A Transferable-experience-based Routing Mechanism for Named Computing Power Network

Yakai Dai, Yi Zhu

***Abstract*—Computing-aware routing is the core technology of Computing Power Network. Through independently determining the forwarding target in each router, computing-aware routing aims at scheduling the computing task to an optimal Computing Service Provider (CSP). However, current computing-aware routing schemes only depend on the collected computing and network resource information, do not consider the experience of CSPs in handling similar tasks, resulting in excessive invocation time. Facing this problem, this paper proposes a novel routing mechanism based on transfer learning experience in named computing power network (NRS-TLE). NRS-TLE introduces the accuracy of trained classification models as the experience of CSP, and design a tcce-function to evaluate the task completion capabilities of different CSPs at NDN routers. Based on the evaluation results of the tcce-function, NDN routers further perform a probabilistic routing scheme. The simulation results show that, comparing with traditional routing scheme SAC-NSGA-II, the computing service invocation time under NRS-TLE will decrease by a minimum of 26.42% and a maximum of 40.22%.**

***Index Terms*—Computing power network, Named data networking, Computing-aware routing, transfer learning.**

I. Introduction

With the advent of the intelligent era, the explosive growth of intelligent applications and devices bring unprecedented traffic pressure of requesting computing power to current Internet. Facing this trend, since 2019, the concept of Computing Power Networks (CPN)[1] is emerged, which aims at improving current network to effectively deal with the soaring computing invocation traffic. To implement CPN, Named Data Networking (NDN)[2] is regarded as a potential solution. For NDN, named addressing is the core difference from IP architecture. Benefitted from this special addressing mechanism with semantic information, NDN router can realize flexible computing service discovery in network layer, and then match the request of invoking computing service with the available computing service provider (CSP) through adopting computing-aware routing mechanism.

Obviously, to achieve effective computing resource allocation and scheduling by NDN router, how to optimize the computing-aware routing is a key problem. For the existing computing-aware routing schemes, the decision of optimal forwarding target is mainly depended on the awareness of network and computing factors, including link available bandwidth to each CSP[3][4][5], link congestion level to each CSP[3][4][5], free computing resource of each CSP[4][5][6][7][8][9], current computing tasks queue length of each CSP[5], and the computational overhead of requested computing service[4], etc. But under some scenarios, especially for the machine learning task with transfer learning support, only considering these explicit factors is not enough, the CSP historical experience of processing the similar task will also affect who is the optimal executor. As shown in Fig.1, three CSPs all provide the image classification model training service, they own the same total computing resource and the same link quality, but they have different free computing resource and different historical experience of training this classification model. Where, we assume that (1) the occupied computing resource of CSP-A, CSP-B and CSP-C is 50%, 20%, 75%, respectively. (2) CSP-A, CSP-B are the freshman to this classification model training task, if they accept the task, they need to retrain the model; CSP-C has performed the similar tasks before and owned the transferable model, if it accepts the task, the model training time is about 90% of retraining using transfer learning. Now, when the user offloads an image lassification model training task to the network, if only considering the network status and available computing resource, the router will prefer to forward the request to CSP-B. But, in fact, the better selection is to forward this request to CSP-C. Although CSP-C is busier than CSP-B, CSP-C can complete this task using less time than CSP-B due to it owns transferable model. From above sample, we can find that, for some model training offloading scenarios, the reasonable computing-aware routing scheme not only depends on the awareness of network and computing resource information, but also need to evaluate the historical experience of processing similar task.

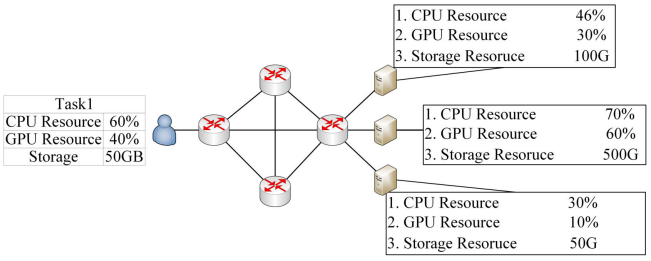


Fig. 1. An example of invocation under the current computing power routing

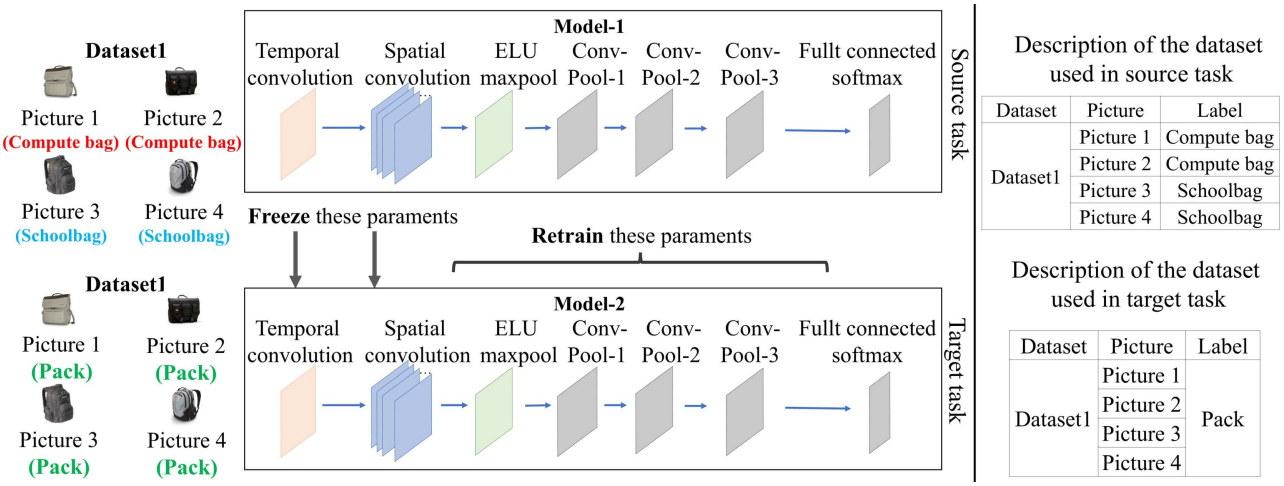
Motivated by the development of transfer learning, we focus on the typical transfer learning scenario of classification model training and then propose a novel computing-aware routing scheme over NDN in this paper, which is named as Named Routing Scheme based on Transfer Learning Experience (NRS-TLE). In NRS-TLE, the historical trained model

Manuscript received April 26, 2024; revised January 25, 2025.

