisions. One of the key challenges in the prediction module is the heavy reliance of prediction models on domain-specific data sets. Self-driving companies operate across various markets, each characterized by unique data domains influenced by differing traffic regulations, driving cultures, and geographical characteristics.

However, the opaque nature of NN models makes it difficult to identify which factors specifically influence the models. Moreover, slight variations in data can significantly alter the parameters of these models. Training domain-specific models is economically burdensome but necessary for self-driving companies to ensure accuracy across different markets. Additionally, in some markets, data collection may be constrained by legal, financial, and resource-related factors, leading to data inefficiency and increasing the difficulty of developing a model specifically for this domain. Thus, a crucial focus in improving the prediction module is to optimize the utilization of existing prediction models. This means if a model can effectively generalize its predictions, successful knowledge transfer could significantly reduce the development costs associated with prediction models. However, this transfer process is complex, as models developed for specific markets are not easily transferrable due to domain specifications.

The rapid advancement of transfer learning may provide solutions to the challenges of domain-specific models. Traditional transfer learning techniques, which reuse models trained on sufficient data from a specific source domain, can reduce biases in domain features and statistics, even when the model is applied to another target domain with only a smaller and potentially less representative dataset available. At first glance, these advancements seem to solve the aforementioned problems. However, a significant roadblock arises in domain adaptation tasks: these methods typically require access to both source and target data domains, and the quantity of this data must be enough to ensure representability. This is because traditional transfer learning studies domain adaptation and domain generation with the assumption that a classifier working well in the source dataset may also function in the target dataset. This is achieved by examining the domain differences between the two datasets or generalizing from the source dataset. Work in deep transfer learning specifically focuses on classifiers that are deep NNs, but these, too, usually require that both source and target datasets be accessible. Generating and accessing both datasets can be expensive, and legal regulations can complicate the transfer of data across different markets. For instance, data protected in one country might not be usable in another, or due to strict regulations, no data could be exported at all. Consequently, these factors demonstrate that traditional transfer learning may not fully address the challenge of domain specificity.

A promising subfield of transfer learning, referred to as network-based transfer, provides a solution to the problem of inaccessible datasets. The basis of this approach lies in the inherent structure of NNs, which extract different levels of features. For instance, generic features can be transferred between domains by replicating high-level parameters of certain networks. Moreover, given a small amount of data in the target domain, these networks can be fine-tuned to exceed the performance achieved by training on the target dataset from scratch. Network-based transfer eliminates the need for direct access to the source dataset or for comparing datasets, as the essential knowledge is already encoded within the network after training and can be transferred to the new domain.

However, network-based transfer does have two major limitations: (1) Pretraining with specific features from the source domain can sometimes be misleading, resulting in performance that is even worse than training from scratch. The success of transfer largely depends on the similarity between the source and target datasets, and there is no practical way to ensure this similarity, nor do technicians know how they can identify potentially harmful features; (2) The issue of parameter selection emerges when only a few parameters are meaningful for transfer, especially in the context of deep networks. Manually searching for the optimal combination of parameters to transfer can be extremely time-consuming. Thus, developing an automatic parameter selector is crucial to optimize network-based transfer in order to improve efficiency and facilitate a positive transfer.

In this paper, an innovative approach is proposed to enhance network-based transfer, specifically for prediction in AVs across international markets. The Automatic Transfer Selector via Reinforcement Learning (ATSRL) is introduced, utilizing a reinforcement learning (RL) agent to automatically select the portions of the network that should be transferred. This technique significantly enhances the efficiency of parameter selection for transfer prediction, providing a solution to the labor-intensive task of manually testing each parameter combination or enduring lengthy training periods to evaluate the impact of prediction transfer. To be more specific, a teacher NN trained on inaccessible source data is used as the starting point, and a set of student networks that pre-load various parameter units from the teacher network is provided. The RL agent treats its current student selection as its state, the next student selection as its action, and a statistical score of a small batch of data on which the student network is trained as its reward. By treating the selection process as an RL problem, the transfer procedure is accelerated since the most promising students are prioritized. The experimental setup involves the application of ATSRL to a temporal convolutional NN (CNN), with the teacher's training performed on sufficient China data and transferred to the student's training on one month's US data, with the focus on specific driving scenarios. The results demonstrate significant improvement in the prediction transfer process.

In summary, this paper makes the following key contributions:

1. RL is leveraged to establish an efficient and safe transfer mechanism for domain-specific prediction models in self-driving vehicles. The method is model-agnostic, implying it can be applied to any specific prediction model.
2. The proposed method is not hindered by the need to collect data from various domains; instead, it requires only the parameters of the prediction model trained with the source data. This approach enables model training even with a limited amount of self-driving data in a new market, and most importantly, the transfer procedure is both fast and safe.
3. Experiments have been conducted using a temporal CNN, an advanced model widely used in the industry. The effectiveness of the proposed method is empirically demonstrated.

The structure of the rest of this paper is as follows: Section 2 delves into related works, specifically discussing transfer learning papers, with a particular emphasis on network-based ones and other automatic transfer works. Section 3 provides the background on transfer learning with a network structure and introduces the basics of RL. In Section 4, the proposed method ATSRL is discussed in detail. Section 5 presents the experimental results, demonstrating the application to the industry-level model. Finally, Section 6 concludes the paper with a discussion of the presented work and potential future research directions.

2. RELATED WORKS

A highly regarded survey paper suggests four categories of modern transfer learning: instance-based, mapping-based, network-based, and adversarial-based[1]. Instance-based methods posit that partial instances from the source domain can be leveraged in the target domain by weight adjustment. For example, some researchers utilize AdaBoost-based technology to adjust weights of source domain instances, filtering out ones dissimilar to the target domain for classification[2]. This line of work has seen improvements in extensions to regression problems[3], faster algorithms[4], and more advanced metric frameworks[5]. However, it assumes access to source instances, which may not be possible due to legal regulations within market-protected self-driving data.

Similar issues arise with mapping-based methods, which rely on establishing a mapping relationship from source to target domains to a new data space with better similarity. Traditional works have applied transfer component analysis[6,7], which has been extended to deep NNs to develop an adaptation module[8]. However, difficulties arise when the prerequisite of statistically measuring both source and target datasets is hampered by restricted access to the source dataset. The unclear legal status of sharing source data statistics can place burden on technicians collaborating across markets, both in terms of cost and manpower.

