EFFICIENT TRANSFER LEARNING FOR HETEROGENEOUS MACHINE LEARNING DOMAINS

By

Zhuangdi Zhu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2022

ABSTRACT

EFFICIENT TRANSFER LEARNING FOR HETEROGENEOUS MACHINE LEARNING DOMAINS

By

Zhuangdi Zhu

Recent advances in deep machine learning hinge on a large amount of labeled data. Such heavy dependence on supervision data impedes the broader application of deep learning in more practical scenarios, where data annotation and labeling can be expensive (e.g. high-frequency trading) or even dangerous (e.g. training autonomous-driving models.) Transfer Learning (TL), equivalently referred to as knowledge transfer, is an effective strategy to confront such challenges. TL, by its definition, distills the external knowledge from relevant domains into the target learning domain, hence requiring fewer supervision resources than learning-from-scratch. TL is beneficial for learning tasks for which the supervision data is limited or even unavailable. It is also an essential property to realize Generalized Artificial Intelligence.

In this thesis, we propose sample-efficient TL approaches using limited, sometimes unreliable resources. We take a deep look into the setting of Reinforcement Learning (RL) and Supervised Learning, and derive solutions for the two domains respectively. Especially, for RL, we focus on a problem setting called imitation learning, where the supervision from the environment is either non-available or scarcely provided, and the learning agent must transfer knowledge from exterior resources, such as demonstration examples of a previously trained expert, to learn a good policy. For supervised learning, we consider a distributed machine learning scheme called Federated Learning (FL), which is a more challenging scenario than traditional machine learning, since the training data is distributed and non-sharable during the learning process. Under this distributed setting, it is imperative to enable TL among distributed learning clients to reach a satisfiable generalization performance. We prove by both theoretical support and extensive experiments that our proposed algorithms can fa-

cilitate the machine learning process with knowledge transfer to achieve higher asymptotic performance, in a principled and more efficient manner than the prior arts.

This thesis is dedicated to my parents, Yi Zhang and Jun Zhu,
as well as my fiancé Tianxudong Tang.

ACKNOWLEDGEMENTS

This piece of writing marks the end of a marvelous journey and the beginning of a new one, none of which would be possible without the support of my advisor, colleagues, friends, and loved ones.

First and foremost, I would like to express my deep gratitude to my research advisor, Dr. Jiayu Zhou. I could not have undertaken this journey without his guidance and help. I thank him for his vision, support, his kind advice on research and beyond. I am also grateful to my other committee members: Dr. Anil K. Jain, Dr. Panning Tan, Dr. Sijia Liu, and Dr. Zhaojian Li, for providing their expertise and informative feedback.

It is my great pleasure to have worked with members of our Illidan Lab, especially Kaixiang Lin, Junyuan Hong, and Mengying Sun. I thank them for their selfless help in improving my skills in research. I would like to extend my sincere thanks to Dr. Philip Quinn for providing me with patient mentoring and warm friendship. Thanks should also go to Dr. Bo Dai, a superb researcher who has been influential to my research. They have all encouraged me for being a qualified PhD candidate.

I would like to shout out to my fiancé, Tang, a kind and loving man who constantly gives me emotional support. He loves me for who I really am and has inspired me to discover the better part of myself. I also thank my friends, especially Xin H, Michelle, Mengmeng, Mimi, Han M, Junyuan H, He L, Yan X, Chuan-Pin C, and Fei Z. They are sources of joyful distractions from my research and work, and together we have shared many treasured memories. Thanks to them, I began to understand that slowly is the fastest way to complete a journey.

I am deeply indebted to my parents who give me a lifetime of love and care. Their belief in me has spiritually supported me during the ups and downs of this PhD journey. They love me deeply by not holding the reins but setting me off on the road to thrive. Special thanks to my younger sister HaoHao, who has grown up so quickly with a good sense of

responsibility. Thanks to her for accompanying our parents when I am not home. I am fortunate to have her as my sibling.

Lastly, I would like to thank my grandfather. His kindness and integrity have influenced me deeply. Hope that I have made him proud. Dear grandpa, may your good soul rest in peace. I miss you so much, and I will always, always love you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

1.1 Motivation

Recent advances in deep machine learning hinges on a large amount of labeled data. Such heavy dependence on supervision data impedes the broader application of deep learning in more practical scenarios, where data annotation and labeling can be expensive (*e.g.* high-frequency trading) or even dangerous (*e.g.* training autonomous-driving models.) ***Transfer Learning (TL)***, which is equivalently referred to as *knowledge transfer*, serves as an effective strategy to confront such challenges. TL, by its definition, distills the external knowledge from relevant domains into the target learning domain, hence requiring fewer supervision resources than learning-from-scratch. It is beneficial for learning tasks for which the supervision data is limited or even unavailable. For example, assisted with knowledge transfer, an auto-driving model trained in a simulation system can be readily deployed to the real world. The performance of such models might be otherwise mitigated due to the distribution drift from simulation to reality.

In fact, the ability to transfer knowledge is essential to realize Generalized Artificial Intelligence. Learning from external expertise has been widely observed among humankind. For instance, toddlers learn to wobble around from following their parents [92], and a beginner player in video gaming may grow into an expert by watching footage of senior players. Such inherent ability, however, can be challenging for AI models to acquire. An AI model that is well trained on a source domain may perform poorly in a target domain of interest, given only a slight change in the data distribution [92]. Therefore, enabling knowledge transfer across different learning scenarios remains an intriguing research topic in machine learning.

Observing the merits of transfer learning, we tackle knowledge transfer in two major machine learning settings: *Reinforcement Learning (RL)* and *Supervised Learning*. Pio-

neer efforts in knowledge transfer have been made for both problem settings. For example, transferring knowledge from external demonstrations to perform imitation learning is shown effective to realize plenty of RL tasks, including robot navigation [90, 85] and game playing [107, 156]. On the other hand, learning domain-transferrable feature representations has enabled domain adaptation in various supervised learning tasks, such as computer vision [179, 120] and natural language processing [37].

Along with their promising results, we notice some non-negligible limitations of prior efforts in transfer learning, mainly in fwo-folds: First, most transfer learning approaches ***hinges on sufficient data*** from the source domain or a relevant domain. For instance, when transferring the knowledge from a teacher model to the student model, prior approaches usually require a large set of data from the source domain, or from a domain with resemblant data distribution. Otherwise, the process of TL is non-feasible due to the lack of such a proxy dataset. Similarly, in the domain of RL, previous TL approaches usually require a large bunch of teacher demonstrations [151], or frequent feedbacks from the teacher policy [145], in order to provide enough guidance for the learning agent. Second, the *learning* ***efficiency*** (in the context of supervised learning), or *sampling efficiency* (in the context of RL), can be further improved for existing TL approaches. Low sampling efficiency can be a crucial problem for RL, which means a learning agent may take a prohibitively long time, exploring trajectories randomly with low returns. Especially, existing TL approaches in RL may still require *on-policy* learning, which is a costly sampling strategy. Analogously, in supervised learning, low learning efficiency could lead to more training iterations before convergence, which are computationally costly, or even incur extra communication burdens to distributed machine learning applications, such as Federated Learning [116].

Observing the potential limitations of prior arts, in this thesis, we propose *sample-efficient* TL approaches using *limited*, sometimes *unreliable* resources in a principled manner. We take a deep look into the setting of RL and *Supervised Learning*, and derive solutions for the two domains respectively. Especially, for RL, we focus on a problem setting called *imitation*

learning, where the supervision from the environment is either non-available or scarcely provided, and the learning agent must transfer knowledge from exterior resources, such as demonstration examples of a previously trained expert, to learn a good policy. For supervised learning, we consider a distributed machine learning scheme called ***Federated Learning (FL)*** [116], which is a more challenging scenario than traditional machine learning, since the training data is distributed and non-sharable during the learning process. Under this distributed setting, it is therefore imperative to enable TL among distributed learning clients to reach a satisfiable generalization performance.

1.2 Overview of Thesis Structure

This section summarizes each of the chapters in this thesis.

In Section 1.3, we present the preliminaries about two different problem settings, *i.e.* RL and supervised federated learning, under which we proposed our transfer learning approaches. We also introduce the principle of *generative adversarial learning* which provides indispensable foundations of our work. In Section 2.1, we clarify the definition of TL in the related machine learning scenarios, then list the terminologies that will be frequently used through this thesis. We also briefly review the recent advances of TL in both RL and supervised learning domains, and point out the distinction between our proposed work and the prior arts before introducing our techniques.

Chapter 3 elaborates the approach *SAIL* , which is an imitation learning approach in a RL problem setting. It realizes knowledge transfer from limited, suboptimal demonstrations to assist the agent learning to achieve expert-level performance with high sample efficiency. In Chapter 4, we present the work of *OPOLO* , which extends the idea of TL from demonstration to a more challenging yet practical setting in RL, *i.e.* to transfer knowledge from *incomplete observations* of expert behavior. In Chapter 5, we shift our focus from RL to an intriguing supervised learning scenario and introduce our work dubbed as *FedGen* , which enables knowledge transfer in FL without the need of accessing training data from the knowl-

edge source. In Chapter 6, we continue the TL study in the context of FL and introduce *FedResCuE*, which enables *self-knowledge distillation* in FL to tackle a systematic heterogeneous FL system with unstable network connections. During FL, knowledge is transferred and preserved in arbitrarily prunable sub-networks of the global model.

In Section 7, we summarize the connection between each of our work, by looking into the following key questions: 1) under which *problem setting* the proposed approach is applicable, 2) what are the *teacher* and *student* regarding the TL process, 3) what kind of *knowledge* has been transferred, and 4) what can be *achieved* upon TL is complete. Next, we explore the potential challenges faced by our TL approaches, point out the open questions that await future research progress, and briefly discuss our ongoing and future work.

1.3 Background and Preliminaries

In this section, we introduce the basic concepts in RL, (supervised) FL, and adversarial generative learning, which composed the cornerstones of our transfer learning research.

1.3.1 Reinforcement Learning (RL)

A typical RL problem can be considered as training an agent to interact with an environment that follows a ***Markov Decision Process* (MPD)** [15]. In an MDP, the agent starts with an initial *state* and performs an *action* accordingly, which yields a *reward* to guide the agent actions. Once the action is taken, the MDP transits to the next state by following the underlying *transition dynamics* of the MDP. The agent accumulates the time-*discounted* rewards along with its interactions with the MDP. A subsequence of interactions is referred to as an *episode*. For MDPs with infinite horizons, one can assume that there are *absorbing states*, such that any action taken upon an absorbing state will only lead to itself and yield zero rewards. All above-mentioned components in the MDP can be represented using a tuple $\mathcal{M} = (\mu_0, \mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{R}, \mathcal{S}_0)$, in which:

- μ_0 is the set of ***initial states***.

- $\mathcal{S}$ is the ***state*** space.
- $\mathcal{A}$ is the ***action*** space.
- $\mathcal{P} : S \times A \times S \to \mathbb{R}$ is the ***transition probability distribution***, where $\mathcal{P}(s'|s,a)$ specifies the probability of the state transitioning to s' upon taking action a from state s.
- $\mathcal{R} : S \times A \times S \to \mathbb{R}$ is the ***reward distribution***, where $\mathcal{R}(s,a,s')$ is the reward an agent can get by taking action a from state s with the next state being s'.
- γ is a discounted factor, with $\gamma \in (0,1]$.
- $\mathcal{S}_0$ is the set of ***absorbing states***.

Figure 1.1 illustrates the above components, and Figure 1.2 is an example of trajectories in an infinite-horizon MDP. An RL agent behaves in $\mathcal{M}$ by following its policy π, which is a mapping from states to actions: $\pi : \mathcal{S} \to \mathcal{A}$. For stochastic policies, $\pi(a|s)$ denotes the probability for agent to take action a from state s. Given an MDP $\mathcal{M}$ and a policy π, one can derive a ***value function*** $V^\pi_\mathcal{M}(s)$, which is defined over the state space:

$$V^\pi_\mathcal{M}(s) = \mathbb{E}\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots; \pi, s\right],$$

where $r_i = \mathcal{R}(s_i, a_i, s_{i+1})$ is the reward that an agent receives by taking action a_i in the i-th state s_i, and the next state transits to s_{i+1}. The expectation $\mathbb{E}$ is taken over the following distriubtion:

$$s_0 \sim \mu_0, a_i \sim \pi(\cdot|s_i), s_{i+1} \sim \mathcal{T}(\cdot|s_i, a_i).$$

The value-function estimates the *quality* of being in state s, by evaluating the expected rewards that an agent can get from s, given that the agent follows policy π in the environment $\mathcal{M}$ afterward. Similar to the value-function, each policy also carries a Q-**function**, which is defined over the state-action space to estimate the quality of taking action a from state s:

$$Q^\pi_\mathcal{M}(s,a) = \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}\left[\mathcal{R}(s,a,s') + \gamma V^\pi_\mathcal{M}(s')\right].$$

The objective of RL is to learn an optimal policy $\pi^*_{\mathcal{M}}$ to maximize the expectation of accumulated rewards, so that: $\forall s \in \mathcal{S}, \pi^*_{\mathcal{M}}(s) = \arg\max_{a \in A} \; Q^*_{\mathcal{M}}(s,a)$, where $Q^*_{\mathcal{M}}(s,a) = \sup_{\pi} \; Q^{\pi}_{\mathcal{M}}(s,a)$.

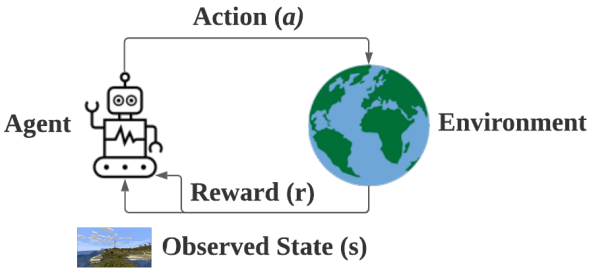


Figure 1.1: Overview of RL.

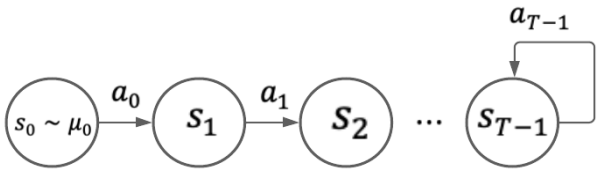


Figure 1.2: An absorbing state (s_{T-1}) denotes the termination of a task in an infinite-horizon MDP.

1.3.2 Federated Learning (FL)

Without ambiguity, in this section, we present a typical FL setting for *supervised learning*, *i.e.*, the general problem of multi-class classification. Let $\mathcal{X} \subset \mathbb{R}^p$ be an instance space, $\mathcal{Z} \subset \mathbb{R}^d$ be a *latent* feature space with $d < p$, and $\mathcal{Y} \subset \mathbb{R}$ be an output space. $\mathcal{T}$ denotes a *domain* which consists of a data distribution $\mathcal{D}$ over $\mathcal{X}$ and a ground-truth *labeling* function $c^* : \mathcal{X} \to \mathcal{Y}$, *i.e.* $\mathcal{T} := \langle \mathcal{D}, c^* \rangle$. Note that we will use the term *domain* and *task* equivalently. A model parameterized by $\boldsymbol{\theta} := [\boldsymbol{\theta}^g; \boldsymbol{\theta}^p]$ consists of two components: a representation learning module $g : \mathcal{X} \to \mathcal{Z}$ parametrized by $\boldsymbol{\theta}^g$, and a predictor $h : \mathcal{Z} \to \triangle^{\mathcal{Y}}$ parameterized by $\boldsymbol{\theta}^h$, where $\triangle^{\mathcal{Y}}$ is the simplex over $\mathcal{Y}$. Given a non-negative, convex loss function $l : \triangle^{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$, the ***risk*** of a model parameterized by $\boldsymbol{\theta}$ on domain $\mathcal{T}$ is defined as $\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) := \mathbb{E}_{x \sim \mathcal{D}} \left[l \left(h(g(x; \boldsymbol{\theta}^g); \boldsymbol{\theta}^h), c^*(x) \right) \right]$.

The objective of FL is to learn a global model parameterized by $\boldsymbol{\theta}$ that minimizes its risk on each of the user tasks $\mathcal{T}_k$ [116]:

$$\min_{\boldsymbol{\theta}} \; \mathbb{E}_{\mathcal{T}_k \in \boldsymbol{\mathcal{T}}} \left[\mathcal{L}_k(\boldsymbol{\theta}) \right], \tag{1.1}$$

where $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_k\}_{k=1}^K$ is the collection of user tasks. We consider all tasks sharing the same labeling rules c^* and loss function l, *i.e.*, $\mathcal{T}_k = \langle \mathcal{D}_k, c^* \rangle$. In practice, Equation 5.1 is empirically optimized by $\min_{\boldsymbol{\theta}} \frac{1}{K}\sum_{k=1}^K \hat{\mathcal{L}}_k(\boldsymbol{\theta})$, where $\hat{\mathcal{L}}_k(\boldsymbol{\theta}) := \frac{1}{|\hat{\mathcal{D}}_k|}\sum_{x_i \in \hat{\mathcal{D}}_k} [l(h(g(x_i; \boldsymbol{\theta}^g); \boldsymbol{\theta}^p), c^*(x_i))]$ is the *empirical* risk over an observable dataset $\hat{\mathcal{D}}_k$. An implied assumption for FL is that the ***global*** data $\hat{\mathcal{D}}$ is distributed to each of the local domains, with $\hat{\mathcal{D}} = \cup\{\hat{\mathcal{D}}_k\}_{k=1}^K$.

1.3.3 Generative Adversarial Learning

Generative Adversarial learning can be traced back to the work of Generative Adversarial Network (GAN) [56], which is a representative solution to generative learning. The goal of GAN is to learn an unknown distribution $p(x)$ using a set of samples from such distribution. More concretely, GAN learns two modules: a discriminator D and a generator G, to jointly optimize the following objective $J(G, D)$:

$$\min_G \max_D J(G, D) := \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim G(z)}[\log(1 - D(G(z)))]. \tag{1.2}$$

The generator G can be later used to synthesize samples to assist downstream model training tasks. GAN performs minimax-optimization on the *Jesen-Shannon divergence* between the ground-truth distribution $p(x)$ and the generated distribution $G(z)$. Recent extensions of GAN have explored other forms of distributional-discrepancy measure, such as the f-divergences [129] or *Wasserstain divergence* [9]. Among different adversarial generative learning approaches, *distribution matching* and *game theory* is at the heart of their foundation. Generative adversarial learning has been applied to solve a variety of problems, such as synthetic data generation [204], adversarial attacks and defenses [192], domain adaptatoin [174], imitation learning [68], etc.

1.3.4 Notations and Terminologies

We list the following terminologies which will be frequently used in this thesis:

1. ***Domain***: we will refer *domain* and *task* equivalently. In the context of RL, a domain denotes a Markov Decision Process $\mathcal{M} = (\mu_0, \mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{R}, \mathcal{S}_0)$. In the context of

supervised learning, a domain $\mathcal{T} = \langle \mathcal{D}, c^* \rangle$ is a composition of a data distribution $\mathcal{D}(x)$ in the input space $\mathcal{X}$ and its target labeling function $c^* : \mathcal{X} \to \triangle^{\mathcal{Y}}$, where $\triangle^{\mathcal{Y}}$ is the simplex of label space $\mathcal{Y}$. Without ambiguity, in this dissertation, we consider a typical supervised learning problem of *multi-class classification*.

2. ***Teacher (Source) and Student (Target)***: We will refer *teacher* and *source* equivalently, and refer *student* and *target* equivalently as well. transfer learning is all about transferring knolwedge from the *teachers (source)* domains to the *student (target)* domains. In the contex of RL, a teacher (student) denotes a policy $\pi : \mathcal{X} \to \mathcal{A}$. In the context of supervised learning, a teacher (student) denotes a predictive model $f = h \circ g : \mathcal{X} \to \triangle^{\mathcal{Y}}$.

Abbreviation	Definition
TL	Transfer Learning
RL	Reinforcement Learning
SL	Supervised Learning
KD	Knowledge Distillation
FL	Federated Learning

Table 1.1: Overview of Abbreviations.

We also present the abbrevations and key notations used in this thesis in Table 1.1 and 1.2.

Symbol	Definition	Context
p	Dimension of input space	SL
d	Dimension of latent space	SL
$\mathcal{X}$	Input Space, $\mathcal{X} \subset \mathbb{R}^p$	SL
x	A vector of input variables, $x \sim \mathcal{X}$	SL
$\mathcal{Z}$	Latent Space, $\mathcal{Z} \subset \mathbb{R}^d$	SL
z	A vector of latent variables, $z \sim \mathcal{Z}$	SL
$\mathcal{Y}$	Output Space, $\mathcal{Y} \subset \mathbb{R}$	SL
y	An output variable, $y \sim \mathcal{Y}$	SL
$\mathcal{D}$	Data Distribution	SL
c^*	Labeling function, $c^* : \mathcal{X} \to \mathcal{Y}$	SL
$\mathcal{T}$	Task, $\mathcal{T} := \langle \mathcal{D}, c^* \rangle$	SL
g	Representation learning module	SL
h	Prediction module	SL
θ	Network paraemeter set	SL & RL
K	Number of (source) domains	SL & RL
$\mathcal{M}$	Markov Decision Process	RL
π	Policy, $\pi : \mathcal{S} \to \mathcal{A}$	RL
τ_π	A trajectory by following policy π, $\tau := \{s_0, a_0, s_1, a_1, \cdots \vert s_{a_t} = \pi(s_t)\}$	RL

Table 1.2: An overview of mathematical notations. A notation can be applicable to either the context Supervised Learning (SL) or Reinforcement Learning (RL).

CHAPTER 2

PROBLEM OVERVIEW: TRANSFER LEARNING

In this chapter, we first present definitions of transfer learning (TL) in two different contexts: reinforcement learning (RL) and supervised learning, respectively. Next, we briefly overview the recent advances of transfer learning approaches in both settings.

2.1 Defitinion of Transfer Learning

2.1.1 Transfer Learning in the Context of Reinforcement Learning

In the context of reinforcement learning, we use $\boldsymbol{\mathcal{M}}_s = \{\mathcal{M}_s^i\}_{i=1}^K$ to denote a set of source domains, and $\mathcal{M}_t$ to denote the target domain. For a minimalistic case, knowledge can transfer between two agents within the same domain, resulting in $|\boldsymbol{\mathcal{M}}_s| = 1$, and $\mathcal{M}_s = \mathcal{M}_t$.

Definition 1. [Transfer Learning in the Context of Reinforcement Learning] *Given a set of* ***source*** *domains $\boldsymbol{\mathcal{M}}_s = \{\mathcal{M}_s^i\}_{i=1}^K$ and a* ***target*** *domain $\mathcal{M}_t$, where each domain is a Markov decision process, transfer learning occurs when an lgorithm A leverages exterior information I_s from $\boldsymbol{\mathcal{M}}_s$ and interior information I_t from $\mathcal{M}_t$ as inputs to generate a policy $\underset{\mathcal{S}^t \to \mathcal{A}^t}{\pi} = A(I_s \sim \boldsymbol{\mathcal{M}}_s, I_t \sim \mathcal{M}_t)$, which achieves higher performance in the target domain $\mathcal{M}_t$, compared with not utilizing the source-domain knowledge I_s.*

In the above definition, we use $A(I_s; I_t)$ to denote the output of a transfer learning algorithm, *i.e.* the learned policy based on information I_s and I_t. One can consider regular RL without transfer learning as an extreme case of the above definition by treating $\boldsymbol{\mathcal{M}}_s = \emptyset$ and $I_s = \emptyset$, so that a policy π is learned purely on the feedback provided by the target domain, *i.e.* $\pi = A(I_t)$. TL is beneficial in improving the RL performance, especially when the supervision from a target domain I_t is lacking or sub-optimal.

We illustrate some eligible types of source-domain knowledge in Figure 2.1, although knowledge from the source domain I_s can also take other forms of supervision.

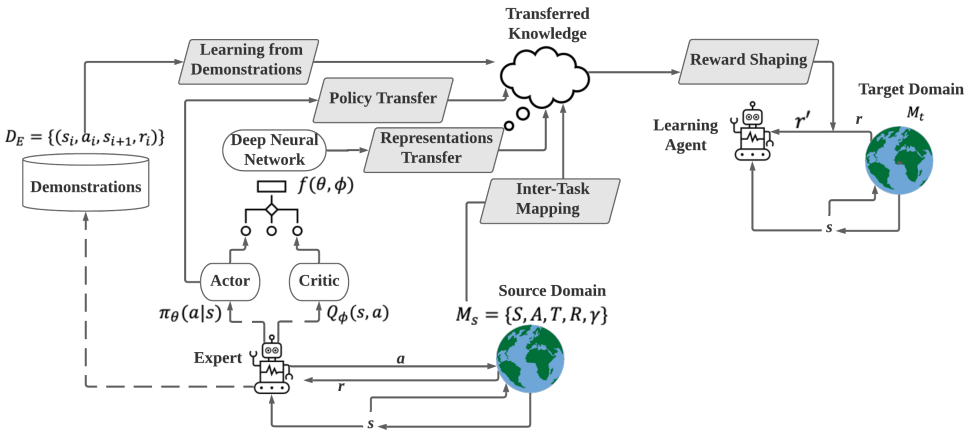


Figure 2.1: TL approaches in RL organized by the format of transferred knowledge.

2.1.2 Transfer Learning in the Context of Supervised Learning

In the context of supervised learning (SL), We use $\boldsymbol{\mathcal{T}}_s = \{\mathcal{T}_s^i\}_{i=1}^K$ to denote a set of source domains, and $\mathcal{T}_t$ to denote the target domain. Below we provide a high-level definition of TL for a supervised learning problem setting:

Definition 2. [Transfer Learning in the Context of Supervised Learning] *Given a set of* ***source*** *domains* $\boldsymbol{\mathcal{T}}_s = \{\mathcal{T}_s^i\}_{i=1}^K$ *and a* ***target*** *domain* $\mathcal{T}_t$*, where each domain is a supervised learning task, with* $\mathcal{T}_s^i = \langle D_s^i, c_i^{s*} \rangle$ *and* $\mathcal{T}_t = \langle D_t, c^{t*} \rangle$*, transfer learning occurs when an lgorithm* A *leverages exterior information* I_s *from* $\boldsymbol{\mathcal{T}}_s$ *and interior information* I_t *from* $\mathcal{T}_t$ *as inputs to generate a predictive model* $\underset{\mathcal{X}^t \to \Delta^{\mathcal{Y}}}{h \circ g} = A(I_s \sim \boldsymbol{\mathcal{T}}_s, I_t \sim \mathcal{T}_t)$*, which achieves lower risk in the target domain* $\mathcal{T}_t$*, compared with not utilizing the source-domain knowledge* I_s*.*

Remark 1 (**Domain of Multi-Class Classification**)**.** *Without losing generality, we also consider that each domain is a multi-class classification problem, although the proposed approaches in this dissertation have the potential to be extended to other scenarios, including regression or multi-label classification.*

2.2 A Glance of Prior Arts

Transfer learning in RL has gained ever-increasing attention due to the recent success of RL in various applications, such as game playing [14] and robotics learning [85]. Traditional transfer learning for RL traces back to a scenario called *behavior cloning*, in which a target policy is trained in a supervised learning manner by leveraging state-action samples from a pretrained teacher [138]. Later approaches are developed to involve the transferred knowledge into the RL learning loop. For instance, in a knowledge transfer approach called *reward shaping* [125], the exterior knowledge can be distilled as a synthetic reward function, which provides extra guidance besides the reward signal from the environment. Based on the form of transferred knowledge, approaches along this line can be mainly categorized into the following: *reward shaping* [182, 38], *learning from demonstrations* [105, 25, 175, 123, 68, 79] *policy transfer* [146, 191, 168, 12], *inter-domain mapping* [60, 7], *representations transfer* [20], etc. There have been prior efforts in summarizing the knowledge transfer techniques in RL [167, 91]. Interested readers are referred to our survey [206] for more recent advances along this line. Our proposed knowledge transfer approaches outstand pior arts in the following aspects: i) our approaches improve the *sample efficiency* in RL by learning from the exterior knowledge in an *off-policy* manner, and ii) we enable knowledge transfer from *accessible* resources, such as sub-optimal or limited teacher demonstrations.

Transfer learning in Supervised Learning is an important machine learning problem that has been studied extensively [132, 207]. This technique has benefited practical supervised learning applications in computer vision, auto-navigation, natural language processing [19, 104], healthcare, etc. While a comprehensive overview of TL in the SL domain requires multi-fold criteria, one angle to investigate TL approaches is to answer the key question of *what* knowledge to transfer. Specifically, appraoches along this line can be mainly organized into three categories: 1) instance based transfer [76, 101, 180, 152], 2) feature representation transfer [174, 203], and 3) model parameter transfer [193, 117]. In instance-based transfer, the training data from the source domain can directly help to learn the target domain by

re-weighting the training samples. In feature representation transfer, the core idea is to learn and leverage latent features that are robust across different domains. For model parameter transfer, the knowledge is conveyed by the parameter set of a teacher model, which is usually more complex in structure than the student model. In Chapter 5, we introduce one of our FL proposed approaches, which can be considered as a hybrid scheme that transfers both model parameters and latent representations. Moreover, in Chapter 6, we investigate how to transfer and preserve knowledge in different model channels of the same model architecture.