This paper is supported by Future Network Scientific Research Fund Project of Jiangsu Province (FNSRFP-2021- YB-49) and National College Students' Innovation and Entrepreneurship Training Program (202310299056Z).

Yakai Dai is an undergraduate student of the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, 212013, China. (e-mail: 1619313299@qq.com)

Yi Zhu is a professor of the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, 212013, China. (e-mail: zhuyi@ujs.edu.cn)



Dataset	Picture	Label
Dataset1	Picture 1	Compute bag
	Picture 2	Compute bag
	Picture 3	Schoolbag
	Picture 4	Schoolbag

Dataset	Picture	Label
Dataset1	Picture 1	Pack
	Picture 2	
	Picture 3	
	Picture 4	

Fig. 2. A simple example about model based transfer learning

accuracy is regarded as an experience. Through introducing the average task queue length and historical experience of CSP, a task completion capability evaluation function (*tcce-function*) is designed for NDN router. Once a classification model training request Interest is received, NDN router will search the available CSPs according to the semantic information carried in the Interest, then evaluate all available CSPs using *tcce-function*. The evaluation results will determine the forwarding probability of each CSP. The simulation results show that, comparing with traditional routing scheme SAC-NSGA-II, the computing service invocation time under NRS-TLE will decrease by a minimum of 26.42% and a maximum of 40.22%.

The contributions of this paper are as follows:

(1) We introduce the accuracy of trained classification models as the experience of CSP, and then design a *tcce-function* for evaluating the task completion capabilities of different CSPs at NDN router. For the model with higher accuracy, it can be better transferred to similar classification task, resulting in better task completion capability of deployed CSP.

(2) Based on the design of *tcce-function*, we propose a probabilistic routing scheme. This scheme will forward the received Interest according to the *tcce-function* results of all available CSPs. Higher *tcce-function* result is, higher forwarding probability is. Due to that our scheme involves the historical experience of CSP, it can provide a more reasonable forwarding decision.

The rest of this paper is organized as follows: Section 2 introduces the model-based transfer learning and named computing service invocation process. In section 3, we first present the network model and relevant parameter settings, then describe the design and running of NRS-TLE in detail. Section 4 shows the simulation results using ndnSIM. Finally, Section 5 concludes the work of this paper.

II. Background

2.1 Model-based Transfer Learning

The model-based transfer learning[12][13][15] is an important branch in the field of current transfer learning. According to the dataset distribution, it can be divided into two categories: (1) the dataset distribution of source task and the target task is the same; (2) the dataset distribution of source task and the target task is different. For the former, the trained model of source task can be directly transferred to the target task. For the latter, an adaptation layer needs to be inserted into the trained model of source task before transferring, which makes the dataset distribution of the source task and target task closer. Due to that the methods and applications of category (1) are relatively mature, we select the classification task under the scenario of category (1) as discussed computing service in this paper.

Fig.2. shows two transferable classification tasks and the transfer processing. Where, the transferable classification task is defined as deep classification model with the same dataset but different labels, the processing method is finetune[14]. As shown in Fig.2, the source task and the target task adopt the same dataset for training, but they are designed for different classification requirements. For the source task, the Picture 1 and Picture 2 is annotated as Computer bag and the Picture 3 and Picture 4 is annotated as Schoolbag. For the target task, both Picture 1, Picture 2, Picture 3 and Picture 4 are annotated as pack. Now the source task has been trained, we can use the trained model (Model-1) to generate the model of target task (Model-2). Using finetune method, we will freeze the early layers responsible for general feature extraction in Model-1, only retrain the last few layers responsible for feature classification, then obtaining Model-2, which is the desired classification model for the target task. Obviously, transferring the existing model to a new task can greatly save training time and computational resource.

Before performing model transfer, the effectiveness of transferring can be determined by evaluating the accuracy of trained model. Here, model accuracy refers to the proportion of correctly identified samples in the total number of samples.

2.2 Named Computing Service Invocation Process

In its early stages, NDN focused on the Internet scenarios of static content delivery. Recently, with the rapid increasing demand of invoking computing services from Internet, extending the NDN router's capability to support named computing service invocation has been an important trend. Here, the typical design is NSC (Named service call) proposed by

[10], which involving three "Interest-Data" transactions to complete the named computing service invocation process. To describe this typical process, we denote these transactions as <I1-D1>, <I2-D2>, and <I3-D3>.

Step 1, The user initiates the computing service invocation: First, the user sends an Interest (I1) with the requested service name to invoke the computing service. After receiving I1, the NDN router forwards it according to the longest prefix matching rule. Due to the Interest carries the requested service name, the named routing operation is equivalent to the computing service discovery in the network. When a CSP running requested computing service receives I1, it replies a Data packet (D1) to confirm the acceptance of this task. Within the D1, the estimated complete time (denoted as time to complete, TTC) and the computational result name are attached. Where, the TTC is estimated by the feature of requested task, current free resources, transmission delay, etc.; the computational result name is equivalent to the address for fetching the result.

Step 2, The CSP retrieves the input parameters or computational data: After replying D1, CSP further sends Interest (I2) to request the input parameters or computational data of this task. Once the user or any in-network node owning the data receives I2, it replies Data (D2) with required parameters or data as response.

Step 3, The user fetches the result: Since receiving D1, the user starts a local timer according to TTC. Once the timer expires, the user sends an Interest (I3) to request the computing result, the name of I3 is derived from the payload of D1. When CSP receives I3, if the task is completed, it will reply the result as a Data packet (D3); otherwise, it will reply an update TTC to user for notifying the new fetching time.

III. Design of NRS-TLE

3.1 Network model and parameter settings

To explore a more reasonable named routing mechanism to support the scenario of model-based transfer learning, we design NRS-TLE in this paper. The network model of NRS-TLE is as shown in Fig.3, which includes three network entities, including CSP, user, and NDN router.

(1)CSP: The CSP provides the computing services of training classification models. For the arrival request of classification model training task, CSP first determines whether the requested task is transferable locally or not. If the arrival task depends on the same dataset as one local trained classification model but require different labels, CSP performs the model-based transfer learning to generate new model with minimized time and resource costs; if there is no similar classification task performed before, CSP will retrain a model for the arrival task. After completing each training task, CSP compares the accuracy of the new model with the previous stored model, then only keep the model with higher accuracy and the annotation file used to achieve the highest accuracy locally.