Contrastingly, neither network-based nor adversarial-based approaches are concerned with the inaccessibility to the source dataset. The aim of both is to identify transferable components within networks. Adversarial-based approaches rely on an adversarial layer, inspired by generative adversarial nets (GAN)[9] to identify transfer components, an advancement from basic network-based approaches. However, adversarial-based approaches[10,11], which incorporate an adversarial layer into the prediction network, substantially modify the structure of the prediction networks. Despite increasing interest, most of the experiments have been conducted using basic CNNs[12].

This paper proposes a method that falls into the network-based transfer category. A key challenge in this category is identifying which components are transferable. For network-based deep transfer learning, certain pre-trained network parameters from the source domain may be reused in the target domain to improve the performance. It is important to note that while some network modules are transferable, others may obstruct transferability[13]. This seminal work has motivated numerous researchers to explore the idea of transferring parts of the network, as opposed to traditional data-centric approaches. This strategy has been proven effective across various areas and is widely employed. For example, knowledge can be transferred from apparently unrelated domains, such as real-world object detection, to gravitational wave signal detection[14]. This demonstrates that NNs are excellent feature extractors for clustering algorithms, even though the data domains may be irrelevant. Different layers extract different data representations, and researchers have used front layers to compute intermediate image representation across various datasets[15], presenting the success of finding generic features of the images. Certainly, not all network-based approaches eliminate the need for direct source data access. In some works, adaptive classifiers and transferable features are jointly learned for both source and target datasets[16], or domain adaptation commonly employed in traditional transfer learning is utilized[17].

Despite the diverse methods offered by network-based approaches given the advancements in modern NNs, and their ability to greatly reduce the need for source data access, these approaches come with their own limitations. Primarily, there is no systematic method to determine which network parts are beneficial for transfer. Using the basic method outlined in[13], it is extremely time-consuming to find transferable network parameters. Moreover, preventing a non-transferable part from hindering the prediction is not straightforward. A great amount of research work focuses on avoiding ”negative transfer”, which involves the transfer of inapplicable modules[18]. The proposed method in this paper builds an automatic selector to ensure the efficient and positive transfer of the network.

A few works have considered automating the process of transfer. For example, there is an overlap between a subfield of meta-learning and network-based transfer learning, wherein the parameters of the network in the source domain may be used to improve the initialization of the one in the target domain. Works in meta-learning aim to learn optimal parameters for networks, with a subset considering reusing parameters from a source domain[19].

Few-shot learning is motivated by the idea of training using only a handful of samples from the target domain, but this approach is model-specific[20].

Model-Agnostic Meta-Learning (MAML) shares the closest motivation to the approach proposed in this paper within the framework of meta-learning, where researchers aim to identify parameters that are most sensitive to task changes and can largely improve the loss[21].

However, MAML makes the assumption that loading all parameters directly as an initialization is beneficial for the target task and that the most promising parameters should be primarily improved. This approach does not fully address the potential for negative transfer that can occur when all parameters are copied and used as initial points. Moreover, more complex and dissimilar datasets may present greater challenges for transfer, as discussed in[22].

The experiments detailed in the renowned paper are simplistic, focusing either on object detection or Mojoco robotics, and may be impractical for real-world scenarios such as self-driving prediction data across different markets. In addition to MAML, a pairwise comparison inspired by meta-learning has been designed to determine what and where to transfer[23].

This involves using a meta-network to learn mappings and weights accordingly for transferable layers. This approach, more robust than MAML, no longer assumes that all parameters are beneficially initialized but instead attempts to select the good ones. However, this method still requires learning the results of all combinations of parameters, leading to an inefficient procedure. In contrast, the method proposed in this paper does not alter the loss or experiment with all combinations of parameters to test transferability. Instead, it judges which parts of the network would be beneficial for transfer by assigning a small score and then proceeds with the promising candidates. This approach is safer and more efficient for transfer for real-world datasets for self-driving prediction.

Other distinct approaches to transfer learning exist. To name a few, unsupervised learning and fine-tuning of transferable base knowledge are enabled through multi-scale convolutional sparse coding[24].

Furthermore, RL algorithms have been utilized in the design of neural networks[25], where a learning agent explores architectures to enhance performance. The complexity of these networks might be better captured by considering non-topological features, as presented in recent studies of complex network structures[26].

However, the method proposed in this paper, while echoing the idea of RL in architecture design, distinguishes itself as the first to employ RL in selecting transferable parameters specifically for complex, real-world datasets, such as self-driving data, in a safe and efficient manner.

3. BACKGROUND

In this section, we briefly introduce the background of Transfer Learning based on the Network Structure and RL.

3.1. Transfer learning with network structure

The transferability of an NN is negatively affected by specific high-level features and difficulties associated with splitting co-adaptive features[13]. In this work, Yosinski proposed a method to find out what features were transferable from the networks trained by source dataset A to the target dataset B. The method required two networks trained on each of the datasets, which were called baseA and baseB.

The control group, referred to as BnB, copied n layers of parameters from baseB, and the rest of the parameters were initialized as usual. For example, when $n = 3$, the performance on dataset B was compared by freezing or not freezing the parameters of layers 0, 1, and 2 to determine if the current parameters could achieve performance equivalent to baseB.

In that case, co-adaptive features that explained potentially worse performance could be found so that the designer would know those features should not be used separately.

The actual transfer work was done by copying parameters from baseA and examining how useful they were compared to the previous. Through analysis and comparison, features could be inferred if they were transferable or not. For example, Yosinski introduced an experiment where performance of the network dropped when freezing 4 and 5 layers of parameters because they were co-adaptive and fragile. Thus, it was anticipated that the performance of transferring parameters of A with 4 and 5 layers dropped. However, the performance drop with 6 and 7 layers transferred was not anticipated because no co-adaptive features affected the performance; thus, the layers were purely not transferrable due to high-level specific features. Most importantly, pre-training with the weights always led to better performance.

However, while all parameters in the work of Yosinski were trainable, the author did not consider the cost of training time.

3.2. Reinforcement learning

A Markov Decision Process (MDP) is a discrete-time mathematical framework used for decision-making, wherein an agent takes an action and receives feedback from the environment at each step.

Markov Decision Process An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is a reward function, $\mathcal{T}(s, a, s') = p(s'|s, a)$ is a transition function, and $\gamma \in [0, 1]$ is a discount factor.

