FedFormer: Contextual Federation with Attention in Reinforcement Learning

Liam Hebert
University of Waterloo
Waterloo, Canada
liam.hebert@uwaterloo.ca

Lukasz Golab
University of Waterloo
Waterloo, Canada
lgolab@uwaterloo.ca

Pascal Poupart
University of Waterloo, Waterloo, Canada
Vector Institute, Toronto, Canada
ppoupart@uwaterloo.ca

Robin Cohen
University of Waterloo
Waterloo, Canada
rcohen@uwaterloo.ca

ABSTRACT

A core issue in multi-agent federated reinforcement learning is defining how to aggregate insights from multiple agents. This is commonly done by taking the average of each participating agent's model weights into one common model (FedAvg). We instead propose FedFormer, a novel federation strategy that utilizes Transformer Attention to contextually aggregate embeddings from models originating from different learner agents. In so doing, we attentively weigh the contributions of other agents with respect to the current agent's environment and learned relationships, thus providing a more effective and efficient federation. We evaluate our methods on the Meta-World environment and find that our approach yields significant improvements over FedAvg and non-federated Soft Actor-Critic single-agent methods. Our results compared to Soft Actor-Critic show that FedFormer achieves higher episodic return while still abiding by the privacy constraints of federated learning. Finally, we also demonstrate improvements in effectiveness with increased agent pools across all methods in certain tasks. This is contrasted by FedAvg, which fails to make noticeable improvements when scaled.

KEYWORDS

Federated Learning, Multi-agent Learning, Transformer Networks

ACM Reference Format:
Liam Hebert, Lukasz Golab, Pascal Poupart, and Robin Cohen. 2023. FedFormer: Contextual Federation with Attention in Reinforcement Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning has become a ubiquitous tool in applications such as autonomous vehicles [26, 27], where vehicles learn to navigate the streets of many complex environments, and Internet-of-Things devices [1, 20, 36]. However, to obtain robust performance, a large collection of training episodes is often needed. In the case of safety-critical systems such as autonomous vehicles, training episodes must also be diverse enough to account for the many possible edge cases that the system could face. As a result, the scale and diversity of data collection needed to achieve adequate performance quickly becomes infeasible for a single party. To account for these challenges, a common technique for data collection relies on crowdsourcing, where training episodes are collected from many agents that explore their local environment and then transmitted to a central controller to train a generalized model [23].

However, this strategy comes with a crucial challenge: privacy. In our example of autonomous vehicles, training data consist of sensitive images of local surroundings, which need to be broadcast over the internet to a centralized source. This presents an opportunity for bad actors to intercept these data for malicious purposes. Motivated by this challenge, privacy-preserving federated machine learning and federated reinforcement learning have been proposed. Here, rather than transmitting training observations, local model weights are transmitted instead. These model weights are then aggregated by a centralized controller to form a new common model, all without seeing any sensitive data.

The current state-of-the-art general method in federated reinforcement learning to combine agents is FedAvg [18]. This strategy takes the weighted average of transmitted model weights across all agents according to the number of observations that an agent has seen. However, there are several drawbacks that come with this approach. In reinforcement learning, policy and critic networks are already prone to instability, which could be further exacerbated by the averaging of model parameters. In addition, the averaging of model weights across multiple agents often prohibits individual exploration in non-identical environments, a challenge when federating models explore diverse environments [23]. An example of this can be seen in autonomous vehicles, where knowledge gained from agents exploring rural roads is less useful to agents exploring urban cities. Despite this, contributions from both sets of agents would still be equally weighted under FedAvg, potentially creating a final agent that performs worse in both environments.

To address these drawbacks, inspired by recent advancements in natural language processing, we propose Federated Transformers (FedFormer), a novel federation policy based on transformer encoder models [5, 32]. Instead of taking the average of model weights, we utilize transformer encoders to learn contextual relationships between agents. We then leverage these learned relationships to contextually federate agents together during inference. This method allows local agents to maintain models specific to their environment,

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

allowing for local exploration, while also contextually including insights from other agents according to their relevance to the current environment. Importantly, these relationships can be learned without divulging confidential metadata, such as geolocation data, health information or other sensitive attributes. In addition, we compute these relationships during each time step, allowing for the dynamic federation as an agent's environment changes throughout the episode. This allows our approach to providing a more efficient aggregation of model insights during inference.

We evaluate FedFormer using the Meta-World environment [37], a benchmark of robotic manipulation tasks. Despite training on heterogeneous environments, our approach brings improved episodic return over FedAvg and over non-federated Soft Actor-Critic (SAC) [11] single-agent methods. This is despite SAC being trained on the entire set of environments as the federated strategies, representing performance without the privacy constraints of the federation. We also demonstrate the onboarding performance of FedFormer, finding that convergence speed increases 5-fold by leveraging pretrained agents in the federation network despite being applied to never-before-seen environments. Notably, these results are obtained without intervention from the other agents, further increasing privacy and effectiveness. When scaled to 10 and 15 agents, FedFormer outperforms the episodic return of FedAvg by a factor of 2.4 and 3.8 at the last epoch, respectively. This illustrates that FedFormer performs well when scaled to increased agent pools (with matching or increased performance), whereas FedAvg fails to make noticeable improvements or presents degraded performance when scaled. In all, our results illustrate that FedFormer performs better than state-of-the-art FedAvg, scales without degradation of performance, and is more effective than single-agent Soft Actor-Critic while still abiding by the privacy constraints of federated learning. Our source code is available at https://github.com/liamhebert/FedFormer.

2 BACKGROUND

2.1 Federated Learning and Federated Reinforcement Learning

There are four key properties of federated learning: distribution, data protection, generality and status equality [23]. Distribution states that federated model training is done in parallel by all participating models. Data protection ensures that the training data held by each participating party is not transmitted to other parties. Generality states that federated models must generalize to diverse environments. Finally, status equality ensures that training favours all participating parties equally, and often with identical infrastructures.