(2)User: The user randomly generates classification model training task, then sends an Interest packet with the task name to invoke the training service. Once a CSP accepts the request, the user will further provide the training data and annotation file to CSP, and retrieve the output model from CSP after task completion.

(3)NDN router: The router implements the named routing mechanism. In NRS-TLE, when receiving an Interest of invoking classification model training service, the router will evaluate the task completion capability of all possible CSPs for this Interest, determine the forwarding probability based on the evaluation results, and then execute the probabilistic forwarding.

To clearly describe our scheme, the followings are some settings and symbol definitions involved in NRS-TLE.

(1) Suppose there are N CSPs in the network, where the $i\text{-}th$ CSP is defined as $P_i, 1 \leq i \leq N$;

(2) Each CSP provides M classification model training services. For the $j\text{-}th$ classification model training service, it is denoted as $Task(j), 1 \leq j \leq M$;

(3) To simplify the task processing, we assume that each CSP stores all required datasets for classification model training. So, they only need to request the annotation file when they deal with the task;

(4) For P_i, its average task queue length is denoted as $Len(i)$, its average task complete time is denoted as $Time(i)$. If we further assume that $ACC(i,j)$ is the historical maximum model accuracy of $Task(j)$ trained by P_i, P_i will update $ACC(i,j)$ after completing a new training of $Task(j)$. If P_i never executes the $Task(j)$ before, $ACC(i,j) = 0$;

(5) For each NDN router, it owns K interfaces and the $k\text{-}th$ interface is denoted as $face(k), 1 \leq k \leq K$;

(6) To support the running of NRS-TLE, NDN router maintains a special table, we name it as State Base. As shown in Fig.3, the State Base records the collected awareness information of CSPs, including their forwarding interfaces, their running status and their providing services information; For P_i, if it is probed by current router, the router will record the following information in the State Base: name of P_i, status indicator of P_i, $Len(i)$, $Time(i)$, each providing service of P_i and the corresponding historical maximum model accuracy $ACC(i,j)$;

(7) We revise the "face" column of Forwarding Interest Table (FIB) to "face/CSP" column. As shown in Fig.3, for the FIB entries of classification model training services, "face/CSP" column records their available CSPs' names; for other entries, "face/CSP" column records their available faces;

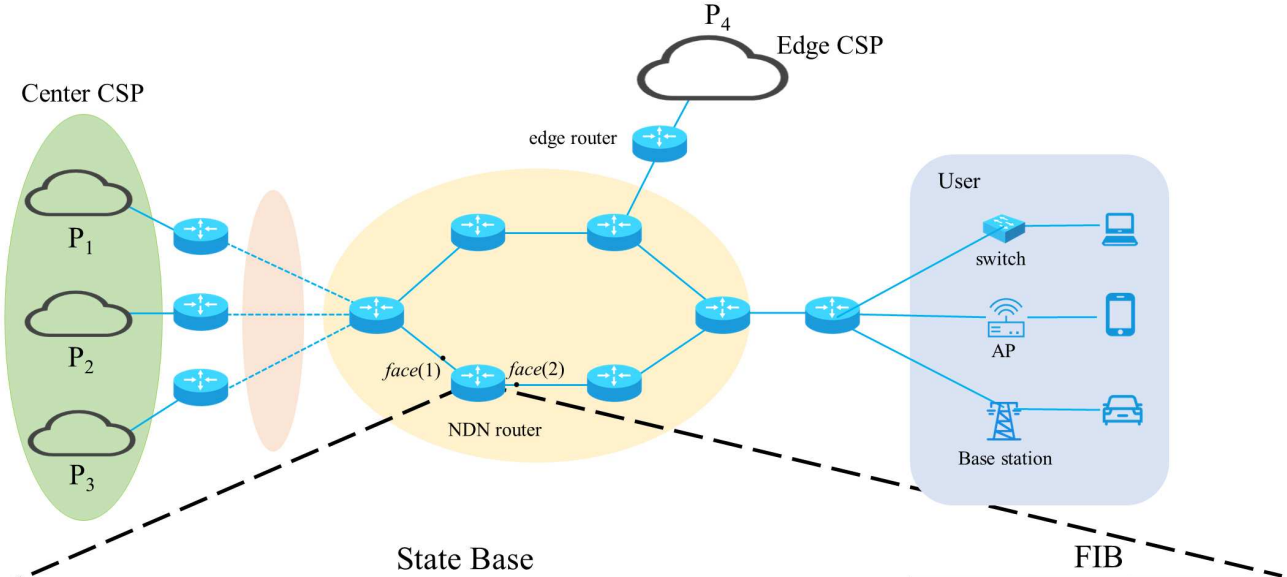
3.2 Description of NRS-TLE

3.2.1 tcce-function: The *tcce-function* is the core of NRS-TLE, which is used to evaluate the capability of P_i to complete $Task(j)$. Now we adopt $f(i,j)$ to denote *tcce-function*, it is calculated by formula

$$f(i,j) = \begin{cases} ACC(i,j) \times \frac{2}{e^{\gamma(\tau(i)-\tau_{expect})}} & \tau(i) > \tau_{expect} \\ ACC(i,j) & \tau(i) \leq \tau_{expect} \end{cases} \quad (1)$$

$$\tau(i) = Time(i) \times Len(i) + Time(i) \quad (2)$$

Where, γ is the discount factor, $0 \leq \gamma \leq 1$; $ACC(i,j)$ is the historical maximum model accuracy of $Task(j)$ trained by P_i; τ_{expect} is the expected complete time of user; τ_{expect} is the total of average queueing time and average execution time at P_i, which is calculated by formula. According to the above design, *tcce-function* both considers the historical



State Base

CSP Name	Status indicator	Face	State			
			Len(*i*)	*Time*(*i*)	*ACC*(*i*, *j*)	Service Name
P_1	seq=1	*face*(1)	2	30min	0.76	/AlexNet/RealWorld
P_2	seq=0	*face*(1)	11	28min	0.68	/AlexNet/RealWorld
P_3	seq=2	*face*(1)	3	26min	0.92	/ResNet/Product
P_4	seq=4	*face*(2)	1	56min	0.56	/ResNet/Product

FIB

Prefix	face/CSP
/AlexNet/RealWorld	P_1
	P_2
/ResNet/Product	P_3
	P_4
/domain/id/n3	*face*(1)

Fig. 3. The network model used in this paper

processing experience of $Task(j)$ at P_i and busy level of P_i. If $\tau(i)$ is less than τ_{expect}, $f(i,j)$ is only determined by $ACC(i,j)$; if $\tau(i)$ is bigger than τ_{expect}, $f(i,j)$ will be decreased with the increasing of $\tau(i)$.