A *policy*, $\pi(s, a) = p(a|s)$, specifies the probability of performing action a at state s. Under a given policy π, the value of a state $V^\pi(s)$ is defined by the expected total discounted reward obtained starting from state s:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t) | s_0 = s, \pi\right] \tag{1}$$

Notice that the next state is sampled from a probability distribution, determined by the current state and action according to the transition function $\mathcal{T}(s_t, a_t, s_{t+1}) = p(s_{t+1}|s_t, a_t)$. Intuitively, this equation recursively sums the rewards obtained from the start to the final state.

The Q-function $Q^\pi(s, a)$ is similarly defined by the expected total discounted reward obtained starting from state s and performing action a:

$$Q^\pi(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')[\mathcal{R}(s') + \gamma \max_{a' \in \mathcal{A}} Q^\pi(s', a')] \tag{2}$$

The Q value thus serves as an estimation of the accumulative reward that can be obtained for a state-action pair. Traditional RL aims to find the optimal policy π^*, which maximizes the expected discounted cumulative reward received for any state. Here, a brief introduction to the algorithms of TD(0) for estimating V, Sarsa (on-policy TD control) for estimating Q, and Deep Q network (DQN) is presented.

TD(0) employs a technique known as bootstrapping, in which it uses an existing estimate for updates. At each step, TD(0) takes an action and observes the reward and subsequent state, which updates the value of the current state, i.e., $V(S) \leftarrow V(S) + \alpha[\mathcal{R} + \gamma V(S') - V(S)]$. TD(0) also utilizes an exploration parameter, where it controls the probability of taking the action given by the policy trained so far or performs a random action as part of the exploration process typical in RL.

Sarsa, instead, considers transitions from state-action pair to state-action pair and learns their Q values. Unlike TD(0), which updates the V values, it updates Q values by $Q(S, A) \leftarrow Q(S, A) + \alpha[\mathcal{R} + \gamma Q(S', A') - Q(S, A)]$.

Similar to TD(0), it utilizes an exploration parameter to select actions based on Q values or to perform random exploration.

DQN uses a convolution NN to approximate Q values. It is known that RL is unstable when an NN (nonlinear function approximator) is used to represent the Q function, but DQN addresses this issue by using a replay buffer that randomizes the data and an iterative update to reduce correlations with the target.

These are the traditional and well-known RL algorithms, and the readers may refer to the original document for more details[27,28].

This paper also does not aim to discuss the diverse advanced RL algorithms, as this work aims to demonstrate the compatibility of the method with any RL algorithms.

4. METHODS

A high-level overview of our proposed method, ATSRL, is shown in Figure 1.

The first block provides a visualization of the initialization process, the recurring second and third blocks represent the core procedure for selecting which models to train, and the final block concludes the procedure by inputting the test data into the various trained model candidates.

Initialization We assume there is a Teacher NN $\mathcal{N}_T$ trained from a dataset that is no longer accessible in the current environment due to legal constraints or technical limitations. However, the features and the network structures are known. Thus, we initialize n student NNs identical to the teacher's model, where n is the number of parameter units that we apply freezing or pretrain-finetuning to. These parameter units can be a single neuron, a layer, or multiple layers of parameters appropriate for the specific structure of the network. For each student $\mathcal{N}_{Si}$, we copy the parameter units from 0 to i from the teacher $\mathcal{N}_T$.

Select a Student We use the following MDP definitions to represent the RL selector agent:

1. $\mathcal{S}$: The current student $\mathcal{N}_{Si}$ being trained
2. $\mathcal{A}$: $\{i + 1 \text{ or } i - 1\}$
3. $\mathcal{R}$: f_1 score of $\mathcal{N}_{Si}$ given a batch of Train Data D_t
4. $\mathcal{T}$: $\text{Prob}(s'|s = \mathcal{N}_{Si}, a = i \pm 1)$
5. γ: 1.0

Our RL selector agent treats the selection process as an RL mission. Each state $s \in \mathcal{S}$ represents the current student being selected for training. A value (V or Q, depending on the implemented RL algorithm) provides the estimated cumulative reward the state can achieve. Given the updated value table, the selector agent chooses an action $a \in A$ that decides what should be the next state, i.e., the next student to be trained. The next student is either the proceeding or the succeeding student with respect to the current student, depending on ± 1. The

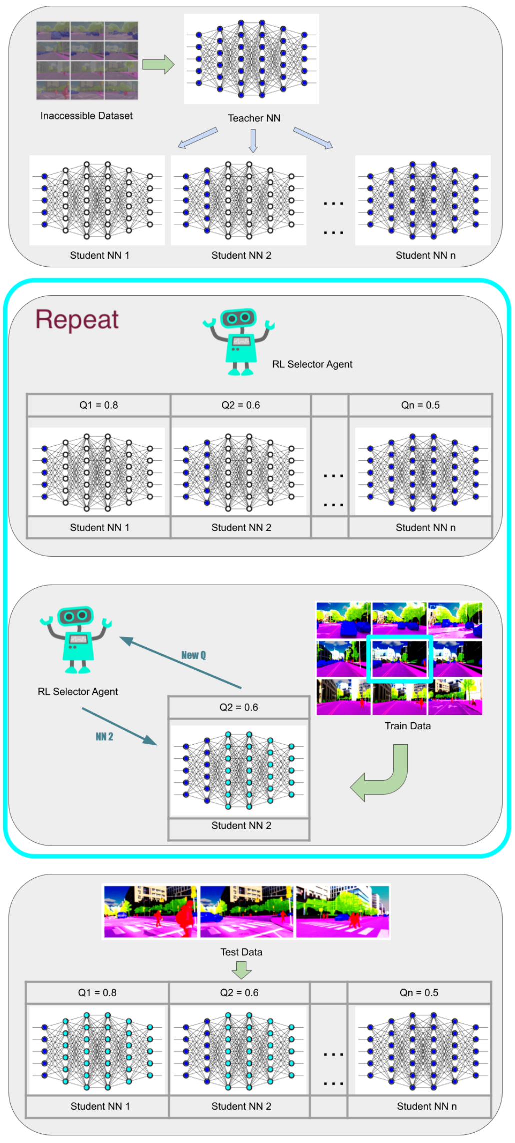

Figure 1. A visualization of the procedure.

reward is the f_1 score the student receives from training a batch of data from the training dataset D(the tth batch).

The transition function is set to be deterministic, and the decay rate is 1.0, i.e., a decay over longer episodes is not considered because it does not affect the outcome.