As a specific supervised learning scenario, FL has achieved wide success in practical applications such as mobile computing and healthcare. As a result, TL learning in FL has recently emerged as an effective approach to tackle user heterogeneity brought by FL. Most existing work along this line is data-dependent [103, 159, 58, 30]. Particularly, [103] proposed **FedDFusion**, which performs TL to refine the global model, assuming that an unlabeled dataset is available with samples from the same or similar domains. Complementary TL efforts have been made to confront data heterogeneity [96, 150]. Specifically, [96] transmits the proxy dataset instead of the model parameters. FedAUX [150] performs *data-dependent* distillation by leveraging an auxiliary dataset to initialize the server model and to weighted-ensemble user models, while FedGen performs knowledge distillation in a data-free manner. FedMix [194] is a *data-augmented* FL framework, where users share their *batch-averaged* data among others to assist local training. On the contrary, in Chapter 5, we proposed *FedGen* , which extracts knowledge from the existing user model parameters and hence faces fewer privacy risks. From a different yet practical perspective, in Chapter 6, we also investigate the role of self-knowledge distillation in achieving robust and efficient FL that confronts heterogeneous systems and faulty network connections.

CHAPTER 3

KNOWLEDGE TRANSFER IN REINFORCEMENT LEARNING FROM SUBOPTIMAL DEMONSTRATIONS

This chapter is based on the following work:

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. *Self Adaptive Imitation Learning: Learning Delayed Rewarded Tasks from Suboptimal Demonstrations.* Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence, 2022.

3.1 Introduction

Reinforcement Learning (RL) is notably advantageous in learning sequential decision-making problems in simulated environments, such as game-playing [119, 156], where massive samples with dense rewards can be accessed at a negligible cost. However, it is challenging to upscale RL to real-world scenarios due to its dependence on immediate reward feedback. For practical applications where rewards are usually delayed in time and sparse in value, RL agents may struggle with high sample complexity, facing difficulties of connecting a long sequence of actions to the feedback received in the far future.

In fact, the ability to learn from delayed feedback is crucial for realizing advanced artificial intelligence [34, 142]. On the one hand, reducing the frequency of reward sampling contributes to a lower interaction complexity for practical applications, such as autonomous-driving [137] and UAV navigation [84]. On the other hand, learning from coarse-grained supervision, such as human preference [90], is rather useful when it is easy to recognize the desired behavior but difficult to explain its rationale by designing delicate reward functions [131].

Recent advances of Imitation Learning (IL) can effectively provide remedies when the environment feedbacks are delayed or even unavailable, by referencing expert demonstrations [68, 87, 88] or policies [145, 162]. In spite of their success, a major limitation of such IL

approaches is that the learned performance is bounded by the given expert. Consequently, when the provided demonstrations are sub-optimal, which is a practical yet more challenging scenario, the IL approaches will induce a sub-optimal policy. In the meantime, some work has been proposed for learning from sub-optimal guidance in delayed rewarded tasks [79, 160, 185, 202, 54]. A shared rationale among them is to augment the environment rewards with synthetic rewards derived from the demonstrations, after which an actor-critic algorithm can take over the policy learning. Although technically effective, these approaches are inherent with twofold limitations. First, *the sub-optimality of teacher demonstrations has not been fully resolved*. Once the learning agent reaches a reasonable performance, the demonstrations will become a bottleneck, leading to negative guidance that contradicts environment feedbacks [112]. Second, *the environment feedback is not well leveraged*. Learning a *critic* function relying on delayed environment rewards can be sample costly, which may provide weak signals to compensate for the sub-optimal demonstrations.

In this paper, we formally consider a problem setting, where an RL agent only has access to a limited number of *sub-optimal* demonstrations in a task with highly *delayed* rewards. Our goal hence is to combine the merits of RL and IL, by exploring the sub-optimal demonstrations that are easier to access in practice, while preserving the chance to explore for better policies guided by the coarse-grained environment feedbacks.

Noticing the challenges of the proposed problem and limitations in prior arts, we propose *Self-Adaptive Imitation Learning (SAIL)*, an off-policy imitation learning approach that strikes a balance between exploitation and exploration to reach high performance. More concretely, we formulate our objective as exploration-driven IL. On the one hand, our approach minimizes the discrepancy between the teacher and the learning policy; on the other hand, it encourages the learning policy to deviate from its previously learned predecessors for better exploration. Specifically, we leverage the delayed feedback from the environment to explore superior self-generated trajectories that surpass the teacher's performance. Those self-generated trajectories are used to replace the suboptimal teachers to constructs a dynamic

target distribution that gradually converges to optimality. An overview of our proposed approach is provided in Figure 3.1. Extensive empirical studies have shown that SAIL achieves significant improvement regarding both sample efficiency and asymptotic performance on various popular benchmarks.

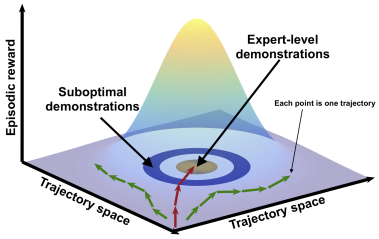


Figure 3.1: Illustraion of *SAIL*: Navigations in red arrows follow the exploration driven IL objective, which approaches to teacher's density distribution while deviating from previous learned ones. It explores more efficiently to reach expertise, compared with random explorations (green arrows).

3.2 Background

Markov Decision Process (MDP) is an ideal environment to formulate RL, which can be defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \mu_0, \mathcal{S}_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action space, $\mathcal{T}(s'|s,a)$ denotes the probability of the environment transitioning from state s to s' upon action a is taken, $r(s,a)$ is the environment reward received by taking action a on state s, $\gamma \in (0,1])$ is a discounted factor, μ_0 is the initial state distribution, and $\mathcal{S}_0$ is the set of terminal states or *absorbing states*. Any absorbing state always transits to itself and yields a reward of zero [164]. Given a trajectory $\tau = \{(s_t, a_t)\}_{t=0}^{\infty}$, we define its return as $R(\tau) = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)$. For an episodic task with a *finite* horizon, its return can be written as $R(\tau) = \sum_{k=0}^{T} \gamma^k r(s_k, a_k)$, where T is the number of steps to reach an absorbing state.

The objective of RL is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected return of its trajectories. Equivalently, this objective can be rephrased as finding a distribution $d_\pi(s,a)$:

$$\max_\pi \ \eta(\pi) := \mathbb{E}_{(s,a)\sim d_\pi(s,a)}\big[r(s,a)\big], \tag{3.1}$$

in which $d_\pi(s,a)$ is the *normalized stationary state-action distribution* of π: $d_\pi(s,a) = (1-\gamma)\mu^\pi(s,a)$, and $\mu^\pi(s,a)$ is the *occupancy measure* of a policy π, defined as:

$$\mu^\pi(s,a) = \sum_{t=0}^{\infty} \gamma^t Pr\big(s_t = s, a_t = a | s_0 \sim \mu_0, a_t \sim \pi(s_t), s_{t+1} \sim \mathcal{T}(s_t, a_t)\big) \text{ [68].}$$

Without ambiguity, we use *density* and *normalized stationary state-action distribution* interchangeably to refer $d_\pi(s,a)$ in this chapter.

Adversarial Imitation Learning addresses IL from the perspective of distribution matching. A representative work along this line is *Generative Adversarial Imitation Learning (GAIL)* [68]. Given a set of demonstrations from an unknown expert policy π_E, GAIL aims to learn a policy π that minimizes the Jensen-Shannon divergence between d_π and d_E:

$$\min_\pi \mathcal{D}_{\text{JS}}[d_\pi(s,a) || d_E(s,a)] - \lambda H(\pi),$$

where d_π and d_E are the densities derived from the learning policy π and the expert policy π_E, and $H(\pi)$ is an entropy regularization term [209, 208].

GAIL applies a saddle-point optimization strategy: it jointly trains a discriminator D and a policy π to optimize the following minimax objective:

$$\min_\pi \max_D \mathbb{E}_{d_\pi(s,a)}[\log(1 - D(s,a))] + \mathbb{E}_{d_E(s,a)}[\log(D(s,a))].$$

In practice, a fixed set of demonstrations from expert densities d_E are given, while samples from d_π are obtained by *on-policy* interactions with the environment.

3.3 Problem Setting

In this chapter, we address the problem of learning in an MDP with *highly delayed* feedbacks. More concretely, in this MDP, an agent learns from the ***trajectory-wise*** reward r_e, which is only non-zero upon reaching an absorbing (terminal) state:

$$r_e(s_t, a_t, s_{t+1}) \neq 0 \Leftrightarrow s_t \notin \mathcal{S}_0, s_{t+1} \in \mathcal{S}_0.$$

Without losing clarity, we use $r_e(\tau)$ to denote the trajectory-wise reward obtained by a trajectory τ. For arbitrary two trajectories τ_i, τ_j, their relative ranking of r_e should align with the task objective:

Assumption 1 (Legitimacy of the Trajectory Rewards)**.** $\forall\ \tau_i, \tau_j, r_e(\tau_i) \geq r_e(\tau_j) \Longrightarrow P_{\pi^*}(\tau_i) \geq P_{\pi^*}(\tau_j)$*, where*

$$P_{\pi^*}(\tau) = \sum_{i=0}^{T} \left(\gamma^i \log \pi^*(a_i|s_i)|\tau := \{(s_0, a_0), (s_1, a_2), \cdots, (s_T, a_T)\}\right)$$

is the extent to which an oracle policy π^* *agrees with a trajectory* τ*.*

Our problem setting provides a generalized framework for a variety of prior arts, including preference-based RL [51, 183] and learning from human feedbacks [119]. Prior work of learning sparse-rewarded tasks with K-step feedbacks [160, 79, 175, 112] can also be reduced to our problem setting, with the advantage that their reward signals are finer-grained and more frequently provided. Compared with an elaborate reward function, trajectory-wise rewards are easier to access and more intuitive to human perception [34, 51], which, however, makes regular RL more challenging.

To alleviate the learning difficulty, we assume that an agent learning a policy π is allowed to leverage external demonstrations $\mathcal{R}_T$ from an unknown teacher policy π_T, which are *sub-optimal* but more accessible than expert demonstrations. For the following of this chapter, we use $d_\pi(s, a)$ and $d_T(s, a)$ to denote the density distribution derived from policy π and π_T, respectively. In practice, d_T is usually approximated from the demonstration data $\mathcal{R}_T$ [209, 49, 68]. Moreover, the learning agent can also access its self-generated transitions cached in a replay buffer $\mathcal{R}_\mathrm{B}$, whose density distribution is $d_B(s, a)$.

3.4 Methodology

3.4.1 Exploration-Driven Objective

We propose an *exploration-driven* IL objective to learn from sub-optimal demonstrations, which is formulated as below:

Objective 1 (Exploration-Driven Imitation Learning)**.**

$$\max_\pi J(\pi) := \underbrace{-\mathbb{D}_{\mathrm{KL}}[d_\pi(s, a)||d_T(s, a)]}_{\textit{Imitation}} + \underbrace{\mathbb{D}_{\mathrm{KL}}[d_\pi(s, a)||d_B(s, a)]}_{\textit{Exploration}},$$

in which $\mathbb{D}_{\mathrm{KL}}$ denotes the KL-divergence between two distributions:

$$\mathbb{D}_{\mathrm{KL}}[p||q] = \mathbb{E}_{p(x)} \log \frac{p(x)}{q(x)}.$$

Objective (1) can be interpreted as joint motivations for imitation and exploration. The first term $-\mathbb{D}_{\mathrm{KL}}[d_\pi(s,a)||d_T(s,a)]$ encourages distribution matching between $d_\pi(s,a)$ and $d_T(s,a)$. The second term $\mathbb{D}_{\mathrm{KL}}[d_\pi(s,a)||d_B(s,a)]$, though counter-intuitive at first sight, serves as an objective for self-exploration. Since $d_B(s,a)$ is the density derived from previously-learned policies, maximizing $\mathbb{D}_{\mathrm{KL}}[d_\pi(s,a)||d_B(s,a)]$ is in favor of visiting state-actions that are rarely seen by previously learned policies, which acts as a repulsive force from $d_B(s,a)$.

Specifically, the proposed objective encourages exploration, which is opposed to a conventional IL objective that solely pursues distribution matching between d_π and d_T:

$$\max_\pi J_{\mathrm{IL}}(\pi) := -\mathbb{D}_{\mathrm{KL}}[d_\pi(s,a)||d_T(s,a)]. \tag{3.2}$$

An optimal solution to Eq (3.2) is a policy that exactly recovers the teacher's density distribution, with $d_\pi(s,a) = d_T(s,a)$ [209]. Given this objective, π is restricted from further exploring density distributions that deviate from d_T, which impedes its potential of generating more superior trajectories. We will verify by empirical studies that optimizing Objective (1) achieves more efficient exploration compared with a pure imitation-driven objective.

3.4.2 Adaptive Learning Target

Following Objective (1), the learning policy has obtained the potential to yield trajectories with performance surpassing the teacher. To fully utilize this self-generated resource, *SAIL* adaptively adjusts the teacher's buffer to replace teacher demonstrations with more superior trajectories sampled from the learning agent, by leveraging the trajectory-wise feedback from the environment. This strategy dynamically improves the lower bound of the teacher's performance. As a result, the density $d_T(s,a)$ of the teacher buffer is approaching an oracle distribution:

Theorem 1. *For a deterministic policy, rewards of its generated trajectories indicate the policy' agreement with an oracle:*

$$\forall \pi_i, \pi_j,\ \mathbb{E}_{\tau\sim\pi_i}\left[r_e(\tau)\right] > \mathbb{E}_{\tau\sim\pi_j}\left[r_e(\tau)\right] \Longrightarrow$$
$$\mathbb{D}_{\mathrm{KL}}\left[\pi_i(a|s)||\pi^*(a|s)\right] < \mathbb{D}_{\mathrm{KL}}\left[\pi_j(a|s)||\pi^*(a|s)\right].$$

Proof. Given arbitrary *deterministic* policies π_i and π_j, *s.t.* $\mathbb{E}_{\tau\sim\pi_i}[r_e(\tau)] > \mathbb{E}_{\tau\sim\pi_j}[r_e(\tau)]$. Based on Assumption 1, one can derive that:

$$\mathbb{E}_{\tau\sim\pi_i}[P_{\pi^*}(\tau)] - \mathbb{E}_{\tau\sim\pi_j}[P_{\pi^*}(\tau)] > 0. \tag{3.3}$$

Next, one can derive that:

$$\begin{aligned}
&\mathbb{D}_{\mathrm{KL}}[\pi_i(a|s)||\pi^*(a|s)] - \mathbb{D}_{\mathrm{KL}}[\pi_j(a|s)||\pi^*(a|s)] \\
=&\mathbb{E}_{d_{\pi_i}(s)\pi_i(a|s)}[\log\frac{\pi_i(a|s)}{\pi^*(a|s)}] - \mathbb{E}_{d_{\pi_j}(s)\pi_j(a|s)}[\log\frac{\pi_j(a|s)}{\pi^*(a|s)}] \\
=&\underbrace{-H[\pi_i(a|s)]}_{0 \text{ for deterministic } \pi_i} - \mathbb{E}_{d_{\pi_i}}[\log\pi^*(a|s)] + \underbrace{H[\pi_j(a|s)]}_{0} + \mathbb{E}_{d_{\pi_j}}[\log\pi^*(a|s)] \\
=&-(1-\gamma)\sum_t \mathbb{E}_{s_t\sim\mu_t^{\pi_i}, a_t\sim\pi_i(s_t)}[\gamma^t\log\pi^*(a_t|s_t)] \\
&+(1-\gamma)\sum_t \mathbb{E}_{s_t\sim\mu_t^{\pi_j}, a_t\sim\pi_j(s_t)}[\gamma^t\log\pi^*(a_t|s_t)] \\
=&-(1-\gamma)\Big(\underbrace{\mathbb{E}_{\tau\sim\pi_i}[P_{\pi_i}(\tau)] - \mathbb{E}_{\tau\sim\pi_j}[P_{\pi_j}(\tau)]\Big)}_{\text{based on Eq equation 3.3}} < 0.
\end{aligned}$$

□

Therefore, when the teacher buffer is updated with more superior trajectories generated by a deterministic policy over time, as in our case, the distribution derived by the teacher buffer is approaching to optimality. Unlike prior art that bundles their critic learning process with environment rewards, we leverage this delayed and coarse-grained feedback to construct a dynamic learning target with increasing superiority, which relieves the bottleneck brought by sub-optimal demonstrations.

3.4.3 Off-Policy Adversarial TD Learning

While our proposed approach is appealing in combining the merits of exploitation and exploration, it is challenging to directly optimize Objective (1). To make it more approachable, we draw a connection from Objective (1) to a conventional RL problem:

Remark 2. *Objective (1) can be rephrased as the following, which is equivalent to a max-return RL objective with* $\log \frac{d_T(s,a)}{d_B(s,a)}$ *in place of the environment rewards:*

$$\max_{\pi} \; J(\pi) := \mathbb{E}_{d_\pi(s,a)} \left[\log \frac{d_T(s,a)}{d_B(s,a)} \right], \tag{3.4}$$

One can consider the optimization of Equation equation 3.4 as a process of policy selection: for the *support* of (s, a) where the teacher has visited more frequently than the previously-learned policies, π is encouraged to build positive densities on those state-actions, leading to $d_\pi(s, a) > 0$ wherever $d_T(s, a) > d_B(s, a)$. Intuitively, this process implies that the agent trusts the teacher more than the previously learned policies.

Based on this insight, we can relate Objective (1) to Temporal-Difference (TD) learning, and solve it under an *actor-critic* framework. To obtain the reward function $\log \frac{d_T(s,a)}{d_B(s,a)}$, we build upon prior arts [68] to learn a discriminator D that optimizes the following:

$$\max_{D:\mathcal{S}\times\mathcal{A}\to(0,1)} \mathbb{E}_{d_B(s,a)}[\log(1 - D(s, a))] + \mathbb{E}_{d_T(s,a)}[\log(D(s, a))]. \tag{3.5}$$

D aims to distinguish between the self-generated data from d_B and the teacher demonstrations from d_T. A well-learned discriminator shall satisfy the following [56]:

$$D^*(s, a) = \frac{d_T(s, a)}{d_T(s, a) + d_B(s, a)}.$$

The output of D with a constant shift, which we found to be more empirically effective, is used to render synthetic rewards to the agent:

$$r'(s, a) = -\log(1 - D(s, a)) \approx \log(\tfrac{d_T(s,a)}{d_B(s,a)} + 1).$$

In the initial training stage, a well-trained discriminator renders higher rewards to teacher demonstrations with $D(s,a) \to 1$, and lower rewards for self-generated samples with $D(s,a) \to 0$. The learning policy is therefore designed to confuse the discriminator by maximizing the shaped accumulated rewards.

To improve sample efficiency, we adopt an *off-policy* learning framework. Our objective is accordingly rephrased to maximize the expectation of Q-values over distributions of a behavior policy β [155, 102]:

$$\max_\theta J_\beta(\pi_\theta) := \int_s d_\beta(s) Q(s, \pi_\theta(s)) d_s = \mathbb{E}_{d_\beta(s)}[Q(s, \pi_\theta(s))], \tag{3.6}$$

where $d_\beta(s)$ is the normalized stationary *state* distribution of β, analogous to the *state-action* distribution. The Q-function is a fixed point solution to the Bellman operation based on the shaped rewards:

$$Q(s,a) = r'(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s'|s,a), a' \sim \pi(s')}[Q(s', a')]. \tag{3.7}$$

Accordingly, the policy-gradient for the actor can be derived as [102]:

$$\nabla_\theta J_\beta(\pi_\theta) \approx \mathbb{E}_{s \sim d_\beta}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a)|_{a=\pi_\theta(s)}]. \tag{3.8}$$

In the next section, we introduce an algorithm that realizes our objective via the above-mentioned off-policy TD learning. It adopts an even more effective sampling approach that further accelerates the learning procedure.

3.4.4 Self-Adaptive Imitation Learning

Combing all the building blocks, we now introduce our approach, dubbed as *Self-Adaptive Imitation Learning (SAIL)*, as described in Algorithm 3.1. *SAIL* maintains two replay-buffers $\mathcal{R}_T$ and $\mathcal{R}_\mathrm{B}$, for caching teacher demonstrations and self-generated transitions, respectively. It jointly learns three components: a discriminator D that serves as a reward provider, a critic Q that minimizes the Bellman error based on the shaped rewards, and an actor π that maximizes the shaped returns. During iterative training, high-quality trajectories

Algorithm 3.1: Self-Adaptive Imitation Learning

Input: teacher replay buffer $\mathcal{R}_T$ with demonstrations,
self-replay-buffer $\mathcal{R}_\mathrm{B}$ with random transitions.
policy π_θ, discriminator D_w, critic Q_ϕ, batch size $N > 0$, coefficient $\alpha > 0$
for $n = 1, \ldots$ **do**
  sample trajectory $\tau \sim \pi_\theta$
  if $r_e(\tau) > \min_{\tau'}\{(r_e(\tau') \mid \tau' \in \mathcal{R}_T\}$) **then**
    $\mathcal{R}_T \leftarrow \mathcal{R}_T \cup \tau$; $\alpha \leftarrow 0$
  else
    $\mathcal{R}_\mathrm{B} \leftarrow \mathcal{R}_\mathrm{B} \cup \tau$
  if $n \mod$ `discriminator-update` $= 0$ **then**
    $\big\{(s_i, a_i, \cdots)\big\}_{i=1}^{N} \sim \mathcal{R}_\mathrm{B}, \big\{(s_i^T, a_i^T, \cdots)\big\}_{i=1}^{N} \sim \mathcal{R}_T$
    update D_w by ascending gradient :
    $\nabla_w \frac{1}{N} \sum_{i=1}^{N} \big[\log D\Big(s_i^T, a_i^T\Big) + \log\Big(1 - D\big(s_i, a_i\big)\Big)\big]$
  if $n \mod$ `Q-update` $= 0$ **then**
    $\{s_i, a_i, s_i'\}_{i=1}^{N} \sim \mathcal{R}_\mathrm{B}, \{s_i^T, a_i^T, s_i^{'T}\}_{i=1}^{N} \sim \mathcal{R}_T$
    $y_i \leftarrow -\log(1 - D(s_i, a_i)) + \gamma \bar{Q}(s_i', \pi(s_i'))$
    $y_i^{'T} \leftarrow -\log(1 - D(s_i^T, a_i^T)) + \gamma \bar{Q}(s_i^{'T}, \pi(s_i^{'T}))$
    update Q_ϕ by minimizing critic loss:
    $J(Q_\phi) = \frac{1-\alpha}{N} \sum_i [(Q_\phi(s_i, a_i) - y_i)^2] + \frac{\alpha}{N} \sum_i [(Q_\phi(s_i^T, a_i^T) - y_i^{'T})^2]$
  if $n \mod$ `policy-update` $= 0$ **then**
    $\big\{(s_i, ., ., ., .)\big\}_{i=1}^{N} \sim \mathcal{R}_\mathrm{B}, \big\{(s_i^T, ., ., ., .)\big\}_{i=1}^{N} \sim \mathcal{R}_T$
    update π by sampled policy gradient:
    $\nabla_\theta J(\pi_\theta) \approx \frac{1-\alpha}{N} \sum_i [\nabla_\theta \pi_\theta(s_i) \nabla_a Q(s_i, a_i)|_{a_i = \pi_\theta(s_i)}] + \frac{\alpha}{N} \sum_i [\nabla_\theta \pi_\theta(s_i^T) \nabla_a Q(s_i^T, a_i^T)|_{a_i^T = \pi_\theta(s_i^T)}]$

generated by the actor are selected to refill the teacher demonstration buffer $\mathcal{R}_T$, while other trajectories are cached in the self-replay buffer $\mathcal{R}_\mathrm{B}$. We highlight three key aspects of *SAIL*:

(1) *Leveraging delayed environment feedback to update teacher buffer* $\mathcal{R}_T$: High-quality trajectories with reward r_e above a threshold C_{d_T} are selected to update the teacher buffer $\mathcal{R}_T$. In practice, C_{d_T} is simultaneously updated as the lowest reward in the teacher buffer $\mathcal{R}_T$: $C_{d_T} = \min_{\tau'}\{r_e(\tau') | \tau' \in \mathcal{R}_T\}$, which guarantees the increasing quality of trajectories in the teacher's buffer.

(2) *Realizing exploration-driven IL with an off-policy discriminator*: Prior art such as *GAIL* relies on *on-policy* training of a discriminator to estimate the ratio of $\frac{d_T(s,a)}{d_\pi(s,a)}$. On the contrary, we learn an *off-policy* discriminator D that aligns with our proposed objective and

encourages efficient exploration, whose effectiveness will be elaborated in the Experiment section.

(3) *Sampling from teacher demonstrations for boosted learning efficiency*: In the initial learning step, we sample from both the teacher dataset $\mathcal{R}_T$ and the self-generated dataset $\mathcal{R}_\mathrm{B}$ to construct a mixed density distribution, which plays the role of d_β in Eq (3.8). More concretely, we derive a mixture distribution: $d_\mathrm{mix} = \alpha d_T + (1-\alpha) d_B$, where α is the ratio of samples from teacher demonstrations. In practice, we initialize $\alpha = 0.5$. Once the learning policy generates trajectories with performance comparable to the teacher, we anneal the value of α to zero.

3.4.4.1 Reasoning of sampling from a mixture of distributions:

Sampling from teacher demonstrations has been studied by other prior arts [175, 142]. Along with the same spirit, the mixture sampling distribution in our case accelerates the IL process by a *behavior-cloning* strategy. To see the rationale, one can rephrase the objective in Equation 3.6 as the following:

$$\max_\theta J_\beta(\pi_\theta) := \alpha \underbrace{\mathbb{E}_{d_T(s)}[Q(s, \pi_\theta(s))]}_{\text{Behavior Cloning}} + (1-\alpha)\mathbb{E}_{d_B(s)}\left[Q(s, \pi_\theta(s))\right].$$

In the early training stage, the discriminator will favor teacher trajectories by assigning them with highest rewards:

$$\mathbb{E}_{d_T(s)}[\max_a Q(s,a)] = \mathbb{E}_{d_T(s,a)}\left[Q(s,a)\right] \equiv \mathbb{E}_{d_T(s)}\left[Q(s, \pi_T(s))\right],$$

which encourages the learning policy to imitate π_T on teacher-visted states $d_T(s)$. We will verify by ablation study that sampling from teacher demonstrations accelerates the process of IL. Given a problem-setting with sub-optimal demonstrations, once the learning agent reaches the teacher-level performance, we relieve this behavior-cloning regularization by annealing α to zero, in order to reinforce the effects of exploration as proposed in our objective.

3.5 Related Work

Our work shares close connections with the following topics:

Imitation Learning (IL) aims to learn from expert demonstrations without accessing environment feedbacks, among which representative examples include ***GAIL*** [68] and its on-policy extensions [79, 185, 49]. Later IL favors off-policy RL frameworks [148, 88]. Especially, ***DAC*** learns a discriminator by off-policy learning and corrects the distribution shifts by importance sampling [87]. In contrast to our approach, the above prior arts are motivated to exactly recover the teacher policy. Our work also draws a subtle connection to *Self-Imitation Learning (SIL)* [130, 59], in that they both utilize self-generated trajectories to build a learning target. However, SIL requires timely feedbacks from the environment to learn a delicate critic, which is in essence on-policy RL, while *SAIL* addresses a different setting by performing exploration-driven IL in an off-policy manner.

Learning from Demonstrations (LfD) facilitates RL by augmenting environment feedbacks with external demonstrations. Prior work relies on demonstrations that are sufficient and optimal [66, 175]. Especially, ***DDPGfD*** leverages a DDPG framework [102] to enable off-policy LfD in continuous spaces [175]. Later approaches, such as ***POfD*** [79], learn from *sub-optimal* demonstrations and trust the environment rewards to learn a critic, whereas demonstrations are only used as auxiliary guidance [160, 202, 54]. In contrast, our approach learns a critic without using environmental rewards, which is more robust especially when environment feedbacks are highly delayed. Some leverage the suboptimal guidance to enforce a policy regularization term, whose effects are gradually decayed to tackle the imperfect guidance [112]. The above problem settings can be considered as relaxed versions of ours with finer-grained feedbacks.

Preference-based RL is a problem setting where the agent learns from the preference of an expert, which saves the necessity of designing elaborated numeric rewards [181, 183]. The preference relations can be over state-actions pairs [51] or over a pair of trajectories $\tau_i \succ \tau_j$ [23], while the former provides more supervision information than the later. Few

prior arts address IL in preference-based RL, except for [23, 21], which tackle IL in an MDP provided with only trajectory-ranked demonstrations but no environment feedbacks. Focusing on a different problem setting, *SAIL* utilizes the self-generated trajectories to build an increasing teacher distribution, which therefore requires fewer teacher demonstrations.

Exploration itself is an independent topic in RL. Classical exploration approaches work by involving randomness into its learning loop [47, 163, 61]. More recent approaches propose to use *intrinsic* rewards for exploration [13, 133]. Especially, [133] proposed ***curiosity-driven*** exploration, a model-based approach which leverages the prediction loss of a transition model as a reward bonus to encourage surprising behavior. Another exploration approach pursues a maximized *information gain* about the agent's belief of the environment [70]. Readers are referred to [115] for a comprehensive discussion on the exploration techniques in RL.

3.6 Experiments

In this section, we study how *SAIL* achieves the objective of imitation learning and exploration in an environment with delayed rewards. Extensive experiments have been conducted to answer the following key questions:

Q1. Is *SAIL* sample-efficient?

Q2. Can *SAIL* surpass the demonstration performance via off-policy exploration?

Q3. Which components in *SAIL* contribute to the exploration or sample efficiency?

Q4: Is *SAIL* robust against different sub-optimal teachers?