3.2.2 ***Status awareness of CSP:*** The status awareness of CSP is the foundation of implementing NRS-TLE. There are two conventional modes to obtain the running information of CSP under NDN architecture. The first mode involves the "Interest-Data" interaction, where CSP will insert its own status into the replied data packet when responding any Interest packet. But this mode will bring heavy transmission overhead. The second mode [11] adopts status advertisement mode, where CSP will periodically broadcast a special Interest packet to announce its status within the network. Usually, the announcement Interest only carries a simple status indicator. For the NDN router, it will actively request the detailed status of CSP only when the current received status indicator is different from previous received indicator. Due to the second mode works with low overhead, we follow this idea to design a status awareness mechanism for NRS-TLE. The entire process consists of two stages.

Stage 1. Initialization Stage

In this stage, CSP broadcasts its own status by publishing an advertisement Interest named "/compute-service-provider-name/seq=0", where seq represents the status indicator and its initial value is set as 0, compute-service-provider-name is the CSP name. Once NDN router receives this initial advertisement, it will send an Interest named "/Request-Status/compute-service-provider-name/seq=0" to retrieve the status information from advertising CSP. As a response, the CSP will reply a Data packet with its deployed classification model training services' name, the maximum model accuracy of each deployed model, current average task queue length, estimated average completion time of deployed services. When the response Data arrives, the NDN router will create new entries in the local State Base and store the received information.

Stage 2. Update Stage

The CSP periodically publishes status advertisement Interest packet. If its status has changed, the status indicator within the advertisement Interest will be increased by 1 on its previous value. When NDN router receives this Interest, the router will search the entry of State Base according to the CSP name, then compare the received status indicator with the local stored status indicator. If they are the same, it means that the status of CSP has not changed, the router can discard this Interest; otherwise, the router should send Interest with name "/Request-Status/compute-service-provider-name/seq=number1" to retrieve the latest status from advertising CSP.

The Fig.4 shows an example of status awareness process.

3.2.3 ***Running of NRS-TLE:*** Based on the design of *tcce-function* and CSP status awareness mechanism, the NRS-TLE running process can be divided into three steps:

Step 1 User initiates a task invocation: The user sends an Interest named "/ModelName/DatasetName/Classification-Training-Service/[Task=j]/Ask" to request a specific classification model training task, where, ModelName is the target model name, DatasetName is the required dataset name, [Task=j] means the invoked task is $Task(j)$. The payload of this invoking Interest includes two parts, τ_{expect} and the annotation file name used to train the model, e.g.,

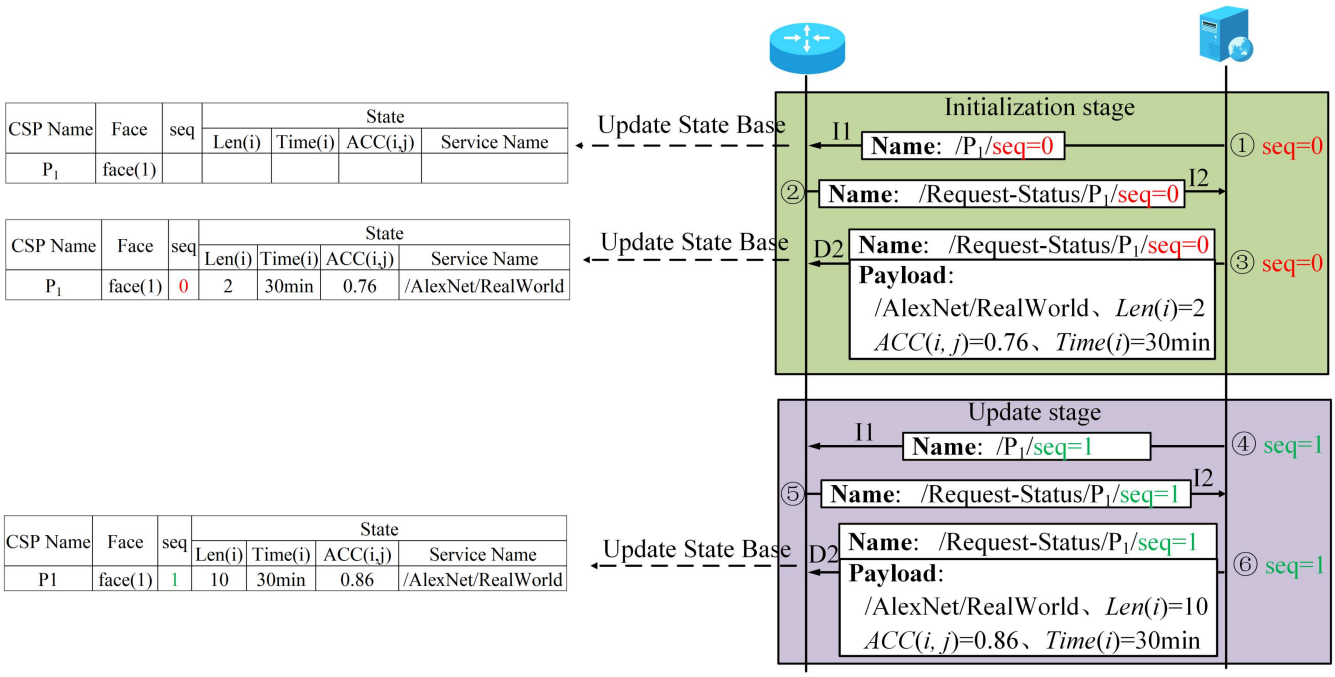


Fig. 4. An example of status awareness process

"/DatasetName/Label/[Task=j]".