Satisfying these constraints, McMahan et al. [18] proposed the FedAvg federation strategy. Given a set of N parties with loss functions $\{\mathcal{F}_i\}_{i=1}^{N}$ and datasets $\{D_i\}_{i=1}^{N} \in D$ interested in formulating a joint cooperative model, FedAvg updates a centralized model as

$$\forall i, w_i(t+1) = w_i(t) - \gamma \nabla F_i(w_i(t)) \tag{1}$$

$$w_g(t+1) = \sum_{i=1}^{N} \frac{|D_i|}{|D|} w_i(t+1) \tag{2}$$

where $w_i(t)$ is the model update computed by agent i at time step t, γ is a fixed learning rate and w_g are the parameters of the centralized model. Under this policy, one can tune how often each model communicates its parameters to the centralized source. This strategy has been transferred to reinforcement learning, with many techniques applying FedAvg to federate policy and critic networks [23]. The most common extension of FedAvg is to introduce domain-specific adjustments to the weighing of model parameters. This has been done in the context of the game Pong [21]; for ioT devices, to optimize edge computation [25, 31, 34], and for the valued, recent application of real-time electric vehicle charging [39].

Another direction in federated reinforcement learning is to utilize metadata about agents to weigh their contributions. Lim et al. [16] and Chu et al. [3] weigh agent contributions according to their average reward. Similarly, Wang et al. [33] propose weighing FedAvg according to metadata such as batch size, average loss, dataset size and hit rate. In each of these prior settings, the training environments are similar, allowing for a non-disruptive application of parameter averaging. We refer to this class of federated methods as Weighted Federated Averaging.

However, it has also been shown that the performance of FedAvg degrades significantly when environments are perturbed and are heterogeneous between agents [23]. This is important since in many prior works, it is assumed that the environments of agents are consistent and non-heterogeneous [25, 31, 34]. For other examples of federated reinforcement learning, we refer the reader to a survey by Qi et al. [23]. Since most of the prior work relies on FedAvg with minor adjustments towards domain-specific goals, different model structures or additional processing steps prior to applying FedAvg, we focus our attention on directly comparing to FedAvg and Weighted FedAvg. This aligns with our focus on proposing a method that can directly replace FedAvg with a focus on applications for federated reinforcement learning.

In this work, we mitigate the downfalls of FedAvg and propose a general framework for federated reinforcement learning that utilizes an attention-based aggregation mechanism to federate agent predictions. Instead of using a hand-made federation strategy, we utilize a transformer model to learn contextual relationships between agents for federation. This approach allows federated reinforcement learning to be applied to tasks where environments are significantly different while still benefiting from the mutual federation.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is based on a formalization of Markov Decision Processes with a set S of states s, a set A of actions a, a transition function $P(s'|s,a)$ that indicates the probability of reaching state s' when executing action a in state s, a reward function $R(s,a) = r \in \Re$ that returns the expected immediate reward of executing a in s, a discount factor $\gamma \in [0,1)$, and a planning horizon h (assumed to be infinite throughout this paper). In RL, an agent optimizes a policy $\pi(a|s)$ that indicates the probability of executing a in s. The value $J^{\pi}(s)$ of executing a policy π when starting in state s is the sum of expected discounted rewards (i.e., $J^{\pi}(s_0) = E_{\pi}[\sum_{t=0}^{h} \gamma^t R(s_t, a_t)]$). Similarly, the value of executing a in s followed by the policy π is $Q^{\pi}(s_0, a_0) = R(s_0, a_0) +$

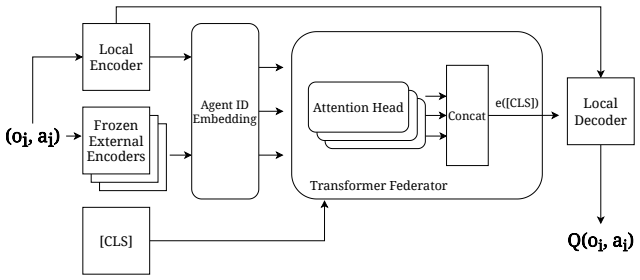

Figure 1: Federated Transformer Q-Function

$\gamma E_\pi[\sum_{t=1}^{h} \gamma^t R(s_t, a_t)]$. An optimal policy π^* achieves the highest expected cumulative reward (i.e., $\pi^* = argmax_\pi J^\pi(s) \forall \pi, s$). However, since the transition distribution and the reward function are unknown, the agent must perform this optimization based on trajectories of states, actions and rewards (i.e., $s_1, a_1, r_1, s_2, a_2, r_2, ...$) that it experiences as it interacts with the environment.

2.2.1 Soft Actor-Critic (SAC). Within reinforcement learning, traditional policy gradient techniques update the parameters θ of a policy π_θ by taking steps in the direction of the gradient of the value function [30, 35]:

$$\nabla_\theta J^{\pi_\theta}(s_t) = \nabla_\theta log(\pi_\theta(a_t|s_t)) \sum_{t'=t}^{\infty} \gamma^{t'-t} R(s_{t'}, a_{t'}) \tag{3}$$

However, a key weakness of this approach is the high variance brought on from the sum of discounted rewards for a given episode $\sum_{t'=t}^{\infty}$, as rewards can vary significantly from episode to episode [15]. To account for this, Actor-Critic methods were proposed, which utilize a function to approximate the sum of expected rewards and train that function separately. This function, called the Q-function or critic function, is trained through off-policy temporal-difference learning by minimizing:

$$\mathcal{L}_Q(\psi) = \mathbb{E}_{(s,a,r,s') \approx D}(Q_\psi(s,a) - y)^2 \tag{4}$$

$$y = r(s,a) + \gamma \mathbb{E}_{a' \approx \pi(s')} Q_{\hat{\psi}(s',a')} \tag{5}$$

where $Q_{\bar{\psi}}$ is a target Q-function (parameterized by ψ), an average of past Q-functions, and D is a replay buffer of past episodes. Recent approaches towards Actor-Critic methods include an entropy term during the policy gradient step to encourage exploration [11]. This new variant, titled Soft Actor-Critic (SAC), proposes a soft policy update given as

$$\nabla_\theta J(\pi_\theta) \tag{6}$$
$$= \mathbb{E}_{s \approx D, a \approx \pi} \nabla_\theta log(\pi_\theta(a|s)(-\alpha log(\pi_\theta(a|s)) + Q_\psi(s,a) - b(s)))$$

where $b(s)$ is a learned state-dependent baseline function.