3.6.1 Setup

We built *SAIL* on a TD3 framework [50] based on *stable-baselines*[1] implementations. It is tested on 4 popular MuJoCo[2] tasks: *Walker2d-v2, Hopper-v2, HalfCheetah-v2*, and *Swimmer-v2*. For each task, we generate teacher demonstrations from a deterministic policy that was pre-trained to be sub-optimal. All experiments are conducted using one imperfect demon-

[1]https://stable-baselines.readthedocs.io/en/master/

[2]https://github.com/openai/mujoco-py

stration trajectory on 5 random seeds, with each trajectory containing no more than 1000 transitions. Models are evaluated after training using 10^6 interaction samples. We defer more details and additional experimental results to the Supplementary.

Note that the original benchmarks are all in dense-reward settings. To construct the delayed rewarded environment as proposed in our chapter, we omit the original rewards such that only episodic feedback is provided upon the completion of a trajectory. To align with Assumption 1, we cache the original return of each trajectory $R(\tau) = \sum_i r(s_i, a_i)$, and downscale it to get a coarser grained supervision, with $r_e(\tau) = \lfloor 0.1 * R(\tau) \rfloor$.

We compare *SAIL* with 5 popular baselines that are mostly applicable to our problem setting: DAC, GAIL, POfD, DDPGfD, and BC, as discussed in the Related Work. For baselines that utilize environment rewards, such as POfD and DDPGfD, we provide them with modified rewards $r_e(s, a)$ upon the completion of each trajectory, instead of the original dense reward $r(s, a)$.

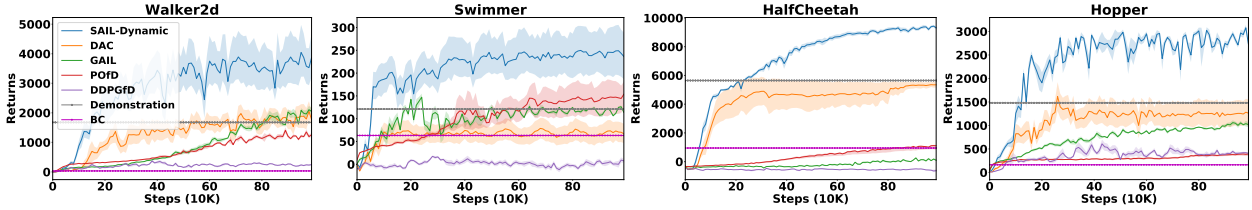


Figure 3.2: Learning curves of *SAIL* and other previous work using one suboptimal demonstration trajectory.

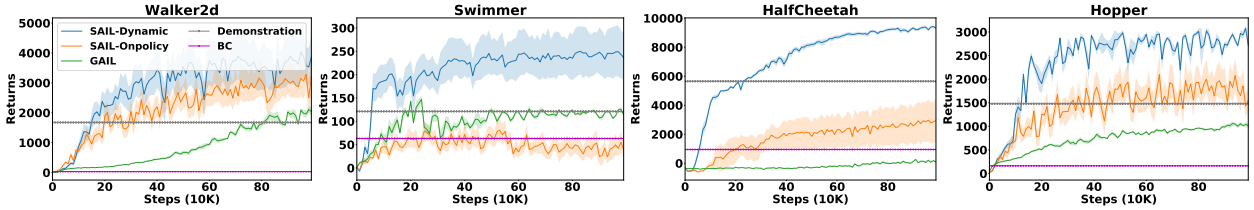


Figure 3.3: Comparing *SAIL* with other on-policy baselines using one suboptimal demonstration trajectory.

3.6.2 Performance on Continuous Action-Space Tasks

Sample efficiency: As the results shown in Figure 3.2, *SAIL* is the only method that performs consistently better in all tasks in terms of both sample efficiency and asymptotic

Benchmark	HalfCheetah	Swimmer	Hopper	Walker
SAIL	**10660.59±105.53**	**309.47±3.0**	**3302.06±14.22**	**5868.53±108.82**
Curiosity Explore	9043.07 ± 165.97	30.87 ± 6.52	3075.46 ± 15.61	5361.78 ± 58.24
Entropy Explore	$8839.32 + 280.47$	65.04 ± 7.66	3079.29 ± 53.25	2792.76 ± 830.20
Teacher Demonstration	5646.71	121.16	1480.69	1675.01

Table 3.1: Off-policy exploration (*SAIL*) achieves higher performance than other exploration approaches.

performance. At the initial stage of the learning, *SAIL* can quickly exploit the suboptimal demonstrations and approach to the demonstration's performance with significantly fewer samples.

Exploration ability: Besides sample efficiency, another advantage of *SAIL* is that it can effectively explore the environment to achieves expert-level performance, even with highly sparse rewards. We observe that prior solutions of learning from environment rewards for exploration, such as POfD and DDPGfD, cannot effectively address our proposed problem setting, as it is sample-costly to learn a meaningful critic from the delayed feedback. Unlike other imitation learning baselines whose performance is limited by the demonstrations, *SAIL* can rapidly surpass the imperfect teacher via constructing a better demonstration buffer.

3.6.3 Effects of Off-Policy Exploration in *SAIL*

Comparison with IL without Exploration: In order to illustrate the benefits of maximizing Objective (1) over a conventional *IL* objective, such as $\mathbb{D}_{\mathrm{KL}}[d_\pi(s,a)||d_T(s,a)]$, we conducted a comparison study where we trained the discriminator using teacher demonstrations τ_T and on-policy self-generated samples τ_π, instead of off-policy samples. This on-policy training scheme is the same as proposed in *GAIL* [68]. In this way, the discriminator can get approximations of $\log(\frac{d_T}{d_\pi})$ instead of $\log(\frac{d_T}{d_B})$. We use the output of this on-policy discriminator to shape rewards, whereas Q and π are still updated in the same off-policy fashion as our proposed approach.

As illustrated in Figure 3.3, compared to *GAIL* (green) which is an on-policy baseline,

SAIL-OnPolicy (orange) still enjoys the benefits of an off-policy learning scheme in general. However, it is less effective compared with our proposed approach. Even when π and Q are learned off-policy, *SAIL-OnPolicy* is slower to surpass the teacher demonstration (dashed gray line), due to its pure imitation-driven objective. *SAIL* enjoys fast improvement in performance not only because of an adaptive teacher demonstration buffer but also because it realizes the exploration-driven optimization.

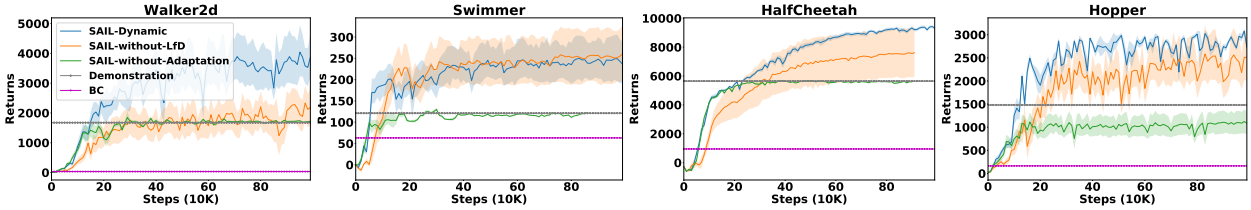


Figure 3.4: Ablation study by removing different algorithmic components from *SAIL*. Only one teacher trajectory is used as demonstration.

Benchmark	**Evaluated Performance / Demonstration Performance**		
HalfCheetah	10660.59 ± 105.53 / 5646.71	10217.00 ± 104.08 / 3598.90	9264.1 ± 163.45 / 875.39
Swimmer	309.47 ± 3.0 / 121.16	367.02 ± 1.11 / 46.82	361.17 ± 1.37 / 33.61
Hopper	3302.06 ± 14.22 / 1480.69	3814.07 ± 10.32 / 665.16	3589.92 ± 12.24 / 282.91
Walker	5868.53 ± 108.82 / 1675.01	4819.20 ± 1240.84 / 484.96	4574.61 ± 71.22 / 255.73

Table 3.2: Using 10^6 interaction samples, performance of *SAIL* is robust regardless of the quality of sub-optimal teacher demonstrations.

Comparison with Other Exploration Approaches: We also compared *SAIL* with its two variants to evaluate the effects of different exploration approaches. In particular, we integrated *SAIL* with a soft-actor-critic [61] RL framework to enable an *entropy-based* exploration. For the other variant, we adopted the idea of random-distillation [27] to create a *curiosity* reward in addition to the reward provided by the discriminator. For both variant versions, we train the discriminator with *on-policy* samples in order to remove the effects of off-policy exploration. Comparison results in Table 3.1 indicate that, adopting an off-policy exploration approach (*SAIL*) is more effective given a fixed number of environment interactions. Entropy-based exploration is prone to high variance, while curiosity-based

exploration, on the other hand, requires learning a forward transition model, and achieves lower final performance.

3.6.4 Ablation Study

We further evaluate *SAIL* by ablation and sensitivity studies to analyze the following aspects:

Effects of learning from expert demonstrations: As shown in Figure 3.4, we observed that sampling from a mixture of teacher data and self-generated data accelerates the learning performance in early training stages. Specifically, the *SAIL-Dynamic* (blue) refers our proposed approach, and *SAIL-without-LfD* (orange) only uses self-generated data to learn policy by setting $\alpha = 0$ constantly. We see that the *SAIL* is superior to *SAIL-without-LfD* in terms of initial performance, which is ascribed to a learning strategy resemblant to *behavior-cloning* when sampling from teacher demonstrations.

Effects of updating teacher demonstration buffers: As shown in Figure 3.4, *SAIL-without-Expert-Adaptation* (green) refers to a variant of *SAIL* which never update the teacher's replay buffer, even when a better trajectory is collected . We can observe that its asymptotic performance is bounded by the teacher's demonstration, which reveals the limitation of most existing IL approaches. One key insight from these results is that, instead of learning critics based on sparse rewards, leveraging the sparse guidance to improve the quality of the teacher can be much more effective in improving the ultimate performance.

Robustness of SAIL on different teacher qualities: To evaluate the robustness of *SAIL* against different teacher performance, we pre-trained a group of teacher policies with varying qualities, ranging from near-randomness to sub-optimality, then used their generated trajectories as demonstrations. For each experiment, we only used one teacher trajectory as demonstrations. Results in Table 3.2 show that *SAIL* can achieve robust performance no matter how sub-optimal the teacher behaves. Powered by both an exploration-driven objective and a self-adaptive learning strategy, *SAIL* can constantly explore with more superior

trajectories to improve its learning target, which results in improving learning performance.

3.7 Summary

In this chapter, we address the problem of reinforcement learning in environments with highly *delayed* rewards given *sub-optimal* demonstrations. To address this challenging problem, we propose a novel objective that encourages exploration-based imitation learning. Towards this objective, we design an effective algorithm called *Self Adaptive Imitation Learning (SAIL)*. The proposed approach is validated to (1) accelerate the agent learning process by fully utilizing teacher demonstration for knowledge transfer, (2) address sample efficiency by off-policy imitation learning, and (3) surpass the imperfect teacher with a large margin by iteratively performing imitation and exploration. Experimental results on challenging locomotion tasks indicate that *SAIL* significantly surpasses state-of-the-arts in terms of both sample efficiency and asymptotic performance.

CHAPTER 4

OFF-POLICY LEARNING FROM OBSERVATIONS: KNOWLEDGE TRANSFER IN REINFORCEMENT LEARNING FROM INCOMPLETE SUPERVISION

Proposed approach in this chapter is based on the following paper:

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. *Off-Policy Imitation Learning from Observations.* Advances in Neural Information Processing Systems 33 (2020).

4.1 Introduction

Imitation Learning (IL) has been widely studied in the reinforcement learning (RL) domain to assist in learning complex tasks by leveraging the experience from expertise [145, 68, 88, 87, 136]. Unlike conventional RL that depends on environment reward feedbacks, IL can purely learn from expert guidance and is therefore crucial for realizing robotic intelligence in practical applications, where demonstrations are usually easier to access than a delicate reward function [156, 119].

Classical IL, or more concretely, Learning from Demonstrations (LfD), assumes that both *states* and *actions* are available as expert demonstrations [1, 68, 88]. Although expert actions can benefit IL by providing elaborated guidance, requiring such information for IL may not always accord with the real world. Actually, collecting demonstrated actions can sometimes be costly or impractical, whereas observations without actions are more accessible resources, such as the camera or sensory logs. Consequently, Learning from Observations (LfO) has been proposed to address the scenario without expert actions [171, 189, 172]. On one hand, LfO is more challenging compared with conventional IL, due to missing finer-grained guidance from actions. On the other hand, LfO is a more practical setting for IL, not only because it capitalizes on previously unusable resources, but also because it reveals the potential to realize advanced artificial intelligence. In fact, learning without action guidance is an inherent ability for human beings. For instance, a novice game player can improve his

skill purely by watching video records of an expert, without knowing what actions have been taken [10].

Among popular LfD and LfO approaches, distribution matching has served as a principled solution [68, 88, 171, 189, 49], which works by interactively estimating and minimizing the discrepancy between two stationary distributions: one generated by the expert, and the other generated by the learning agent. To correctly estimate the distribution discrepancy, traditional approaches require *on-policy* interactions with the environment whenever the agent policy gets updated. This inefficient sampling strategy impedes wide applications of IL to scenarios where accessing transitions are expensive [147, 118]. The same challenge is aggravated in LfO, as more explorations by the agent are needed to cope with the lack of action guidance.

Towards sample efficiency, some off-policy IL solutions have been proposed to leverage transitions cached in a replay buffer. Mostly designed for LfD, these methods either lack theoretical guarantee by ignoring a potential distribution drift [87, 149, 169], or hinge on the knowledge of expert actions to enable off-policy distribution matching [88], which makes their approach inapplicable to LfO.

To address the aforementioned limitations, in this work, we propose a LfO approach that improves sample efficiency in a principled manner. Specifically, we derive an upper bound of the LfO objective which dispenses with the need of knowing expert actions and can be fully optimized with *off-policy* learning. To further accelerate the learning procedure, we combine our objective with a regularization term, which is validated to pursue distribution matching between the expert and the agent from a *mode-covering* perspective. Under a mild assumption of a deterministic environment, we show that the regularization can be enforced by learning an inverse action model. We call our approach *OPOLO* (*O*ff *PO*licy *L*earning from *O*bservations). Extensive experiments on popular benchmarks show that *OPOLO* achieves state-of-the-art in terms of both asymptotic performance and sample efficiency.

4.2 Background

We consider learning an agent in an environment of Markov Decision Process (MDP) [164], which can be defined as a tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, p_0)$. Particularly, $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces; P is the state transition probability, with $P(s'|s,a)$ indicating the probability of transitioning from s to s' upon action a; r is the reward function, with $r(s,a)$ the immediate reward for taking action a on state s; Without ambiguity, we consider an MDP with *infinite* horizons, with $0 < \gamma < 1$ as a discounted factor; p_0 is the initial state distribution. An agent follows its policy $\pi : \mathcal{S} \to \mathcal{A}$ to interact with this MDP with an objective of maximizing its expected return:

$$\max J_{\mathrm{RL}}(\pi) := \mathbb{E}_{s_0 \sim p_0, a_i \sim \pi(\cdot|s_i), s_{i+1} \sim P(\cdot|s_i,a_i), \forall 0 \le i \le t} \Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \Big] = \mathbb{E}_{(s,a) \sim \mu^\pi(s,a)} \Big[r(s,a) \Big],$$

in which $\mu^\pi(s,a)$ is the *stationary state-action distribution* induced by π, as defined in Table 4.1.

Learning from demonstrations (LfD) is a problem setting in which an agent is provided with a fixed dataset of expert demonstrations as guidance, without accessing the environment rewards. The demonstrations $\mathcal{R}_E$ contain sequences of both *states* and *actions* generated by an expert policy π_E: $\mathcal{R}_E = \{(s_0, a_0), (s_1, a_1), \cdots | a_i \sim \pi_E(\cdot|s_i), s_{i+1} \sim P(\cdot|s_i, a_i)\}$. Without ambiguity, we assume that the expert and agent are from the same MDP.

Among LfD approaches, distribution matching has been a popular choice, which minimizes the discrepancy between two stationary *state-action* distributions: one is $\mu^E(s,a)$ induced by the expert, and the other is $\mu^\pi(s,a)$ induced by the agent. Without loss of generality, we consider KL-divergence as the discrepancy measure for distribution matching, although any f-divergences can serve as a legitimate choice [68, 186, 129] :

$$\min J_{\mathrm{LfD}}(\pi) := \mathbb{D}_{\mathrm{KL}}[\mu^\pi(s,a) || \mu^E(s,a)]. \tag{4.1}$$

Learning from observations (LfO) is a more challenging scenario where expert guidance $\mathcal{R}_E$ contains only *states*. Accordingly, applying distribution matching to solve LfO

yields a different objective that involves *state-transition* distributions [189, 105, 171]:

$$\min J_{\text{LfO}}(\pi) := \mathbb{D}_{\text{KL}}[\mu^{\pi}(s, s')||\mu^{E}(s, s')]. \tag{4.2}$$

There exists a close connection between LfO and LfD objectives. In particular, the discrepancy between two objectives can be derived precisely as follows:

$$\mathbb{D}_{\text{KL}}[\mu^{\pi}(a|s, s')||\mu^{E}(a|s, s')] = \mathbb{D}_{\text{KL}}[\mu^{\pi}(s, a)||\mu^{E}(s, a)] - \mathbb{D}_{\text{KL}}[\mu^{\pi}(s, s')||\mu^{E}(s, s')]. \tag{4.3}$$

Remark 3. *In a non-injective MDP, the discrepancy of* $\mathbb{D}_{\text{KL}}[\mu^{\pi}(a|s, s')||\mu^{E}(a|s, s')]$ *cannot be optimized without knowing expert actions. In a deterministic and injective MDP, it satisfies that* $\forall\ \pi : \mathcal{S} \to \mathcal{A},\ \mathbb{D}_{\text{KL}}[\mu^{\pi}(a|s, s')||\mu^{E}(a|s, s')] = 0.$

Despite the potential gap between these two objectives, the LfO objective in Eq (4.2) is still intuitive and valid, as it emphasizes on recovering the expert's influence on the environment by encouraging the agent to yield the desired *state-transitions*, regardless of the immediate behavior that leads to those transitions. In this work, we follow this rationale and consider Eq (4.2) as our learning objective, which has also been widely adopted by prior art [171, 170, 161, 22]. We will show later that pursuing this objective is sufficient to recover expertise for various challenging tasks.

A common limitation of existing LfO and LfD approaches relies in their inefficient optimization. Work along this line usually adopts a GAN-style strategy [56] to perform distribution matching. Take the representative work of *GAIL* [68] as an example, in which a discriminator $x : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and a generator $\pi : \mathcal{S} \to \mathcal{A}$ are jointly learned to optimize a dual form of the original LfD objective:

$$\min_{\pi} \max_{x} J_{\text{GAIL}}(\pi, x) := \mathbb{E}_{\mu^{E}(s,a)}[\log(x(s, a))] + \mathbb{E}_{\mu^{\pi}(s,a)}[\log(1 - x(s, a))].$$

During optimization, *on-policy* transitions in the MDP are used to estimate expectations over μ^{π}. It requires new environment interactions whenever π gets updated and is thus sample inefficient. This inconvenience is echoed in the work of LfO, which inherits the same spirit of on-policy learning [189, 171]. In pursuit of sample efficiency, some off-policy solutions

	State Distribution	State-Action Distribution	Joint Distribution	Transition Distribution	Inverse-Action Distribution
Notation	$\mu^\pi(s)$	$\mu^\pi(s,a)$	$\mu^\pi(s,a,s')$	$\mu^\pi(s,s')$	$\mu^\pi(a\|s,s')$
Support	$\mathcal{S}$	$\mathcal{S}\times\mathcal{A}$	$\mathcal{S}\times\mathcal{A}\times\mathcal{S}$	$\mathcal{S}\times\mathcal{S}$	$\mathcal{A}\times\mathcal{S}\times\mathcal{S}$
Definition	$(1-\gamma)\sum_{t=1}^{\infty}\gamma^t\mu_t^\pi(s)$	$\mu^\pi(s)\pi(a\|s)$	$\mu^\pi(s,a)P(s'\|s,a)$	$\int_{\mathcal{A}}\mu^\pi(s,a,s')da$	$\frac{\mu^\pi(s,a)P(s'\|s,a)}{\mu^\pi(s,s')}$

Table 4.1: Summarization on different stationary distributions, with $\mu_t^\pi(s) = p(s_t = s|s_0 \sim p_0(\cdot), a_i \sim \pi(\cdot|s_i), s_{i+1} \sim P(\cdot|s_i, a_i)),\ \forall i < t)$.

have been proposed. These methods, however, either lack theoretical guarantee [169, 87], or rely on the expert actions [87, 88], which makes them inapplicable to LfO.

To improve the sample efficiency of LfO with a principled solution, in the next section we show how we explicitly introduce an off-policy distribution into the LfO objective, from which we derive a feasible upper-bound that enables *off-policy* optimization without the need of accessing expert actions.

4.3 *OPOLO*: Off-Policy Learning from Observations

4.3.1 Surrogate Objective

The idea of re-using cached transitions to improve sample efficiency has been adopted by many RL algorithms [119, 61, 50, 102]. In the same spirit, we start by introducing an *off-policy* distribution $\mu^R(s,a)$, which is induced by a dataset $\mathcal{R}$ of historical transitions. Choosing KL-divergence as a discrepancy measure, we obtain an upper-bound of the LfO objective by involving $\mu^R(s,a)$:

$$\mathbb{D}_{\mathrm{KL}}\left[\mu^\pi(s,s')||\mu^E(s,s')\right] \leq \mathbb{E}_{\mu^\pi(s,s')}\left[\log\frac{\mu^R(s,s')}{\mu^E(s,s')}\right] + \mathbb{D}_{\mathrm{KL}}\left[\mu^\pi(s,a)||\mu^R(s,a)\right]. \tag{4.4}$$

As a result, the LfO objective can be optimized by minimizing the RHS of Eq (4.4). Although widely adopted for its interpretability, KL divergence can be tricky to estimate due to issues of biased gradients [41, 88]. To avoid the potential difficulty in optimization, we further substitute the term $\mathbb{D}_{\mathrm{KL}}[\mu^\pi(s,a)||\mu^R(s,a)]$ in Eq (4.4) by a more aggressive f-

divergence, with $f(x) = \frac{1}{2}x^2$, which serves as an upper-bound of the KL-divergence:

$$\mathbb{D}_{\mathrm{KL}}[P||Q] \leq \mathcal{D}_f[P||Q]. \tag{4.5}$$

Our choice of f-divergence can be considered as a variant of Pearson χ^2-divergence with a constant shift, which has also been adopted as a valid measure of distribution discrepancies [121, 122]. Compared with KL-divergence, this f-divergence enables unbiased estimation without deteriorating the optimality, whose advantages will become increasingly visible in Section 4.3.2.

Built upon the above transformations, we reach an objective that serves as an effective upper-bound of $\mathbb{D}_{\mathrm{KL}}[\mu^\pi(s,s')||\mu^E(s,s')]$:

$$\min_\pi J_{\mathrm{opolo}}(\pi) := \mathbb{E}_{\mu^\pi(s,s')}\left[\log \frac{\mu^R(s,s')}{\mu^E(s,s')}\right] + \mathcal{D}_f[\mu^\pi(s,a)||\mu^R(s,a)]. \tag{4.6}$$

4.3.2 Off-Policy Transformation

Optimization Eq (4.6) is still *on-policy* and induces additional challenges through the term $\mathcal{D}_f[\mu^\pi(s,a)||\mu^R(s,a)]$. However, we show that it can be readily transformed into off-policy learning. We first leverage the dual-form of an f-divergence [127]:

$$-\mathcal{D}_f[\mu^\pi(s,a)||\mu^R(s,a)] = \inf_{x:S\times A\to R} \mathbb{E}_{(s,a)\sim\mu^\pi}[-x(s,a)] + \mathbb{E}_{(s,a)\sim\mu^R}[f_*(x(s,a))],$$

and use this dual transformation to rewrite Eq (4.6):

$$\begin{aligned}\min_\pi J_{\mathrm{opolo}}(\pi) &\equiv \max_\pi \ \mathbb{E}_{\mu^\pi(s,s')}\left[-\log \frac{\mu^R(s,s')}{\mu^E(s,s')}\right] - \mathcal{D}_f[\mu^\pi(s,a)||\mu^R(s,a)] \\ &\equiv \max_\pi \min_{x:\mathcal{S}\times\mathcal{A}\to R} J_{\mathrm{opolo}}(\pi,x) := \mathbb{E}_{\mu^\pi(s,a,s')}\left[\log \frac{\mu^E(s,s')}{\mu^R(s,s')} - x(s,a)\right] + \mathbb{E}_{\mu^R(s,a)}[f_*(x(s,a))].\end{aligned} \tag{4.7}$$

If we consider a synthetic reward as $r(s,a,s') = \log \frac{\mu^E(s,s')}{\mu^R(s,s')} - x(s,a)$, the first term in Eq (4.7) resembles an RL return function: $\hat{J}(\pi) = \mathbb{E}_{(s,a,s')\sim\mu^\pi(s,a,s')}[r(s,a,s')]$. Observing this similarity, we turn to learn a Q-function by applying a change of variables:

$$Q(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a),a'\sim\pi(\cdot|s')}\left[-x(s,a) + \log \frac{\mu^E(s,s')}{\mu^R(s,s')} + \gamma Q(s',a')\right].$$

Equivalently, this Q function is a fixed point of a variant Bellman operator $\mathcal{B}^\pi Q$:

$$Q(s,a) = -x(s,a) + \mathbb{E}_{s'\sim P(\cdot|s,a),a'\sim\pi(\cdot|s')}\left[\log\frac{\mu^E(s,s')}{\mu^R(s,s')} + \gamma Q(s',a')\right] = -x(s,a) + \mathcal{B}^\pi Q(s,a).$$

Rewriting $x(s,a) = (\mathcal{B}^\pi Q - Q)(s,a)$ and applying it back to Eq (4.7), we finally remove the *on-policy* expectation by a series of telescoping:

$$\begin{aligned}
&\max_\pi \min_{x:S\times A\to R} J_{\text{opolo}}(\pi, x) \equiv \max_\pi \min_{Q:S\times A\to R} J_{\text{opolo}}(\pi, Q) \\
&:= \mathbb{E}_{(s,a,s')\sim\mu^\pi(s,a,s')}[\log\frac{\mu^E(s,s')}{\mu^R(s,s')} - (\mathcal{B}^\pi Q - Q)(s,a)] + \mathbb{E}_{(s,a)\sim\mu^R(s,a)}[f_*((\mathcal{B}^\pi Q - Q)(s,a))] \\
&= (1-\gamma)\mathbb{E}_{s_0\sim p_0, a_0\sim\pi(\cdot|s_0)}[Q(s_0,a_0)] + \mathbb{E}_{(s,a)\sim\mu^R(s,a)}[f_*((\mathcal{B}^\pi Q - Q)(s,a))]. \qquad (4.8)
\end{aligned}$$

A similar rationale has also been the key component of *distribution error correction* (DICE) [121, 122, 201]. Based on the above transformation, we propose our main objective:

$$\max_\pi \min_{Q:S\times A\to R} J_{\text{opolo}}(\pi, Q) := (1-\gamma)\mathbb{E}_{s_0\sim p_0, a_0\sim\pi(\cdot|s_0)}[Q(s_0,a_0)] + \mathbb{E}_{\mu^R(s,a)}[f_*((\mathcal{B}^\pi Q - Q)(s,a))]. \qquad (4.9)$$

Specifically, when $f(x) = f^*(x) = \frac{1}{2}x^2$, the second term $\mathbb{E}_{\mu^R(s,a)}[f_*((\mathcal{B}^\pi Q - Q)(s,a))]$ is reminiscent of an Bellman error, for which we can have unbiased estimation by mini-batch gradients.

Given access to the *off-policy* distribution $\mu^R(s,a)$ and the initial distribution p_0, optimization equation 4.9 can be efficiently realized once we resolve the term $\log\frac{\mu^E(s,s')}{\mu^R(s,s')}$ contained in $\mathcal{B}^\pi Q(s,a)$.

4.3.3 Adversarial Training with Off-Policy Experience

We can take the advantage of GAN training [56] to estimate the term $\log\frac{\mu^E(s,s')}{\mu^R(s,s')}$ inside $\mathcal{B}^\pi Q(s,a)$, by learning a discriminator D:

$$\max_{D:\mathcal{S}\times\mathcal{S}\to\mathbb{R}} \mathbb{E}_{(s,s')\sim\mu^E(s,s')}\Big[\log(D(s,s'))\Big] + \mathbb{E}_{(s,s')\sim\mu^R(s,s')}\Big[\log(1-D(s,s'))\Big],$$

which upon training to optimality, satisfies $\log(\frac{\mu_E(s,s')}{\mu^R(s,s')}) = \log D^*(s,s') - \log(1-D^*(s,s'))$. Unlike prior art [68, 171, 87] that requires estimating the ratio of $\log\frac{\mu^E}{\mu^\pi}$, the discriminator

in our case is designed to be off-policy in accordance with our proposed objective. Up to this step, optimization equation 4.9 can be achieved by interactively optimizing Q, π, and D with pure off-policy learning.

4.3.4 Policy Regularization as Forward Distribution Matching

Optimization 4.9 essentially minimizes an upper-bound of the *inverse* KL divergence:

$$\mathbb{D}_{\text{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')],$$

which is known to encourage a *mode-seeking* behavior [55]. Although mode-seeking is more robust to covariate-drift than *mode-covering* (such as behavior cloning), it requires sufficient explorations to find a reasonable state-distribution, especially at early learning stages. On the other hand, a mode-covering strategy has merits in quickly minimizing discrepancies on the *expert* distribution, by optimizing a *forward* KL-divergence such as $\mathbb{D}_{\text{KL}}[\pi_E(a|s)||\pi(a|s)]$.