Step 2 NDN router implements probabilistic forwarding: After receiving invoking Interest, the NDN router searches the matched entry in FIB based on the longest prefix matching rule and obtains all available forwarding faces or CSPs. If the matched prefix belongs to the classification model training task, the router further searches the State Base according to the obtained CSP name one by one. For each available CSP, the router should calculate the *tcce-function* to evaluate its task completion capability. Then, the router scores each available face by summing all *tcce-function* values from corresponding CSPs. Suppose for the requested task, there is L CSPs recorded in the FIB entry, where the L CSPs are connected by K interfaces, $L \leq N$. If the connected CSP set of the k-th interface is S_k, the score of k-th interface is expressed as formula(3):

$$Score[face(k)] = \sum_{P_i \in S_k} f(i,j) \tag{3}$$

Then, the forwarding probability is determined by formula(4):

$$Probability[face(k)] = \frac{Score[face(k)]}{\sum_{k=1}^{K} Score[face(k)]} \tag{4}$$

Step 3 CSP responses the task: Suppose that the invoking Interest is forwarded to P_i. Once P_i receives the Interest, P_i will firstly compare the local stored model with requested task.

1. If local stored annotation file is the required file used for this training, that means the local stored model is the requested training result. Under this situation, P_i can directly reply the local model to user without repetitive training.

2. If local stored annotation file is not the required file used for this training, P_i must extract the payload "/DatasetName/Label/[Task=j]" from the received Interest, then use the extracted name to build an Interest to retrieve the annotation file from user. After fetching the annotation file, P_i will check the transferability of this classification task.

(1) If the classification model of target dataset is trained before, P_i performs the transfer learning to generate a new model for this task.

(2) If the classification model of target dataset is not trained before, P_i will retain a model for the task.

IV. Performance Evaluation of NRS-TLE

In this section, we will evaluate the performance of NRS-TLE from the aspects of Average Invocation Time, Average Training Quality, Average Task Queue Length, Forwarding Task Number, Occupation Rate of CPU Resource and Forwarding Probability:

1. Average Invocation Time refers to the time from initiating a task request to receiving the computational result. It consists of three components, the network transmission delay, the task queuing time, and the task execution time. In our simulation, we use ndnSIM tool to simulate the network transmission delay and task queuing time. However, for task execution time, it is difficult to derive from ndnSIM. Therefore, we run the classification model training experiment separately on a server (16 core Intel Xeon 8163 CPU and 32G memory), then measure the task execution time.

2. Average Training Quality is the statistical of all obtained accuracy for each model, it is derived from the real training experiment on the server.

3. Average Task Queue Length is the average tasks number of waiting for processing in each CSP, which is used to evaluate the congestion level of CSPs. To derive this indicator, we record the task queue length of each CPS every 200 simulation second from simulation begin to the end, then calculating the average value of all records.

4. Occupation Rate of CPU Resource is the ratio of the CPU resource consumed by different tasks to the maximum

CPU resource of the CSPs. This indicator discloses the usage of computing resource by each type of task under different schemes, and the number of tasks that can be processed by the CSPs in parallel.

5. Forwarding Probability will be get from the Forwarding Task Number. This indicator reveals the probability of a task being forwarded from different interfaces. We will calculate the probability according to the different tasks' numbers forwarded from different interfaces.

6. Forwarding Task Number is the forwarding number of different tasks to each CSP during the simulation. This indicator discloses the forwarding selection of different tasks at router side and is used to evaluate the rationality of different routing scheme.

The comparison scheme adopted in our simulation is SAC-NSGA-II proposed in[4], which is a typical routing scheme based on computing-network resource awareness. In SAC-NSGA-II, the router collects the information of transmission delay (each link) and computational resource utilization (each CSP), then run genetic algorithm to determine the optimal forwarding target. For the CSP under SAC-NSGA-II, it doesn't support transfer learning and only executes the task by retraining the model.

The parameter settings of our simulation are as follows.

1. The simulation topology is shown in Fig.5, including 1 user, 4 CSPs (named as S1, S2, S3, S4), and 3 NDN routers (named as R1, R2, R3). The maximum CPU resource of each CSP is 2.5×10^9cycle/s. For each link, its delay is set as 10ms and its bandwidth is set as 1Gbps.

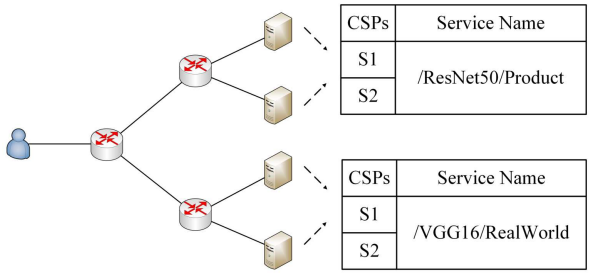


Fig. 5. The simulation topology

2. The datasets used for classification model training experiment are RealWorld[12] and Product[12]. The size of RealWorld dataset is 742MB and the size of Product dataset is 106M. For both two datasets, we set 25 annotation methods[12] respectively. In our experiment, the annotation file names of RealWorld are defined as RealWorld1-RealWorld25; the annotation file names of Product are defined as Product1-Product25.

3. S1 and S2 provide the classification model training service of "/ResNet50/Product", which owns 23.5 million parameters[16]. S3 and S4 provide the classification model training service of "/VGG16/RealWorld", which owns 138 million parameters[17].

4. For the CSP, it only stores the model and annotation file associated with the highest accuracy. That means the local stored model and annotation file will be replaced if the new trained model has higher accuracy.

5. The user follows uniform distribution to select service and annotation file from 50 "service-annotation" combinations, then generates Interest to request the selected classification training task, the request interval is 50 simulation second.

Based on the aforementioned settings, we first run training experiments on the server to obtain the average transfer learning time and retraining time for two types of services, while recording the model accuracy for each "service-annotation" combination under transfer learning. Next, we run network simulation program in ndnSIM to obtain the average transmission delay and the average task queuing time. The network simulation results are obtained by statistical of 1000 random user requests.

The final results of Average Invocation Time are as shown in Fig.6. We can clearly see that the average invocation time under NRS-TLE is obviously lower than that under SAC-NSGA-II. For "/ResNet50/Product" service, the average invocation time of two schemes is 5675.8s and 8405.08s respectively. For "/VGG16/RealWorld" service, the average invocation time of two schemes is 7258.17s and 12609.7s respectively. The reason is that NRS-TLE utilizes the transfer learning method to avoid of retraining the whole model.