3 METHODS

Our FedFormer model adapts the SAC algorithm by introducing a contextual federated Q-function (Section 2.2.1). Our only core modification from standard Soft Actor-Critic is the usage of double Q-Networks to help model stability [9].

The overall architecture for our federated Q-function can be seen in Figure 1. Given a network of N agents, we initialize the Q-network of each agent with $N-1$ frozen external encoder networks, each representing the other agents, and one local encoder network. We also initialize a local transformer encoder and a final output decoder network. Each encoder network consists of the same infrastructure, with the exception of each external network having gradients disabled.

The first step during inference is to generate encodings from each encoder network for the same action-observation pair. Following inspiration from BERT, we first encode agent identity through element-wise added learned embeddings according to the encoder network that generated that representation [5] (IdEmbedding). This allows the aggregator network to identify the source of each encoding to potentially aid in learning which agents are known to be more relevant to the current agent. We also include a special `[CLS]` embedding to the input set to encode the global representation of the transformer network. This stage of encodings E is thus given as

$$E = \{FF_i(o,a) + IdEmbedding(i), \forall i \in N, CLSEmbedding\} \tag{7}$$

where FF_i is the encoder network for agent i. We then feed our set of encodings E into a transformer encoder network. Given this set of embeddings, each layer of our transformer encoder network computes:

$$Q = EW_Q,\ K = EW_K,\ V = EW_V, \tag{8}$$

$$Attn(E) = softmax(\frac{QE^\top}{\sqrt{d}})V \tag{9}$$

where $Attn(E)$ is an attention-weighted representation of E and W_Q, W_K, W_V are all learnable weight matrices representing the Query, Key and Value embeddings of the input. In our work, we utilize a two-layer transformer encoder architecture, but this can be tuned in future work.

After aggregating each representation through our transformer encoder, we concatenate the transformed encoding of the `[CLS]` token ($e_{[CLS]}$) with the local encoder representation $FF_l((o,a))$ into a final decoder network, predicting the Q-value for that action-observation pair for agent i. This is given as:

$$Q_i(o,a) = FF_d(e(\texttt{[CLS]}), FF_{local}(o,a)) \tag{10}$$

Since each agent has its own transformer encoder network that is trained according to the agent's own local loss function, the `[CLS]` encoding is therefore trained to be a global representation relevant to the task of that agent. During back-propagation, we fine-tune end-to-end apart from the external encoder networks.

At the end of every epoch, the frozen external networks are updated with the local encoder networks from the other agents. Since each agent needs only to transmit the weights of its local encoder network to the other agents, network bandwidth is increased compared to a FedAvg-based strategy, but not proportionally due to the increase in model complexity. In practice, network bandwidth is kept negligible due to the small size of local encoder networks. For our experiments, each agent was required to upload 1.8MB and download 1.8MB $\times N-1$, which is negligible under modern networks. In addition, we only replace the encoders in the Q-function, not the target Q-function, as the target Q-function is eventually replaced by the Q-function regardless as part of Soft Actor-Critic [11].

An interesting aspect of our architecture is the ease of agent onboarding, the process by which agents are introduced into the federation network. Since transformer networks can scale to a variety of input sizes, onboarding new agents consist of simply adding a new encoder network to the set of external encoder networks. This is significantly different from FedAvg, which could introduce untrained noise into other networks through parameter averaging.

4 RESULTS

4.1 Experimental Design

To evaluate FedFormer, we utilize the Meta-World benchmark [37], which includes a set of robotic manipulation tasks, each able to be parameterized with random variables. As a result, we can initialize a wide variety of diverse heterogeneous environments while maintaining the same task, a challenge for traditional FedAvg methods. For this work, we evaluate our methods on a set of seven tasks that concern operating a robotic arm to push buttons, open and close windows, doors, and drawers as well as reach specific destinations, among other complex tasks. [1]

We compare our solution against Federated Averaging (FedAvg) [18], Weighted FedAvg (FedWeightedAvg) as implemented in Chu et al. [3] and non-federated Soft Actor-Critic (SAC) [11]. As discussed in Section 2, recent advances in federated reinforcement learning are grounded in federated averaging with many domain-specific adjustments. Therefore, we focus our evaluation on comparing underlying federation strategies as our method can behave as a direct replacement for FedAvg. We also compare our methods against SAC to measure performance degradation by preserving privacy in a federated setting. Our experiments against SAC are trained on a centralized set of environments used by the federation methods. It is important to note that performance degradation is expected due to the restrictions of privacy preservation. As such, matching or exceeding non-federated methods is an extremely difficult benchmark and is often referred to as an upper limit for federated learning [23].

In each experiment, we follow the same hyper-parameters as proposed for SAC in the Meta-World paper [37], with the exception that we train for 250 epochs with 600 training iterations during each epoch due to computing limitations. We utilize a batch size of 1200 steps and keep a replay buffer of size 10^6. Each agent comprising the evaluated federation strategies learns from a set of five sampled environment configurations and is tested on a different set of five sampled environments. As such, each agent has access to a small constrained set of heterogeneous environments. Both federation strategies (FedAvg and our FedFormer) have five participating agents unless otherwise stated. For non-federated SAC, the single agent learns from the set of all sampled environments: 25 environments for training and 25 environments for testing. To ensure the robustness of our experiments, we utilize ten random seeds for each method and average the results with error bars reflecting standard error, represented as shaded areas in the figures.

In each experiment, we use a three-layer feed-forward network with two output heads to predict the mean and variance of a tanh distribution over the action space as the policy network. For FedFormer, our Q-network consists of a two-layer transformer model with a hidden size of 400. Each encoder and decoder network consists of a three-layer feed-forward network with each layer having a hidden size of 400 except for the final decoder layer, which consists of a single output unit representing the Q-value. For FedAvg and SAC, the Q-network consists of a three-layer feed-forward network each with a hidden size of 400, apart from the last layer which has a single output unit representing the Q-Value. We include a detailed list of all hyperparameters in Table 1.