To combine the advantages of both, in this section we show how we further speed up the learning procedure from a *mode-covering* perspective, without deteriorating the efficacy of our main objective. To achieve this goal, we first derive an optimizable lower-bound from a *mode-covering* objective:

$$\mathbb{D}_{\text{KL}}[\pi_E(a|s)||\pi(a|s)] = \mathbb{D}_{\text{KL}}[\mu^{E}(s'|s)||\mu^{\pi}(s'|s)] + \mathbb{D}_{\text{KL}}[\mu^{E}(a|s,s')||\mu^{\pi}(a|s,s')], \tag{4.10}$$

in which we define $\mu^{\pi}(s'|s) = \int_{\mathcal{A}} \pi(a|s)P(s'|s,a)da$ as the *conditional* state transition distribution induced by π, likewise for $\mu^{E}(s'|s)$.

Similar to Remark 3, the discrepancy $\mathbb{D}_{\text{KL}}[\mu^{E}(a|s,s')||\mu^{\pi}(a|s,s')]$ is not optimizable without knowing expert actions. However, under some mild assumptions, we found it feasible to optimize the other term $\mathbb{D}_{\text{KL}}[\mu^{E}(s'|s)||\mu^{\pi}(s'|s)]$ by enforcing a policy regularization:

Remark 4. *In a deterministic MDP, assuming the support of $\mu^{E}(s,s')$ is covered by $\mu^{R}(s,s')$, s.t. $\mu^{E}(s,s') > 0 \implies \mu^{R}(s,s') > 0$, then regulating policy using $\mu^{R}(\cdot|s,s')$ minimizes the **forward** distributional divergence $\mathbb{D}_{\text{KL}}[\mu^{E}(s'|s)||\mu^{\pi}(s'|s)]$:*

$$\exists \tilde{\pi} : \mathcal{S} \to \mathcal{A},\ \textit{s.t.}\ \forall (s,s') \sim \mu^{E}(s,s'),\ \tilde{\pi}(\cdot|s) \propto \mu^{R}(\cdot|s,s') \implies \tilde{\pi} = \arg\min_{\pi} \mathbb{D}_{\text{KL}}[\mu^{E}(s'|s)||\mu^{\pi}(s'|s)].$$

Intuitively, when expert labels are unavailable, this regularization can be considered as performing *states matching*, by encouraging the policy to yield actions that lead to desired footprints. Given a transition $s \to s'$ from the *expert observations*, a conditional distribution $\mu^R(\cdot|s, s')$ only has *support* on actions that yield this transition $s \to s'$. Therefore, following this regularization avoids the policy from drifting to undesired states.

In practice, we can estimate $\mu^R(\cdot|s, s')$ by learning an inverse action model P_I using off-policy transitions from $\mu^R(s, a, s')$ to optimize the following:

$$\max_{P_I:\mathcal{S}\times\mathcal{S}\to\mathcal{A}} -\mathbb{D}_{\mathrm{KL}}[\mu^R(a|s, s')||P_I(a|s, s')] \equiv \max_{P_I:\mathcal{S}\times\mathcal{S}\to\mathcal{A}} \mathbb{E}_{(s,a,s')\sim\mu^R(s,a,s')}[\log P_I(a|s, s')]. \tag{4.11}$$

4.3.5 Algorithm

Based on all the abovementioned building blocks, we now introduce *OPOLO* in Algorithm 4.1. *OPOLO* involves learning a policy π, a critic Q, a discriminator D, and an inverse action regularizer P_I, all of which can be done through off-policy training.

In particular, π and Q is jointly learned to find a saddle-point solution to optimization equation 4.9. The discriminator D assists this process by estimating a density ratio $\log \frac{\mu^E(s,s')}{\mu^R(s,s')}$. For better empirical performance, we adopt $-\log(1 - D(s, s'))$ as the discriminator's output, which corresponds to a constant shift inside the logarithm term, in that $\log(\frac{\mu^E(s,s')}{\mu^R(s,s')} + 1) = -\log(1 - D^*(s, s'))$. The inverse action model P_I serves as a regularizer to infer proper actions on the *expert observation distribution* to encourage *mode-covering* .

4.4 Related Work

The recent development on imitation learning can be divided into two categories:

Learning from Demonstrations (LfD) traces back to behavior cloning (***BC***) [138], in which a policy is pre-trained to minimize the prediction error on expert demonstrations. This approach is inherent with issues such as distribution shift and regret propagations. To address these limitations, [145] proposed a no-regret IL approach called *DAgger*, which however requires online access to oracle corrections. More recent LfD approaches favor Inverse reinforcement learning (***IRL***) [1], which work by seeking a reward function that

Algorithm 4.1: Off-Policy Learning from Observations

Input: expert observations $\mathcal{R}_T$, off-policy-transitions $\mathcal{R}$, initial states $\mathcal{S}_0$, f- function,
policy π_θ, critic Q_ϕ, discriminator D_w, inverse action model $P_{I\varphi}$, learning rate α.
for $n = 1, \ldots$ **do**
  sample trajectory $\tau \sim \pi_\theta$, $\mathcal{R} \leftarrow \mathcal{R} \cup \tau$
  update D_w: $w \leftarrow w + \alpha \hat{\mathbb{E}}_{(s,s')\sim\mathcal{R}_E}[\nabla_w \log(D_w(s,s'))] + \hat{\mathbb{E}}_{(s,s')\sim\mathcal{R}}[\nabla_w \log(1 - D_w(s,s'))]$.
    set $r(s,s') = -\log(1 - D_w(s,s'))$.
  update $P_{I\varphi}$: $\varphi \leftarrow \varphi + \alpha \hat{\mathbb{E}}_{(s,a,s')\sim\mathcal{R}}[\nabla_\varphi \log(P_{I\varphi}(a|s,s'))]$.
  update π_θ and Q_ϕ :
  $J(\pi_\theta, Q_\phi) = (1-\gamma)\hat{\mathbb{E}}_{s\sim\mathcal{S}_0}[Q_\phi(s,\pi_\theta(s))] + \hat{\mathbb{E}}_{(s,a,s')\sim\mathcal{R}}\Big[f^*\Big(r(s,s') + \gamma Q_\phi(s',\pi_\theta(s')) - Q_\phi(s,a)\Big)\Big]$.
  $J_{\mathrm{Reg}}(\pi_\theta) = \mathbb{E}_{(s,s')\sim\mathcal{R}_E, a\sim P_{I\varphi}(\cdot|s,s')}[\log \pi_\theta(a|s)]$.
    $\phi \leftarrow \phi - \alpha J_{\nabla\phi}(\pi_\theta, Q_\phi)$; $\theta \leftarrow \theta + \alpha\big(J_{\nabla\theta}(\pi_\theta, Q_\phi) + J_{\nabla\theta} J_{\mathrm{Reg}}(\pi_\theta)\big)$.

guarantees the superiority of expert demonstrations, based on which regular RL algorithms can be used to learn a policy [165, 209]. A representative instantiation of IRL is Generative Adversarial Imitation Learning (*GAIL*) [68]. It defines IL as a distribution matching problem and leverages the GAN technique [56] to minimize the Jensen-Shannon divergence between distributions induced by the expert and the learning policy. The success of *GAIL* has inspired a variety of other work, including those adopting different RL frameworks [87], or choosing different divergence measures [49, 136, 9] to enhance the effectiveness of imitation learning. Most work along this line focuses on *on-policy* learning, which is a sample-costly strategy.

As an *off-policy* extension of *GAIL* , *DAC* [87] improves the sample efficiency by re-using previous samples stored in a relay buffer rather than on-policy transitions. Similar ideas of reusing cached transitions can be found in [149]. One limitation of these approaches is that they neglected the discrepancy induced when replacing the on-policy distribution with off-policy approximations, which results in a deviation from their proposed objective. Another off-policy imitation learning approach is ValueDICE [88], which inherits the idea of DICE [121] to transform an on-policy LfD objective to an off-policy one. This approach, however, requires the information of expert actions, which otherwise makes off-policy estimation unreachable in a model-free setting. Therefore, their approach is not directly applicable to

LfO.

Learning from Observations (LfO) tackles a more challenging scenario where expert actions are unavailable. Work alone this line falls into *model-free* and *model-based* approaches. *GAIfO* [171] is a model-free solution that applies the principle of *GAIL* to learn a discriminator with state-only inputs. *IDDM* [189] further analyzed the theoretical gap between the LfD and LfO objectives, and proved that a lower-bound of this gap can be somewhat alleviated by maximizing the mutual-information between $(s, (a, s'))$, given an on-policy distribution $\mu^{\pi}(s, a, s')$. Its performance is comparable to *GAIL*. [22] assumed that the given observation sequences are ranked by superiority, based on which a reward function is designed for policy learning. Similar to *GAIL*, the sample efficiency of these approaches is suboptimal due to their *on-policy* strategy.

Model-based LfO can be further organized into learning a *forward* [161, 44] dynamics model or an *inverse* action model [169, 105]. Especially, [161] proposed a forward model solution to learn time-dependent policies for finite-horizon tasks, in which the number of policies to be learned equals the number of transition steps. This approach may not be suitable for tasks with long or infinite horizons. Behavior cloning from observations (*BCO*) [169] learns an inverse model to infer actions missing from the expert dataset, after which behavior cloning is applied to learn a policy. Besides the common issues faced by BC, this strategy does not guarantee that the ground-truth expert actions can be recovered, unless is a deterministic and injective MDP is assumed. Some other recent work focused on different problem settings than ours, in which the expert observations are collected with different transition dynamics [52] or from different viewpoints [105, 107, 157]. Readers are referred to [172] for further discussions of LfO.

4.5 Experiments

We compare *OPOLO* against state-of-the-art LfD and LfO approaches on MuJuCo benchmarks, which are locomotion tasks in continuous state-action space. In accordance with our assumption in Sec 4.3.4, these tasks have *deterministic* dynamics. Original rewards are re-

moved from all benchmarks to fit into an IL scenario. For each task, we collect 4 trajectories from a pre-trained expert policy. All illustrated results are evaluated across 5 random seeds.

Baselines: We compared *SAIL* against 7 baselines. We first selected 5 representative approaches from prior work: *GAIL* (on-policy LfD), *DAC* (off-policy LfD), *ValueDICE* (off-policy LfD), *GAIfO* (on-policy LfO), and *BCO* (off-policy LfO). We further designed two strong *off-policy* approaches, Specifically, we built *DACfO*, which is a variation of *DAC* that learns the discriminator on (s, s') instead of (s, a), and *ValueDICEfO*, which is built based on *ValueDICE*. Instead of using ground-truth expert actions, *ValueDICEfO* learns an inverse model by optimizing Eq (4.11), and uses the approximated actions generated by the inverse model to fit an LfO problem setting. To the best of our knowledge, *DACfO* and *ValueDICEfO* have not been investigated by any prior art. Among these baselines, *GAIL*, *DAC*, and *ValueDICE* are provided with both expert *states* and *actions*, while all other approaches only have access to expert *states*. More experimental details can be found in the supplementary material.

Our experiments focus on answering the following important questions:

1. *Asymptotic performance*: Is *OPOLO* able to achieve expert-level performance given a limited number of expert observations?
2. *Sample efficiency*: Can *OPOLO* recover expert policy using less interactions with the environment, compared with the state-of-the-art?
3. *Effects of the inverse action regularization*: Does the inverse action regularization useful in speeding up the imitation learning process?
4. *Sensitivity of the choice of f-divergence*: Can *OPOLO* perform well given different f functions?

4.5.1 Performance Comparison

OPOLO can recover expert performance given a fixed budget of expert observations. As shown in Figure 4.1, *OPOLO* reaches (near) optimal performance in all benchmarks. For

simpler tasks such as *Swimmer* and *InvertedPendulum*, most baselines can successfully recover expertise. For other complex tasks with high state-action space, on-policy baselines, such as *GAIL* and *GAIfO*, are struggling to reach their asymptotic performance within a limited number of interactions, As shown in Figure 4.2, the off-policy baseline *BCO* is prone to sub-optimality due to its behavior cloning-like strategy, On the other hand, the performance of *ValueDICEfO* can be deteriorated by potential action-drifts, as the inferred actions are not guaranteed to recover expertise. For fair comparison, performance of all *off-policy* approaches are summarized in Table 4.2 given a fixed number of interaction steps.

The asymptotic performance of *OPOLO* is 1) superior to *DACfO* and *ValueDICEfO*, 2) comparable to *DAC*, and 3) is more robust against overfitting compared with *ValueDICE*, whereas both *DAC* and *ValueDICE* enjoy the advantage of off-policy learning and extra action guidance.

Env	**HalfCheetah**	**Hopper**	**Walker**	**Swimmer**	**Ant**
BCO	3881.10±938.81	1845.66±628.41	421.24±135.18	256.88±4.52	1529.54±980.86
OPOLO-x	**7632.80±128.88**	3581.85±19.08	3947.72±97.88	246.62±1.56	5112.04±321.42
OPOLO	7336.96±117.89	3517.39±25.16	3803.00±979.85	257.38±4.28	**5783.57±651.98**
DAC	6900.00±131.24	3534.42±10.27	**4131.05±174.13**	232.12±2.04	5424.28±594.82
DACfO	7035.63±444.14	3522.95±93.15	3033.02±207.63	185.28±2.67	4920.76±872.66
ValueDICE	5696.94±2116.94	**3591.37±8.60**	1641.58±1230.73	262.73±7.76	3486.87±1232.25
ValueDICEfO	4770.37±644.49	3579.51±10.23	431.00±140.87	**265.05±3.45**	75.08±400.87
Expert	7561.78±181.41	3589.88±2.43	3752.67±192.80	259.52±1.92	5544.65±76.11
$(\mathcal{S}, \mathcal{A})$	$(17, 6)$	$(11, 3)$	$(17, 6))$	$(8, 2)$	$(111, 8)$

Table 4.2: Evaluated performance of *off-policy* approaches. Results are averaged over 50 trajectories.

4.5.2 Sample Efficiency

OPOLO is comparable with and sometimes superior to *DAC* in all evaluated tasks, and is much more sample-efficient than *on-policy* baselines. As shown in Figure 4.1, the sample-efficiency of *OPOLO* is emphasized by benchmarks with high state-action dimensions. In particular, for tasks such as *Ant* or *HalfCheetah*, the performance curves of *on-policy* baselines are barely improved at early learning stages. One intuition is that they need more explorations to build the current support of the learning policy, which cannot benefit from

cached transitions. For these challenging tasks, *OPOLO* is even more sample-efficient than *DAC* that has the guidance of expert actions. We ascribe this improvement to the *mode-covering* regularization of *OPOLO* enforced by its inverse action model, whose effect will be further analyzed in Sec 6.5.6. Meanwhile, other off-policy approaches such as *BCO* and *ValueDICEfO*, are prone to overfitting and performance degradation (as shown in Figure 4.2), which indicates that the effect of the inverse model alone is not sufficient to recover expertise. On the other hand, the *ValueDICE* algorithm, although being sample-efficient, is not designed to address LfO and requires expert actions.

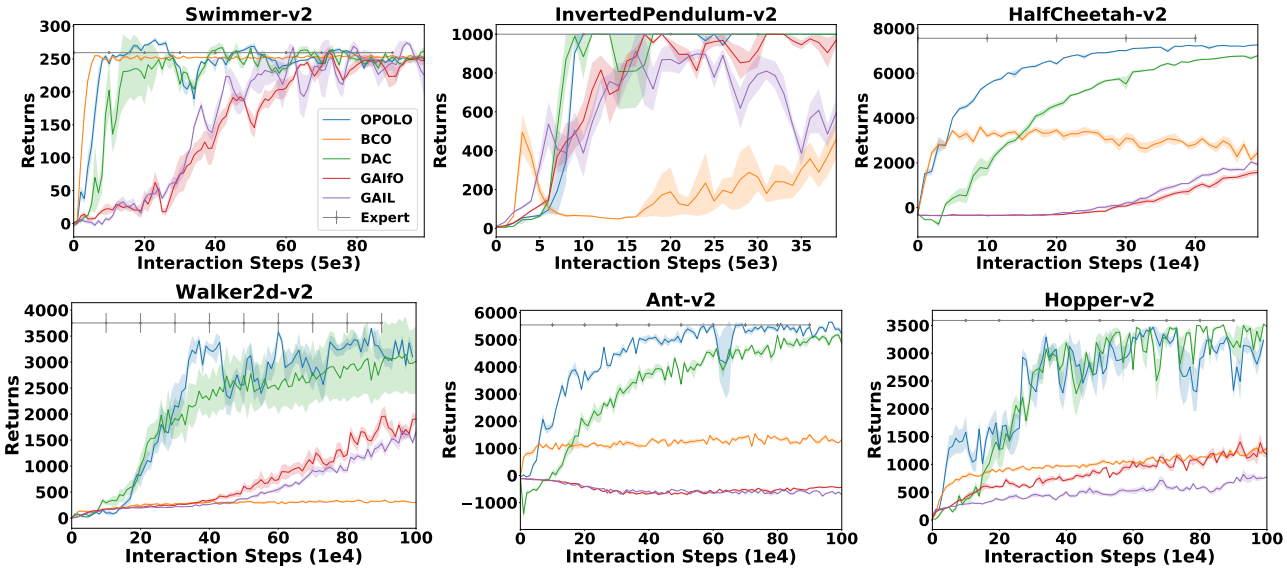


Figure 4.1: Interaction steps versus learning performance. Compared with others, our proposed approach (*OPOLO*) is the most sample-efficient to reach expert-level performance (Grey horizontal line).

4.5.3 Ablation Study

In this section, we further analyze the effects of the inverse action regularization by a group of ablation studies. Especially, we implement a variant of *OPOLO* that does not learn an inverse action model to regulate the policy update. We compare this approach, dubbed as *OPOLO-x*, against our original approach as well as the *DAC* algorithm.

Effects on Sample efficiency: Performance curves in Figure 4.3 show that removing the inverse action regularization from *OPOLO* slightly affects its sample-efficiency, although the degraded version is still comparable to *DAC*. This impact is more visible in challeng-

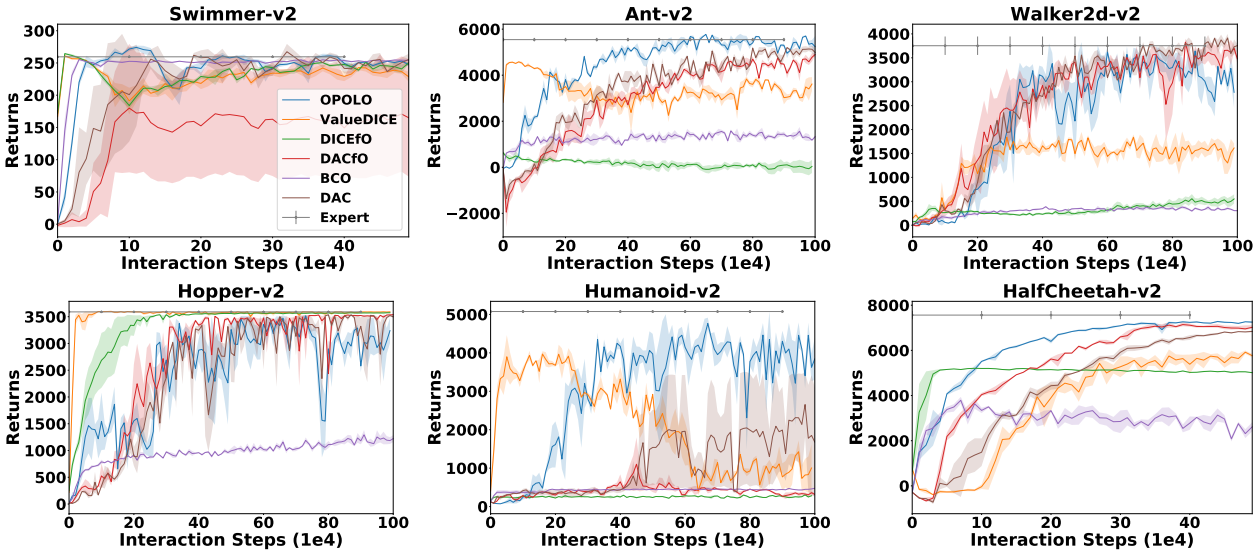


Figure 4.2: Compared with strong off-policy baselines, *OPOLO* is the only one that consistently achieves competitive performance across all tasks, without accessing expert actions.

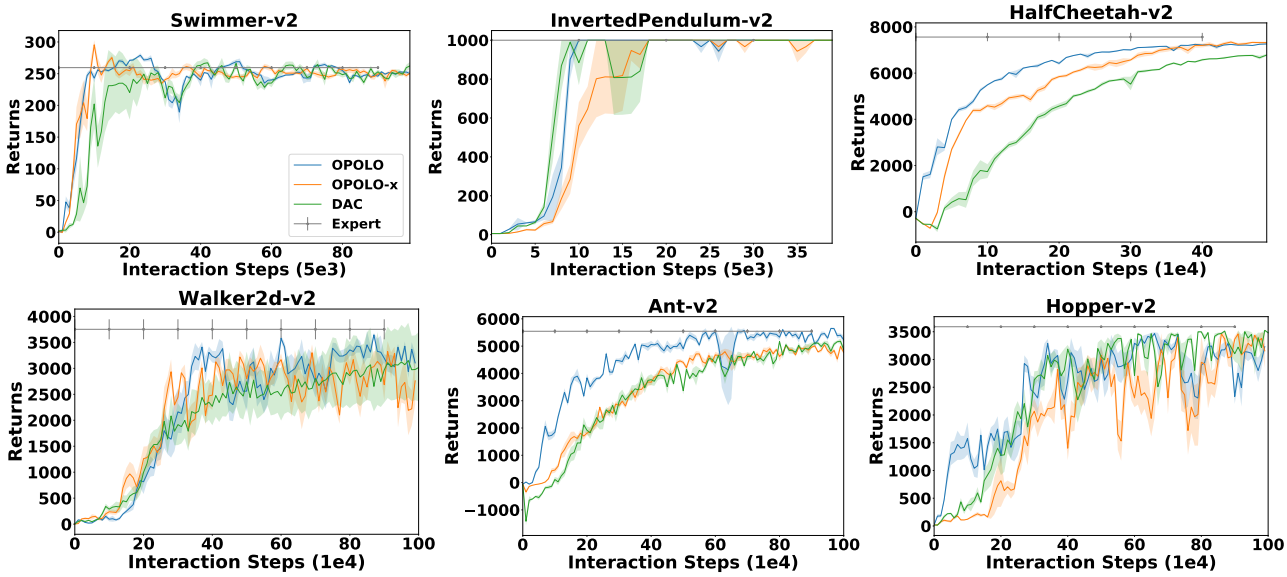


Figure 4.3: Removing the inverse action regularization (*OPOLO-x*) results in slight efficiency drop, although its performance is still comparable to *DAC* and *OPOLO*.

ing tasks such as *HalfCheetah* and *Ant*. From another perspective, the same phenomenon indicates that an inverse action regularization is beneficial for accelerating the IL process, especially for games with high observation spaces. An intuitive exploration is that, while our main objective serves as a driving force for *mode-seeking*, a regularization term assists by encouraging the policy to perform *mode-covering*. Combing these two motivations leads to a more efficient learning strategy.

Effects on Performance: Given a reasonable number of transition steps, the effects of an inverse-action model are less obvious regarding the asymptotic performance. As shown in Table 4.2, *OPOLO-x* is mostly comparable to *OPOLO* and *DAC*. This implies that the effect of the *state-covering* regularization will gradually fade out once the policy learns a reasonable state distribution. From another perspective, it indicates that following our main objective alone is sufficient to recover expert-level performance. Comparing with *BCO* which uses the inverse model solely for behavior cloning, we find it more effective when serving as a regularization to assist distribution matching from a *forward* direction.

4.5.4 Sensitivity Analysis

To analyze the effects of different f-functions on the performance of the proposed approach, we explored a family of f-divergence where $f(x) = \frac{1}{p}|x|^p, f^*(y) = \frac{1}{q}|y|^q, \text{s.t. } \frac{1}{p} + \frac{1}{q} = 1, p, q > 1$, as adopted by *DualDICE* [121]. Evaluation results show that *OPOLO* yields reasonable performance across different f-functions, although our choice ($q = p = 2$) turns out to be most stable. Results using the *Ant* task is illustrated in Figure 4.

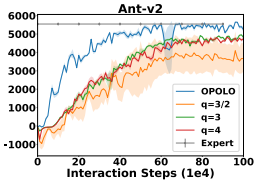


Figure 4.4: Performance given different f-functions.

4.6 Summary

Towards sample-efficient imitation learning from observations (LfO), we proposed a *principled* approach that performs imitation learning by accessing only a limited number of expert observations. We derived an upper bound of the original LfO objective to enable efficient off-policy optimization, and augment the objective with an inverse action model regulariza-

tion to speeds up the learning procedure. Extensive empirical studies are done to validate the proposed approach.

CHAPTER 5

DATA FREE KNOWLEDGE TRANSFER FOR HETEROGENEOUS FEDERATED LEARNING

This chapter is based on the work accompanying the following paper:

Zhu, Zhuangdi, Junyuan Hong, and Jiayu Zhou. *Data-Free Knowledge Distillation for Heterogeneous Federated Learning.* Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

5.1 Introduction

Federated Learning (FL) is an effective machine learning approach that enables the decentralization of computing and data resources. Classical FL, represented by FedAvg [116], obtains an aggregated model by iteratively averaging the parameters of distributed local user models, therefore omits the need of accessing their data. Serving as a communication-efficient and privacy-preserving learning scheme, FL has shown its potential to facilitate real-world applications, including healthcare [154], biometrics [3], and natural language processing [63, 6], to name just a few.

Along with its promising prospect, FL faces practical challenges from data ***heterogeneity*** [98], in that user data from real-world is usually *non-iid* distributed, which inherently induces deflected local optimum [80]. Moreover, the permutation-invariant property of deep neural networks has further increased the heterogeneity among user models [198, 178]. Thus, performing element-wise averaging of local models, as adopted by most existing FL approaches, may not induce an ideal global model [99, 98].

A variety of efforts have been made to tackle user heterogeneity, mainly from two complementary perspectives: one focuses on stabilizing local training, by regulating the deviation of local models from a global model over the *parameter space* [98, 40, 80]. This approach may not fully leverage the underlying knowledge across user models, whose diversity suggests

informative structural differences of their local data and thus deserves more investigation. Another aims to improve the efficacy of model aggregation [198, 30], among which *knowledge distillation* has emerged as an effective solution [103, 96]. Provided with an unlabeled dataset as the proxy, knowledge distillation alleviates the model drift issue induced by heterogeneity, by enriching the global model with the ensemble knowledge from local models, which is shown to be more effective than simple parameter-averaging. However, the prerequisite of a *proxy data* can leave such an approach infeasible for many applications, where a carefully engineered dataset may not always be available on the server. Moreover, by only refining the global model, the inherent heterogeneity among user models is not fully addressed, which may in turn affect the quality of the knowledge ensemble, especially if they are biased due to limited local data [82], which is a typical case for FL.

Observing the challenge in the presence of user heterogeneity and the limitations of prior art, in this work, we propose a ***data-free*** knowledge distillation approach for FL, dubbed as FedGen (*Fe*derated *D*istillation via *Gen*erative Learning). Specifically, FedGen learns a generative model derived solely from the prediction rules of user models, which, given a target label, can yield feature representations that are consistent with the ensemble of user predictions. This generator is later broadcasted to users, escorting their model training with augmented samples over the latent space, which embodies the distilled knowledge from other peer users. Given a latent space with a dimension much smaller than the input space, the generator learned by FedGen can be lightweight, introducing minimal overhead to the current FL framework.

The proposed FedGen enjoys multifold benefits: i) It extracts the knowledge out of users which was otherwise mitigated after model averaging, without depending on any external data. ii) Contrary to certain prior work that only refines the global model, our approach directly regulates local model updating using the extracted knowledge. We show that such knowledge imposes an inductive bias to local models, leading to better generalization performance under *non-iid* data distributions. iii) Furthermore, the proposed approach is ready to

address more challenging FL scenarios, where sharing *entire* model parameters is impractical due to privacy or communication constraints, since the proposed approach only requires the prediction layer of local models for knowledge extraction.

Extensive empirical studies echoed by theoretical elaborations show that, our proposed approach yields a global model with better generalization performance using fewer communication rounds, compared with the state-of-the-art.

5.2 Notations and Preliminaries

Without ambiguity, in this work, we discuss a typical FL setting for *supervised learning*, i.e., the general problem of multi-class classification. Let $\mathcal{X} \subset \mathbb{R}^p$ be an instance space, $\mathcal{Z} \subset \mathbb{R}^d$ be a *latent* feature space with $d < p$, and $\mathcal{Y} \subset \mathbb{R}$ be an output space. $\mathcal{T}$ denotes a *domain* which consists of a data distribution $\mathcal{D}$ over $\mathcal{X}$ and a ground-truth *labeling* function $c^* : \mathcal{X} \to \mathcal{Y}$, *i.e.* $\mathcal{T} := \langle \mathcal{D}, c^* \rangle$. Note that we will use the term *domain* and *task* equivalently. A model parameterized by $\boldsymbol{\theta} := [\boldsymbol{\theta}^f; \boldsymbol{\theta}^p]$ consists of two components: a feature extractor $f : \mathcal{X} \to \mathcal{Z}$ parametrized by $\boldsymbol{\theta}^f$, and a predictor $h : \mathcal{Z} \to \triangle^{\mathcal{Y}}$ parameterized by $\boldsymbol{\theta}^p$, where $\triangle^{\mathcal{Y}}$ is the simplex over $\mathcal{Y}$. Given a non-negative, convex loss function $l : \triangle^{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$, the ***risk*** of a model parameterized by $\boldsymbol{\theta}$ on domain $\mathcal{T}$ is defined as $\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) := \mathbb{E}_{x \sim \mathcal{D}} \left[l \left(h(f(x; \boldsymbol{\theta}^f); \boldsymbol{\theta}^p), c^*(x) \right) \right]$.