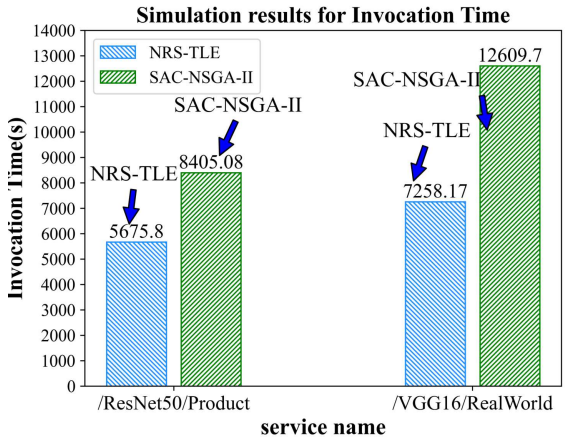


Fig. 6. Simulation results for the Average Invocation Time

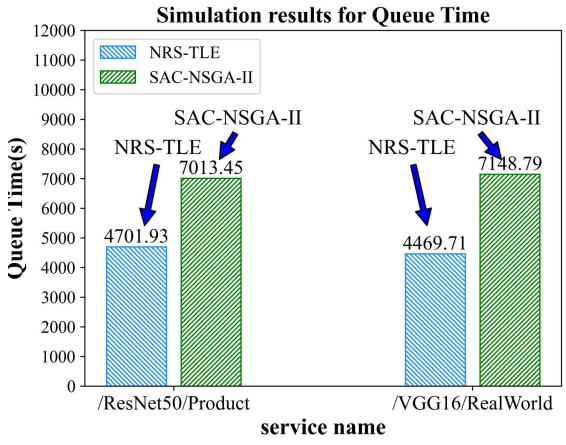


Fig. 7. Simulation results for the Average Queue Time

Fig.7 and Fig.8 shows the detailed components of Invocation Time. Fig.7 shows the detailed queue time under two schemes. Fig.8 further shows the detailed task execution time under two schemes, comparing with "/ResNet50/Product", "/VGG16/RealWorld" service under NRS-TLE can save

more training time, the percentage of time saved is approximately 48.93%. Due to that the parameters required retraining are depended the source model accuracy under transfer learning mode, while NRS-TLE always selects the CSP with higher model accuracy to forward, it can achieve more time benefit when the requested model owning more parameters.

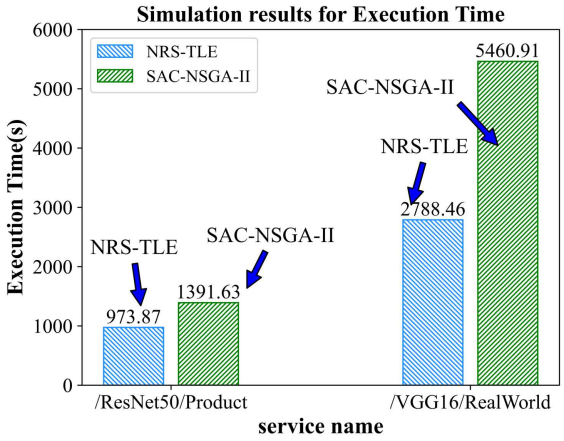


Fig. 8. Simulation results for the detailed task Execution Time

About the Average Training Quality under two schemes, Fig.9 gives the experiment results. As shown in Fig.9, the average trained model accuracies of NRS-TLE and SAC-NSGA-II are similar, but NRS-TLE has a slight and marginal advantage over SAC-NSGA-II. For "/ResNet50/Product" service, the accuracy difference of two schemes is about 1.09%. For "/VGG16/RealWorld" service, the accuracy difference of two schemes is about 2.93%. This further reveals that NRS-TLE can achieve the similar training results as the SAC-NSGA-II scheme in a shorter time.

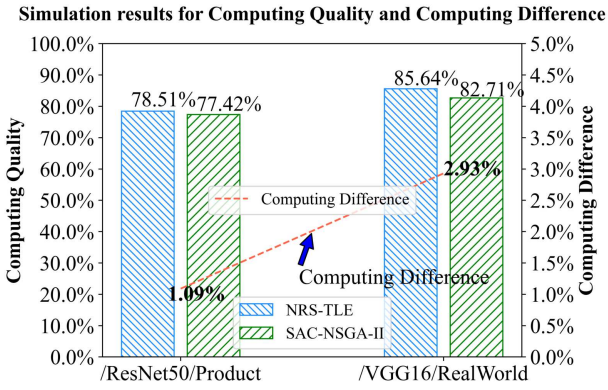


Fig. 9. Simulation results for the Average Training Quality

From the load balance aspect, Fig.10 shows the Average Task Queue Length of 4 CSPs. Under SAC-NSGA-II, the average queuing task number of S1, S2, S3, S4 is 6 tasks, 5 tasks, 5 tasks, 5 tasks, respectively, their queuing lengths are almost same. Under NRS-TLE, the average queuing task number of 4 CSPs has a little difference, the minimum queuing length is 3 tasks (S4), the maximum queuing length is 9 tasks (S2), but the difference is not high. So, we can deduce that, SAC-NSGA-II can better balance the tasks for 4 CSPs, and NRS-TLE can also maintain the fundamental load balance.

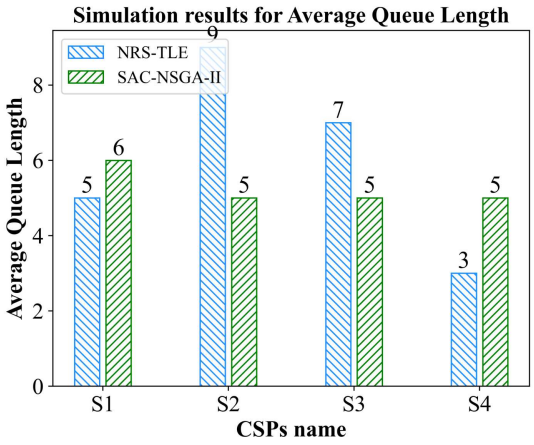


Fig. 10. Simulation results for Average Queue Length

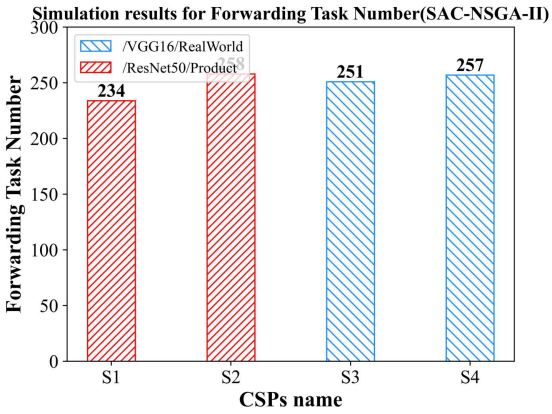


Fig. 11. Simulation results for Forwarding Task Number(SAC-NSGA-II)