Table 1: Hyperparameters Used

Parameter	Value
Batch Size	1200
Number of Epochs	250
Path Length	500
Gradient Steps per Epoch	600
Discount Factor	0.99
Optimizer	Adam
Policy Hidden Sizes	(256, 256, 256)
Policy Activation Function	ReLU
Policy Learning Rate	3×10^{-4}
Policy Minimum Standard Deviation	e^{-20}
Policy Maximum Standard Deviation	e^{2}
Soft Target Interpolation	5×10^{-3}
Use Automatic Entropy Tuning	True
FedFormer	
Transformer Layers	2
Transformer Heads	4
Transformer Hidden Sizes	256
Encoder Hidden Size	(256, 256, 256)
Decoder Hidden Size	(256, 256, 256)
Batch Normalization	True
Activation Function	ReLU
Q-Function Learning Rate	3×10^{-4}
SAC and FedAvg	
Hidden Size	(256, 256, 256)
Activation Function	ReLU
Q-Function Learning Rate	3×10^{-4}

To implement our work, we modified an implementation of SAC in RLKit [24] with additional modules borrowed from Garage [4] and PyTorch [22]. RLKit and Garage are licensed under the MIT license[2] and Pytorch is licensed under the BSD-3 "Revised" license [3]. Experiments were run in parallel on a computing cluster consisting of RTX 3080, RTX 2080Ti, V100 and P100 GPUs, depending on availability.

4.2 Meta-World Evaluation

In the first set of experiments, we compare the performance of FedFormer (blue) against FedAvg [18] (orange), Weighted FedAvg [3] (red) and non-Federated SAC [11] (green) on seven Meta-World

[1] While the Meta-World benchmark can assess how well a meta-learning agent improves over a collection of tasks, we instead evaluate how well our solution performs on each of the several different tasks individually.

[2] https://spdx.org/licenses/MIT.html

[3] https://github.com/pytorch/pytorch/blob/master/LICENSE

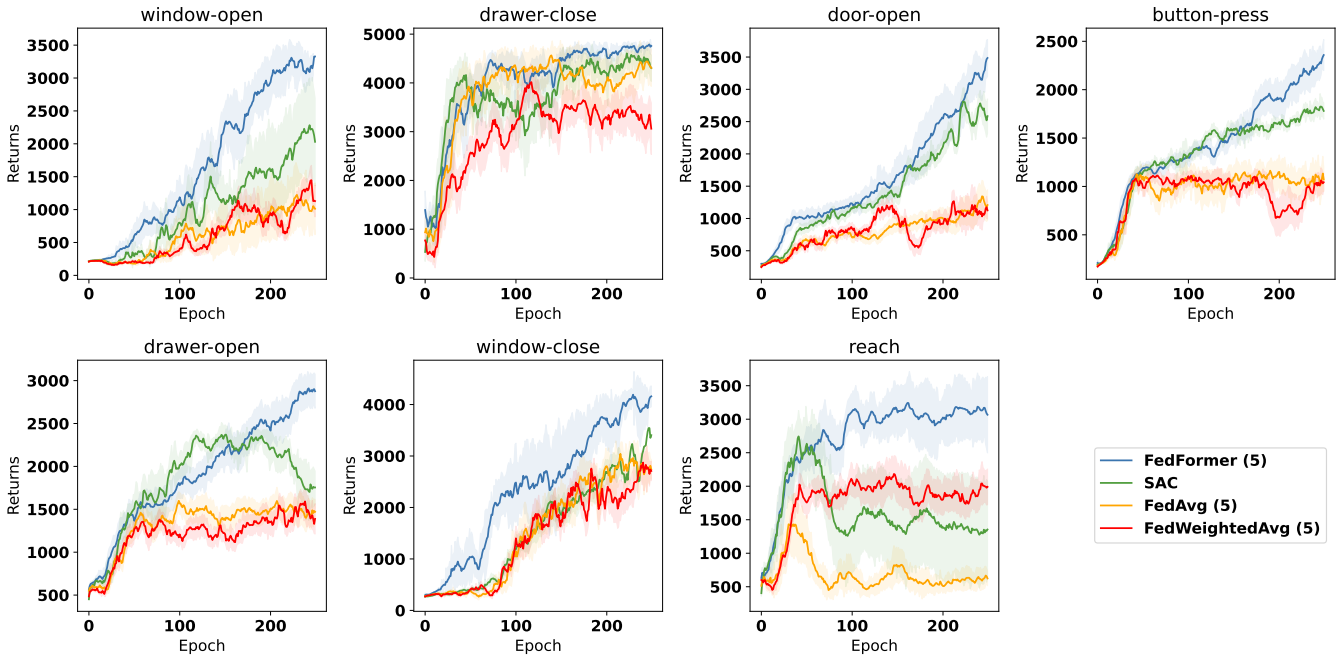

Figure 2: Performance on Meta-World tasks

tasks. Figure 2 shows the results, where the X-axis represents training epochs and the Y-axis is the average episodic return. Improvements in the X-axis indicate that the agent is able to learn faster, and the Y-axis indicates enhanced performance, where the agent outperforms other agents. Further, separation in standard error (shaded area) between each method indicates improved performance based on 10 evaluated seeds.

FedFormer outperforms FedAvg and Weighted FedAvg and matches or exceeds the performance of non-federated SAC in all of our evaluated tasks. As demonstrated by the standard error bars, FedFormer demonstrates consistently superior performance to FedAvg across six of the seven evaluated tasks. The only exception is the `drawer-close` task, in which all methods achieved similar returns.