Federated Learning aims to learn a global model parameterized by $\boldsymbol{\theta}$ that minimizes its risk on each of the user tasks $\mathcal{T}_k$ [116]:

$$\min_{\boldsymbol{\theta}} \ \mathbb{E}_{\mathcal{T}_k \in \boldsymbol{\mathcal{T}}} \left[\mathcal{L}_k(\boldsymbol{\theta}) \right], \tag{5.1}$$

where $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_k\}_{k=1}^K$ is the collection of user tasks. We consider all tasks sharing the same labeling rules c^* and loss function l, *i.e.*, $\mathcal{T}_k = \langle \mathcal{D}_k, c^* \rangle$. In practice, Equation 5.1 is empirically optimized by $\min_{\boldsymbol{\theta}} \ \frac{1}{K} \sum_{k=1}^K \hat{\mathcal{L}}_k(\boldsymbol{\theta})$, where $\hat{\mathcal{L}}_k(\boldsymbol{\theta}) := \frac{1}{|\hat{\mathcal{D}}_k|} \sum_{x_i \in \hat{\mathcal{D}}_k} \left[l(h(f(x_i; \boldsymbol{\theta}^f); \boldsymbol{\theta}^p), c^*(x_i)) \right]$ is the *empirical* risk over an observable dataset $\hat{\mathcal{D}}_k$. An implied assumption for FL is that the ***global*** data $\hat{\mathcal{D}}$ is distributed to each of the local domains, with $\hat{\mathcal{D}} = \cup \{\hat{\mathcal{D}}_k\}_{k=1}^K$.

Knowledge Distillation (KD) is also referred as a *teacher-student* paradigm, with the goal of learning a lightweight student model using knowledge distilled from one or more powerful teachers [26, 11]. Typical KD leverages a ***proxy*** dataset $\hat{\mathcal{D}}_{\mathrm{P}}$ to minimize the discrepancy between the logits outputs from the teacher model $\boldsymbol{\theta}_T$ and the student model $\boldsymbol{\theta}_S$, respectively. A representative choice is to use Kullback-Leibler divergence to measure such discrepancy [67]:

$$\min_{\boldsymbol{\theta}_S} \ \mathbb{E}_{x\sim\hat{\mathcal{D}}_{\mathrm{P}}}\left[\mathbb{D}_{\mathrm{KL}}\left[\sigma(g(f(x;\boldsymbol{\theta}_T^f);\boldsymbol{\theta}_T^p)\|\sigma(g(f(x;\boldsymbol{\theta}_S^f);\boldsymbol{\theta}_S^p)\right]\right],$$

where $g(\cdot)$ is the logits output of an predictor h, and $\sigma(\cdot)$ is the non-linear activation applied to such logits, *i.e.* $h(z;\boldsymbol{\theta}^p)=\sigma(g(z;\boldsymbol{\theta}^p))$.

The idea of KD has been extended to FL to tackle user heterogeneity [103, 30], by treating each user model $\boldsymbol{\theta}_k$ as the *teacher*, whose information is aggregated into the *student* (global) model $\boldsymbol{\theta}$ to improve its generalization performance:

$$\min_{\boldsymbol{\theta}} \ \mathbb{E}_{x\sim\hat{\mathcal{D}}_{\mathrm{P}}}\left[\mathbb{D}_{\mathrm{KL}}[\sigma(\frac{1}{K}\sum_{k=1}^{K} g(f(x;\boldsymbol{\theta}_k^f);\boldsymbol{\theta}_k^p))\|\sigma(g(f(x;\boldsymbol{\theta}^f);\boldsymbol{\theta}^p)]\right].$$

One primary limitation of the above approach resides in its dependence on a proxy dataset $\hat{\mathcal{D}}_{\mathrm{P}}$, the choice of which needs delicate consideration and plays a key role in the distillation performance [103]. Next, we show how we make KD feasible for FL in a *data-free* manner.

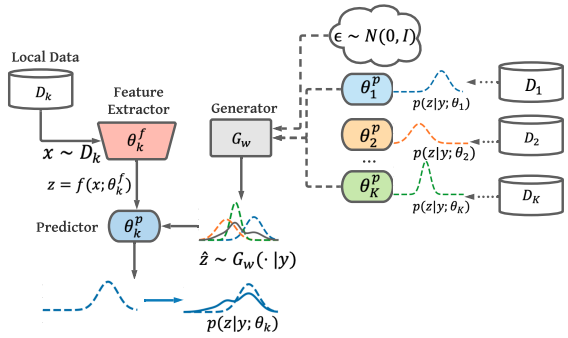


Figure 5.1: Overview of FEDGEN: a generator $G_{\boldsymbol{w}}(\cdot|y)$ is learned by the server to aggregate information from different local clients without observing their data. The generator is then sent to local users, whose knowledge is distilled to user models to adjust their interpretations of a good feature distribution.

Algorithm 5.1: Data Free Federated Distillation via Generalized Learning

Require: Tasks $\{\mathcal{T}_k\}_{k=1}^K$;
  Global parameters $\boldsymbol{\theta}$, local parameters $\{\boldsymbol{\theta}_k\}_{k=1}^K$;
  Generator parameter $\boldsymbol{w}$; $\hat{p}(y)$ uniformly initialized;
  Learning rate α, β, local steps T, batch size B, local label counter c_k.
repeat
  Server selects active users $\mathcal{A}$ uniformly at random, then broadcast $\boldsymbol{w}, \boldsymbol{\theta}$, $\hat{p}(y)$ to $\mathcal{A}$.
  for all user $k \in \mathcal{A}$ in parallel **do**
    $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}$,
    for $t = 1, \ldots, T$ **do**
      $\{x_i, y_i\}_{i=1}^B \sim \mathcal{T}_k$, $\{\hat{z}_i \sim G_{\boldsymbol{w}}(\cdot|\hat{y}_i), \hat{y}_i \sim \hat{p}(y)\}_{i=1}^B$.
      Update label counter c_k.
    $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \beta \nabla_{\boldsymbol{\theta}_k} J(\boldsymbol{\theta}_k)$. ▷ Optimize Equation 5.5
    User sends $\boldsymbol{\theta}_k$, c_k back to server.
  Server updates $\boldsymbol{\theta} \leftarrow \frac{1}{|\mathcal{A}|} \sum_{k \in \mathcal{A}} \boldsymbol{\theta}_k$, and $\hat{p}(y)$ based on $\{c_k\}_{k \in \mathcal{A}}$.
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$. ▷ Optimize Equation 5.4
until training stop

5.3 FedGen: Data Free Federated Distillation via Generative Learning

In this section, we elaborate our proposed approach with a summary shown in Algorithm 5.1. An overview of its learning procedure in illustrated in Figure 5.1.

5.3.1 Knowledge Extraction

Our core idea is to extract knowledge about the global view of data distribution, which is otherwise non-observable by conventional FL, and distill such knowledge to local models to guide their learning. We first consider learning a conditional distribution $Q^* : \mathcal{Y} \rightarrow \mathcal{X}$ to characterize such knowledge, which is consistent with the ground-truth data distributions:

$$Q^* = \underset{Q:\mathcal{Y} \rightarrow \mathcal{X}}{\arg\max} \ \mathbb{E}_{y \sim p(y)} \mathbb{E}_{x \sim Q(x|y)} [\log p(y|x)], \tag{5.2}$$

where $p(y)$ and $p(y|x)$ are the ground-truth *prior* and *posterior* distributions of the target labels, respectively, both of which are unknown. To make Equation 5.2 optimizable w.r.t Q, we replace $p(y)$ and $p(x|y)$ with their empirical approximations. First, we estimate $p(y)$ as:

$$\hat{p}(y) \propto \sum\nolimits_k \mathbb{E}_{x \sim \hat{\mathcal{D}}_k} [\mathrm{I}(c^*(x) = y)],$$

where $\mathrm{I}(\cdot)$ is an indicator function, and $\hat{\mathcal{D}}_k$ is the observable data for domain $\mathcal{T}_k$. In practice, $\hat{p}(y)$ can be obtained by requiring the training label counts from users during the model uploading phase. Next, we approximate $p(y|x)$ using the ensemble wisdom from user models:

$$\log \hat{p}(y|x) \propto \frac{1}{K} \sum\nolimits_{k=1}^{K} \log p(y|x; \boldsymbol{\theta}_k).$$

Equipped with the above approximations, directly optimizing Equation 5.2 over the input space $\mathcal{X}$ can still be prohibitive: it brings computation overloads when $\mathcal{X}$ is of high dimension, and may also leak information about the user data profile. A more approachable idea is hence to recover an *induced* distribution $G^* : \mathcal{Y} \to \mathcal{Z}$ over a latent space, which is more compact than the raw data space and can alleviate certain privacy-related concerns:

$$G^* = \underset{G:\mathcal{Y}\to\mathcal{Z}}{\arg\max}\ \mathbb{E}_{y\sim\hat{p}(y)}\mathbb{E}_{z\sim G(z|y)}\left[\sum_{k=1}^{K} \log p(y|z; \boldsymbol{\theta}_k^p)\right]. \tag{5.3}$$

Following the above reasoning, we aim to perform knowledge extraction by learning a conditional generator G, parameterized by $\boldsymbol{w}$ to optimize the following objective:

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}) := \mathbb{E}_{y\sim\hat{p}(y)}\mathbb{E}_{z\sim G_{\boldsymbol{w}}(z|y)}\left[l(\sigma(\frac{1}{K}\sum_{k=1}^{K} g(z; \boldsymbol{\theta}_k^p)), y)\right], \tag{5.4}$$

where g and σ are the logit-output and the activation function as defined in Section 5.2. Given an arbitrary sample y, optimizing Equation 5.4 only requires access to the predictor modules $\boldsymbol{\theta}_k^p$ of user models. Specifically, to enable diversified outputs from $G(\cdot|y)$, we introduce a noise vector $\epsilon \sim \mathcal{N}(0, I)$ to the generator, which is resemblant to the re-parameterization technique proposed by prior art [83], so that $z \sim G_{\boldsymbol{w}}(\cdot|y) \equiv G_{\boldsymbol{w}}(y, \epsilon|\epsilon \sim \mathcal{N}(0, I))$. We discuss more implementation details in the supplementary.

Given arbitrary target labels y, the proposed generator can yield feature representations $z \sim G_{\boldsymbol{w}}(\cdot|y)$ that induce ideal predictions from the ensemble of user models. In other words, the generator approximates an induced image of a *consensual* distribution, which is consistent with the user data from a global view.

5.3.2 Knowledge Distillation

The learned generator $G_{\boldsymbol{w}}$ is then broadcasted to local users, so that each user model can sample from G_w to obtain augmented representations $z \sim G_w(\cdot|y)$ over the feature space. As a result, the objective of a local model $\boldsymbol{\theta}_k$ is altered to maximize the probability that it yields ideal predictions for the augmented samples:

$$\min_{\boldsymbol{\theta}_k} \; J(\boldsymbol{\theta}_k) := \hat{\mathcal{L}}_k(\boldsymbol{\theta}_k) + \hat{\mathbb{E}}_{y\sim\hat{p}(y), z\sim G_{\boldsymbol{w}}(z|y)} \left[l(h(z;\boldsymbol{\theta}_k^p); y) \right], \tag{5.5}$$

where $\hat{\mathcal{L}}_k(\boldsymbol{\theta}_k) := \frac{1}{|\hat{\mathcal{D}}_k|} \sum_{x_i \in \hat{\mathcal{D}}_k} \left[l(h(f(x_i;\boldsymbol{\theta}_k^f);\boldsymbol{\theta}_k^p), c^*(x_i)) \right]$ is the empirical risk given local data $\hat{\mathcal{D}}_k$. We show later that the augmented samples can introduce inductive bias to local users, reinforcing their model learning with a better generalization performance.

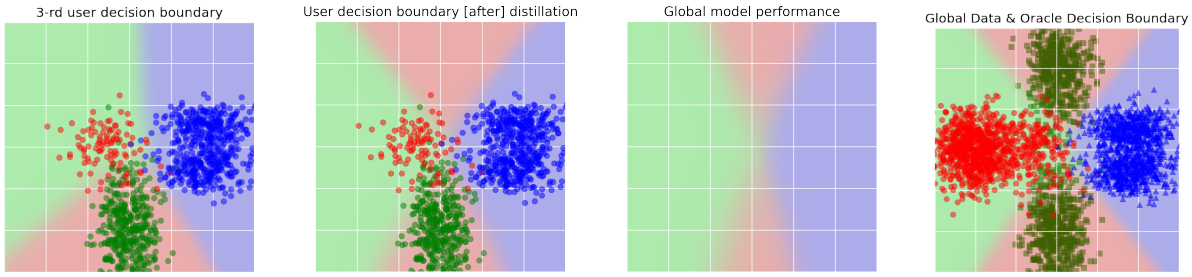
Up to this end, our proposed approach has realized data-free knowledge distillation, by interactively learning a lightweight generator that primarily depends on the prediction rule of local models, and leveraging the generator to convey consensual knowledge to local users. We justify in Section 5.6.2 that our approach can effectively handle user heterogeneity in FL, which also enjoys theoretical advantages as analyzed in Section 5.4.

5.3.3 Extensions for Flexible Parameter Sharing

In addition to tackling data heterogeneity, FedGen can also handle a challenging FL scenario where sharing the entire model is against communication or privacy prerequisites. On one hand, advanced networks with deep feature extraction layers typically contain millions of parameters [65, 24], which bring significant burdens to communication. On the other hand, it has been shown feasible to backdoor regular FL approaches [177]. For practical FL applications such as healthcare or finance, sharing entire model parameters may be associated with considerable privacy risks, as discussed in prior work [64].

FedGen is ready to alleviate those problems, by sharing only the prediction layer $\boldsymbol{\theta}_k^p$ of local models, which is the primary information needed to optimizing Equation 5.4, while keeping the feature extractor $\boldsymbol{\theta}_k^f$ localized. This partial sharing paradigm is more efficient, and at the same time less vulnerable to data leakage, as compared with a strategy that shares

the entire model. Empirical study in Section 5.6.4 shows that, FEDGEN significantly benefits local users, even without sharing feature extraction modules. We defer the algorithmic summary of this variant approach to the supplementary.



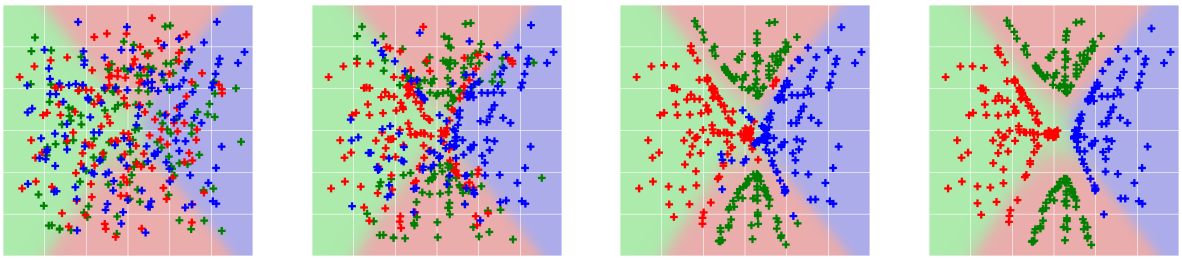
(a) Decision boundary ***before*** KD.

(b) Decision boundary ***after*** KD.

(c) Global model decision boundary.

(d) *Oracle* decision boundary.

Figure 5.2: After KD, accuracy has improved from 81.2% to 98.4% for one user (Fig 5.2a - Fig 5.2b), while a global model obtained by parameter-averaging (*without* KD) has 93.2% accuracy (Fig 5.2c), compared with an oracle model with 98.6% accuracy (Fig 5.2d).



(a) Randomly initialized $r(x|y)$.

(b) $r(x|y)$ learned after 50 batches.

(c) $r(x|y)$ learned after 150 batches.

(d) $r(x|y)$ learned after 250 batches.

Figure 5.3: Samples from the generator gradually approaches to real distribution, where each user model (teacher) sees limited, disjoint local data. The background color indicates oracle decision boundaries learned over the global data.

5.4 FEDGEN Analysis

In this section, we provide multiple perspectives to understand our proposed approach. We first visualize *what* knowledge is learned and distilled by FEDGEN, then analyze *why* the distilled knowledge is favorable, from the viewpoint of *distribution matching* and *domain adaptation*, respectively. We primarily focus on interpreting the rationale behind FEDGEN and leave detailed discussion and derivations to the supplementary.

5.4.1 Knowledge Distillation for Inductive Bias

We illustrate the KD process in FEDGEN on a FL prototype, which contains three users, each assigned with a disjoint dataset $\hat{\mathcal{D}}_k, k \in \{1, 2, 3\}$. When trained using only the local data, a user model is prone to learn biased decision boundaries (See Figure 5.2a).

Next, a generator $G_w(\cdot|y)$ is learned based on the prediction rule of user models. For clear visualizations, we learn $G_w(\cdot|y)$ on the raw feature space $\mathcal{Y} \to \mathcal{X} \subset \mathbb{R}^2$ instead of a latent space. As shown in Figure 5.3, $r(x|y)$, which denotes the distribution derived from $G_w(x|y)$, gradually coincides with the ground-truth $p(x|y)$ (Figure 5.2d), even when the individual local models are biased. In other words, $G_w(x|y)$ can fuse the aggregated information from user models to approximate a global data distribution.

We then let users sample from $G_w(x|y)$, which serves as an inductive bias for users with limited data. As a result, each user can observe beyond its own training data and adjust their decision boundaries to approach to the ensemble wisdom (Figure 5.2b).

5.4.2 Knowledge Distillation for Distribution Matching

A notable difference between FEDGEN and prior work is that the knowledge is distilled to user models instead of the global model. As a result, the distilled knowledge, which conveys inductive bias to users, can directly regulate their learning by performing *distribution matching* over the latent space $\mathcal{Z}$:

Remark 5. *Let $p(y)$ be the prior distribution of labels, and $r(z|y) : \mathcal{Y} \to \mathcal{Z}$ be the conditional distribution derived from generator $G_{\boldsymbol{w}}$. Then regulating a user model $\boldsymbol{\theta}_k$ using samples from $r(z|y)$ can minimize the conditional KL-divergence between two distributions, derived from the generator and the user, respectively:*

$$\max_{\boldsymbol{\theta}_k} \ \mathbb{E}_{y\sim p(y), z\sim r(z|y)} \left[\log p(y|z; \boldsymbol{\theta}_k)\right] \equiv \min_{\boldsymbol{\theta}_k} \ \mathbb{D}_{\mathrm{KL}}[r(z|y) \| p(z|y; \boldsymbol{\theta}_k)], \tag{5.6}$$

where we define $p(z|y; \boldsymbol{\theta}_k)$ as the probability that the input feature to the predictor $\boldsymbol{\theta}_k$ is z given that it yields a label y. In practice, Equation 5.6 is optimized by using empirical samples from the generator: $\{(z, y)|y \sim \hat{p}(y), z \sim G_w(z|y)\}$, which is consistent with the

second term of the local model objective (Equation 5.5), in that $\forall\ y \in \mathcal{Y}$:

$$\max_{\boldsymbol{\theta}_k}\ \mathbb{E}_{z\sim r(z|y)}\left[\log p(y|z;\boldsymbol{\theta}_k)\right] \approx \min_{\boldsymbol{\theta}_k}\ \mathbb{E}_{z\sim G_{\boldsymbol{w}}(z|y)}\left[l(h(z;\boldsymbol{\theta}_k^p);y)\right].$$

Distinguished from prior work that applies weight regularization to local models [98, 40], FedGen can serve as an alternative and compatible solution to address user heterogeneity, which inherently bridges the gap among user models w.r.t their interpretations of an ideal feature distribution.

5.4.3 Knowledge Distillation for Improved Generalization

One can also draw a theoretical connection from the knowledge learned by FedGen to an improved generalization bound. To see this, we first present a performance bound for the aggregated model in FL, which is built upon prior arts from *domain adaptation* [17, 16]:

Theorem 2. (**Generalization Bounds for FL**) *Consider an FL system with K users. Let $\mathcal{T}_k = \langle \mathcal{D}_k, c^* \rangle$ and $\mathcal{T} = \langle \mathcal{D}, c^* \rangle$ be the k-th local domain and the global domain, respectively. Let $\mathcal{R}: \mathcal{X} \to \mathcal{Z}$ be a feature extraction function that is simultaneously shared among users. Denote h_k the hypothesis learned on domain $\mathcal{T}_k$, and $h = \frac{1}{K}\sum_{k=1}^{K} h_k$ the global ensemble of user hypotheses. Then with probability at least $1-\delta$:*

$$\begin{aligned}\mathcal{L}_{\mathcal{T}}(h) &\equiv \mathcal{L}_{\mathcal{T}}\left(\frac{1}{K}\sum_k h_k\right)\\ &\leq \frac{1}{K}\sum_k \hat{\mathcal{L}}_{\mathcal{T}_k}(h_k) + \frac{1}{K}\sum_k \left(d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \lambda_k\right) + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4K}{\delta}\right)},\end{aligned}$$

*where $\hat{\mathcal{L}}_{\mathcal{T}_k}(h_k)$ is the empirical risk on $\mathcal{T}_k$, $\lambda_k := \min_h(\mathcal{L}_{\mathcal{T}_k}(h) + \mathcal{L}_{\mathcal{T}}(h))$ denotes an oracle performance. $d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}})$ denotes the divergence measured over a symmetric-difference hypothesis space. $\tilde{\mathcal{D}}_k$ and $\tilde{\mathcal{D}}$ is the **induced** image of $\mathcal{D}_k$ and $\mathcal{D}$ over $\mathcal{R}$, respectively, s.t. $\mathbb{E}_{z\sim\tilde{\mathcal{D}}_k}[\mathcal{B}(z)] = \mathbb{E}_{x\sim\mathcal{D}_k}[\mathcal{B}(\mathcal{R}(x))]$ given a probability event $\mathcal{B}$, and so for $\tilde{\mathcal{D}}$.*

Specifically, $\mathcal{L}_{\mathcal{T}}(h)$ is usually considered as an ideal upper-bound for the global model in FL [135, 103]. Two key implications can be derived from Theorem 2: i) Large user ***heterogeneity*** leads to high distribution divergence ($d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}})$), which undermines the

quality of the global model; ii) More empirical samples (m) are favorable to the generalization performance, which softens the numerical constraints.

In other words, the generalization performance can be improved by enriching local users with augmented data that aligns with the global distribution:

Corollary 1. *Let $\mathcal{T}$, $\mathcal{T}_k$, $\mathcal{R}$ defined as in Theorem 2. $\mathcal{D}_{\mathrm{A}}$ denotes an **augmented** distribution, and $\mathcal{D}'_k = \frac{1}{2}(\mathcal{D}_k + \mathcal{D}_{\mathrm{A}})$ is a **mixture** of distributions. Accordingly, $\tilde{\mathcal{D}}_{\mathrm{A}}$, $\tilde{\mathcal{D}}'_k$ denotes the **induced** image of $\mathcal{D}_{\mathrm{A}}$, $\mathcal{D}'_k$ over $\mathcal{R}$, respectively. Let $\hat{\mathcal{D}}'_k = \hat{\mathcal{D}}_k \cup \hat{\mathcal{D}}_{\mathrm{A}}$ be an empirical dataset of $\mathcal{D}'_k$, with $|\hat{\mathcal{D}}_k| = m$, $|\hat{\mathcal{D}}'_k| = m' > m$. If $d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}})$ is bounded, s.t $\exists\ \epsilon > 0, d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}}) \leq \epsilon$, then with probability $1 - \delta$:*

$$\mathcal{L}_{\mathcal{T}}(h) \leq \frac{1}{K}\sum_k \mathcal{L}_{\mathcal{T}'_k}(h_k) + \frac{1}{K}\sum_k (d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}}) + \lambda'_k) + \sqrt{\frac{4}{m'}\left(d\log\frac{2em'}{d} + \log\frac{4K}{\delta}\right)}, \tag{5.7}$$

where $\mathcal{T}'_k = \{\mathcal{D}'_k, c^*\}$ is the updated local domain, $d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}}) \leq d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}})$ when ϵ is small, and $\sqrt{\frac{4}{m'}(d\log\frac{2em'}{d} + \log\frac{4K}{\delta})} < \sqrt{\frac{4}{m}(d\log\frac{2em}{d} + \log\frac{4K}{\delta})}$.

Such an augmented distribution $\mathcal{D}_{\mathrm{A}}$ can facilitate FL from multiple aspects: not only does it relax the numerical constraints with more empirical samples ($m' > m$), but it also reduces the discrepancy between the local and global feature distributions ($d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}})$). This finding coincides the merits of FedDGen: since the generator $G_w(z|y)$ is learned to recover an aggregated distribution over the feature space, one can treat samples from the generator $\{z|y \sim \hat{p}(y), z \sim G_w(z|y)\}$ as the augmented data from $\tilde{\mathcal{D}}_{\mathrm{A}}$, which naturally has a small deviation from the global induced distribution $\tilde{\mathcal{D}}$. More rigorous analysis along this line is left to our future work. We elaborate the role of such an augmentation distribution $\mathcal{D}_A$ in the supplementary.

5.5 Related Work

Federated Learning (FL) is first proposed by [116] as a decentralized machine learning paradigm. Subsequent work along this line tackles different challenges faced by FL, including heterogeneity [80, 98, 113], privacy [43, 2], communication efficiency [58, 86], and convergence analysis [77, 139, 197]. Specifically, a wealth of work has been proposed to handle

user ***heterogeneity***, by regularizing model weight updates [98], allowing personalized user models [45, 40], or introducing new model aggregation schemes [198, 113]. We refer readers to [97] for an organized discussion of recent progress on FL.

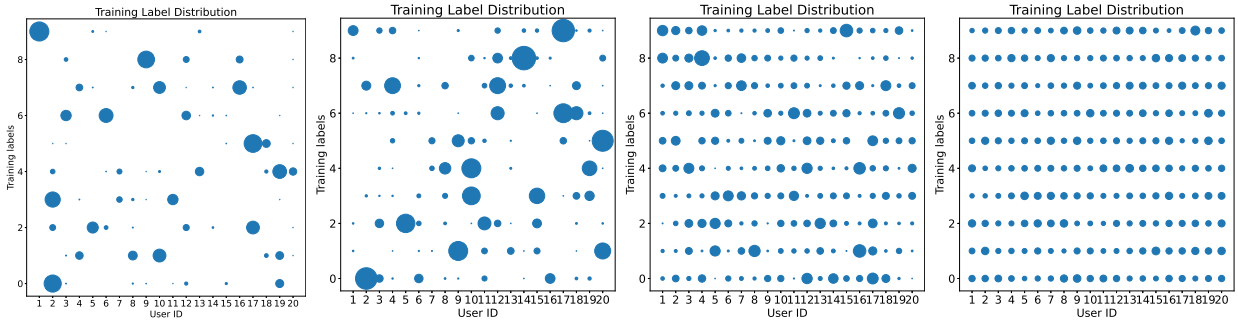
Knowledge Distillation (KD) is a technique to compress knowledge from one or more teacher models into an empty student [67, 26, 11, 74]. Conventional KD hinges on a proxy dataset [67]. More recent work enables KD with fewer data involved, such as dataset distillation [180], or core-data selection [173, 152]. Later there emerges data-free KD approaches which aim to reconstruct samples used for training the teacher [193, 117]. Particularly, [110] extracts the meta-data from the teacher's activation layers. [193] learns a conditional generator that yields samples which maximize the teacher's prediction probability of a target label. Along with the same spirit, [117] learns a generator by adversarial training. Different from prior work, we learn a generative model that is tailored for FL, by ensembling the knowledge of multiple user models over the latent space, which is more lightweight for learning and communication.

Knowledge Distillation in Federated Learning has recently emerged as an effective approach to tackle user heterogeneity. Most existing work is data-dependent [103, 159, 58, 30]. Particularly, [103] proposed **FedDFusion**, which performs KD to refine the global model, assuming that an unlabeled dataset is available with samples from the same or similar domains. Complementary KD efforts have been made to confront data heterogeneity [96, 150]. Specifically, [96] transmits the proxy dataset instead of the model parameters. FedAUX [150] performs *data-dependent* distillation by leveraging an auxiliary dataset to initialize the server model and to weighted-ensemble user models, while FedGen performs knowledge distillation in a data-free manner. FedMix [194] is a *data-augmented* FL framework, where users share their *batch-averaged* data among others to assist local training. On the country, FedGen extracts knowledge from the existing user model parameters, which faces fewer privacy risks. **FedDistill** (Federated Distillation) is proposed by [153] which extracts from user models the statistics of the logit-vector outputs, and shares this meta-data

to users for KD. We provide detailed comparisons with work along this line in Section 5.6.

5.6 Experiments

In this section, we compare the performance of our proposed approach with other key related work. We leave implementation details and extended experimental results to the supplementary.



(a) $\alpha = 0.05$ (b) $\alpha = 0.1$ (c) $\alpha = 1$ (d) $\alpha = 10$

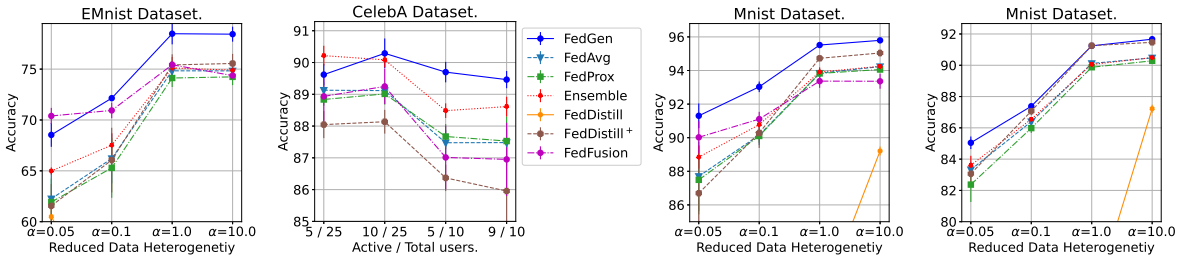
Figure 5.4: Visualization of statistical heterogeneity among users on MNIST dataset, where the x-axis indicates user IDs, the y-axis indicates class labels, and the size of scattered points indicates the number of training samples for a label available to that user.