Another important simulation results are the Forwarding Task Number for 4 CSPs. Fig.11 gives the forwarding number of two type tasks to 4 CSPs under SAC-NSGA-II, while Fig.12 gives the forwarding number of two type tasks to 4 CSPs under NRS-TLE. From the simulation results, the forwarding results under two schemes are totally different. Under NRS-TLE, the tasks of "/ResNet50/Product"are mainly forwarded to S2, only few tasks are forwarded to S1; the tasks of "/VGG16/RealWorld"are mainly forwarded to S3, only few tasks are forwarded to S4. But under SAC-NSGA-II, the forwarding proportion of "/ResNet50/Product" task are almost same for S1 and S2, while the forwarding proportion of "/VGG16/RealWorld" task are also almost same for S3 and S4. The results show that, NRS-TLE will provide more reasonable routing selection. Because NRS-TLE forwards the training task according to the local model accuracy stored in different CSPs, it can markedly shorten the training result retrieving time.

Fig.13 shows the simulation results of Occupation Rate of CPU Resource. From the simulation results, the occupation rate of CPU resources in the NRS-TLE is much lower than that in the SAC-NSGA-II. Because of the lower occupation rate of a single task in the NRS-TLE, the number of tasks that can be processed by CSPs parallelly in the NRS-TLE is higher than that in the SAC-NSGA-II, and the CSPs in the NRS-TLE can process tasks more efficiently.

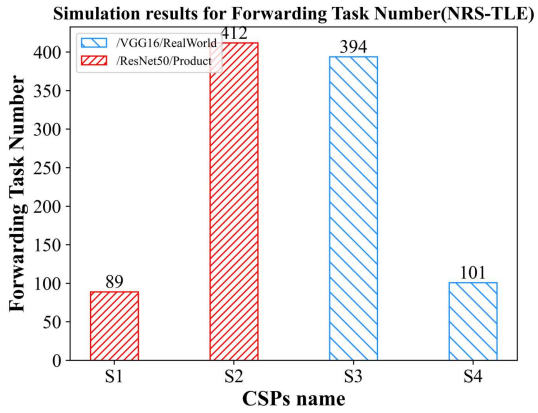


Fig. 12. Simulation results for Forwarding Task Number(NRS-TLE)

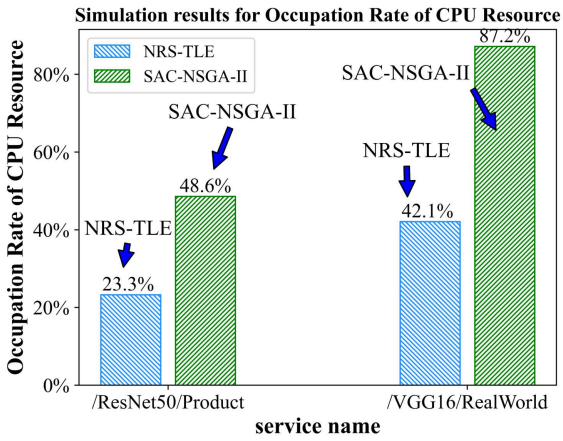


Fig. 13. Simulation results for Occupation Rate of CPU Resource

Fig.14 and Fig.15 shows the simulation results of Forwarding Probability of R1. From the simulation results, we can get 2 conclusions: (1)In NRS-TLE, most tasks of "/ResNet50/Product" will be routed to R2 and most tasks of "/VGG16/RealWorld" will be routed to R3. NRS-TLE can route according to the experience of different CSPs. (2)In SAC-NSGA-II, two kinds of tasks will be routed to R2 and R3 with the same probability. SAC-NSGA-II is unable to route according to the experience of different CSPs.

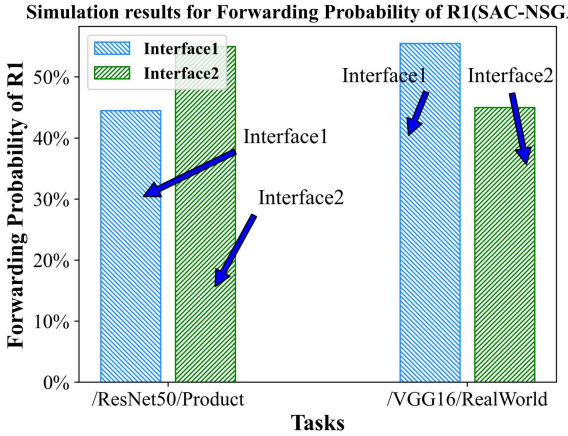


Fig. 14. Simulation results for Forwarding Probability of R1(SAC-NSGA-II)

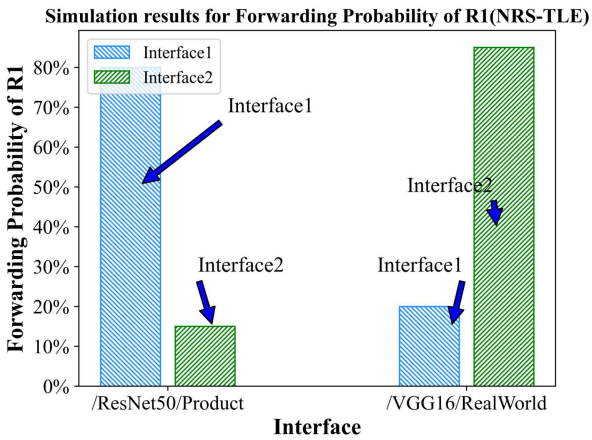


Fig. 15. Simulation results for Forwarding Probability of R1(NRS-TLE)

Fig.16 and Fig.17 shows the simulation results of Forwarding Probability of R2. From the results, we can get 2 conclusions:

(1)In NRS-TLE, the routing probability in different interfaces is equal beacuse the service in S1 and S2 is the same. The difference of NRS-TLE performance and SAC-NSGA-II performance is not significant.

(2)In the worst-case scenario, the performance of NRS-TLE and SAC-NSGA-II is the same.

