Cascade: Enhancing Reinforcement Learning with Curriculum Federated Learning and Interference Avoidance — A Case Study in Adaptive Bitrate Selection

Salma Emara, Daniel Liu, Fei Wang, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

***Abstract*—Current reinforcement learning (RL) algorithms, particularly RL-based networking algorithms, demonstrate significant potential for overcoming limitations of manually-tuned heuristics. However, RL-based algorithms are known to be sample inefficient and may not perform well in a wide range of environments. Federated reinforcement learning (FRL) aims to enhance sample efficiency and improve model performance across a wide variety of environments. Nevertheless, many existing approaches neglect the challenges posed by the dynamic nature of training sample distributions in RL and the heterogeneity of data across clients in FRL. This may restrain the broader applicability of FRL algorithms. Addressing these gaps, we propose *Cascade*, the first curriculum federated reinforcement learning framework with interference avoidance, and we study *Cascade* in the context of RL-based adaptive bitrate (ABR) selection algorithms. To eliminate interference between two or more interfering tasks from different clients in FRL, we propose an interference avoidance technique that penalizes changes to model parameters important for other clients. We extensively evaluate Cascade in a wide range of network environments. Our experiments show that Cascade outperforms the state-of-the-art federated learning settings by a minimum of 21% in average reward and 15% in model skewness. These findings highlight the efficacy of *Cascade* and underscore the potential of enhancing FRL.**

***Index Terms*—Federated reinforcement learning, curriculum learning, reinforcement learning, adaptive bitrate selection.**

I. Introduction

Recently, deep reinforcement learning (DRL) has revolutionized several engineering solutions, outperforming the state-of-the-art in areas such as adaptive bitrate selection, exemplified by Pensieve [1]. Given the success of DRL, there is a growing demand to address its associated problems, such as sample inefficiency and poor model performance in a wide range of environments.

Federated Reinforcement Learning (FRL) [2] solves several issues in applying DRL to several domains. For example, FRL mitigates the sample inefficiency challenge by training agents on several sample trajectories generated by multiple clients. Also, it does not expose the privacy of clients by sharing their trajectories to other clients, since in FRL only model updates are shared not the raw trajectories.

Despite the promising potential of FRL in many applications such as resource network management [3], there are existing challenges in using FRL to train a **DRL-based networking algorithm**. These challenges include time-changing data distributions, i.e. non-stationary data distributions, on clients. This is caused by dynamic network environments, and the changing policy/model during the training process, which changes the distribution of experiences observed.

Another challenge in FRL, inherited from RL, is the difficulty for DRL agents to learn effectively when trained across a wide variety of environments. Similarly, in FRL, the aggregation of knowledge from diverse experiences by the server to form a global model presents challenges in achieving a well-performing, converged global model.

To this end, given the evidence that curriculum learning aids reinforcement learning algorithms in diverse network environments [4], [5], we propose *Cascade*, the first FRL framework that introduces curriculum learning with interference avoidance to improve the performance of an RL-based algorithm on a wide variety of environments. In our work, we extensively evaluate Cascade on the adaptive bitrate (ABR) selection problem.

The key contributions of our work as follows. Firstly, we empirically show that using curriculum learning enhances the performance of FRL for ABR using the `FedAvg` algorithm. Secondly, we formalize a curriculum learning solution for FRL and test it in the context of ABR. Applying curriculum learning in federated learning is challenging because, unlike traditional curriculum learning which relies on raw trajectories or task identifiers for sequencing tasks, the federated learning server lacks access to these elements. Finally, we propose interference avoidance techniques to avoid aggregating interfering gradients. This enhances the performance of the converged model across various environments.

II. Preliminaries, Related Work and Motivation

We study our proposed algorithm Cascade on the adaptive bitrate (ABR) selection problem. ABR serves pre-recorded videos and livestreams to a broad array of clients over the Internet [6], [7]. Similar to RL-based algorithms, in RL-based ABR selection, at each time step t, the agent observes a state s_t, and selects an action a_t according to its policy $\pi(a|s)$. In

RL-based ABR, s_t is the dynamics of the past and current throughput, playback buffer and portion of unwatched video over one video session, and a_t is the video chunk bitrate. In return, the agent receives the next state s_{t+1} and a reward r_t. In RL-based ABR, reward is structured to be a linear combination of the selected bitrate, change in bitrate and rebuffering time, which reflects the quality of experience of the user. A higher bitrate means a higher resolution and higher bandwidth required. The last terminal step that ends the episode is time step T, and the agent's goal is to maximize the total accumulated reward $R_t = \sum_{k=0}^{T} \gamma^k r_{t+k}$, where the discounted factor is $\gamma \in (0,1]$. The state value $V^{\pi}(s) = \mathbb{E}[R_t | s_t = s]$ is the expected return from state s following policy π.