		Top-1 Test Accuracy.						
Dataset	Setting	FEDAVG	FEDPROX	FEDENSEMBLE	FEDDISTILL	FEDDISTILL$^+$	FEDDFUSION	FEDGEN
MNIST, T=20	$\alpha = 0.05$	87.70±2.07	87.49±2.05	88.85±0.68	70.56±1.24	86.70±2.27	90.02±0.96	**91.30±0.74**
	$\alpha = 0.1$	90.16±0.59	90.10±0.39	90.78±0.39	64.11± 1.36	90.28±0.89	91.11±0.43	**93.03±0.32**
	$\alpha = 1$	93.84±0.25	93.83 ± 0.29	93.91±0.28	79.88±0.66	94.73±0.15	93.37±0.40	**95.52±0.07**
CELEBA, T=20	$r = 5/10$	87.48±0.39	87.67±0.39	88.48±0.23	76.68±1.23	86.37±0.41	87.01±1.00	**89.70±0.32**
	$r = 5/25$	89.13±0.25	88.84±0.19	**90.22±0.31**	74.99±1.57	88.05± 0.43	88.93±0.79	89.62±0.34
	$r = 10/25$	89.12±0.20	89.01±0.33	90.08±0.24	75.88±1.17	88.14±0.37	89.25±0.56	**90.29±0.47**
EMNIST, T=20	$\alpha = 0.05$	62.25±2.82	61.93±2.31	64.99±0.35	60.49±1.27	61.56±2.15	**70.40±0.79**	68.53±1.17
	$\alpha = 0.1$	66.21±2.43	65.29±2.94	67.53±1.19	50.32±1.39	66.06±3.18	70.94±0.76	**72.15±0.21**
	$\alpha = 10$	74.83± 0.69	74.24±0.81	74.90±0.80	54.77±0.33	75.55 ±0.94	74.36±0.40	**78.43±0.74**
EMNIST, α=1	$T = 20$	74.83±0.99	74.12±0.88	75.12±1.07	46.19±0.70	75.41±1.05	75.43±0.37	**78.48±1.04**
	$T = 40$	77.02±1.09	75.93 ±0.95	77.68±0.98	46.72±0.73	78.12±0.90	77.58±0.37	**78.92± 0.73**

Table 5.1: Performance overview. For MNIST and EMNIST, a *smaller* α indicates *higher* heterogeneity. For CELEBA, r denotes the ratio between active users and total users. T denotes the number of local training steps.

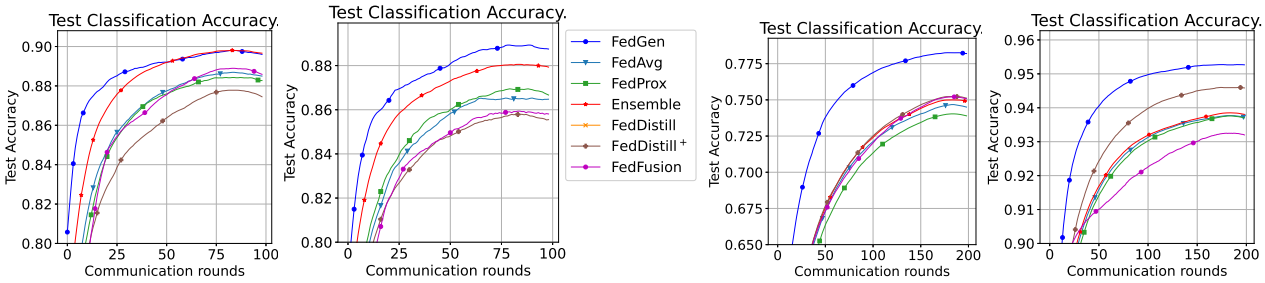
5.6.1 Setup

Baselines: In addition to **FEDAVG** [116], **FEDPROX** regularizes the local model training with a proximal term in the model objective [98]. **FEDENSEMBLE** extends FEDAVG



(a) EMnist Data. (b) CelebA Dataset. (c) MNISTon CNN. (d) Mnist on MLP.

Figure 5.5: Visualized performance w.r.t data heterogeneity.



(a) CelebA,$r = 0.4$. (b) CelebA, $r = 0.5$. (c) EMnist, $\alpha = 1$. (d) Mnist, $\alpha = 1$.

Figure 5.6: Selected learning curves, averaged over 3 random seeds.

to ensemble the prediction output of all user models. **FedDFusion** is a data-based KD approach [103], for which we provide unlabeled training samples as the proxy dataset. **FedDistill** [75] is a data-free KD approach which shares label-wise average of logit-vectors among users. It does not share network parameters and therefore experiences non-negligible performance drops compared with other baselines. For a fair comparison, we derive a baseline from FedDistill, which shares both model parameters and the label-wise logit vectors. We name this stronger baseline as **FedDistill**$^+$.

Dataset: We conduct experiments on three image datasets: **Mnist** [94], **EMnist** [35], and **CelebA** [109], as suggested by the Leaf FL benchmark [29]. Among them, Mnist and EMnist dataset is for digit and character image classifications, and CelebA is a celebrity-face dataset which is used to learn a binary-classification task, *i.e.* to predict whether the celebrity in the picture is smiling.

Configurations: Unless otherwise mentioned, we run 200 global communication rounds, with 20 user models in total and an active-user ratio $r = 50\%$. We adopt a local updating step $T = 20$, and each step uses a mini batch with size $B = 32$. We use at most 50% of

the total training dataset and distribute it to user models, and use all testing dataset for performance evaluation. For the classifier, we follow the network architecture of [116], and treat the last MLP layer as the predictor $\boldsymbol{\theta}_k^p$ and all previous layers as the feature extractor $\boldsymbol{\theta}_k^f$. The generator $G_{\boldsymbol{w}}$ is MLP based. It takes a noise vector ϵ and an one-hot label vector y as the input, which, after a hidden layer with dimension d_h, outputs a feature representation with dimension d. To further increase the diversity of the generator output, we also leverage the idea of *diversity loss* from prior work [114] to train the generator model.

User heterogeneity: for MNIST and EMNIST dataset, we follow prior arts [103, 71] to model non-iid data distributions using a Dirichlet distribution $\mathbf{Dir}(\alpha)$, in which a smaller α indicates higher data heterogeneity, as it makes the distribution of $p_k(y)$ more biased for a user k. We visualize the effects of adopting different α on the statistical heterogeneity for the MNIST dataset in Figure 5.4. For CELEBA, the raw data is naturally non-iid distributed. We further increase the data heterogeneity by aggregating pictures belonging to different celebrities into disjoint groups, with each group assigned to one user.

5.6.2 Performance Overview:

From Table 6.2, we can observe that FEDGEN outperforms other baselines with a considerable margin.

Impacts of data heterogeneity: FEDGEN is the only algorithm that is robust against different levels of user heterogeneity while consistently performs well. As shown in Figure 5.5, the gain of FEDGEN is more notable when the data distributions are highly heterogeneous (with a small α). This result verifies our motivations, since the advantage of FEDGEN is induced from the knowledge distilled to local users, which mitigates the discrepancy of latent distributions across users. This knowledge is otherwise not accessible by baselines such as FEDAVG or FEDPROX.

As one of the most competitive baselines, the advantage of FEDDFUSION vanishes as data heterogeneity becomes mitigated, which gradually becomes comparable to FEDAVG, as shown in Figure 5.5a and Figure 5.5c. Unlike FEDDFUSION, the performance gain of our

approach is consistently significant, which outperforms FedDFusion in most cases. This discrepancy implies that our proposed approach, which directly distills the knowledge to user models, can be more effective than fine-tuning the global model using a proxy dataset, especially when the distilled knowledge contains inductive bias to guide local model learning.

As a data-free KD baseline, FedDistill experiences non-negligible performance drops, which implies the importance of parameter sharing in FL. FedDistill$^+$, on the other hand, is vulnerable to data heterogeneities. As shown in Table 6.2, it can outperform FedAvg when data distributions are near-iid (*e.g.* when $\alpha \geq 1$), thanks to the shared logit statistics as the distilled knowledge, but performs worse than FedAvg when α gets smaller, which indicates that sharing such meta-data alone may not be effective enough to confront user heterogeneity.

FedEnsemble enjoys the benefit of ensemble predictions from all user models, although its gain is less significant compared with FedGen. We ascribe the leading performance of our approach to the better-generalized performance of local models. Guided by the distilled knowledge, a user model in FedGen can quickly jump out of its local optimum, whose aggregation can be better than the ensemble of potentially biased models as in FedEnsemble.

Learning efficiency: As shown in Figure 5.6, FedGen has the most rapid learning curves to reach a performance and outperforms other baselines. Although FedDFusion enjoys a learning efficiency higher than other baselines under certain data settings, due to the advantages induced from proxy data, our approach can directly benefit each local user with actively learned knowledge, whose effect is more explicit and consistent (More illustrations in supplementary).

Comments on sharing generative model: Given a compact latent space, the generative model can be lightweight for learning or downloading. In practice, we use a generator network with 2 compact MLP layers, whose parameter size is small compared with the user classification model. The above empirical results also indicate that the leading performance gain combined with a faster convergence rate can trade off the communication load brought

by sharing a generative model.

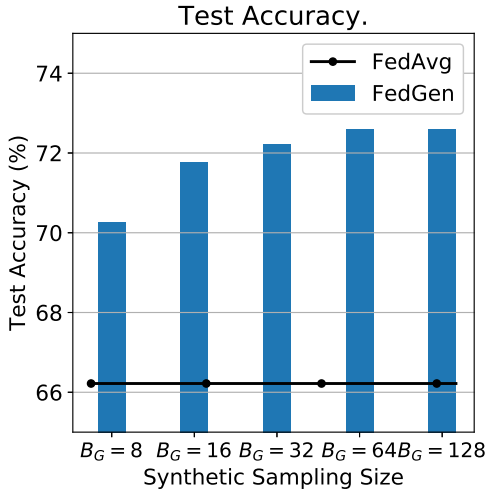


Figure 5.7: Effects of synthetic samples.

	Effects of the Generator Network Structure.				
$[d_\epsilon,\ d_h]$	$[64, 256]$	$[32, 256]$	$[32, 128]$	$[16, 128]$	$[32, 64]$
Accuracy(%)	FedAvg=66.22±2.58				
FedGen	71.61±0.25	72.09±0.46	72.43±0.57	72.01±0.76	70.98±0.85

Table 5.2: Effects of the generator's network structure, using EMnist dataset with $\alpha = 0.1$.

	Performance w.r.t synthetic sample sizes.				
G **sampling size**	$B_G = 8$	$B_G = 16$	$B_G = 32$	$B_G = 64$	$B_G = 128$
Training time (ms)	FedAvg= 47.66 ± 1.68				
FedGen	57.20±2.22	57.39±2.21	58.17±2.24	58.91±2.29	60.06±2.32

Table 5.3: Effects of the number of synthetic samples, using EMnist dataset with $\alpha = 0.1$.

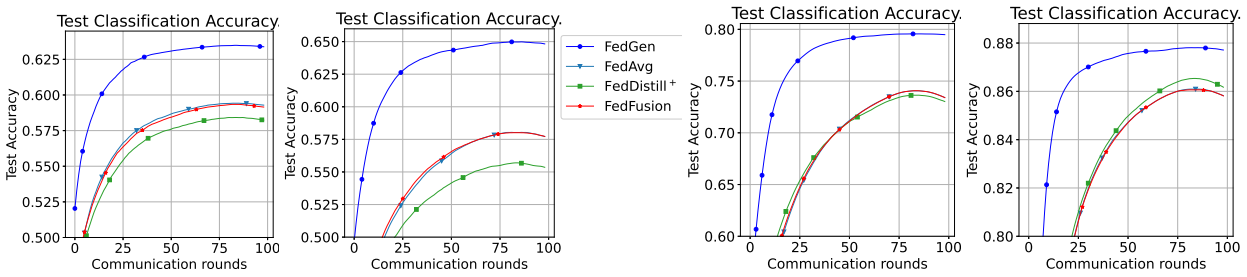
5.6.3 Sensitivity Analysis

Impacts of straggler users: We explore different numbers of total users versus active users on the CelebA dataset, with the active ratios r ranging from 0.2 to 0.9. Figure 5.5b shows that our approach is next to FedEnsemble when the number of straggler users are high ($r = 0.2$, with 5 out 25 active users per learning round), and is consistently better than all baselines w.r.t to the asymptotic performance given a moderate number of active users. Combined with Figure 5.6a and Figure 5.6b, one can observe that our approach requires

much less communication rounds to reach high performance, regardless of the setting of straggler users.

Effects of different network architectures: we conduct analysis on the MNIST dataset, using both CNN and MLP network architectures. As shown in Figure 5.5d and Figure 5.5c, the outstanding performance of FEDGEN is consistent across two different network settings, although the overall performance trained with CNN networks is noticeably higher than those with MLP networks.

Effects of communication frequency: We explore different local updating steps T on the EMNIST, so that a higher T means longer communication delays before the global communication. Results in Table 6.2 indicates that our approach is robust against different levels of communication delays (See supplementary for more results).



(a) $\alpha = 0.05$ (b) $\alpha = 0.1$ (c) $\alpha = 1$ (d) $\alpha = 10$

Figure 5.8: Learning curves on MNIST with limited parameter sharing.

Top 1 Accuracy (%)				
Algorithms	FEDAVG	FEDDISTILL$^+$	FEDDFUSION	FEDGEN
$\alpha = 0.05$	59.67±0.76	58.83±0.62	59.62±0.84	**63.60±0.63**
$\alpha = 0.1$	58.39±0.74	56.25±0.98	58.38±0.81	**65.42±0.29**
$\alpha = 1$	74.49± 0.57	74.24±0.60	74.51±0.55	**79.72±0.52**
$\alpha = 10$	86.35±0.60	86.89±0.26	86.28±0.69	**87.92±0.46**

Table 5.4: Performance overview on the MNIST dataset, by only sharing the last prediction layer.

Effects of the generator's network architecture and sampling size: Extended analysis has verified that FEDGEN is *robust* across different generator network architectures (Table 5.2). Moreover, sampling synthetic data from the generator only adds minor training

workload to local users (Table 5.2). The gain of FEDGEN over FEDAVG is consistently remarkable given different synthetic sample sizes, whereas a sufficient number of synthetic samples brings even better performance (Figure 5.7). Especially, in Table 5.2, we explored different dimensions for the input noise (d_ϵ) and the hidden layer (d_h) of the generator while keeping its output layer dimension fixed (*i.e.* the dimension of the feature space $\mathcal{Z}$). Table 5.3 shows the training time for one local update, averaged across users and the communication rounds. B_G denotes the number of *synthetic* samples used for each mini-batch optimization. By default, we set $B_G = B$, and B is the number of *real* samples drawn from the local dataset (see Algorithm 1).

5.6.4 Extensions to Flexible Parameter Sharing

Motivated to alleviate privacy and communication concerns, FEDGEN is ready to benefit distributed learning without sharing entire model parameters. To explore this potential, we conduct a case study on FEDAVG, FEDDISTILL$^+$, and FEDGEN, where user models share only the last prediction layer and keep their feature extraction layers localized. Note that FEDDFUSION is not designed to address FL with partial parameter sharing, which requires entire user models for KD. For a fair comparison, we modify FEDDFUSION to let it upload entire user models during the model aggregation phase, but disable the downloading of feature extractors, so that the server model can still be fine-tuned using the proxy data.

Results in Table 5.4 show that our approach consistently outperforms other baselines by a remarkable margin, the trend of which is more significant given high data heterogeneity (Figure 5.8). Its distinguished performance from FEDDFUSION verifies the efficacy of data-free distillation under this challenging scenario. These promising results show that FEDGEN has the potential to further reduce communication workload, not only by fast convergence but also by a flexible parameter sharing strategy.

5.7 Summary

In this chapter, we propose an FL paradigm that enables efficient knowledge distillation to address user heterogeneity without requiring any external data. Extensive empirical experiments, guided by theoretical implications, have shown that our proposed approach can benefit federated learning with better generalization performance using fewer communication rounds, compared with the state-of-the-art.

CHAPTER 6

FEDRESCUE: SELF-KNOWLEDGE DISTILLATION FOR RESILIENT AND COMMUNICATION EFFICIENT FEDERATED LEARNING

This chapter is based on our research accompanied by the following paper:

Zhu, Zhuangdi, Junyuan Hong, Steve Drew, and Jiayu Zhou. *Resilient and Communication Efficient Learning for Heterogeneous Federated Learning.* Proceedings of the 39th International Conference on Machine Learning, 2022.

6.1 Introduction

Federated Learning (FL) is a decentralized machine learning scheme that eliminates private data sharing on participating devices. Recent years witnessed effervescent development of FL in varied domains, including healthcare [144], computer vision [106], natural language processing [63, 116], and Internet of things (IoT) [81, 42], to name just a few. The rapid adoption and deployment of edge computing have enabled computing to be even closer to the source of the data and users. The growing demand for privacy-aware and low-latency machine learning at the edge makes FL a natural fit.

The diversities among participating devices and their network topologies are phenomenal, which imposed significant challenges of *statistical* and *system* heterogeneity to FL. The statistical heterogeneity in FL has been extensively tackled by techniques such as regularized optimization [40], customized model aggregation [178], and data augmentation [205]. In comparison, system heterogeneity is induced by significant gaps in memory capacities and transmission bandwidth among edge devices, the impact of which is under-explored. Moreover, traditional FL hinges on reliable connections, where model parameters are transmitted between edge devices and a central server without packet loss. This prerequisite can be prohibitive for practical edge-based applications, including autonomous driving and IoT, where devices such as wearable devices and vehicles can frequently opt-in, opt-out, or move

around. Under faulty network connections, prior FL solutions may become fragile when the model parameters fail to be intactly shared among active users due to transmission interruptions. This connection uncertainty is *bidirectional*, which exists either when a participant downloads or uploads parameter updates, leading to nonnegligible information loss of the participating devices. To the best of our knowledge, few pioneer efforts have been made to address the transmission uncertainty in FL, leaving most FL learning schemes at potential risk of undermined performance.

Observing the ***system heterogeneity*** and ***connection uncertainty*** in FL, in this paper, we propose an FL framework that addresses both challenges simultaneously. In our approach, edge devices learn a ***prunable*** neural network by ***self-distillation***, such that a model can be structurally pruned to submodels that contain adequate knowledge of the learning domain. Towards effective optimization, we propose ***progressive learning***, which articulates the knowledge representation in the model in a nested and progressive structure. This strategy enables FL participants with diverging model architectures to fully devote their knowledge to model aggregation. Furthermore, powered with a ***sequential model transmission*** paradigm, our approach is especially beneficial in amortizing the risk of connection interruptions, since the partially transmitted model parameters before interruption can still contribute self-contained domain knowledge to the recipient.

To the best of our knowledge, we are the first to address both system heterogeneity and connection uncertainty in FL by progressively learning self-distilled networks. Extensive empirical studies have verified that, our proposed approach, dubbed as ***FedResCuE***, is ***Res***ilient to unstable transmission connections while ***C***omm***u***nication ***E***fficient under system heterogeneity, which achieves high asymptotic performance compared with the state-of-the-art.

6.2 Problem Setting

6.2.1 Prelimnaries of Federated Learning

Without loss of generality, we consider a learning setting that addresses a representative problem of *multi-class classification*. Let $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_k\}_{k=1}^K$ denote the learning domains of edge devices, where a domain $\mathcal{T}_k = \langle \mathcal{X}_k \times \mathcal{Y}_k \rangle$ is defined by a joint input and output distribution. Let $\boldsymbol{\theta}_k$ represent *local* model parameters, and $\boldsymbol{w}$ *global* model parameters, which are usually obtained via parameter-wise averaging on $\{\boldsymbol{\theta}_k\}_{k=1}^K$ [116]. Denote $\mathcal{L} : \nabla^{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}^+$ the loss function recgonized by all domains, where $\nabla^{\mathcal{Y}}$ is a *simplex* over $\mathcal{Y}$. The objective of FL is to learn a global model that generalizes well on all devices: $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_{\mathcal{T}_k \sim \boldsymbol{\mathcal{T}}}[\mathcal{L}(f(\mathcal{X}_k; \boldsymbol{w}), \mathcal{Y}_k)]$, which is approximated by empirical data in practice:

$$\hat{\boldsymbol{w}}^* = \arg\min_{\boldsymbol{w}} \frac{1}{K} \sum\nolimits_{k=1}^{K} \left[\frac{1}{n_k} \sum\nolimits_{i=1}^{n_k} \mathcal{L}(f(x_k^i; \boldsymbol{w}), y_k^i) \right],$$

where $\{x_k^i, y_k^i\}_{i=1}^{n_k} \subset \mathcal{T}_k$. A FL system typically involves four iterative phases: i) the *downloading* phase, when the server broadcasts a global model to active users; ii) the *local learning* phase, when active users update their local model parameters; iii) the *uploading* phase, when active users send parameters back to the server, and iv) the *aggregation* phase, when the server derives a global model using user-uploaded parameters.

6.2.2 Learning with System Heterogeneity

In this paper, we tackle FL under system heterogeneity, where edge devices can learn local models $\boldsymbol{\theta}^k$ with various network architectures due to different capacities in memory and transmission bandwidth [69, 39]. To enable FL with system heterogeneity, participants will first agree on the largest model architecture (*i.e.* a $\times 1$ network), while smaller models in this system are treated as the pruned versions (*i.e.* a $\times p$ network) with a pruning ratio $p < 1$. In Deep Neural Networks (DNNs), such pruning is manifested as reducing the number of

channels or filters. For instance, the weight matrix $\boldsymbol{w} \in \mathbb{R}^{m \times n}$ in a Convolution or Linear layer l will be pruned to $\boldsymbol{w}[: m \times p, : n \times p]$ by ratio p. [1]

This strategy leaves one potential drawback to model *aggregation*, in that the global model is obtained via parameter-wise averaging on edge models with heterogeneous sizes. Naively learning and aggregating such model parameters may ignore the divergence in their feature extraction patterns induced by architecture heterogeneity, leading to impaired global model performance. Consequently, the knowledge uploaded by users with diverging model sizes might not be absorbed well by their peers.

6.2.3 Learning with Unstable Network Connection

Another challenge tackled in this paper is the *connection instability* in FL, which differs from the ***straggler*** issue as discussed in prior art [143, 98]. The former refers to *active* users transmitting *partial* instead of complete model parameters due to connection interruption, while the latter results from *inactive* users that did not participate in model learning. Connection interruption may occur bidirectionally either during the downloading phase or the uploading phase. It is a common issue that can be induced by multiple factors, including bandwidth, transmission power, noisy density, and interference [31, 126], yet enough effort has been made to address it. A naive solution to connection interruption is to ignore the faulty connected devices and treat them as stragglers [32, 98], which may waste the local learning of those devices that could otherwise be leveraged to improve the global model.

6.3 Resilient and Communication Efficient FL

Towards addressing the challenges of system heterogeneity and unstable network connections, we aim to learn neural networks that can be *structurally decomposed* for learning, inference, and transmission. Observing the natural property of deep neural networks, we propose to *vertically* decompose a model as a sequence of *columns*, while a column can be one or

[1] Without losing generality, in this paper, we unify the pruning ratio p for all layers in a model, although such pruning can be extended to choose different ratios p_l for different layers l.

more model channels described in Section 6.2.2, depending on the desired granularity. This proposed paradigm enjoys twofold benefits:

- During local learning, the predictive knowledge is *progressively* captured in model columns and can be incrementally enriched with more column connections, which ***benefits FL with system heterogeneity***, in that the knowledge from heterogeneous models can be structurally aligned in the global model.
- During FL synchronization, model columns are *sequentially* transmitted between the server and the edge device. It makes FL ***resilient to unstable connections***, since losing parts of the tailing columns upon interruption does not lead to a catastrophic undermining of domain representations, and a lightweight submodel that is successfully transmitted still contains intact predictive knowledge. Moreover, this *divided-and-transmit* strategy is also in accord with lower-level transmission protocols. We demonstrate this process in Figure 6.1.

Orthogonal to our work, there are personalized FL approaches, which either divide a model *horizontally* then transmit the feature extraction layers [8], or selectively transmit parameters with *unordered* structures [158]. Contrarily, in our FL paradigm, the received columns can be readily assembled for learning and inference. Therefore, instead of discarding the partially received parameters upon connection interruption, they can be effectively utilized for global aggregation or local model initialization.

6.3.1 Learning Self-Distilled Local Models

We aim to learn a model that can be structurally decomposed, arbitrarily *prunable* with reduced *columns*, and dispenses with the need for fine-tuning. We name such a model *self-distilled*, which more concretely, shall satisfy the following objective:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} \ \mathcal{L}(f(\mathcal{X};\boldsymbol{\theta}),\mathcal{Y}) + \mathop{\mathbb{E}}_{p\sim\mathcal{P}}\left[\mathcal{L}(f(\mathcal{X};\boldsymbol{\theta}_{\times p}),\mathcal{Y}) + \mathbb{D}_{\mathrm{KL}}[f(\mathcal{X};\boldsymbol{\theta})\|f(\mathcal{X};\boldsymbol{\theta}_{\times p})]\right], \tag{6.1}$$

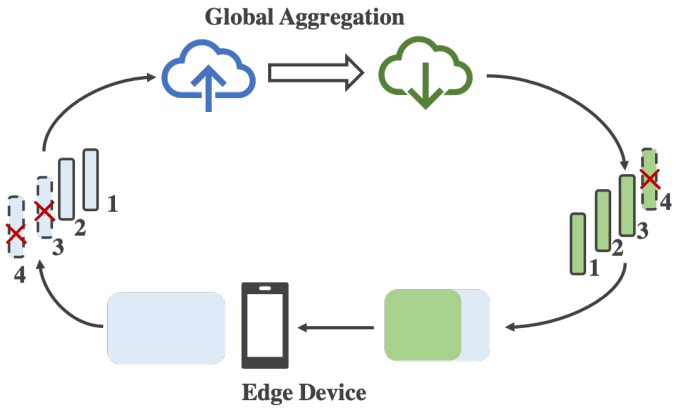


Figure 6.1: Model parameters are divided and learned as *columns*, which are then transmitted *sequentially* between the server and clients, until than an interruption occurs to one column, or when all columns are transmitted successfully.

where Θ denotes the parameter space; $\mathcal{X} \times \mathcal{Y}$ is the learning domain; $\mathcal{P} = \{p | p \leq 1, \forall \boldsymbol{\theta} \in \Theta \ f(\mathcal{X}; \boldsymbol{\theta}_{\times p}) \subseteq \nabla^{\mathcal{Y}}\}$ is the set of legitimate pruning ratios; $\mathbb{D}_{\mathrm{KL}}[p \| q]$ denotes the KL-divergence between distribution p and q.

The notion of *self-distillation* is embodied by two components in Equation 6.1: The first is $\mathcal{L}(f(\mathcal{X}; \boldsymbol{\theta}_{\times p}), \mathcal{Y})$, which induces the largest model $\boldsymbol{\theta}$ to maintain arbitrary submodels $\boldsymbol{\theta}_{\times p}$ that are effective for the learning domain. The other is $\mathbb{D}_{\mathrm{KL}}[f(\mathcal{X}; \boldsymbol{\theta}) \| f(\mathcal{X}; \boldsymbol{\theta}_{\times p})]$, in that it encourages the predictive knowledge captured by $\boldsymbol{\theta}$ (teacher), which is manifested as the learned posterior $p(\cdot | \mathcal{X}; \boldsymbol{\theta}) \propto f(\mathcal{X}; \boldsymbol{\theta})$, to be distilled to the submodel $\boldsymbol{\theta}_{\times p}$ (student) by distribution matching. We verify in Section 6.5.6 that, the $\mathbb{D}_{\mathrm{KL}}$ term is especially beneficial when a submodel itself is not representative enough to capture sophisticated features due to a limited number of channels. Hence the posterior distribution from the teacher serves as finer-grained guidance in addition to the label supervision. Moreover, not only is the learned self-distilled network robust against *connection loss*, it also benefits FL under *system heterogeneity*, as the knowledge of an edge model with a larger structure can be adequately conveyed by its submodels, which will be aggregated by the server in the next round and shared with users of smaller model capacities as inductive bias.

6.3.2 Effective Optimization via Progressive Learning

Optimizing Equation 6.1 might be prohibitive at first sight, as multiple submodels are bundled within the same network structure, while updating $\boldsymbol{\theta}_{\times p_i}$ may interfere with its nested submodels $\boldsymbol{\theta}_{\times p_j}$ when $p_j < p_i$. Towards effective optimization, we propose an approach that learns a self-distilled model that *incrementally* builds up its feature representation by involving more model *columns*.