Update a Student After the selector agent enters the current state, i.e., selects which student N_{Si}, the student NN is updated. For example, in the third block of Figure 1, student 2 is selected. It is given a small batch of training and validation data from the entire training dataset. The network of student 2 could either freeze the first two parameter units or apply pretrain-finetuning to them.

Regardless, the remaining parameter units are updated, and thus, with the f_1 score (or other meaningful statistics), the selector agent also updates the corresponding value of student 2.

Then, the selector agent repeats the process to find the next student to train.

Test all Students After the RL selector agent has completed the training for each of the student networks, all the students are tested with the testing data.

The best model among all the candidates is then selected as the final output student model.

Complexity Analysis The time complexity of the ATSRL method can be mainly attributed to the number of student networks and the training process for each of these networks. In the worst-case scenario, if we consider each student network to be trained independently, the time complexity is proportional to $O(n \times T)$, where n is the number of student networks, and T is the time required to train each network. However, the actual runtime could be considerably less, as the process of training student networks may be optimized by transfer learning from the teacher network and from each other, and only the most promising student is trained fully with partial training data. The space complexity, on the other hand, depends on the number of student networks and the size of each network and can be approximated as $O(n \times M)$, where M is the memory requirement for each network.

f_1 **Score Briefing** The f_1 score is a crucial statistic in binary classifications, representing a balance between precision (the ratio of true positives to all predicted positives) and recall (the ratio of true positives to all actual positives). Calculated as the harmonic mean of precision and recall, the f_1 score ranges from 0 to 1, with 1 indicating perfect precision and recall and 0 showing a total absence of either. Though other statistics can be considered as needed, depending on the characteristics of the data, the f_1 score is most widely utilized in real-world datasets.

Algorithm 1 Automatic transfer selector via reinforcement learning (ATSRL)

```
Input: Train Data $D = \{D_1, D_2...D_m\}$ where $m$ is the number of fold $D$ is split, Teacher NN $\mathcal{N}_T$
Initialize $V(s_i)$ or $Q(s_i, a)$, $s_{t0}$, $f_1\_best$, Student NN $\mathcal{N}_{Si}, i \in \{1, 2...n\}$
                                        ▹ with $1, 2...i$ parameter units copied from $\mathcal{N}_T$ to $\mathcal{N}_{Si}$
(optional) lock $1, 2...i$ parameter units of $\mathcal{N}_{Si}$
while $D_t \neq \phi$ do
    $D_t \leftarrow D_j \in D$
    $f_1 \leftarrow$ Train$(\mathcal{N}_{St}, D_t)$
    Update $(Q(s_t), f_1)$
    if $f_1 > f_1\_best$ then
        save $\mathcal{N}_{St}$
    end if
    $a_t \leftarrow \max_a Q(s_t, a)$
    $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$
end while
```

The pesudocode is presented more formally in Algorithm 1. The inputs of the algorithm include the training data $D = \{D_1, D_2...D_m\}$ and the teacher network $\mathcal{N}_T$.

The training data D is divided into smaller batches, each used for fine-tuning in each step.

The current state s_i represents the current student network $\mathcal{N}_{Si}$ being trained, requiring respective initializations of their values.

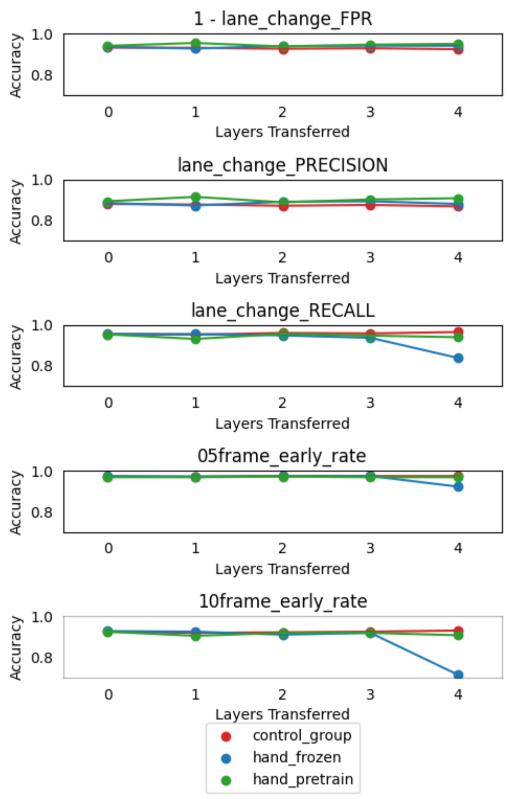

Figure 2. Compare freezing and pretraining residual blocks by hand.

s_{t0} denotes the initial student selected, and f_1_best is a list of the best observed f_1 scores or other relevant statistics for each student.

Besides, parameter units are transferred from $\mathcal{N}_T$ to $\mathcal{N}_{Si}$.

At each time step, a batch from the training data is selected.

Repeated traversals over the data are permissible, as student models may encounter the same batch of data multiple times. The selected student is trained with this data batch, and the resulting f_1 score is recorded. This f_1 score enables the update of the current state's value.

If the score outperforms previously observed scores for this state, this model $\mathcal{N}_{S_t}$ is saved. Finally, the action and the next state are selected based on the current value and transition.

5. EXPERIMENT RESULTS

The data utilized for the experiment is a month's worth of truck driving data from the United States, specifically related to the cutting-in scenario. This data is divided into training + validation and testing datasets at a ratio of 0.8 : 0.2.

The input data consists of a sequence of 21 features of length 10, processed for the cutting-in scenario. The output data is a binary integer, indicating whether or not cutting-in occurs.