The RL algorithm we use in this paper is the advantage actor-critic (A2C) algorithm [8]. In A2C, we train an actor and critic parameterized by θ and ϕ, respectively. The critic estimates the value function $V(s;\phi)$ and the actor learns a policy $\pi(a|s;\theta)$ distribution over actions to maximize the expected return suggested by the critic. The critic is updated by minimizing the critic loss described as:

$$L(\phi) = \mathbb{E}[(V(s) - \mathbb{E}_{a\sim\pi(.|s)}[r(s,a) + \gamma V(s')|s])^2], \quad (1)$$

and the actor is updated in the direction of $\nabla_{\theta'} J$ denoted as:

$$\begin{aligned} \nabla_{\theta'} J = \nabla_{\theta'} \log(\pi(a_t|s_t;\theta'))(R_t - V^{\pi}(s_t;\phi)) \\ + \nabla_{\theta'} H(\pi(s_t;\theta')), \end{aligned} \quad (2)$$

where H is the entropy [8].

In the integration of RL-ABR into FL, clients run RL algorithms to train an ABR algorithm collaboratively over different network environments. Training ABR using FRL can enhance sample efficiency and improve convergence to a better asymptotic behavior than using centralized training. In addition, FRL can preserve the privacy of clients as their local statistics will not be shared.

Why and why not curriculum learning? Despite the discussed benefits of incorporating FL into RL-ABR, clients may experience diverse network environments due to data heterogeneity, which makes RL struggle to converge to an optimal model in such a setting. However, recently proposed curriculum reinforcement learning has shown improvements in model convergence when training across a wide variety of environments [4]. In addition, recently curriculum federated learning has shown potential in improving global model performance when data distributions are heterogeneous [9]This insight gives us the motivation to apply curriculum learning to FRL.

The traditional concept of curriculum learning requires that the model uses data samples with gradually increasing difficulty for training. "easy" experiences are supposed to be not "too easy" to be useless to learning, or not "too difficult" [10]. Curriculum learning focuses on easy tasks first and acquires knowledge that can be transferred to more sophisticated knowledge.

For the FRL-ABR problem, we claim the easy knowledge acquired from learning in less dynamic and less noisy environments will improve the action space exploration in more sophisticated environments, and hence improve the overall model's performance. Hence, our goal is to create a curriculum that aggregates knowledge gained from training in "easy" environments, first.

Developing a curriculum learning algorithm requires knowledge about the level of difficulty of environments. Using a metric to quantify that, the algorithm can gradually increase the difficulty of training environments to focus on more rewarding environments [10]. Many existing curriculum learning methods [4], [5] depend on parameterizing tasks, and based on some representation of the parameters of the tasks, they form a sequence of tasks, i.e. a curriculum. This requires access to the experiences observed. However, in FL, features of tasks on each client are hidden from the server. Hence, it is challenging for the FL server to know which clients have agents running on "easy" environments to form a curriculum.

Moreover, curriculum learning requires focusing more on some environments than others at some points in time. Therefore, the server may aggregate the updates of some clients more than others, leading to the skewness of the performance of the global model to some network environments. This happens when previously acquired knowledge interferes with new knowledge to be acquired, thus inhibiting the learning of newer tasks.

As we develop our curriculum learning algorithm for FRL, we want to ensure: (i) building a curriculum without exchanging private knowledge of environments on clients, and (ii) preventing the skewness of the model towards easy tasks learned in initial training stages.

III. Curriculum Learning for Federated Reinforcement Learning

In this work, we propose a curriculum learning algorithm that runs on the server and the algorithm decides which clients' updates to aggregate. The server uses a difficulty score to decide on which client updates to aggregate. This difficulty score is calculated by each client, and the clients communicate to the server the difficulty score along with their local updates.