Specifically, we first sample a batch of *ordered* pruning ratios $\hat{P} = [p_i | p_i \in \mathcal{P}, p_i < p_{i+1}\ \forall i < S, p_S = 1]_{i=1}^{S}$, then adaptively optimize each sampled submodel towards its objective function. Once a smaller submodel is updated (*e.g.* $\boldsymbol{\theta}_{\times p_1}$), we fix its parameters and update parameters in the subsequent model *columns* (*e.g.* $\boldsymbol{\theta}_{\times p_2} \backslash \boldsymbol{\theta}_{\times p_1}$). This learning scheme leverages the idea of *coordinate descent* [184], which works by successively optimizing one coordinate while fixing the others. As illustrated in Figure 6.2, predictive knowledge is learned *progressively* by adding more *lateral* connections. More concretely, we update a sampled $\boldsymbol{\theta}_{\times p_i}$ as the following:

$$\boldsymbol{\theta}_{\times p_i} \leftarrow \boldsymbol{\theta}_{\times p_i} - \eta \nabla_{\{\boldsymbol{\theta}_{\times p_i} \backslash \boldsymbol{\theta}_{\times p_{i-1}}\}} J(x; \boldsymbol{\theta}_{\times p_i}), \tag{6.2}$$

where $J(x; \boldsymbol{\theta}_{\times p_i})$ is the objective function for the current submodel $\boldsymbol{\theta}_{\times p_i}$; η is the learning rate, and $\nabla_{\{\boldsymbol{\theta}_{p_i} \backslash \boldsymbol{\theta}_{\times p_{i-1}}\}}$ denotes the gradients *w.r.t.* parameters that are included in $\boldsymbol{\theta}_{\times p_i}$ but not in $\boldsymbol{\theta}_{\times p_{i-1}}$. In particular, we tailor the objective for each submodel $\boldsymbol{\theta}_{\times p_i}$ as the following:

$$J(x; \boldsymbol{\theta}_{\times p_i}) = \mathcal{L}(f(x; \boldsymbol{\theta}_{\times p_i}), y) + \alpha_i \mathbb{D}_{\mathrm{KL}}[f(x; \bar{\boldsymbol{\theta}}) \| f(x; \boldsymbol{\theta}_{\times p_i})], \tag{6.3}$$

where $\bar{\boldsymbol{\theta}}$ is a constant cache of the largest network learned from the last iteration, which serves as the teacher; $\alpha_i := \mathbb{I}[p_i < 1]$ indicates the necessity of knowledge distillation, which renders 0 if the tail of the model columns is sampled (*i.e.* $p_i = 1$), and 1 otherwise. Once all sampled submodels have been visited, we perform an one-time back-propagation to update the teacher. We summarize this model learning approach in Algorithm 6.1.

Besides being readily prunable, the merits of progressive learning are multifold. First, it accelerates training by adaptively reaching a good *initialization*, which resembles *meta-*

learning techniques [46]. Second, it alleviates the overfitting issue especially when the teacher model structure is surplus given insufficient training data, which we verify in Section 6.5.4. Furthermore, it also alleviates the permutation-invariant issue in deep neural networks [178] by inducing nested and ordered domain representations captured in submodels.

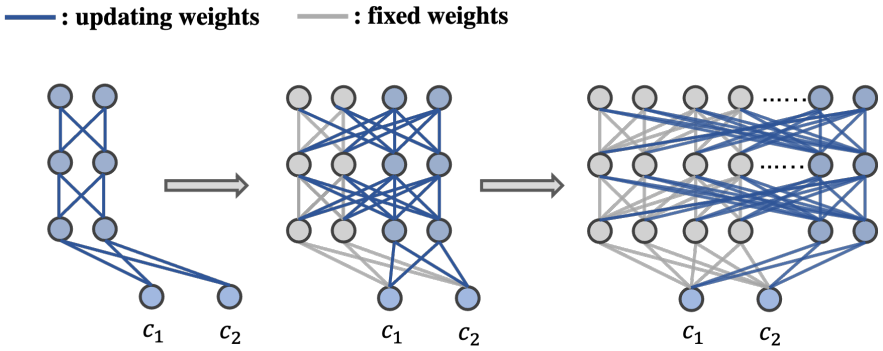


Figure 6.2: A self-distilled network is learned via progressively updating *columns* of parameters.

Algorithm 6.1: Progressive Self-Distillation

Inputs: Training dataset $D \subset \mathcal{X} \times \mathcal{Y}$; model with parameter set $\boldsymbol{\theta} \in \Theta$; pruning ratios $\mathcal{P}$, learning rate η, loss function $\mathcal{L}$, constant $S \leq |\mathcal{P}|$.
repeat
  Sample batch of $x, y \sim D$.
  Sample *ordered* ratios $\hat{P} = [p_i | p_i \in \mathcal{P}, p_i < p_{i+1}\ \forall i < S, p_S = 1]_{i=1}^{S}$
  $\bar{\boldsymbol{\theta}} \leftarrow$ stop_gradient($\boldsymbol{\theta}$), $\boldsymbol{\theta}_{\times 0} = \emptyset$.
  for $p_i \sim \hat{P}$ **do**
    $g_i \leftarrow \nabla_{\{\boldsymbol{\theta}_{\times p_i} \setminus \boldsymbol{\theta}_{\times p_{i-1}}\}} J(x; \boldsymbol{\theta}_{\times p_i})$ ▷(Equation 6.3)
    $\boldsymbol{\theta}_{\times p_i} \leftarrow \boldsymbol{\theta}_{\times p_i} - \eta * g_i$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta * \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(x; \boldsymbol{\theta}), y)$.
until training stop
Return $\boldsymbol{\theta}$

Case Study on the Effects of Progressive Learning: We illustrate with a preliminary study that, progressively learning a model enjoys the benefits of (1) finding a good initialization for the learning domain and (2) alleviating knowledge forgetting.

In this prototype, we divide a convolution model evenly into two columns and use the first half $\boldsymbol{\theta}_{\times 0.5}$ to learn on a small subset of the MNIST image data. Next, we learn on the SYNTHETIC images with a same data size, by updating $\boldsymbol{\theta}_{\times 1.0} \setminus \boldsymbol{\theta}_{\times 0.5}$ while keeping parame-

ters in $\boldsymbol{\theta}_{\times 0.5}$ fixed. This approach is compared with another variant (*overwriting*), which learns MNIST using $\boldsymbol{\theta}_{\times 0.5}$ then learns SYNTHETIC using the entire model $\boldsymbol{\theta}_{\times 1.0}$. As shown in Table 6.1, where results are averaged over 6 random seeds, the progressively learned $\times 1.0$ model outperforms its counterpart on both domains. Overwriting $\boldsymbol{\theta}_{\times 0.5}$, on the other hand, may disrupt such initialization brought by learning on the MNIST domain. This also leads to non-negligible forgetting of previously learned representations.

Accuracy (%) on MNIST and SYNTHETIC.		
Domain	Progressive learning	Overwriting
MNIST	**77.95±7.93**	38.17±28.19
SYNTHETIC	**69.48±1.93**	42.30±32.30

Table 6.1: Progressive *vs.* overwriting learning.

6.3.3 Proposed Federated Algorithm: *FedResCuE*

Before introducing our FL paradigm, we refine our algorithm with two more techniques to further tackle system heterogeneity and connection instability.

Heterogeneous Model Aggregation: In our FL system, edge users will initially agree on the maximal network architecture and the legitimate pruning ratios $\mathcal{P}$. Next, edge users can choose their maximal local model size with a capacity ratio $p_k \in \mathcal{P}$. During *downloading* (*uploading*) phases, user k with ratio p_k will receive (send) at most $\times p_k$ of model parameters, depending on the network connection. During the *aggregation* phase, active users will only contribute to global parameters that are within their uploading ratios. Accordingly, in the next learning round $t+1$, $\forall\ p_i \in \mathcal{P}$, the global model parameters are derived as follows:

$$\boldsymbol{w}^{t+1}_{\times p_i} \backslash \boldsymbol{w}^{t+1}_{\times p_{i-1}} = \frac{1}{|\mathcal{A}^t_{p_i}|} \sum_{k \in \mathcal{A}^t_{p_i}} \boldsymbol{\theta}^{k,t}_{\times p_i} \backslash \boldsymbol{\theta}^{k,t}_{\times p_{i-1}}, \tag{6.4}$$

where $\mathcal{A}^t_{p_i} = \{k | k \in \mathcal{A}^t,\ \mathbb{I}[\mathcal{R}(\boldsymbol{\theta}^{k,t}) > p_i]\}$; $\mathcal{A}^t$ denotes the active users from the last round, and $\mathcal{R}(\boldsymbol{\theta}^{k,t})$ is the uploaded network size of user k.

Local Model Padding: Due to unstable network connections, the local device is prone to receiving only partial of the global model during the *downloading* phase. To compensate for the missing global parameters, we leverage the local parameters cached from the last round to *pad* the initialized model to avoid catastrophic information loss. More concretely, at learning round t, we initialize the local model for the k-th user as follows:

$$\boldsymbol{\theta}^{k,t} \leftarrow \boldsymbol{w}^{t}_{\times p_k^{d,t}} \cup \{\boldsymbol{\theta}^{k,t-1}_{\times p_k} \backslash \boldsymbol{\theta}^{k,t-1}_{\times p_k^{d,t}}\}, \tag{6.5}$$

where $\boldsymbol{w}^{t}_{\times p_k^{d,t}}$ denotes the global parameters downloaded by user, and $p_k^{d,t}$ is the downloading ratio, which is possibly smaller than p_k.

Built upon the above techniques, we now present the proposed *FedResCuE* in Algorithm 6.2. In our algorithm, the workload introduced by progressive learning is lightweight, since the column-wise gradient update has omitted the need for repeated calculation for small submodels, as opposed to some prior arts [196]. Moreover, as elaborated in Section 6.5.6.2, *FedResCuE* can obtain superior performance with a small sampling frequency (S). Our approach is also communication efficient, which not only provides flexibility for users to select affordable model architectures but also requires much fewer communication rounds than prior work to reach the predefined performance, as verified in Section 6.5.5.

Algorithm 6.2: Resilient and Communication-Efficient Federated Learning

Inputs: Tasks $\{D_k\}_{k=1}^{K}$; global model $\boldsymbol{w}$ with parameter space Θ; legit pruning ratios $\mathcal{P}$, user capacity ratios $\{p_k\}_{k=1}^{K}$, models initialized as $\{\boldsymbol{\theta}^{k,0} := \boldsymbol{w}_{\times p_k}\}_{k=1}^{K}$. learning rate η, loss function $\mathcal{L}$, sampling frequency S, epochs T.
for $t \in [T]$ **do**
    Aggregate global parameters $\boldsymbol{w}^t$ via Equation 6.4.
    Broadcast $\boldsymbol{w}^t$ to active users $\mathcal{A}^t$.
    for user $k \in \mathcal{A}^t$ in parallel **do**
        Download $\boldsymbol{w}^t$ with downloading ratio $p_k^{d,t} \leq p_k$ depending on the connection quality.
        $\boldsymbol{\theta}^{k,t} = \boldsymbol{w}^{t}_{\times p_k^{d,t}} \cup \{\boldsymbol{\theta}^{k,t-1}_{\times p_k} \backslash \boldsymbol{\theta}^{k,t-1}_{\times p_k^{d,t}}\}$ ($\triangleright$ model initialization via Equation 6.5)
        $\boldsymbol{\theta}^{k,t} \leftarrow$ Algorithm 6.1 $(D_k, \boldsymbol{\theta}^{k,t}, \mathcal{P}, \eta, \mathcal{L}, S)$
        Upload $\boldsymbol{\theta}^{k,t}$, with uploading ratio $p_k^{u,t} \leq p_k$ depending on the connection quality.

6.4 Related Work

Systematic Heterogeneity in FL is a rising challenge induced by the emergence of FL applications to wireless communications and IoT, where participating devices have varying capacities in computation and transmission. Some work enables heterogeneous model architectures by sharing model predictions on a public dataset instead of sharing model parameters [166, 75], at the cost of non-negligible performance degradation. Some approaches allow users to share partial network layers, leaving potential opportunities for adopting different architectures for unshared layers [205, 8]. In general, yet enough efforts have been made to effectively tackle FL with heterogeneous model architectures, except for a few pioneers such as ***FedHetero*** [39] and ***FjORD*** [69], which are extensively analyzed in Section 6.5.

Network Pruning has long been studied in non-FL scenarios, which aims to prune a lightweight model from a larger one with the maximal knowledge reserved. Prior approaches usually require *fine-tuning* using label supervisions [62, 95, 111, 36], Later there emerge *zero-short* pruning [196, 28]. Approaches include *structured* pruning that reduces model channels [190, 196, 195, 28]. Orthogonal approaches include *early exit* [199, 200], which learns *horizontally* pruned networks with reduced number of neural layers. Other pruning strategies follow the lottery ticket hypothesis [48, 140, 108]. Most prior work derives one submodel after pruning, whereas the ***slimmable*** learning [196] and [28] makes arbitrarily submodels prunable. The idea of prunable models has been applied to FL to tackle system heterogeneity by *FjORD* [69]. To the best of our knowledge, we are the first to apply *progressive learning* to address both system heterogeneity and connection instability in FL.

Resilient and Communication Efficient FL addresses FL under restricted or unstable connection bandwidths [143, 57, 69]. *FedProx* [98] tackles *stragglers* via a regularized objective. Other scheduling-based approaches assume that the server has control over the active users [143, 32, 128]. Not much work has mentioned the faulty connections issues. Some chooses to naively drop the faulty connected edge devices [32, 143]. [57] and [188] suggest to use stale model parameters for the disconnected users. Meanwhile, there are complementary

efforts that compress transmitted model parameters by quantization [4, 5] or sketching [73]. Some approaches focus on asynchronous communication [33, 187]. Our algorithm can be potentially combined with related work to further improve resiliency and communication efficiency in FL.

6.5 Evaluation

In this section, we conduct extensive experiments to answer the following key questions, leaving more experimental details to the supplementary:

1. Is *FedResCuE* resilient to system heterogeneity and unstable network connections?
2. Is *FedResCuE* communication-efficient to reach satisfactory performance with fewer synchronization rounds, compared with the state-of-the-art?
3. Which components of *FedResCuE* have contributed to its *resiliency* and *communication efficiency*?

Results: Experiments below show that *FedResCuE* notably outperforms related work in communication efficiency and asymptotic performance. Its superiority is consistent across different FL settings, and become more prominent under *insufficient* training data, *heterogeneous* model architectures, and *unstable* network connections.

6.5.1 Experiment Setup

Dataset: We use CelebA [89] to simulate edge users with *i.i.d.* data distributions. We also apply DigitsFive [134] to simulate users with statistical heterogeneity, which is a multi-domain benchmark with five image datasets: MNIST [93], SVHN [124], USPS [72], Synthetic, and MNIST-M [53].

Models: We build a ResNet neural network [65] for learning the CelebA domain, and build a model consisting of 3 Conv2d layers followed by 3 Linear layers for learning

DigitsFive domains. To enable effective model aggregation under system heterogeneity, we perform careful treatments on the BatchNorm layers.

Compared Approaches: In addition to ***FedAvg*** [116], we compare ***FedResCuE*** against the following approaches that tackle system heterogeneity: i) ***FedHetero*** [39] extended *FedAvg* to allow edge devices with different model sizes; ii) ***FjORD*** [69] learns prunable local models without *progressive learning*; iii) ***FedSlim*** is a proposed baseline in this paper, in which models are locally updated by following the **slimmable** training [196] and globally aggregated as *FedAvg*.

Training: Active users will sync with the server after a complete *epoch* of local training. For faulty connection settings (Section 6.5.3), the *local padding* strategy is applied to *all* evaluated algorithms for fair comparisons. For the CelebA domain, training data is *i.i.d.* sampled and assigned to 20 total users. For the DigitsFive domains, we assign each domain data to 2 unique users. For both types of experiments, 5 active users are randomly selected per communication round. We also evaluate the algorithmic performance given different sizes of training data in Section 6.5.4.

Evaluation: Unless otherwise specified, the performance is reported using the global model on *all available testing data*, which is evaluated every 2 communication rounds. Results are averaged over 3 random seeds. Asymptotic performance is reported after 300 rounds for CelebA, and 100 rounds for DigitsFive.

Global Model Accuracy (%) Evaluated on CelebA, Stable Network Connection.							
Training Data	User Capacity	Evaluated Model	*FedAvg*	*FedHetero*	*FjORD*	*FedSlim*	*FedResCuE*
100%	$\forall k\ p_k = 1$ (uniform)	$\boldsymbol{w}_{\times 1}$	81.06±0.63	-	80.57±0.91	81.14±0.76	**81.39±0.20**
		$\boldsymbol{w}_{\times 0.25}$	18.57±0.64	-	69.94±0.65	70.47±0.61	**71.19±0.19**
	$\forall k\ p_k \sim P_C$ (cluster)	$\boldsymbol{w}_{\times 1}$	-	76.80±0.53	75.71±0.47	77.49±0.40	**78.22±0.41**
		$\boldsymbol{w}_{\times 0.25}$	-	68.56±0.51	70.98±0.75	73.22±0.34	**73.25±0.47**
20%	$\forall k\ p_k = 1$ (uniform)	$\boldsymbol{w}_{\times 1}$	68.03±0.50	-	67.89±1.47	67.96±0.72	**71.27±0.27**
		$\boldsymbol{w}_{\times 0.25}$	16.47±2.24	-	**61.38±1.69**	59.56±1.39	61.12±1.35
	$\forall k\ p_k \sim P_C$ (cluster)	$\boldsymbol{w}_{\times 1}$	-	59.38±0.41	62.43±1.65	59.53±0.86	**64.53±1.06**
		$\boldsymbol{w}_{\times 0.25}$	-	55.41±0.39	61.86±1.21	58.31±0.23	**61.98±0.85**

Table 6.2: We report best performance from different S for applicable approaches (See Section 6.5.6.2).

6.5.2 Performance Under System Heterogeneity

We apply CelebA data to explore two system settings: 1) the *uniform* setting, where all edge devices maintain the same network architecture; and 2) the *cluster* setting, where user model capacities are randomly sampled from a set $P_C \subset \mathcal{P}$ to represent system heterogeneity. **Results:** As shown in Table 6.2 and Table 6.3, *FedResCuE* consistently exceeds other approaches in asymptotic performance, especially under a heterogeneous (*cluster*) system. In particular, the advantage of *FedResCuE* on the $\times 0.25$ global model demonstrates its benefits to small-capacity users compared to related work, in that *FedResCuE* helps predictive knowledge be distilled from larger models into their nested sub-models, which will be eventually shared by users with smaller model sizes. In the meantime, *FedResCuE* is also more effective than *FjORD* and *FedSlim*, which is largely ascribed to the benefit of its progressive learning, as opposed to a batch-gradient update scheme in prior art.

Moreover, when system heterogeneity is bundled with connection instability (Table 6.3), enabling system heterogeneity in FL naturally introduces resiliency to connection instability, which can be revealed by the performance gain of *FedHetero* over *FedAvg* in Table 6.3. In fact, *FedHetero* can be treated as *macro*-level prunable training, although the resiliency brought by diversified model architectures is less effective than learning self-distilled networks.

Global Model Accuracy (%) Evaluated on CelebA Under Connection Loss.						
User Capacity	Evaluated Model	*FedAvg*	*FedHetero*	*FjORD*	*FedSlim*	***FedResCuE***
uniform	$\boldsymbol{w}_{\times 1}$	50.36±2.17	-	61.79±1.62	57.31±1.27	**70.02±0.40**
	$\boldsymbol{w}_{\times 0.25}$	12.58±0.51	-	60.20±1.67	55.33±0.89	**67.40±0.84**
cluster	$\boldsymbol{w}_{\times 1}$	-	60.92±1.33	64.52±0.60	62.35±1.76	**69.78±0.74**
	$\boldsymbol{w}_{\times 0.25}$	-	59.70±0.64	64.11±0.41	61.77±1.62	**68.83±1.00**

Table 6.3: Performance under faulty connections, given 100% of training data and $0.1 \leq er \leq 0.2$.

Accuracy(%) on DigitsFive, Given 5% Training Data.					
Domain	SVHN	Syn	USPS	MNIST	MNIST-M
Local	45.88	62.04	89.95	85.84	61.99
FedAvg	**69.54**	80.17	94.38	93.61	75.22
FedSlim	66.77	78.95	94.00	93.80	73.95
FjORD	49.72	57.12	68.60	68.01	56.29
FedResCuE	69.32	**80.57**	**95.17**	**95.05**	**76.89**

Table 6.4: *FedResCuE* is the most robust algorithm given heterogeneous data and domain-dependent connection error.

6.5.3 Performance Under Unstable Connections

To simulate the unstable connection scenario, a connection error rate $\boldsymbol{er}$ is generated dynamically to denote the probability that the current column transmission is interrupted. When transmitting one network, we traverse all columns until one column is interrupted based on probability er, or when all columns have been successfully transmitted. Note that the connection loss occurs *bidirectionally*. Hence an edge device may receive or upload models with a smaller size than its assigned architecture. In practice, we set the size of a transmission *column* to be $0.125\times$ of the global network. We apply both CelebA and DigitsFive to explore unstable connection scenarios. For experiments on the CelebA domain, er is dynamically sampled, with $er \sim [0.1, 0.2]$. For experiments on DigitsFive domains, er is set to be a constant depending on the specific domain.

Results: Under faulty network connections, *FedResCuE* consistently outperforms other approaches under both *i.i.d.* and heterogeneous data distributions. Given the CelebA domain (Table 6.3), *FedResCuE* achieves higher accuracy than *FedSlim* and *FjORD* with a significant margin, which we ascribe to both its progressive learning procedure and the proposed optimization objective. In fact, given unstable connections, a smaller network architecture turns out to be more reliable, whose transmission is less likely to be interrupted. *FedResCuE* can facilitate FL in this scenario, as it follows a progressive scheme to gradually learn the larger network, which captures the complementary representation built upon its nested smaller submodels. On the other hand, both *FedResCuE* and *FjORD* are more competent than *FedSlim*, which indicates that solely performing distillation from the teacher ($\times 1.0$

model) to student (submodel), as *FedSlim* performs, may be insufficient to deliver reliable submodels, especially given a model with information staleness caused by transmission loss. Contrarily, the optimization of *FedResCuE* (Equation 6.1), which encourages both knowledge distillation and submodel-learning with label supervision, is a more robust strategy.

As shown in Table 6.4, under domain-dependent connection errors, *FedResCuE* also shows consistent robustness against heterogeneous statistical distributions. Note that a small training dataset from DigitsFive is applied to ensure the necessity of FL. Hence *Local* learning without sharing parameters yields worse performance than FL. The performance gain of *FedResCuE* resides in both the small ($\times 0.25$) and the large ($\times 1.0$) model. Contrarily, *FjORD* and *FedSlim* may underperform *FedAvg* when evaluated using the $\times 1.0$ model, indicating their potential drawback given insufficient data, which we investigate more in Section 6.5.4.

6.5.4 Performance Given Insufficient Training Data

To analyze the impacts of data sufficiency, we assign 100% and 20% of the CelebA to users for training, respectively, assuming a stable network connection ($er = 0$).

Results: As shown in Table 6.2, when training data is sufficient with *i.i.d.* distributions, all algorithms perform comparably well. However, given only 20% of the training data, both *FjORD* and *FedSlim* slightly underperform *FedAvg* when evaluated using the $\times 1.0$ model, while *FedResCuE* remarkably outperforms all others. In fact, the progressive parameter update in *FedResCuE* can make a subnetwork a good *initialization* for the encompassing larger submodel, which is analogous to meta-learning that delivers an effective model with fewer shots of training. This is especially beneficial in the lack of training data. Contrarily, *FjORD* and *FedSlim*, which adopt batch-gradient updates for learning prunable models, may struggle with the interference of noisy gradients, which can be further amplified in a potentially overfit model.

6.5.5 Evaluation of Communication Efficiency

We analyze the communication efficiency via the number of FL synchronization rounds for the global model to reach a reasonable performance. As shown in Table 6.5, under system heterogeneity (*i.e.* the *cluster* setting), *FedResCuE* constantly learns faster to obtain a predefined accuracy, requiring fewer communication rounds than all its peers. The communication efficiency of *FedResCuE* also resides in its flexibility in user model architectures, in that devices that choose a small model architecture can be further benefited by transmitting fewer parameters per communication round. Although other baselines *e.g. FedHetero* also enable system heterogeneity, they require non-negligible more communication rounds to perform comparably to *FedResCuE*. Accompanying performance curves of the $\times 1.0$ model are visualized in Figure 6.3.

Communication Efficiency on CelebA dataset.					
Acc	Model Size	*FedHetero*	*FjORD*	*FedSlim*	*FedResCuE*
100 % training data, $0.1 \leq er \leq 0.2$.					
60%	$\boldsymbol{w}_{\times 0.5}$	256.7	218.0	253.3	**124.7**
20 % training data, $er = 0$					
55%	$\boldsymbol{w}_{\times 0.5}$	180.7	156.0	192.0	**96.0**

Table 6.5: *FedResCuE* requires notably fewer communication rounds to reach the predefined accuracy (Acc).

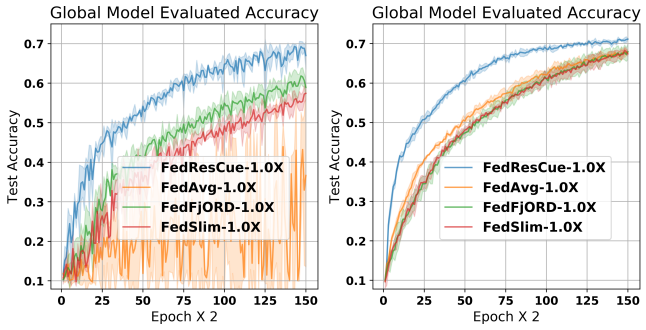


Figure 6.3: Evaluation curves for the $\times 1.0$ model, with $0.1 \leq er \leq 0.2$ (left) and 20% training data (right).

6.5.6 Sensitivity Analysis

6.5.6.1 Effects of Knowledge Distillation

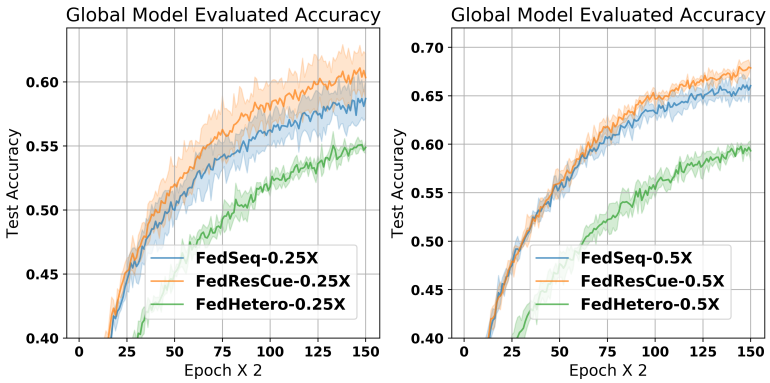
To analyze the role of knowledge distillation in our model learning, we design a variant of our approach called *FedSeq* to compare against *FedResCuE*. Particularly, the optimization objective of *FedSeq* does not require minimizing the KL-divergence between the teacher $\bar{\boldsymbol{\theta}}$ and a student $\boldsymbol{\theta}_{\times p_i}$, *i.e.* it always sets the term α_i to 0 in Equation 6.3.

Results: Knowledge distillation is especially beneficial to smaller submodels, whereas the gap between *FedSeq* and *FedResCuE* gradually diminishes when evaluating using larger submodels. As illustrated in Figure 6.4, where 20% of the CelebA training data is given, both approaches are learning comparably well in initial training stages, while *FedResCuE* converges to higher asymptotic performance. The learning curves demonstrate that it is beneficial to distill representation knowledge from a large, complete model to smaller submodels, in that the larger model has more channels to capture refined domain knowledge.

6.5.6.2 Impacts of Submodel Sampling

Sampling frequency, denoted as $S = |\hat{P}|$ in Algorithm 6.1, is the number of submodels sampled per batch update. A key question regarding *FedResCuE* is: *how does S affect the learning performance?* This question is equally intriguing to *FedSlim* and *FjORD*, both of which require submodel sampling. To answer this question, we traverse different choices of S, while the sampling granularity is set to be $\times 0.05$ of the largest model width.

Results: As shown in Figure 6.5, *FedResCuE* is constantly the most robust under different S. In the meantime, a moderate number of ratio sampling (*e.g.* $S = 4$) benefit most evaluated algorithms, while oversampling with a large S causes non-negligible performance degradation on *FedSlim* and *FjORD*. Their over-sensitivity to S can be induced by their batch-gradient update scheme, which may cause the gradients *w.r.t.* larger submodels to interfere with those of smaller ones when aggregating all gradients in one batch, hence undermining model performance. On the contrary, *FedResCuE* alleviates such issues by using a progressive



(a) $\times_{0.25}$ model. (b) $\times_{0.5}$ model.

Figure 6.4: KD in *FedResCuE* benefits smaller submodels.

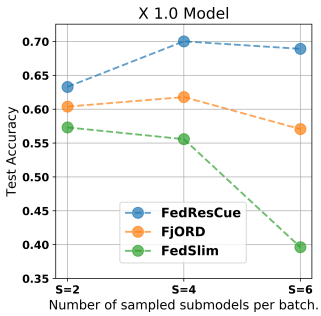


Figure 6.5: Impacts of sampling frequency S.

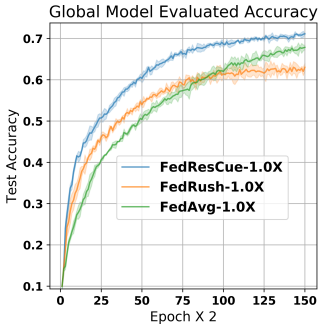


Figure 6.6: Effects of progressive learning.

learning scheme that decouples such mutual impacts.

6.5.6.3 Effects of Progressive Learning

To evaluate the efficacy of the progressive learning scheme, we compare *FedResCuE* against an intuitive alternative named *FedRush*, which directly updates the sampled submodel without freezing the preceding parameters.