The temporal convolutional neural network (TCN)[29] served as the prediction model. TCN creates a hierar-

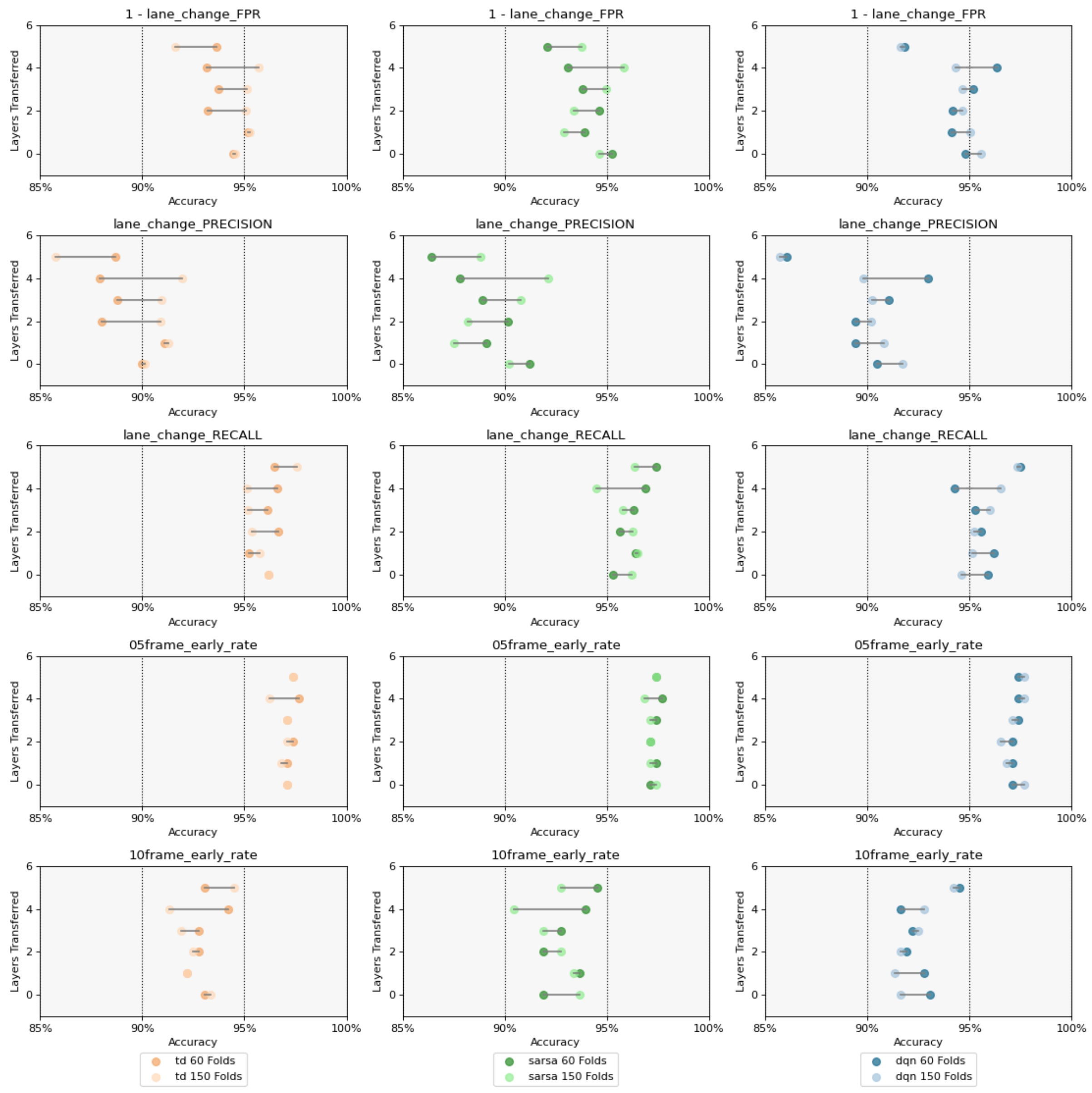

Figure 3. Compare RL variations with 60 folds and 150 folds of data to train.

chical structure that integrates temporal features into a CNN.

Each internal node of the tree-like structure is a residual block, composed of multiple layers of dilated convolution, activation, and normalization. Due to this structure, the initial consideration in the experiment was to freeze or pretrain residual blocks instead of individual layers.

The TCN used in this case has a depth of 4, signifying the presence of 4 residual blocks plus extraneous parameters that could be selected as the transfer target. A model was trained using the training + validation data and was denoted as ScratchUS. Additionally, a prediction model trained in China was obtained. Although the features were identical, the data from China was inaccessible for the experiment.

This model was referred to as TeacherCN. The goal was to leverage both TeacherCN and the existing US

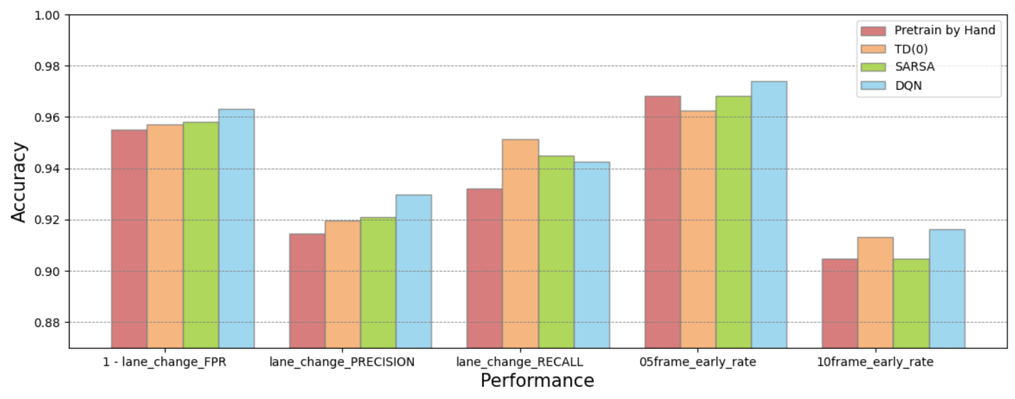

Figure 4. Compare best RL variations. RL: reinforcement learning.