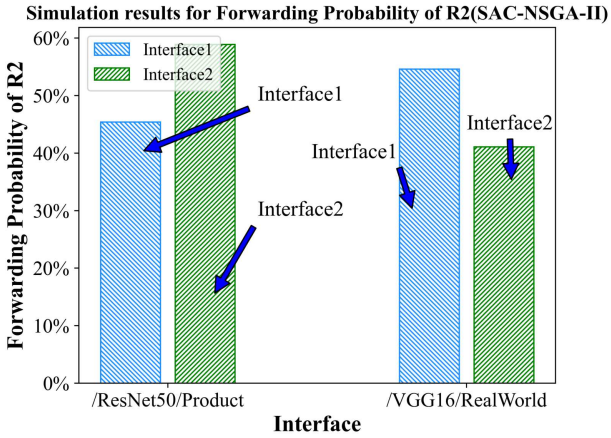


Fig. 16. Simulation results for Forwarding Probability of R2(SAC-NSGA-II)

V. Conclusions

Different of the existing routing schemes in Computing Power Networks only considering the computation-network resource, this paper takes into account the experience of CSPs in handling similar tasks. Specifically, for the scenario of classification model training, a novel routing mechanism based on transfer learning experience in named computing power network (NRS-TLE) is proposed. NRS-TLE utilizes the accuracy of CSPs historical trained models to measure their experience. A *tcce-function* is designed to evaluate the task processing capabilities of different CSPs based on this experience. Subsequently, a probability-based routing scheme is devised, prioritizing the assignment of user requests for model classification training to the CSP with better training experience. Simulation and experimental results

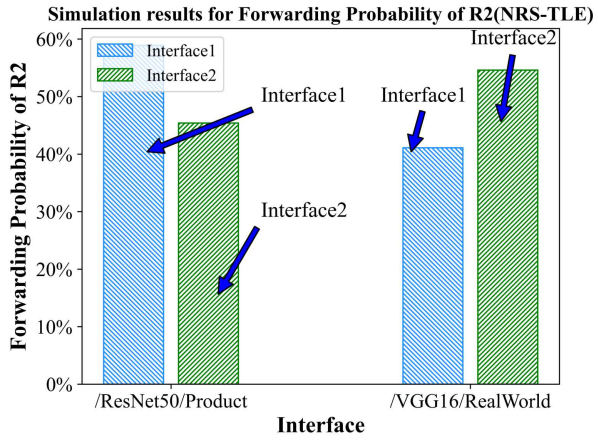


Fig. 17. Simulation results for Forwarding Probability of R2(NRS-TLE)

show that, comparing with existing routing scheme that do not consider CSP experience, NRS-TLE not only achieves slightly superior model accuracy but also significantly reduces the required training time.

With the advancement of transfer learning technology, various types of intelligent applications in the future will increasingly utilize the historical experience to enhance learning efficiency. Consequently, the routing decisions in network will gradually incorporate the evaluation of CSPs' internal experience. Although this paper only focuses on the research of classification model training in a particular application scenario, the exploratory work presented here will provide valuable insights for designing next-generation routing schemes in Computing Power Network.

REFERENCES

[1] Xiongyan Tang, Chang Cao, Youxiang Wang, Shuai Zhang, et al, "Computing power network: The architecture of convergence of computing and networking towards 6G requirement," *China communications*, vol. 18, no. 2, pp. 175-185, 2021.

[2] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, et al, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66-73, 2014.

[3] Xingnan Li, Jiangang Lu, Peiming Zhang, et al, "A GNN-Based Routing and Scheduling Mechanism for Multi-domain Computing First Network," *2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS)*, pp. 153-163, 2023.

[4] Yunpeng Xie, Xiaoyao Huang, Jingchun Li, et al, "Computing Power Network: Multi-Objective Optimization-Based Routing," *Sensors*, vol. 23, no. 15, pp. 6702, 2023.

[5] Xiaoyao Huang, Bo Lei, Min Wei, Guoliang Ji, et al, "Task Value Aware Optimization of Routing for Computing Power Network," *2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1-6, 2023.

[6] Chuanfu Zhang, Wen Li, Jing Chen, Jing Ge, Di Wang, et al, "A Global Task Scheduling Method Based on Network Measurement and Prediction in Computing Power Networks," *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 2837-2839, 2023.

[7] Jialu Zhao, Kun Xie, Huiyue Sun, Chengcheng Li, and Yasheng Zhang, "An MA-PPO based Task Scheduling Method for Computing First Networks," *Proceedings of the 2022 5th International Conference on Telecommunications and Communication Engineering*, pp. 260-265, 2023.

[8] Micha Król, Spyridon Mastorakis, David Oran, Dirk Kutscher, "Compute First Networking: Distributed Computing meets ICN," *ICN'19: Proceedings of the 6th ACM Conference on Information-Centric Networking*, pp. 67-77, 2019.

[9] Tong Sun, Chao Ma, Zechun Li, and Kun Yang, "Cloud Computing-based Parallel Deep Reinforcement Learning Energy Management Strategy for Connected PHEVs," *Engineering Letters*, vol. 32, no. 6, pp. 1210-1220, 2024.

[10] Daniel Meirovitch, LiXia Zhang, "NSC-Named Service Calls, or a Remote Procedure Call for NDN," *https://named-data.net/publications/techreports/ndn-tr-0074-1-nsc/*, 2021.

[11] Uthra Ambalavanan, Dennis Grewe, Naresh Nayak, et al, "DICer: Distributed coordination for in-network computations," *Proceedings of the 9th ACM Conference on Information-Centric Networking*, pp. 45-44, 2022.

[12] Ling Shao, Fan Zhu, Xuelong Li, et al, "Transfer Learning for Visual Categorization: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019-1034, 2015.

[13] Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, Jiayu Zhou, et al, "Transfer Learning in Deep Reinforcement Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 109, pp. 13344-13362, 2023.

[14] Xin Zhang, Wen Xie, Ziqi Dai, Jun Rao, Haokun Wen, et al "Finetuning Language Models for Multimodal Question Answering," *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 9420-9424, 2023.

[15] Baoxin Zhang, Dan Yang, Yang Liu, and Yu Zhang, "Graph Contrastive Learning with Knowledge Transfer for Recommendation," *Engineering Letters*, vol. 32, no. 3, pp. 477-487, 2024.

[16] Karen Simonyan, Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, pp. 1-14, 2015.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, et al, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016.