The difficulty score measurement is produced during the RL training process. We study the suitability of different potential metrics. We propose the following three different metrics, each with strengths and weaknesses.

Loss: During the training process, every $t_{\max}$ steps or until the episode terminates, we calculate the actor loss and critic loss to update the actor and critic. The critic loss $L(\phi)$, also known as TD-error or prediction error, represents the critic error in estimating the value of a state. The actor loss $-\nabla'_{\theta} J$ represents the gradient step taken in gradient ascent. We are assuming that if a client observes a lower critic or actor loss, it means that they are closer to the converged model.

Rate of change in loss: Another metric that can indicate how noisy or difficult an environment is the rate at which loss decreases. If the rate is fast, even if the model did not converge, there is a high probability there were easy experiences to gain and the samples caused efficient learning.

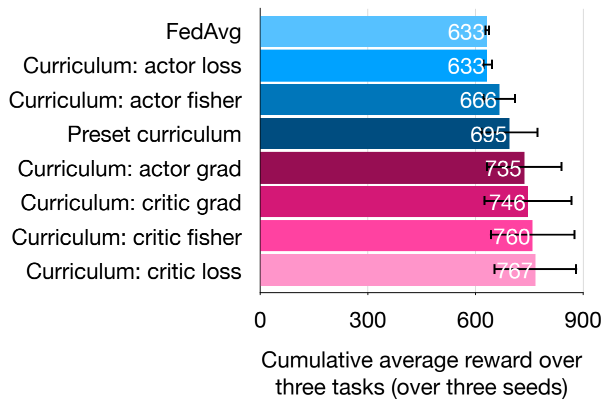

Fig. 1: The cumulative average reward over three tasks of the final global model after 20 communication rounds of different difficulty score metrics.

Calculating the rate of change in losses requires fitting a linear line to the critic or actor losses observed each update. The gradient of the fitted line is the rate of change in loss. This requires minimizing the following mean squared error:

$$MSE = \sum_{i=1}^{N} |r \cdot i - l_i|^2, \tag{3}$$

where r is the rate of change in critic or actor loss – r_{critic} or r_{actor} respectively, l_i is the critic or actor loss in update i – $L_i(\phi)$ or $(\nabla_\theta J)_i$ respectively, and N is the number of updates in the current round. Using least squares regression (or any other simple method) to find the rate of change in loss has a time complexity of $\mathcal{O}(N)$.

Sum of the diagonals of Fisher information matrix: The Fisher Information Matrix F (FIM) (i) is equivalent to the second derivative of the loss near a minimum, *i.e.* Hessian matrix (ii) can be easily computed from first order derivatives, and (iii) is positive semi-definite. The second derivative of the loss near a minimum represents the curvature, and if we sum all the diagonals of F, we can get a sense of the curvature of all the model parameters. If this sum is close to zero, it shows that the model converged.

Clients trained on “easy” tasks will be referred to as “elite clients”. Depending on the difficulty score metric used, elite clients will vary. The server selects the top βth percentile clients for aggregation in the first round. The percentile β is initially set to 20 and incrementally increased by 15 in each round until it reaches 100. To study the suitability of each of the aforementioned difficulty score metrics, we set up three FL clients each running a different throughput trace file collected on a bus simulating a network environment with (i) stable throughput, (ii) steadily varying throughput, and (iii) dynamic throughput, respectively.

In Fig. 1, we plot the cumulative average reward of the final global model after 20 communication rounds. The legend labels the algorithms compared according to the difficulty score metrics used. A “Preset curriculum” refers to a naive curriculum where the server considers a network environment with a more stable throughput as an easier task.

We observe that regardless of the difficulty score metric used, all types of curriculum learning perform similar or better than `FedAvg`. This behavior manifests that curriculum learning aids the exploration of a better action space by building a better model parameter space during the training process. Using actor model metrics yielded a weaker performance than metrics of the critic model, since the critic is faster in capturing knowledge of the environment during training.

We observe closer the rate of convergence of the experiments using critic model metrics, and “Curriculum: critic grad” shows the quickest rate of convergence. Since the server automatically selects clients with a quick rate of change in critic loss, the global model aggregates models that learn quickly and hence the system learns quickly too. We decide to use curriculum learning that uses the rate of change in critic loss “Curriculum: critic grad” in our proposed algorithm “Cascade”.