The performance difference between FedFormer and FedAvg increases as training progresses. At the final epoch, FedFormer achieves 2.17x (`door-open`), 1.77x (`drawer-open`), 1.48x (`window-close`), 2.25x (`button-press`), 3.41x (`reach`) and 2.60x (`window-open`) performance gains over FedAvg. The only task where FedAvg performs comparably to FedFormer is on the `drawer-close` task. We also see that FedAvg behaves similarly to FedWeightedAvg with the exception of `window-close`, where FedWeightedAvg achieves much stronger performance.

Notably, increases in effectiveness can be seen in the `button-press`, `drawer-open` and `door-open` tasks, where FedAvg appears to plateau early in performance around epoch 50. This is contrasted by FedFormer, which continues to grow in performance. We hypothesize that this can be attributed to stifled exploration in FedAvg, a key feature of reinforcement learning that differs from traditional machine learning. When an individual agent makes considerable gains in FedAvg, it is forced to average its parameters with those of the other agents, which could be performing much worse. This is overcome by FedFormer, which allows for attentive and selective aggregation between agents according to their learned relevance.

Most surprising are the results compared to non-federated SAC. In the `door-open`, `reach`, `window-open` and `window-close` tasks, FedFormer consistently exceeds the performance of SAC at each epoch, robust to standard error. Focusing on the final epoch, we also obtain 1.56x performance gains in the `drawer-open` task, 1.36x performance gains in the `button-press` task, and 1.58x performance gains in the `window-open` task. In each other task, we obtain similar performance to SAC. Recall that SAC agents are centralized and train on the entire set of environments without preserving privacy, whereas FedFormer abides by the privacy constraints of federated learning with a distributed pool of agents. We attribute many of these gains to our transformer aggregation network, which can contextually leverage the individual expertise of agents without forcing a centralized model.

4.3 Scalability

Next, we measure the ability of our method to scale to larger sets of agents. To do this, we compare the results of both federation strategies (FedAvg and our FedFormer) when scaled to 5, 10 and 15 agents (Figure 3) [4]. As was the case in our tests in Section 4.2, each agent is given five sampled environments for training and five sampled environments for testing without replacement. Since Meta-World generates 50 environments per task, some agents will have overlapping tasks for training that will be testing tasks for others, but never in the same agent. It is assumed that including

[4] Although we evaluate our methods up to 15 agents, we can scale up to 512 agents (or 1024 with sparse variants) due to our usage of transformers

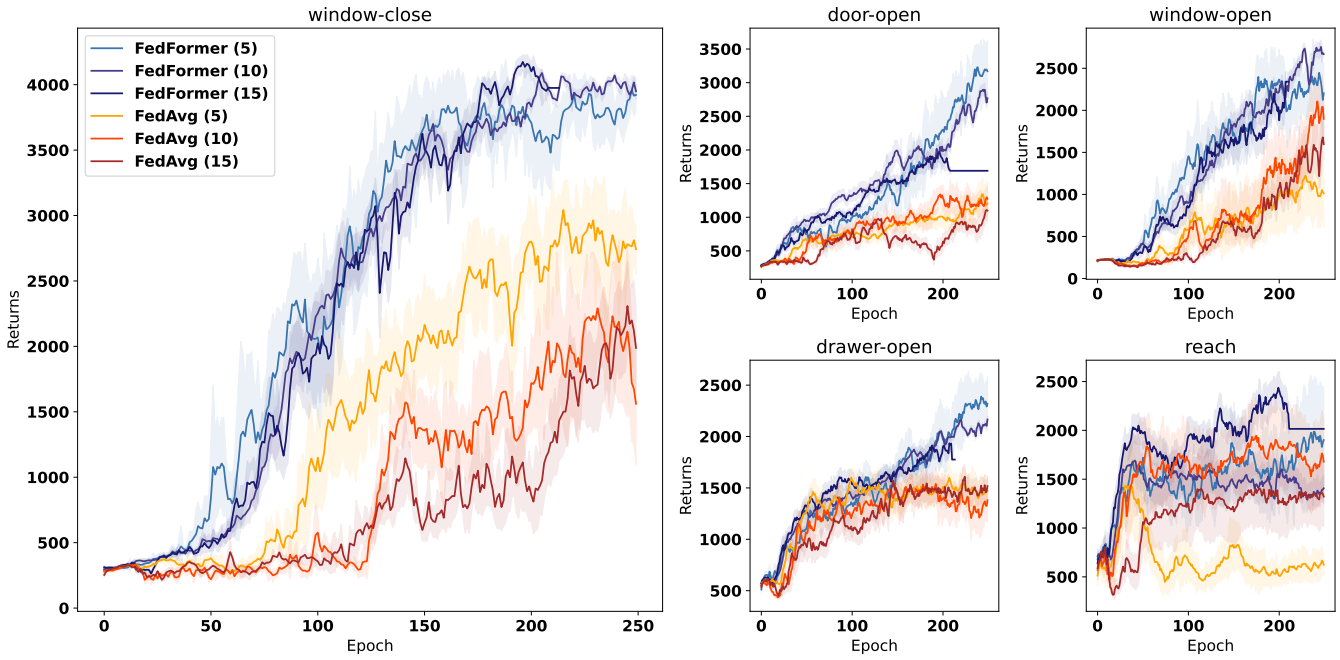

Figure 3: Performance when scaling federation methods to 5, 10 and 15 agents

more agents in federated learning should result in more robust models due to the larger availability of training data. However, it is also important to note that the performance reported is the average performance of *all* agents. Therefore, all agents must perform well in order to achieve a high metric in our results. It is important to note that scaling poses a more difficult problem due to the increased heterogeneity of the environments that each agent is tasked with.

We find that increasing the number of agents results in better performance in FedFormer and worse performance for FedAvg. Focusing on the final epoch and when utilizing 10 agents, we can see performance improvements of 2.23x (`window-close`), 2.95x (`door-open`), 1.47x (`drawer-open`), 1.83x (`reach`) and 2.70x (`window-open`) compared to FedAvg. We see similar gains with 15 agents, where we can see performance improvements of 1.98x (`window-close`), 2.75x (`door-open`), 1.62x (`drawer-open`), and 1.88x (`window-open`). We also see that FedAvg fails to meaningfully scale above FedAvg (5), with FedAvg (5) and FedAvg (10) performing comparably or worse across all tasks. Many of the results reported are also robust to standard error. We hypothesize the poor performance of FedAvg is due to many agents attempting to explore the environment simultaneously but subsequently being forced into a single representation. In our experimentation, we found FedWeightedAvg to exhibit similar trends as FedAvg with marginal improvements as the number of agents increased.