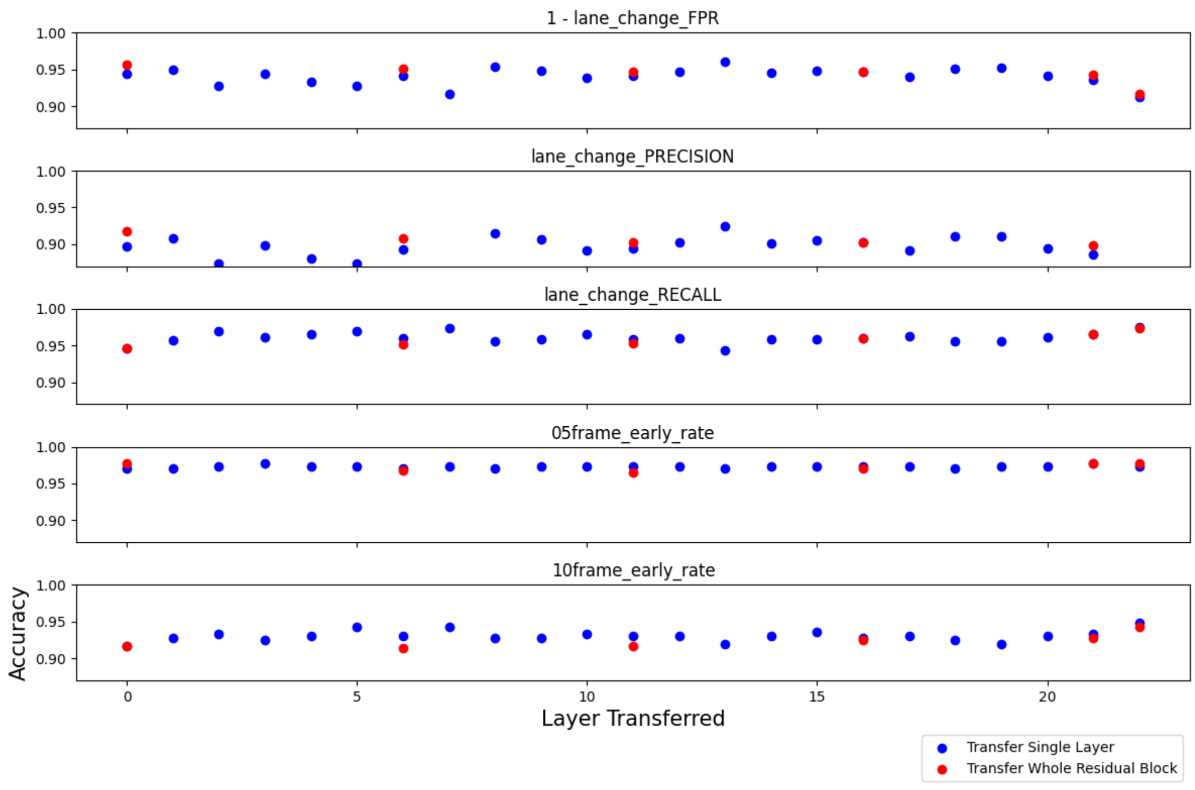

Figure 5. Layer level transfer.

training + validation data to create a new transferred model that would surpass ScratchUS in performance on the testing data.

The chosen metrics were the five most important statistics in self-driving prediction:

1. Lane Change False positive Rate (FPR): FPR of accurately predicting the lane change
2. Lane Change Precision: precision of lane change
3. Lane Change Recall: recall of lane change
4. 05 Frame Early Rate: prediction correctness using the 5 latest frames

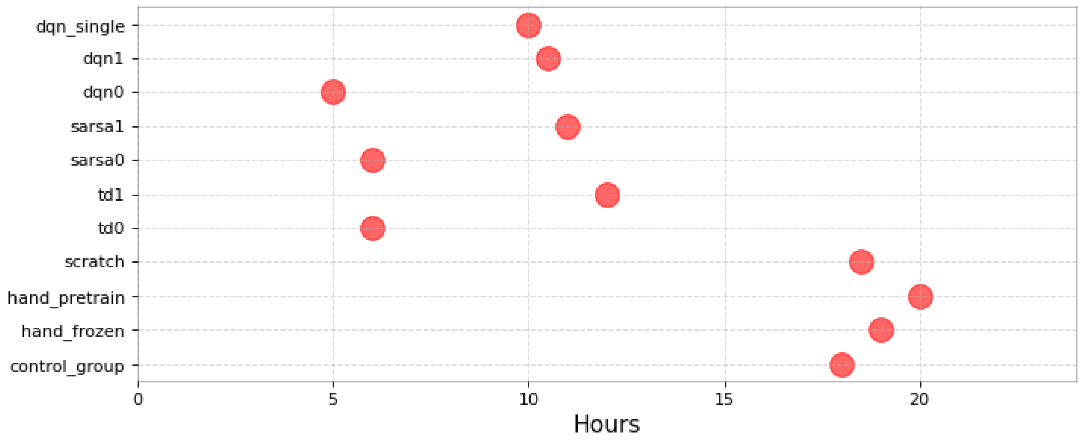

Figure 6. Runtime of the experiments.

5. 10 Frame Early Rate: prediction correctness using the 10 latest frames

FPR is a critical statistic in the context of lane change predictions in autonomous driving. It measures the percentage of instances in which a lane change is incorrectly predicted; that is, when the prediction system inaccurately indicates that a lane change will occur when, in fact, it does not. Minimizing FPR is crucial because an excessive number of false alarms can lead to unnecessary or inappropriate responses from the vehicle, potentially compromising safety and operational efficiency.

All subsequent experiments were conducted on the same testing data, utilizing the above statistics. A total of 16382 data points were used for testing. For convenience in visualization, 1 - FPR was displayed instead of FPR. These statistics are empirically informative for the other teams in the general planning-control-prediction-fuel-economy group at PlusAI and are thus more meaningful than other indicators such as validation loss.

5.1. Traditional network transfer by hand

The manual freezing and pretraining of residual blocks were first examined. The result is shown in Figure 2. The control group (red dots) utilized the model directly trained by the training set of data (ScratchUS) to investigate potential co-adaptive features.

The results revealed that as residual block 0, residual block 01, residual block 012, residual block 0123, and all parameters were gradually frozen, the red dots remained nearly unchanged.

This suggested limited co-adaptive features that might impact the result. Subsequently, TeacherCN was used as the teacher model, and the same manual freezing procedure was repeated (blue dots). Almost every freezing procedure remains unchanged except when all parameters are frozen (with x index at 4). In this case, it was inferred that all parameters of residual blocks are transferable except for the remaining parameters.

This further suggested that TeacherCN was not directly applicable to the data collected in the US, as the lane change recall and 10 frame early rate were extremely poor.

Manual pretraining with TeacherCN (green dots) served as a sanity check. As expected, it performed slightly better than the control group and manual freezing of residual blocks.

Notably, pretraining and finetuning with all parameters again proved to be an exception (with x index at 4), which could also be inferred from the blue dots. For other transfer options, performance was slightly better than the control group.

The rest of the experiment aimed to enhance this advantage using the proposed methods. Each point on the plot represented an independent training procedure, with the training + validation data divided into 3 folds, each taken by each training. Fifty episodes were assigned for each fold to ensure convergence. All training procedures used the same 3 folds.