IV. Interference Avoidance between Client Updates

In addition to aiding the exploration of a better action space with curriculum learning, in Cascade, we want to ensure prior learned knowledge is not interfering with learning new experiences. This interference may lead to model skewness. To minimize interference during training, we calculate the importance weight of each parameter in the neural network to each client. Knowing important model parameters for each client, we can penalize the changes to these parameters in the local loss function of other clients. This will inhibit clients from causing interfering gradients and will reduce model skewness.

At the beginning of a communication round, each client receives the global model θ_{t-1}, which includes the critic and actor. After one training round, each client i will communicate the model updates to the global server $\Delta\theta_{t-1,i}$, which is the change to the initial global model from the previous training round. The server will aggregate the model updates from clients and report the global model to clients as $\theta_{t-1} + \frac{1}{I}\sum_{j=1}^{I} \Delta\theta_{t-1,j}$, where I is the number of clients aggregated. The server will also communicate to the clients if their updates were aggregated or not and the value of I. This will help each client to calculate the importance weight matrix Ω, which contains information about which model updates are important for other clients. If the client's update was aggregated, Ω is calculated as in Eqn. 4.

$$\Omega_i = \theta_t - [\theta_{t-1} + \frac{1}{I}\Delta\theta_{t-1,i}] = \frac{1}{I}\sum_{j=1, j\neq i}^{I} \Delta\theta_j \tag{4}$$

Each client will have Ω_i at the beginning of each round – except the first round. Ω_i represents a summary of changes other clients created to the model. Large changes to certain parameters signal that these are critical parameters in the training of other clients. While parameters that are changed incrementally are not important to other clients. Hence, changing critical parameters to other clients will interfere with their learning process and should be avoided. While changing less critical parameters will not erase or reverse the knowledge acquired by the other clients.

We propose adding a regularization term to the training of the critic and actor of our clients such that we penalize changes to important parameters. Eqn 5 represents the local clients model (actor or critic) loss, respectively:

$$l_{i,\text{model}} = L_i(\theta) + \lambda \Omega_{i,t,model}(\theta_i - \theta_t)^2, \tag{5}$$

where θ_i is the current model parameters for client i, θ_t are the model parameters received at the beginning of round t, $\Omega_{i,t,model}$ are importance weights calculated for the models for client i at the beginning of the round t. Adding a regularization term above to the local clients' loss will be referred to as "interference avoidance" or IA for short and we will show the ablation study of IA in the next section. A summary of all steps of the Cascade algorithm is displayed in Alg. 1.

Algorithm 1 Cascade Algorithm

```
1:  procedure LOCAL UPDATE
2:      Input: node index $i$, models $\phi_{t-1}$, $\theta_{t-1}$
3:      Except 1st round, get $\Omega_{i,t,critic}$ and $\Omega_{i,t,actor}$ as (4)
4:      for all episodes do
5:          for all updates do
6:              Get $L(\phi_t)$ and $\nabla_{\theta_t} J$ as (1) and (2)
7:              Get $l_{i,\text{critic}}$ and $l_{i,\text{actor}}$ as (5)
8:              Update $\phi_t$ and $\theta_t$ using $l_{i,\text{critic}}$ and $l_{i,\text{actor}}$
9:      Get $r_{\text{critic},i}$ using (3)
10:     return $r_{\text{critic},i}$, $\Delta\phi_{i,t}$, $\Delta\theta_{i,t}$
11: procedure GLOBAL UPDATE
12:     Input: difficulty score and local updates $r_{\text{critic},i}$, $\Delta\phi_{i,t}$, $\Delta\theta_{i,t}$
13:     Aggregate the elite $\beta$%ile clients, $\Delta\phi_t = \sum_i^I \frac{1}{I}\Delta\phi_{i,t}$, $\Delta\theta_t = \sum_i^I \frac{1}{I}\Delta\theta_{i,t}$, where $I$ is the number of clients with $r_{\text{critic},i} < \beta$%ile
14:     Get $\phi_t = \phi_{t-1} + \Delta\phi_t$ and $\theta_t = \theta_{t-1} + \Delta\theta_t$
15:     return $\phi_t$, $\theta_t$
```

Communication overhead: In summary, the communication overhead of Cascade is two scalars communicated from the server to each client, which is I and if their updates were aggregated into the global model to help the clients calculate Ω. From clients to servers, apart from sending the model updates, the clients send their difficulty level metric, which is also a scalar value.