As a result, we conclude that our federation strategy can efficiently scale to include multiple agents. This result aligns with previous work in transformer models, which has shown the ability to reason over larger sets of embeddings [5].

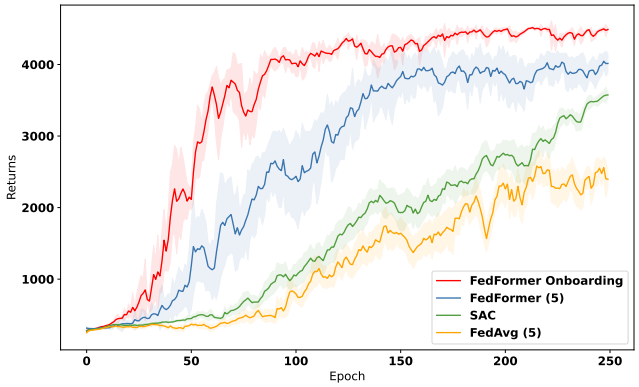

Figure 4: Performance of onboarding agents into pre-trained federation networks on the `window-close` task

4.4 Agent Onboarding

In real-life use cases, agents are often introduced at random intervals into federation strategies. When an agent joins an established federation network, the network must adapt to account for the new environments introduced by the new agent. In the case of FedAvg, this can be a significant setback as model averaging can introduce noise into existing converged models when the models originate from heterogeneous environments. In addition, FedAvg can require all agents to restart training to keep up with the modifications of the new agent.

We now evaluate the onboarding strategy of FedFormer, which overcomes all these challenges (Section 3). To do this, we train a

FedFormer network on the window-close task and save the resulting converged encoder networks. We then train a single new agent on five never-before-seen environments utilizing the saved encoder networks of the previous agents as the external network set. The main difference here is that we do not update the external networks throughout training by the federation. In our experiments, we evaluate our methods by utilizing the resulting trained networks from each seed of the window-close task. The results for this experiment compared against FedFormer, SAC and FedAvg can be seen in Figure 4.

We see that the onboarded agent (red) drastically benefits from the pre-trained encoder networks. Within the first 60 epochs, the new agent obtains near the equivalent performance of the original FedFormer network after it had trained for 250 epochs, obtaining a $\approx$ 5x efficiency increase. We also obtain 6.5x performance gains over SAC and 11.23x performance gains over FedAvg at epoch 60, demonstrating the increased speed of convergence. In addition, performance continues to improve beyond this point, obtaining a 1.24x increase over FedFormer at epoch 250. It is important to note that this onboarding process was done without any further training from the other agents.

4.5 Ablation Study

To evaluate the usefulness of attention in our federation network, we have conducted additional experiments replacing the Transformer network in FedFormer with a three-layer non-attention multi-layer perceptron model (FedMLP). To do this, we concatenate the embeddings generated by each of the encoder networks as the input to the MLP. Then, the output layer of the MLP generates an embedding of size 256, matching the dimensionality of the CLS embeddings created by FedFormer. Just like FedFormer, this embedding is then decoded by a local decoder network to generate the respective Q value for that state-action pair (Figure 1). This baseline is similar to work by Zhuo et al. [40], which proposed concatenating the Q-network embeddings of two agents into a global MLP. Our results when scaling this method from 5, 10 and 15 agents can be seen in Figure 5, following the procedure outlined in Section 4.3. It is important to note that scaling poses a more difficult problem due to the increased heterogeneity of the environments that each agent is tasked with.

In each of the tasks evaluated, we see the dominant performance of FedFormer over the FedMLP baseline. This can be most evidently seen in the window-close task, where FedFormer (5) often achieves over 2x the performance of FedMLP (5) across each epoch. It is important to note the large differences between the two methods when scaled to include larger agent pools. The largest difference can again be seen on the window-close task, where the performance of FedMLP drops by 2.4x when scaled to 10 agents and 3.8x when scaled to 15 agents. This is contrasted by FedFormer, where the performance is either maintained or increased with increased agents. These results indicate the importance of the attention mechanism for FedFormer, especially in relation to scalability. With attention, it is possible to contextually select and aggregate the insights of participating agents in relation to the needs of the current agent. This is contrasted by FedMLP, which must aggregate the insights of all agents, including agents that provide irrelevant input. The problem of noisy inputs is then further exacerbated with increased agent counts, as seen in Figure 5.

5 RELATED WORK

A similar field to federated reinforcement learning is multi-agent reinforcement learning (MARL). MARL focuses on tasks where multiple agents interact in the same environment and share insights with each other to solve a cooperative or competitive task [28]. An important challenge with these systems is the non-stationarity of the environment, where each agent has their own perspective on the environment and actions committed by a single agent can affect the environment for another [38]. Therefore, it becomes critical that multiple agents can efficiently share information, similar to federated learning [8, 17, 29]. However, MARL allows for the constant flow of observation information between agents and often uses a centralized model to plan the actions of each agent [12, 17], therefore making the methods unsuitable for federated learning.

One MARL approach related to our work is Multi-Agent-Attention-Critic (MAAC), proposed by Iqbal and Sha [14]. This method proposed utilizing transformer models to coordinate cooperative agents to solve a collective task. During each timestep, each agent broadcasts their local observation-action pairs to a centralized model. These inputs are then aggregated using a transformer encoder into a set of Q-Values corresponding to each agent, which is then broadcast back to each agent. As a result, each resulting Q-value is created with respect to each other agent's observations and the model is fine-tuned end-to-end to optimize a joint goal.