5.2. RL variations trained over time

Figure 3 illustrated how five important statistics changed over time, from iterating 60 folds of data to 150 folds. The entire training + validation data was divided into 30 folds, representing the performance for 2 and 5 iterations of traversing the entire data. As the number of folds increased, fewer data entries were contained in each fold, and convergence time was accordingly shortened to 30 episodes for each fold. Moreover, in contrast to providing the same 3 folds to all the training procedures, only the model represented by the current state was given one of the 30 folds of data and trained to convergence at a time. Three RL algorithm variations were experimented with: TD(0) (in orange), SARSA (in green), and DQN (in blue). Similar to Figure 2, but reversed, the y-axis displayed the transferred residual blocks, and the x-axis presented the accuracy. Darker colors represented 60 folds, while the lighter colors represented 150 folds. For the lane change FPR, regardless of what residual blocks were transferred, DQN clearly performed better than the other two. Increasing the number of folds did not necessarily improve the FPR.

For TD and SARSA, the difference was more pronounced. This was also similar for lane change precision, where DQN outperformed the other two and showed smaller improvement over time. This inferred that DQN trained the fastest and best, and no additional folds were necessary.

For the remaining three statistics of lane change recall, 05 frame early rate, and 10 frame early rate, DQN behaved in a general manner. Small variances were observed among the specific residual blocks selected for transfer according to those statistics. Interestingly, though SARSA selected actions based on the estimation of Q values, it did not demonstrate more advantage than TD(0) where the selection process was largely random.

It could be concluded that DQN performed better than the other two variations, but SARSA did not seem to be more advantageous as a general observation. However, the best model trained by SARSA outperformed TD(0), as will be shown in the next sub-section.

5.3. Best models of RL variations

From Figure 2 and Figure 3, it might be difficult to define what constitutes the best model since there were five statistics of interest.

A high early rate might also correlate with a high FPR rate, making it challenging to decide which one should be sacrificed for the trade-off.

However, from Figure 3, it was also observed that pretraining and finetuning with residual blocks of 0123 resulted in the lowest FPR, highest precision, and 05 frame early rate, while at the same time, the other two statistics were within the top 3. This observation was consistent with Figure 2, though the advantage was small.

It was inferred that the algorithm designed with RL agents was capable of pointing out the best candidate as well.

In this case, transferring residual blocks of 0123 was treated as the best, and its RL variations and manual pretrain were compared in Figure 4. Notably, DQN (in blue) was found to be optimal as it achieved the highest number among four in all statistics except for lane change recall.

Yet, it was still better than manual pretraining selection.

TD(0) and SARSA showed alternate advantages over each other, generally doing a better job than pretraining manually.

One possible explanation was that multiple small batches allowed for better finetuning. However, the primary focus was not on surpassing the manual pretraining but on improving the auto-selection process.

In summary, the DQN transfer model with residual blocks of 0123 was retained as the best and further compared with a layer-level selection model in the next sub-section.

5.4. Layer-level transfer

Up until now, pre-training and finetuning were performed with the parameters of entire residual blocks - one reason was due to the structure of TCN, and the second was that traditional manual pretraining and finetuning made it too laborious to approach optimal transfer by hand. The results in the previous subsections validated the methodology, leading to an exploration of whether pre-training and finetuning with parameters of single layers could yield even better results. Additionally, the number of folds was increased to 50, while the episode of each fold was decreased to 20 to achieve convergence. Similarly, each time, only the model with specific layers transferred was trained. In line with transferring by residual blocks, layer-level transfer referred to transferring parameters of the 1st, 1st + 2nd, 1st + 2nd + 3rd, ... until all the parameters. Therefore, with some layers transferred, they were exactly the same as transferring the residual blocks as previous. In the results shown in Figure 5, the x axis referred to layers transferred, and the y axis showed the accuracy. The blue dots and red dots denoted transferring the single layer or whole residual blocks, respectively, where they had the same x axis, which meant the corresponding residual blocks were composed of the layers. Most red dots were close to the blue dots, representing the same layers to transfer. To clarify, there was no interest in boosting the corresponding blue dots to perform significantly better than the red ones. It was anticipated that the slight difference was due to training variance. The goal of this experiment was to find out if there were candidate layers that could be transferred and outperform the existing performance, i.e., could any blue dots be found that were better among all candidates of blue and red dots.

The answer was affirmative, as readers could easily observe that red dots were never the optimal among the five statistics. Therefore, with the proposed algorithm, it was claimed that the selection became more efficient for this transfer model.

5.5. Runtime comparison

A comparison of the runtime of the experiments was also performed, as shown in Figure 6.

One significant challenge with manual pretraining and finetuning is its labor-intensive nature.

Selecting with RL agents can save time because the advantage of NNs is that finetuning a small amount of data can significantly enhance performance.

With less data, training converges faster, and the use of various batches of data prevents overfitting.

It is worth noting that the result was empirical, with statistics provided in the previous sections. In this comparison, "dqn0" and "dqn1" referred to the DQN selector with 60 and 150 folds, respectively, and the same applied to "sarsa" and "td". The result showed that with the same amount of data used for training (5 iterations traversing the entire training + validation data, with various batch sizes), the DQN selector with the single layer transferred was the most efficient.

6. CONCLUSION

This paper introduced innovative methods of ATSRL applied to the self-driving prediction model domain. There was a successful use of a network trained in China, despite no access to the China data, to assist the learning of a prediction network for US data. This process proved more efficient and effective than manual training, and it also surpassed pre-training and fine-tuning conducted solely using US data.

Future work could expand the current scenario to include not only cut-in but also cut-out and merge. The claim was also made that the algorithm is not limited to the temporal convolutional network. While it is the most recent module in the company infrastructure and hence served as the test bed, future plans include extending to newer models as they become available and conducting further comparisons. Furthermore, plans to perform experiments with a teacher model either from the US or China are in place to assist students with a small amount of German data. This will likely prove to be a more challenging task given the differences in traffic rules and culture between Germany and both China and the US.

The focus of this work was largely practical and aimed at improving the existing architecture. Thus, an extensive comparison with a broad range of network-based transfer learning methods was not performed. Additionally, the most advanced RL algorithms were not considered in the study, as the aim was to validate the practicability of the current approach. These represent the limitations and two key areas for further exploration in future work.

7. PATENTS

Notice of Allowance (US) has been received in Spring 2023.

The study is accompanied by a patent application in the self-driving domain. As the patent has been approved, it can limit the scope for introducing more experiments or utilizing the most advanced RL algorithms for transfer in the current work. Thus, further comparisons with benchmarks and incorporating advanced RL algorithms will be considered for future research endeavors. This approach will ensure the patent application process is not disrupted while continuing to advance and refine the work.