V. Experimental results

A. Implementation and experimental setup

Cascade is built on top of Plato[1], an open-source federated learning framework built to emulate real-life scenarios. We use the Pensieve [1] RL-interface implementation in Park [11], which simulates the buffer, playback and a network trace. Cascade can be applied to training other networking algorithms too, such as congestion control or networking adaptive coding, due to its usage of the training dynamics of a model instead of the features of a networking algorithm.

[1]https://github.com/TL-System/plato

Clients are trained and tested on real-life traces collected by streaming a pre-recorded video over 290 traces from FCC broadband measurements (labeled "FCC") and 310 cellular traces (labeled "Norway") [12]. During the training process, each client has one of 6 different training distributions of traces that simulate a real-world throughput observed by an agent. Each client trains for 400 episodes, where for every episode one trace file is randomly sampled from 10 files from the same network environment. Each episode has a length of 490 steps, as set by Park [11]. After training for a fixed number of episodes, clients report their local updates and the difficulty score of Cascade, r_{critic}, to the server. After the server aggregates the local clients' updates based on the global model aggregation algorithm, the server will test the global model on the 10 traces for each client.

Baselines. We compare Cascade with a number of baselines. They are (i) RL-ABR: the training is conducted in a centralized behavior, sampling one trace file every episode from the training distributions of all clients combined; (ii) `FedAvg` [13]: clients are randomly selected and their local updates are aggregated with equal weights; (iii) `FedADP` [14]: clients are randomly selected but their local updates of clients are weighted according to the alignment of their gradient vector with the gradient vector of the global model.

Evaluation metrics We analyze the performance of Cascade and other FL algorithms using four main metrics: (i) *cumulative* average reward, which is the sum of the average rewards achieved over the training distribution of each client, (ii) the model skewness calculated across the training distribution of each client, (iii) the average reward of the converged model evaluated over 30 traces, which were not used during the training distribution, and (iv) the convergence speed defined as the number of rounds to reach the average reward of the last 20 rounds. The last 20 rounds have stable reward values.

B. Number of clients participating

We train Cascade and other algorithms over three different settings of experiments with: 6, 12 and 18 clients participating. Each experiment is run over three different seeds, and the average reward over three seeds is reported. At the end of each training round, we get the average reward over all the traces in each of the 6 training environments and add them to get the cumulative average reward. Note that the average reward for each distribution is used only for evaluation, not for training.

As shown in Table I, Cascade's asymptotic reward, which is the average cumulative average reward in the last 30 rounds, is 510, which is 23.8% and 20.9% higher than `FedAvg` and `FedADP`, respectively. This was at the expense of a lower convergence speed. In addition, the model skewness of Cascade is lower by 15% and 21% compared to `FedAvg` and

FedADP. This illustrates that Cascade is better at avoiding interference between clients' knowledge and does not favor some clients more than others.

Limitations if clients have the same environment distribution. With 12 and 18 clients participating, the 6 environment distributions are replicated twice and thrice, respectively. Through replicating environment distributions, we aim to study the effect of interference avoidance between clients running on similar environment distributions. Intuitively, the model will have parameters important for each client and few parameters to be shared, *i.e.* Cascade avoids sharing of knowledge across clients with similar tasks. This decreases the asymptotic reward by 3–5% compared to FedAvg and FedADP since the model is not used efficiently. In terms of model skewness and convergence speed, Cascade is within 3–5% comparable behavior with FedAvg and FedADP as observed in Table I. Therefore, in the training stage, we encourage selecting distant clients to reduce the chance of having clients with similar environment distributions.

In Fig. 2, we plot the cumulative average reward over the 6 training distributions vs. the training round for 70 rounds for the case of 6 clients participating. When 6 clients were participating, we observe that Cascade has the highest cumulative reward asymptotically. It is noteworthy that in Fig. 2 RL-ABR converges to an unstable asymptotic reward demonstrating the challenge in learning while observing a diverse set of environments. All federated learning algorithms demonstrate a stable convergence, which reveals the benefit of federated reinforcement learning in settings where learning happens in a wide range of environments.