While our approach also utilizes transformer models, there are significant differences, notwithstanding the obvious constraints of privacy. The transformer model in MAAC is used for cooperative learning, where the central model coordinates observation embeddings from many agents towards a shared goal. In our work, each agent has their own decentralized transformer network that is optimized for aggregating embeddings toward that agent's goal. In addition, the input to each agent is a set of embeddings encoding the same local observation but with different federated encoder networks, whereas MAAC requires observation sharing from each agent during each timestep. We also include elements from natural language processing, such as [CLS] tokens to encode global representations and position encodings [5].

6 CONCLUSION

We presented a novel federation strategy based on transformer models, FedFormer. Our technique addresses concerns about utilizing federated learning towards reinforcement learning, which is typically done through the FedAvg method of averaging parameters. Instead, our method utilizes attention to compute contextual relationships between agents without compromising the ability of individual agents to explore the environment.

Our method outperforms FedAvg and single-agent Soft Actor-Critic (SAC), with FedFormer outperforming FedAvg by a factor of 4.4 and SAC by a factor of 1.75. In addition, we demonstrate that FedFormer can effectively scale to more agents, with gains of 2.4x and 3.8x over FedAvg with 10 and 15 agents respectively. This contrasts with FedAvg, which has negligible performance gains and at times performs worse. Lastly, we also demonstrate the ease

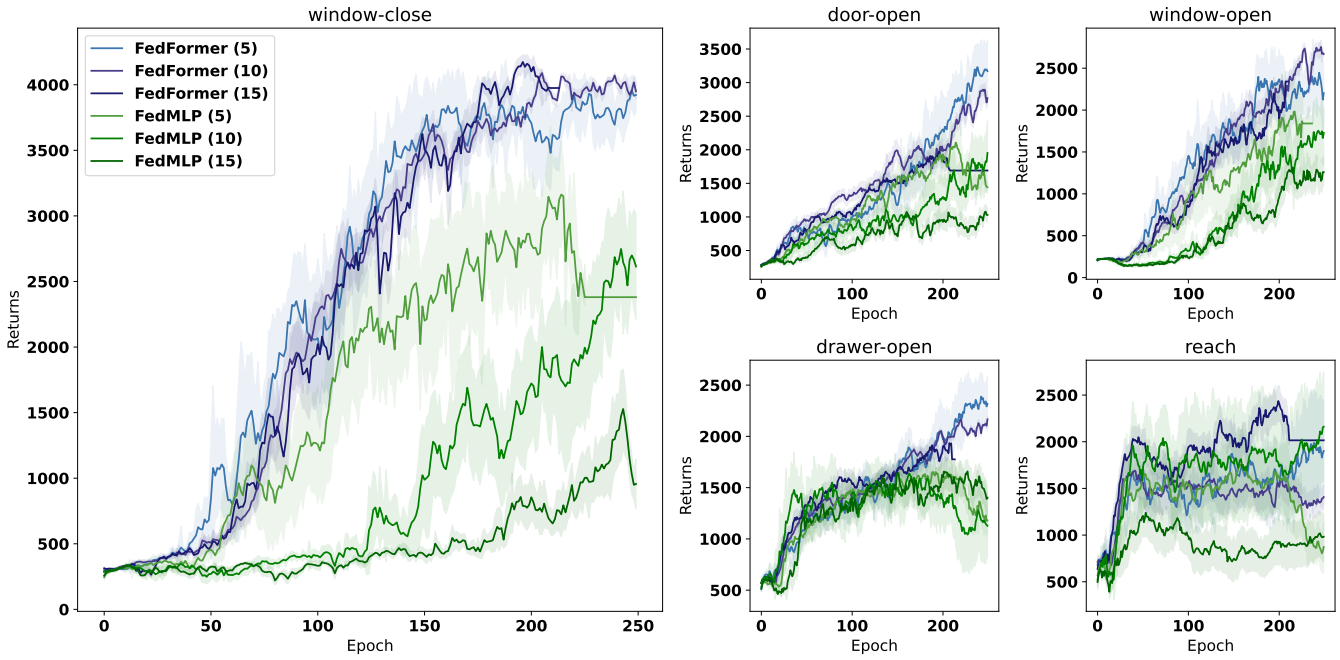

Figure 5: Scaling Performance when replacing Transformer Network with MLP Network

of agent onboarding in our methods, resulting in 5x convergence speeds by leveraging pre-trained models without sacrificing the performance of other models. All of this is achieved while still abiding by the privacy constraints of federated learning.

The primary limitation of our work is related to computational latency. Previous work in transformer models states a computational limit of $\approx$ 512 embeddings when computing attention [5]. This is due to the quadratic increase in latency with additional embeddings. Further, each new agent will add an encoder network to the model. As a result, this limits the number of agents that can participate as part of the federation network. However, this limit can be addressed by utilizing domain-specific agent sampling strategies, such as those explored by Ren et al. [25], Wang et al. [34] and Tehrani et al. [31].

With our advancements with introducing FedFormer, there are several future work directions that could be pursued. First, since our approach does not rely on averaging parameters, it would be interesting to explore the use of deep learning models to represent the encoder networks [19]. This could potentially allow for the federation to tackle more complicated tasks involving computer vision and partial observability by leveraging deep encoders such as convolutional neural networks and recurrent neural networks. The utilization of these models would have been previously difficult due to instability brought on by parameter averaging [23].

Another interesting direction for future work is to explore related work in supervised heterogeneous federated learning. This area proposes techniques to group and filter agents, including to address tolerance to faults from byzantine agents [10]. These techniques support learning from a set of changing heterogeneous environments but often assume that the environments (datasets) of individual agents remain consistent [10]. While this is often not the case in federated reinforcement learning, these techniques still present interesting insights to explore. For example, Huang et al. [13] could be adapted to cluster agents based on their Q-values, Fang and Ye [7] could be repurposed to address the detection of byzantine agents. The work of Fan et al. [6] and Chen et al. [2] will also become relevant. Byzantine filtering layers proposed by Fan et al. [6] could be applied on the input encoder networks; Chen's insights into theoretical guarantees may inspire us to expand beyond our current experimental validation. Since both Fan et al. [6] and Chen et al. [2] are grounded in FedAvg, using FedFormer instead with these approaches may provide important steps forward.