DECLARATIONS

Acknowledgments
The authors would like to thank Dr. Joseph Campbell and Dr. Simon Stepputtis who are postdoc fellows in Advanced Agent-Robotics Technology Research led by Prof. Katia Sycara for their insights and discussions.

Authors' contributions
Made substantial contributions to conception and design of the study and performed data analysis and interpretation: Guo Y
Performed data acquisition, as well as provided administrative, technical, and material support: Wang Y, Yang IH, Sycara K

Availability of data and materials
The data is owned by PlusAI, Inc. where the first author performed her internship project on self-driving.

Financial support and sponsorship

This work was supported by PlusAI and by DARPA award HR001120C0036 and by AFRL/AFOSR award FA9550-18-1-0251 and AFOSR FA9550-18-1-0097 and ARL W911NF-19-2-0146.

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Tan C, Sun F, Kong T, et al. A Survey on Deep Transfer Learning. In: Artificial Neural Networks and Machine Learning – ICANN 2018. Springer International Publishing; 2018. pp. 270–79. DOI
2. Dai W, Yang Q, Xue GR, Yu Y. Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning. ACM; 2007. DOI
3. Gupta S, Bi J, Liu Y, Wildani A. Boosting for regression transfer via importance sampling. *Int J Data Sci Anal* 2023. DOI
4. Yao Y, Doretto G. Boosting for transfer learning with multiple sources. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE; 2010. DOI
5. Xu Y, Pan SJ, Xiong H, et al. A Unified Framework for Metric Transfer Learning. *IEEE Trans Knowl Data Eng* 2017;29:1158–71. DOI
6. Zhang J, Li W, Ogunbona P. Joint geometrical and statistical alignment for visual domain adaptation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017. DOI
7. Pan SJ, Tsang IW, Kwok JT, Yang Q. Domain adaptation via transfer component analysis. *IEEE Trans Neural Netw* 2011;22:199–210. DOI
8. Long M, Cao Y, Cao Z, Wang J, Jordan MI. Transferable Representation Learning with Deep Adaptation Networks. *IEEE Trans Pattern Anal Mach Intell* 2019;41:3071-85. DOI
9. Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks. *Commun ACM* 2020;63:139-44. DOI
10. Tzeng E, Hoffman J, Saenko K, Darrell T. Adversarial discriminative domain adaptation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017. DOI
11. Ganin Y, Ustinova E, Ajakan H, et al. Domain-adversarial training of neural networks. In: Domain Adaptation in Computer Vision Applications. Springer International Publishing; 2017. pp. 189–209. DOI
12. Tzeng E, Hoffman J, Darrell T, Saenko K. Simultaneous deep transfer across domains and tasks. In: 2015 IEEE International Conference on Computer Vision (ICCV). IEEE; 2015. DOI
13. Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? *Advances in neural information processing systems* 2014;27. Available from: https://proceedings.neurips.cc/paper_files/paper/2014/hash/375c71349b295fbe2dcdca9206f20a06-Abstract.html. [Last accessed on 27 Aug 2023]
14. George D, Shen H, Huerta EA. Classification and unsupervised clustering of LIGO data with Deep Transfer Learning. *Phys Rev D* 2018;97. DOI
15. Oquab M, Bottou L, Laptev I, Sivic J. Learning and transferring mid-level image representations using convolutional neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 2014. DOI
16. Long M, Zhu H, Wang J, Jordan MI. Unsupervised domain adaptation with residual transfer networks. *NeurIPS* 2016;29. Available from: http://ise.thss.tsinghua.edu.cn/~mlong/doc/residual-transfer-network-nips16.pdf. [Last accessed on 27 Aug 2023]
17. Zhu H, Long M, Wang J, Cao Y. Deep Hashing Network for Efficient Similarity Retrieval. In: Proceedings of the AAAI Conference on Artificial Intelligence 2016 mar;30. DOI
18. Zhang W, Deng L, Zhang L, Wu D. A survey on negative transfer. *IEEE/CAA J Automa Sin* 2023;10:305–29. DOI
19. Huisman M, Van Rijn JN, Plaat A. A survey of deep meta-learning. *Artif Intell Rev* 2021;54:4483–541. DOI
20. Rezende DJ, Mohamed S, Danihelka I, Gregor K, Wierstra D. One-shot generalization in deep generative models. In: International conference on machine learning. PMLR; 2016. pp. 1521–29. Available from: http://proceedings.mlr.press/v48/rezende16.pdf. [Last accessed on 27 Aug 2023]
21. Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning. PMLR; 2017. pp. 1126–35. Available from: http://proceedings.mlr.press/v70/finn17a/finn17a.pdf. [Last accessed on 27 Aug

2023]
22. Campbell J, Guo Y, Xie F, Stepputtis S, Sycara K. Introspective action advising for interpretable transfer learning. In: Conference on Lifelong Learning Agents; 2023. Available from: https://arxiv.org/pdf/2306.12314.pdf. [Last accessed on 27 Aug 2023]
23. Jang Y, Lee H, Hwang SJ, Shin J. Learning what and where to transfer. In: International Conference on Machine Learning. PMLR; 2019. pp. 3030–39. Available from: http://proceedings.mlr.press/v97/jang19b/jang19b.pdf. [Last accessed on 27 Aug 2023]
24. Chang H, Han J, Zhong C, Snijders AM, Mao JH. Unsupervised Transfer Learning via Multi-Scale Convolutional Sparse Coding for Biomedical Applications. *IEEE Trans Pattern Anal Mach Intell* 2018;40:1182–94. DOI
25. Baker B, Gupta O, Naik N, Raskar R. Designing neural network architectures using reinforcement learning. In: International Conference on Learning Representations; 2016. Available from: https://arxiv.org/pdf/1611.02167.pdf. [Last accessed on 27 Aug 2023]
26. Shang Y. Feature-enriched core percolation in multiplex networks. *Phys Rev E* 2022;106:054314. DOI
27. Sutton RS, Barto AG. Reinforcement Learning: An Introduction. *IEEE Trans Neural Netw* 1998;9:1054-54. DOI
28. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015;518:529-33. DOI
29. Bai S, Kolter JZ, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:180301271* 2018. Available from: https://arxiv.org/pdf/1803.01271.pdf. [Last accessed on 27 Aug 2023]