TABLE I: The asymptotic cumulative average reward, the number of training rounds required to reach this reward, and model skewness for Cascade vs. FedAvg and FedADP

	# of clients	Asymptotic reward **(higher)**	Rounds to converge **(lower)**	Model skewness **(lower)**
FedAvg	6	412	37	373
FedADP	6	422	**27**	402
Cascade	6	**510**	41	**319**
FedAvg	12	**445**	33	**340**
FedADP	12	420	**30**	386
Cascade	12	420	**30**	360
FedAvg	18	**425**	40	397
FedADP	18	424	**27**	383
Cascade	18	411	33	**371**

C. Generalization

We test the generalization of the converged models, when 6 clients were participating, from all baselines with Cascade. In Fig. 3, we plot the average reward of each algorithm including RL-ABR on 30 different traces from different network environments that were not in the training environment distribution. We observe that Cascade is having almost 45% higher average reward compared to FL algorithms: FedAvg and FedADP. This exhibits how curriculum learning can help in generalization in FL.

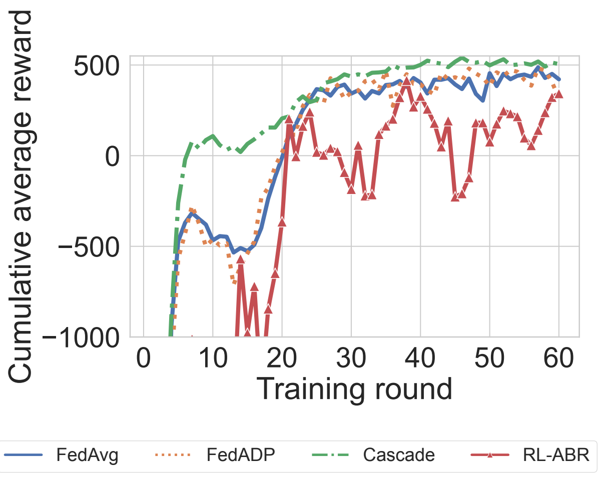

Fig. 2: Comparison of cumulative average reward over 6 different distributions across different algorithms for 6 clients selected for global aggregation.

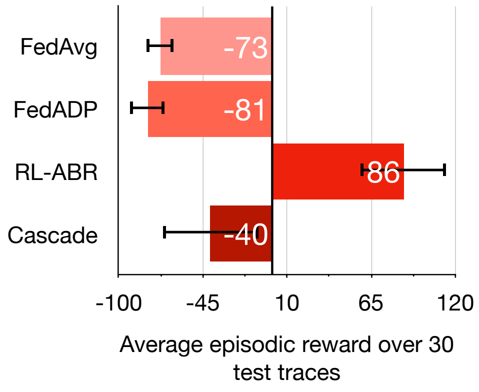

Fig. 3: Comparison of generalization results across different algorithms when 6 clients participate in global aggregation

However, RL-ABR shows better generalization than all federated learning algorithms. Our insight is that RL-ABR avoids overfitting to specific network environments by training *sequentially* on diverse settings, unlike federated learning, where each client's overfitting may lead to a globally overfitted model.

D. Ablation Studies

Additionally, we study the effect of curriculum learning and interference avoidance, if each is used solely. We study the effect of λ, varying it from 0 to 3, on asymptotic behavior, convergence, model skewness and generalization when 6 clients are participating. When $\lambda = 0$, curriculum learning is used without interference avoidance. "IA $\lambda = 1$" refers to interference avoidance without curriculum learning.

In Fig. 4 and Table II, we observe Cascade with $\lambda = 1$ having the best asymptotic reward, model skewness and average reward over out-of-training distribution (or test reward). Increasing λ beyond 1 seems to avoid interference excessively

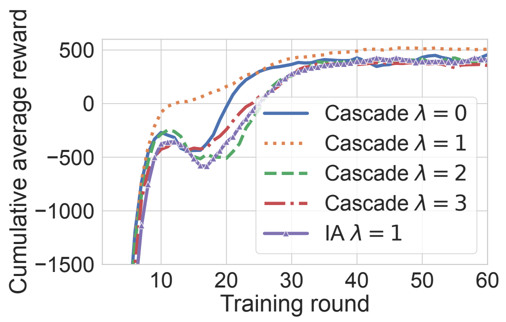

Fig. 4: Effect of λ and curriculum learning when $K = 6$, where IA is only interference avoidance without curriculum learning and Cascade λ is curriculum learning with interference avoidance but with different λ

to limit the accumulation of knowledge of tasks. Hence, the asymptotic reward decreases with increase in λ. Interestingly, interference avoidance alone has a similar asymptotic reward to `FedADP`, since both try to avoid interference between any two clients. By weighting updates according to their alignment with gradient updates, `FedADP` avoids aggregating interfering gradients, and IA penalizes changes to parameters that may interfere with progress of other clients.