Lastly, our work opens the door for advances in existing FedAvg-based architectures. From IoT devices to self-driving cars, we believe that our methods can serve as a direct stand-in for FedAvg to provide better performance. Further evaluation of these tasks with the addition of domain-specific adjustments such as those proposed in previous work [21, 25, 31, 34] would be especially interesting. We hope that our work can be used to improve federated reinforcement learning and introduce advancements in deep learning to this critical field, especially as privacy becomes harder to obtain in favour of performance.

ACKNOWLEDGMENTS

The authors thank the Natural Sciences and Engineering Research Council of Canada, the Canada Research Chairs Program, the Vector Institute and the University of Waterloo Cheriton Scholarship for financial support. We are also grateful to the reviewers for their valued feedback on the paper.

REFERENCES

[1] Fanyu Bu and Xin Wang. 2019. A smart agriculture IoT system based on deep reinforcement learning. *Future Generation Computer Systems* 99 (2019), 500–507.

[2] Yiding Chen, Xuezhou Zhang, Kaiqing Zhang, Mengdi Wang, and Xiaojin Zhu. 2022. Byzantine-Robust Online and Offline Distributed Reinforcement Learning. *arXiv preprint arXiv:2206.00165* (2022).

[3] Yunfei Chu, Zhinong Wei, Xicheng Fang, Sheng Chen, and Yizhou Zhou. 2022. A Multiagent Federated Reinforcement Learning Approach for Plug-In Electric Vehicle Fleet Charging Coordination in a Residential Community. *IEEE Access* 10 (2022), 98535–98548. https://doi.org/10.1109/ACCESS.2022.3206020

[4] Garage Contributors. 2019. Garage: A toolkit for reproducible reinforcement learning research.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186.

[6] Xiaofeng Fan, Yining Ma, Zhongxiang Dai, Wei Jing, Cheston Tan, and Bryan Kian Hsiang Low. 2021. Fault-tolerant federated reinforcement learning with theoretical guarantee. *Advances in Neural Information Processing Systems* 34 (2021), 1007–1021.

[7] Xiuwen Fang and Mang Ye. 2022. Robust Federated Learning With Noisy and Heterogeneous Clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10072–10081.

[8] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems* 29 (2016).

[9] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[10] Dashan Gao, Xin Yao, and Qiang Yang. 2022. A Survey on Heterogeneous Federated Learning. *arXiv preprint arXiv:2210.04505* (2022).

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[12] Yanlin Han and Piotr Gmytrasiewicz. 2019. IPOMDP-Net: A Deep Neural Network for Partially Observable Multi-Agent Planning Using Interactive POMDPs. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press.

[13] Wenke Huang, Mang Ye, and Bo Du. 2022. Learn From Others and Be Yourself in Heterogeneous Federated Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10143–10153.

[14] Shariq Iqbal and Fei Sha. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2961–2970.

[15] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[16] Hyun-Kyo Lim, Ju-Bong Kim, Joo-Seong Heo, and Youn-Hee Han. 2020. Federated Reinforcement Learning for Training Control Policies on Multiple IoT Devices. *Sensors* 20, 5 (2020).

[17] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, and Zhibo Gong. 2018. Modelling the dynamic joint policy of teammates with attention multi-agent DDPG. *arXiv preprint arXiv:1811.07029* (2018).

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[20] Mehdi Mohammadi, Ala Al-Fuqaha, Mohsen Guizani, and Jun-Seok Oh. 2017. Semisupervised deep reinforcement learning in support of IoT and smart city services. *IEEE Internet of Things Journal* 5, 2 (2017), 624–635.

[21] Chetan Nadiger, Anil Kumar, and Sherine Abdelhak. 2019. Federated reinforcement learning for fast personalization. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, 123–127.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[23] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. 2021. Federated reinforcement learning: Techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887* (2021).

[24] RAIL-Berkeley and RLKit contributors. 2021. RLKit: Reinforcement learning framework and algorithms implemented in PyTorch.

[25] Jianji Ren, Haichao Wang, Tingting Hou, Shuai Zheng, and Chaosheng Tang. 2019. Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things. *IEEE Access* 7 (2019), 69194–69201.

[26] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep Reinforcement Learning framework for Autonomous Driving. *Electronic Imaging* 29, 19 (2017), 70–76.

[27] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295* (2016).

[28] Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383.

[29] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. *Advances in neural information processing systems* 29 (2016).

[30] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[31] Peyman Tehrani, Francesco Restuccia, and Marco Levorato. 2021. Federated Deep Reinforcement Learning for the Distributed Control of NextG Wireless Networks. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 248–253.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[33] Xiaofei Wang, Ruibin Li, Chenyang Wang, Xiuhua Li, Tarik Taleb, and Victor CM Leung. 2020. Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. *IEEE Journal on Selected Areas in Communications* 39, 1 (2020), 154–169.

[34] Xiaofei Wang, Chenyang Wang, Xiuhua Li, Victor CM Leung, and Tarik Taleb. 2020. Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching. *IEEE Internet of Things Journal* 7, 10 (2020), 9441–9455.

[35] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3 (1992), 229–256.

[36] Xiong Xiong, Kan Zheng, Lei Lei, and Lu Hou. 2020. Resource allocation based on deep reinforcement learning in IoT edge computing. *IEEE Journal on Selected Areas in Communications* 38, 6 (2020), 1133–1146.

[37] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*. PMLR, 1094–1100.

[38] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.

[39] Zixuan Zhang, Yuning Jiang, Yuanming Shi, Ye Shi, and Wei Chen. 2022. Federated Reinforcement Learning for Real-Time Electric Vehicle Charging and Discharging Control. In *2022 IEEE Globecom Workshops (GC Wkshps)*. 1717–1722. https://doi.org/10.1109/GCWkshps56602.2022.10008598

[40] Hankz Hankui Zhuo, Wenfeng Feng, Yufeng Lin, Qian Xu, and Qiang Yang. 2019. Federated deep reinforcement learning. *arXiv preprint arXiv:1901.08277* (2019).