Our insight is that only avoiding interference suppresses knowledge accumulation in cases where there is transferable knowledge between clients. Avoiding interference prematurely before transferable easy knowledge is gained would inhibit learning. When curriculum learning is used, important transferable easy knowledge is gained first. Then, interference avoidance comes into action with more clients' models aggregating in Cascade. Clients use the common knowledge gained in the early stages of curriculum learning to advance the overall knowledge of the model without interfering with other clients.

TABLE II: The asymptotic cumulative average reward, the number of training rounds required to reach this reward, model skewness and average test reward over out-of-training distribution traces for Cascade with different λ and without curriculum learning

	Asymptotic reward **(higher)**	Rounds to converge **(lower)**	Model skewness **(lower)**	Test reward **(higher)**
Cascade $\lambda = 0$	408	**31**	351	-69
Cascade $\lambda = 1$	**510**	41	**319**	**-40**
Cascade $\lambda = 2$	396	33	382	-70
Cascade $\lambda = 3$	368	**31**	360	-61
IA $\lambda = 1$	422	35	382	-83

VI. Concluding Remarks

In this work, we propose Cascade, a new federated reinforcement learning framework for ABR, powered by curriculum learning and interference avoidance. We show that aggregating all clients simultaneously may inhibit learning as in `Fedavg`, and we empirically show that if we choose clients' models that were trained on easy environments to aggregate, we would achieve a higher knowledge accumulation on all tasks. In Cascade, we have clients report their difficulty score, which is the rate at which critic loss is decreasing, and the server aggregates fast-learning clients since they were trained in easy environments. To avoid interference between clients in FL, we develop a simple yet effective technique to penalize changes to model parameters that are important to other clients. Our experiments show that Cascade outperforms many federated learning algorithms by 20% in asymptotic performance and 21% in model skewness. This was at a cost of slower convergence rate, almost 11%.

References

[1] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun. (ACM SIGCOMM)*, 2017.

[2] H. H. Zhuo, W. Feng, Q. Xu, Q. Yang, and Y. Lin, "Federated deep reinforcement learning," *arXiv preprint arXiv:1901.08277*, 2019.

[3] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multi-timescale resource management for multi-access edge computing in 5g ultra dense network," *IEEE Internet of Things Journal*, vol. 8, pp. 2238 – 2251, 2020.

[4] Z. Xia, Y. Zhou, F. Y. Yan, and J. Jiang, "Automatic curriculum generation for learning adaptation in networking," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun. (ACM SIGCOMM)*, 2022.

[5] S. Emara, F. Wang, B. Li, and T. Zeyl, "Pareto: Fair congestion control with online reinforcement learning," *IEEE Transactions on Network Science and Engineering*, pp. 1–18, 2022.

[6] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *Proc. 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2020, pp. 495–511.

[7] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun. (ACM SIGCOMM)*, 2015, p. 325 – 338.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd International Conference on Machine Learning (ICML)*, 2016.

[9] S. Vahidian, S. Kadaveru, W. Baek, W. Wang, V. Kungurtsev, C. Chen, M. Shah, and B. Lin, "When do curricula work in federated learning?" in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[10] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th International Conference on Machine Learning (ICML)*, 2009.

[11] H. Mao, P. Negi, A. Narayan, H. Wang, J. Yang, H. Wang, R. Marcus, R. Addanki, M. K. Shirkoohi, S. He, V. Nathan, F. Cangialosi, S. Venkatakrishnan, W.-H. Weng, S. Han, T. Kraska, and M. Alizadeh, "Park: An open platform for learning-augmented computer systems," in *Proc. 33rd Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[12] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *Proc. Multimedia Systems Conference (MMSys)*, 2013.

[13] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[14] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking (TCCN)*, vol. 7, no. 4, pp. 1078–1088, 2021.