Results: As shown in Figure 6.6, a rush gradient update as in *FedRush* leads to notably undermined performance. Particularly, when updating the $\boldsymbol{\theta}_{\times p_{i+1}}$ model, *FedRush* overwrites the parameters in $\boldsymbol{\theta}_{\times p_i}$, which could otherwise serve as a good initialization to support the subsequent submodels. Contrarily, *FedResCuE* learns more effectively by avoiding the potential information forgetting.

6.6 Summary

In this chapter, we propose a self-distillation approach which enables resilient knowledge sharing in heterogeneous federated learning setting. Our proposed, dubbed as *FedResCuE*, addresses both system heterogeneity and unstable network connections, by learning *self-distilled* networks in a *progressive* manner, which proves to be communication-efficient with higher performance in the proposed FL settings compared with the state-of-the-art.

CHAPTER 7

OVERVIEW AND OPEN QUESTIONS

In this chapter, we first present a systematic comparison of the proposed transfer learning approaches, discuss their contributions as well as potential limitations. We also provide a few thoughts on the limitations of our transfer learning and list a few research directions that are worth more effort in the near future.

We compare the recipes of each of our TL works in Table 7.1. Although these approaches are proposed to address heterogeneous machine learning problems, they share the same core idea of TL from accessible, usually *limited* or *unreliable* knowledge source to assist target domain learning in a *sample-efficient* manner.

In spite of their promising results, one limitation of the proposed approaches is that they are not designed to assist learning for ***unseen domains***. In fact, for the RL approaches *OPOLO* and *SAIL* , we assume that the source and target Markov decision process share the same underlying components, such as reward logics and state transition probabilities. This problem setting is analogous to the one where the child learns to walk by imitating adults in a world that follows the same physical rules. For *FedGen* which tackles a Federated Learning setting, each client task is served as the source and the target domain simultaneously. The data distributions of different clients are heterogeneous but remain stable through the learning process.

Another limitation of our work is that our approach might be vulnerable to ***negative transfer***. By now we assume that the knowledge source is beneficial and relevant to the target domain learning. Effective schemes to detect irrelevant or even adversarial source domains remain to be proposed to improve the robustness of our transfer learning approaches.

Algorithm:	*OPOLO*	*SAIL*	*FedGen*	*FedResCuE*
Problem setting:	RL without external rewards.	RL with sparse and delayed rewards.	Federated Supervised Learning without extra training data on the server.	Federated Supervised Learning, with system heterogeneity and unstable network connections.
Difference between source and target domains	$\lvert\boldsymbol{\mathcal{M}}_s\rvert = 1, \mathcal{M}_s = \mathcal{M}_t$.	$\lvert\boldsymbol{\mathcal{M}}_s\rvert = 1, \mathcal{M}_s = \mathcal{M}_t$.	Each FL client $\mathcal{T}_k \in \{\mathcal{T}_s\}_{i=1}^K$ is a source and target domain at the same time. $\mathcal{T}_i \neq \mathcal{T}_j \ \forall i, j \in [K], i \neq j$	Similar to *FedGen* , with $\mathcal{T}_i$ not necessarily equal $\mathcal{T}_j \ \forall i, j \in [K], i \neq j$.
Carrier of transferred knowledge	Examples of ***expert*** visited ***states*** without expert actions: $\mathcal{R}_T = (s_0, s_1, s_2, \cdots)$.	Examples of ***sub-optimal*** demonstrations of both ***states*** and ***actions***: $\mathcal{R}_T = (s_0, a_0, s_1, a_1, s_2, a_2, \cdots)$.	a conditional generator $G : \mathcal{Y} \to \mathcal{Z}$ learned from the network parameters of client models.	Parameter of submodel $\boldsymbol{\theta}_p$ learned by self-distillation, with $p \leq 1$.
Challenges of transferring such knowledge	***No action*** guidance is available; Inefficient sampling due to on-policy learning.	Teacher demonstrations are ***suboptimal***, hence traditional appraoches lead to suboptimal performance.	Local user data distribution is ***heterogeneous*** and ***unavailable*** to the server.	1) FL clients learn models with different *architectures*. 2) *Unstable* Network connection.
Solutions	Optimizing towards an ***off-policy*** objective that omits the need of knowing expert actions.	An iterative process of ***exploitation and exploration***: exploiting teacher dmonstrations to reach reasonable performance; exploring for better self-generated demonstrations to repalce the teacher.	Learning and transmitting a generator purely out of the client model parameters to approximate $p(z\lvert y)$, *i.e.* the global latent feature distribution.	1) Model channels are *vertically* decomposed and sequentailly transmitted. 2) Learn self-distilled models that are knowledge preserving, and robust to connection drop.

Table 7.1: An overview comparison of the proposed TL approaches.

Table 7.1 (cont'd)

Distilled knowledge	Near-optimal state-transisin distributions of the teacher domain: $\mu_E(s, s')$	Sub-optimal state-action distributions of the teacher domain $\mu_T(s, a)$.	Latent distribution $p(z\|y)$ of the global data.	Feature representations reserved in the $1.0\times$ model.
Transfer learning results	Recover expert-level performance via efficient off-policy learning.	Supass teacher demonstrations to reach near-optimal performance with high sample efficiency.	Ensemble knowledge from heterogeneous user data distribution to achieve high global performance.	Enabled system heterogeneity in FL system. Learned self-distilled models that are readily prunable and robust to connection loss.

APPENDICES

APPENDIX A

APPENDIX FOR *OPOLO*

For all the following derivations, we use $\mathbb{D}_{\mathrm{KL}}[P(X)||Q(X)]$ to denote the KL-divergence between two distributions P and Q:

$$\mathbb{D}_{\mathrm{KL}}[P(X)||Q(X)] = \mathbb{E}_{x\sim p(x)} \log \frac{p(x)}{q(x)} = \int_X p(x) \log \frac{p(x)}{q(x)} dx.$$

Accordingly, when $P(X|Z)$ and $Q(X|Z)$ are *conditional* distributions, $\mathbb{D}_{\mathrm{KL}}[P||Q]$ denotes their *conditional* KL-divergence:

$$\mathbb{D}_{\mathrm{KL}}[P(X|Z)||Q(X|Z)] = \int_{Z\times X} p(z)p(x|z) \log \frac{p(x|z)}{q(x|z)} dxdz.$$

For simplicity, we will equivalently use $\mathbb{E}_{x\sim p(x)}[\cdot]$ and $\mathbb{E}_{p(x)}[\cdot]$ to denote certain expectation in which x is sampled from the distribution $P(X)$.

A.0.1 Derivation of the Surrogate Objective

We first refer Lemma 1 from [189] for a complete presentation:

Lemma 1.

$$\mathbb{D}_{\mathrm{KL}}[\mu^\pi(s,a,s')||\mu^E(s,a,s')] = \mathbb{D}_{\mathrm{KL}}[\mu^\pi(s,a)||\mu^E(s,a))].$$

Proof.

$$\begin{aligned}
\mathbb{D}_{\mathrm{KL}}[\mu^\pi(s,a,s')||\mu^E(s,a,s')] &= \int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^\pi(s,a,s') \log \frac{\mu^\pi(s,a)\cdot P(s'|s,a)}{\mu^E(s,a)\cdot P(s'|s,a)} ds'dads \\
&= \int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^\pi(s,a,s') \log \frac{\mu^\pi(s,a)}{\mu^E(s,a)} ds'dads \\
&= \int_{\mathcal{S}\times\mathcal{A}} \mu^\pi(s,a) \log \frac{\mu^\pi(s,a)}{\mu^E(s,a)} dads \\
&= \mathbb{D}_{\mathrm{KL}}[\mu^\pi(s,a)||\mu^E(s,a)].
\end{aligned}$$

□

Lemma 2.

$$\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')] \leq \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,a)||\mu^{E}(s,a)].$$

Proof. As defined in Table 4.1, $\mu^{\pi}(a|s,s')$ is the inverse-action transition probability induced by policy π:

$$\mu^{\pi}(a|s,s') = \frac{\mu^{\pi}(s,a,s')}{\mu^{\pi}(s,s')} = \frac{\cancel{\mu^{\pi}(s)}\pi(a|s)P(s'|s,a)}{\int_{\mathcal{A}}\cancel{\mu^{\pi}(s)}\pi(\bar{a}|s)P(s'|s,\bar{a})d\bar{a}} = \frac{\pi(a|s)P(s'|s,a)}{\int_{\mathcal{A}}\pi(\bar{a}|s)P(s'|s,\bar{a})d\bar{a}}.$$

Based on this notion, we can derive:

$$\begin{aligned}
&\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,a)||\mu^{E}(s,a)] \\
=&\underbrace{\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,a,s')||\mu^{E}(s,a,s')]}_{\text{Lemma 1}} \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}\mu^{\pi}(s,a,s')\log\frac{\mu^{\pi}(s,a,s')}{\mu^{E}(s,a,s')}ds'dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}\mu^{\pi}(s,s')\mu^{\pi}(a|s,s')\log\frac{\mu^{\pi}(s,s')\times\mu^{\pi}(a|s,s')}{\mu^{E}(s,s')\times\mu^{E}(a|s,s')}ds'dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}\mu^{\pi}(s,s')\mu^{\pi}(a|s,s')\log\frac{\mu^{\pi}(s,s')}{\mu^{E}(s,s')}ds'dads \\
&+\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}\mu^{\pi}(s,s')\mu^{\pi}(a|s,s')\log\frac{\mu^{\pi}(a|s,s')}{\mu^{E}(a|s,s')}ds'dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}}\mu^{\pi}(s,s')\log\frac{\mu^{\pi}(s,s')}{\mu^{E}(s,s')}ds'ds + \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(a|s,s')||\mu^{E}(a|s,s')] \\
=&\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')] + \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(a|s,s')||\mu^{E}(a|s,s')] \qquad \text{(A.1)} \\
\geq&\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')].
\end{aligned}$$

□

Based on Lemma2, we can derive the upper-bound of our original objective:

Theorem 3 (Surrogate Objective as the Divergence Upper-bound)**.**

$$\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')] \leq \mathbb{E}_{\mu^{\pi}(s,s')}[\log\frac{\mu^{R}(s,s')}{\mu^{E}(s,s')}] + \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,a)||\mu^{R}(s,a)].$$

Proof.

$$
\begin{aligned}
\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')] &= \int_{\mathcal{S}\times\mathcal{S}} \mu^{\pi}(s,s') \log \frac{\mu^{\pi}(s,s')}{\mu^{E}(s,s')} dsds' \\
&= \int_{\mathcal{S}\times\mathcal{S}} \mu^{\pi}(s,s') \log \Big(\frac{\mu^{R}(s,s')}{\mu^{E}(s,s')} \times \frac{\mu^{\pi}(s,s')}{\mu^{R}(s,s')} \Big) dsds' \\
&= \int_{\mathcal{S}\times\mathcal{S}} \mu^{\pi}(s,s') \log \frac{\mu^{R}(s,s')}{\mu^{E}(s,s')} dsds' \\
&+ \int_{\mathcal{S}\times\mathcal{A}} \mu^{\pi}(s,s') \log \frac{\mu^{\pi}(s,s')}{\mu^{R}(s,s')} dsds' \\
&= \mathbb{E}_{\mu^{\pi}(s,s')}[\log \frac{\mu^{R}(s,s')}{\mu^{E}(s,s')}] + \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{R}(s,s')] \\
&\leq \mathbb{E}_{\mu^{\pi}(s,s')}[\log \frac{\mu^{R}(s,s')}{\mu^{E}(s,s')}] + \underbrace{\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,a)||\mu^{R}(s,a)]}_{\text{derived from Lemma 2}}.
\end{aligned}
$$

□

A.0.2 Connections between LfO and LfD

Theorem 4.

$$\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(a|s,s')||\mu^{E}(a|s,s')] = \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,a)||\mu^{E}(s,a)] - \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')].$$

Proof. We can refer Eq (A.1) from the proof of Lemma 2:

$$\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,a)||\mu^{E}(s,a)] = \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(s,s')||\mu^{E}(s,s')] + \mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(a|s,s')||\mu^{E}(a|s,s')].$$

□

A.0.3 An Unoptimizable Gap Between LfO and LfD

Remark 3: *In a non-injective MDP, the discrepancy of* $\mathbb{D}_{\mathrm{KL}}[\mu^{\pi}(a|s,s')||\mu^{E}(a|s,s')]$ *cannot be optimized without knowing expert actions.*

Proof. We provide proof with a counter-example. Consider a non-injective MDP in a tabular case, whose transition dynamics is shown in Table A.1, with $|\mathcal{S}| = 3$, and $|\mathcal{A}| = 4$. Especially,

P	a_0	a_1	a_2	a_3
$P(s_1\|s_1,\cdot)$	0	1	0	0
$P(s_2\|s_1,\cdot)$	**1**	0	**1**	0
$P(s_3\|s_1,\cdot)$	0	0	0	1
$P(s_1\|s_2,\cdot)$	0	1	0	0
$P(s_2\|s_2,\cdot)$	0	0	1	0
$P(s_3\|s_2,\cdot)$	0	0	0	1
$P(s_1\|s_3,\cdot)$	0	1	0	0
$P(s_2\|s_3,\cdot)$	0	0	1	0
$P(s_3\|s_3,\cdot)$	0	0	0	1

Table A.1: A deterministic but non-injective MDP.

π	s_1	s_2	s_3
a_0	**0.5**	0	0
a_1	0	0	1
a_2	**0.5**	0	0
a_3	0	1	0

Table A.2: Learning Policy π.

π_E	s_1	s_2	s_3
a_0	0	0	0
a_1	0	0	1
a_2	**1**	0	0
a_3	0	1	0

Table A.3: Expert Policy π_E.

there exists two actions which lead to the same deterministic transition, *i.e.* for $s_1, s_2 \in \mathcal{S}$, $\exists\ a_0, a_2 \in \mathcal{A}$, *s.t.* $P(s_2|s_1, a_2) = P(s_2|s_1, a_0) = 1$, as illustrated in Figure A.1.

In this MDP, there is an *expert* policy π_E as listed in Table A.3. Trajectories generated by this expert are illustrated as blue lines in Figure A.1. In a LfO scenario, a learning agent only has access to sequences of states visited by the expert: $\mathcal{R}_T = \{s_1, s_2, s_3, s_1, s_2, s_3, \cdots\}$, without knowing what actions have been taken by the expert.

Based on the given observations $\mathcal{R}_T$, a policy π can only satisfy the state distribution matching with $\mathbb{D}_{\text{KL}}[\mu^\pi(s, s')||\mu^E(s, s')] = 0$, but unable to optimize $\mathbb{D}_{\text{KL}}[\mu^\pi(a|s, s)||\mu^E(a|s, s)]$, as both a_0 and a_2 lead to a deterministic transition of $s_1 \to s_2$. In lack of expert actions, the best guess for a learning policy is to equally distribute action probabilities with $\pi(a_0|s_1) = (a_2|s_1) = 0.5$. which results in $\mu^\pi(a_0|s_1, s_2) = \mu^\pi(a_2|s_1, s_2) = 0.5$, whereas $\mu^E(a_2|s_1, s_2) = 1$, $\mu^E(a_0|s_0, s_1) = 0$. Consequently, we reach at $\mathbb{D}_{\text{KL}}[\mu^\pi(a|s, s')||\mu^E(a|s, s')] > 0$. □

Remark: *In a deterministic and injective MDP, it satisfies that* $\forall\ \pi : \mathcal{S} \to \mathcal{A}$, $\mathbb{D}_{\text{KL}}[\mu^\pi(a|s, s')||\mu^E(a|s, s')] = 0$.

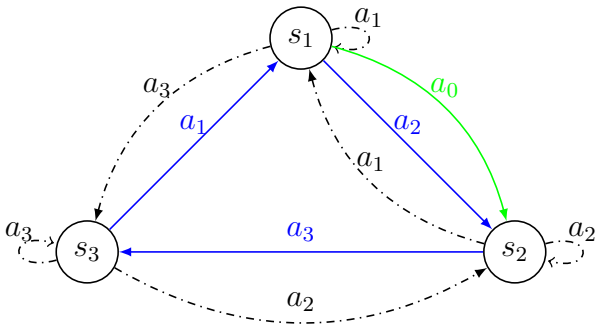


Figure A.1: Transition of an non-injective MDP.

We provide proof in a finite, **discrete** state-action space, although the conclusion is valid to extend to continuous cases.

Proof. In a deterministic and injective MDP, we can interpret the transition dynamics with a *deterministic* function g:

$\exists g : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, *s.t.* $\forall\ (s, a, s')$, $g(s, a) = s' \iff P(s'|s, a) = 1$, and $g(s, a) \neq s' \iff P(s'|s, a) = 0$.

since this MDP is also injective, given arbitrary policy π and a transition $s \to s', (s, s') \sim \mu^\pi(s, s')$, there exists one and only action a which satisfies $g(s, a) = s', P(s'|s, a) = 1$.

Accordingly, $\mu^\pi(a|s, s') = \frac{\pi(a|s)P(s'|s,a)}{\mathbb{E}_{\tilde{a}\sim\pi(\cdot|s)}[P(s'|s,a)]} = \mathbb{1}[g(s, a) = s']$ depends only on the transition dynamics, where $\mathbb{1}(x)$ is an indicator function. The same conclusion applies to $\mu^E(a|s, s')$ as well. Therefore, we reach at:

$$
\begin{aligned}
\forall\ \pi : \mathcal{S} \to \mathcal{A},\ & \mathbb{D}_{\mathrm{KL}}[\mu^\pi(a|s, s')||\mu^E(a|s, s')] \\
=& \mathbb{E}_{\mu^\pi(s,a,s')}\big[\log \frac{\mathbb{1}[g(s, a) = s']}{\mathbb{1}[g(s, a) = s']}\big] \\
=& \mathbb{E}_{\mu^\pi(s,a,s')}\big[\log \frac{1}{1}\big] = 0.
\end{aligned}
$$

□

A.0.4 Upper Bound of the KL-Divergence

Theorem 5. *For two arbitrary distributions P and Q, and an f-divergence with $f(x) = \frac{1}{2}x^2$, it satisfies that $\mathbb{D}_{\mathrm{KL}}[P||Q] \leq \mathcal{D}_f[P||Q]$.*

Proof. Given two distributions P and Q, their density ratio is denoted as $w_{p|q}$, with $w_{p|q} = \frac{p(x)}{q(x)} \geq 0$. If we consider a function $g(w) = w\log(w) - \frac{1}{2}w^2$, $g(w)$ is constantly decreasing when $w \in (0, \infty)$, as $\frac{\partial g}{\partial w} = \log w + 1 - w \leq 0 \ \forall w \geq 0$.

Since KL-Divergence is a special case of f-divergence with $f_{\mathbb{D}_{\mathrm{KL}}}(x) = x\log x$, it is sufficient to show that:

$$\begin{aligned}
\mathcal{D}_{\mathbb{D}_{\mathrm{KL}}}[P||Q] - \mathcal{D}_f[P||Q] &= \int_{\mathcal{X}} q(x)\Big(w_{p/q}\log(w_{p/q}) - \frac{1}{2}(w_{p/q})^2\Big)dx \\
&\leq \int_{\mathcal{X}} q(x) \sup_{w\in(0,+\infty)} (w\log(w) - \frac{1}{2}w^2)dx \\
&= \int_{\mathcal{X}} q(x) \lim_{w\to 0^+} (w\log(w) - \frac{1}{2}w^2)dx \\
&= 0.
\end{aligned}$$

□

A.0.5 Forward Distribution Matching

A.0.5.1 Lower-bound of the BC Objective

Theorem 6.

$$\mathbb{D}_{\mathrm{KL}}[\pi_E(a|s)||\pi(a|s)] = \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^\pi(s'|s)] + \mathbb{D}_{\mathrm{KL}}[\mu^E(a|s,s')||\mu^\pi(a|s,s')].$$

Proof. Based on the definition of $\mu^\pi(a|s,s')$ in Table 4.1:

$$\mu^\pi(a|s,s') = \frac{\pi(a|s)P(s'|s,a)}{\int_{\mathcal{A}} \pi(\bar{a}|s)P(s'|s,\bar{a})d\bar{a}} = \frac{\pi(a|s)P(s'|s,a)}{\mu^\pi(s'|s)}, \tag{A.2}$$

and similar for $\mu^E(a|s,s')$, we can derive at the following:

$$
\begin{aligned}
&\mathbb{D}_{\mathrm{KL}}[\pi_E(a|s)||\pi(a|s)] \\
=&\int_{\mathcal{S}\times\mathcal{A}} \mu^E(s)\pi_E(a|s)\log\frac{\pi_E(a|s)}{\pi(a|s)}dads \\
=&\int_{\mathcal{S}\times\mathcal{A}} \mu^E(s,a)\log\frac{\pi_E(a|s)}{\pi(a|s)}dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^E(s,a)P(s'|s,a)\log\frac{\pi_E(a|s)P(s'|s,a)}{\pi(a|s)P(s'|s,a)}ds'dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^E(s,a,s')\log\frac{\pi_E(a|s)P(s'|s,a)}{\pi(a|s)P(s'|s,a)}ds'dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^E(s,a,s')\underbrace{\log\frac{\mu^E(a|s,s')\mu^E(s'|s)}{\mu^\pi(a|s,s')\mu^\pi(s'|s)}}_{\text{Eq (A.2)}}ds'dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^E(s,a,s')\Big(\log\frac{\mu^E(a|s,s')}{\mu^\pi(a|s,s')}+\log\frac{\mu^E(s'|s)}{\mu^\pi(s'|s)}\Big)ds'dads \\
=&\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^E(s,a,s')\log\frac{\mu^E(a|s,s')}{\mu^\pi(a|s,s')}ds'dads+\int_{\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mu^E(s,a,s')\log\frac{\mu^E(s'|s)}{\mu^\pi(s'|s)}ds'dads \\
=&\mathbb{D}_{\mathrm{KL}}[\mu^E(a|s,s')||\mu^\pi(a|s,s')]+\mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^\pi(s'|s)].
\end{aligned}
$$

□

A.0.5.2 Policy Regularization as A Forward Distribution Matching

Without loss of generality, in this section we provide proof based on a finite, **discrete** state-action space.

Assumption 2 (Deterministic MDP)**.** *$\exists g : \mathcal{S}\times\mathcal{A}\to\mathcal{S}$ a deterministic function, s.t. $\forall\ (s,a,s')$, $g(s,a)\neq s' \iff P(s'|s,a)=0$, and $g(s,a)=s' \iff P(s'|s,a)=1$.*

Based on Assumption 2, we have the following:

Corollary 2. *In a deterministic MDP, $\forall\ \pi:\mathcal{S}\to\mathcal{A},\ \mu^\pi(a|s,s')>0 \Longrightarrow P(a|s,s')=1$.*

Proof. $\mu^\pi(a|s,s')\propto\pi(a|s)P(s'|s,a)>0 \Longrightarrow P(s'|s,a)>0$. Based on Assumption 2, it holds that $g(s,a)=s'$, therefore $P(s'|s,a)=1$. □

Assumption 3 (Support Coverage)**.** *The support of expert transition distribution* $\mu^E(s, s')$ *is covered by* $\mu^R(s, s')$*:*

$$\mu^E(s, s') > 0 \Longrightarrow \mu^R(s, s') > 0.$$

Combing Corollary 2 and Assumption 3, we can reach at the following:

Corollary 3. $\forall (s, s') \sim \mu^E(s, s'), \mu^R(a|s, s') > 0 \Longrightarrow P(a|s, s') = 1.$

Lemma 3. *Given a policy* $\hat{\pi}$*, s.t.* $\forall (s, s') \sim \mu^E(s, s')$*,* $\hat{\pi}(a|s) \propto \mu^R(a|s, s')$*, then it satisfies that:*

$$\forall \pi : \mathcal{S} \to \mathcal{A},\ \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^{\pi}(s'|s)] \geq \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^{\hat{\pi}}(s'|s)].$$

Proof. In a discrete state-action space, $\mu^{\pi}(s'|s)$ can be denoted as $\mu^{\pi}(s'|s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[P(s'|s, a)]$, and the similar for $\mu^{\hat{\pi}}(s'|s)$:

$$\begin{aligned}
&\mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^{\hat{\pi}}(s'|s)] - \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^{\pi}(s'|s)] \\
=&\mathbb{E}_{\mu^E(s,s')}\left[\log \frac{\mu^E(s'|s)}{\mu^{\hat{\pi}}(s'|s)} - \log \frac{\mu^E(s'|s)}{\mu^{\pi}(s'|s)}\right] \\
=&\mathbb{E}_{\mu^E(s,s')}\left[\log \mu^{\pi}(s'|s)] - \log \mu^{\hat{\pi}}(s'|s)\right] \\
=&\mathbb{E}_{\mu^E(s,s')}\left[\log \mathbb{E}_{a\sim\pi(\cdot|s)}[P(s'|s, a)]\right] - \mathbb{E}_{\mu^E(s,s')}\left[\log \mathbb{E}_{a\sim\hat{\pi}(\cdot|s)}[P(s'|s, a)]\right] \\
=&\mathbb{E}_{\mu^E(s,s')}\left[\log \mathbb{E}_{a\sim\pi(\cdot|s)}[P(s'|s, a)]\right] - \mathbb{E}_{\mu^E(s,s')}\left[\log \mathbb{E}_{a\sim\mu^R(\cdot|s,s')}[P(s'|s, a)]\right] \\
=&\mathbb{E}_{\mu^E(s,s')}\left[\log \mathbb{E}_{a\sim\pi(\cdot|s)}[P(s'|s, a)]\right] - \mathbb{E}_{\mu^E(s,s')}\underbrace{\left[\log \mathbb{E}_{a\sim\mu^R(\cdot|s,s')}[1]\right]}_{\text{Corollary 3}} \\
=&\mathbb{E}_{\mu^E(s,s')}\left[\log \mathbb{E}_{a\sim\pi(\cdot|s)}[P(s'|s, a)]\right] \\
\leq&\mathbb{E}_{\mu^E(s,s')}\left[\log \mathbb{E}_{a\sim\pi(\cdot|s)}[1]\right] \\
=&0.
\end{aligned}$$

□

Remark 4. *In a deterministic MDP, assuming the support of* $\mu^E(s, s')$ *is covered by* $\mu^R(s, s)$*, s.t.* $\mu^E(s, s') > 0 \Longrightarrow \mu^R(s, s') > 0$*, then regulating policy using* $\mu^R(\cdot|s, s')$ *can*

minimize $\mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^\pi(s'|s)]$*:*

$$\exists \tilde{\pi} : \mathcal{S} \to \mathcal{A},\ s.t.\ \forall (s,s') \sim \mu^E(s,s'),\ \tilde{\pi}(\cdot|s) \propto \mu^R(\cdot|s,s') \Longrightarrow \tilde{\pi} = \arg\min_\pi \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^\pi(s'|s)].$$

Proof. Based on Lemma 3, we have that:

$$\forall \pi : \mathcal{S} \to \mathcal{A},\ \ \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^\pi(s'|s)] \geq \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^{\tilde{\pi}}(s'|s)].$$

Therefore, $\tilde{\pi} = \arg\min_\pi \mathbb{D}_{\mathrm{KL}}[\mu^E(s'|s)||\mu^\pi(s'|s)]$.

□

A.0.5.3 Estimating the Inverse Action Distribution

Theorem 7.

$$\max_{P_I : \mathcal{S}\times\mathcal{S}\to\mathcal{A}} -\mathbb{D}_{\mathrm{KL}}[\mu^R(a|s,s')||P_I(a|s,s')] \equiv \max_{P_I : \mathcal{S}\times\mathcal{S}\to\mathcal{A}} \mathbb{E}_{(s,a,s')\sim\mu^R(s,a,s')}[\log P_I(a|s,s')].$$

Proof.

$$\begin{aligned}
&- \mathbb{D}_{\mathrm{KL}}[\mu^R(a|s,s')||P_I(a|s,s')] \\
&= -\int_{\mathcal{S}\times\mathcal{S}\times\mathcal{A}} \mu^R(s,s')\mu^R(a|s,s') \log \frac{\mu^R(a|s,s')}{P_I(a|s,s')} dadsds' \\
&= -\int_{\mathcal{S}\times\mathcal{S}\times\mathcal{A}} \mu^R(s,s')\mu^R(a|s,s') \Big(\log \mu^R(a|s,s') - \log P_I(a|s,s') \Big) dadsds' \\
&= \underbrace{H[\mu^R(a|s,s')]}_{\text{fixed } w.r.t.\ P_I} + \int_{\mathcal{S}\times\mathcal{S}\times\mathcal{A}} \mu^R(s,s')\mu^R(a|s,s') \log P_I(a|s,s') dadsds' \\
&= \underbrace{H[\mu^R(a|s,s')]}_{\text{fixed } w.r.t.\ P_I} + \mathbb{E}_{\mu^R(s,a,s')}[\log P_I(a|s,s')].
\end{aligned}$$

□

Note that we use $H[\mu^R(a|s,s')]$ to denote the conditional entropy of $\mu^R(a|s,s')$, with $H[\mu^R(a|s,s')] = \mathbb{E}_{\mu^R(s,a,s')}[-\log \mu^R(a|s,s')]$.

A.0.6 Derivation of Eq (4.8):

$J_{\text{opolo}}(\pi, Q) = \mathbb{E}_{(s,a,s')\sim\mu^\pi(s,a,s')}[r(s,s') - (\mathcal{B}^\pi Q - Q)(s,a)] + \mathbb{E}_{(s,a)\sim\mu^R(s,a)}[f_*((\mathcal{B}^\pi Q - Q)(s,a))]$,

where $\mathcal{B}^\pi Q(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a),a'\sim\pi(\cdot|s')}\Big[r(s,s') + \gamma Q(s',a')\Big]$, and $r(s,s') = \log\frac{\mu^E(s,s')}{\mu^R(s,s')}$.

Proof. The first term in the RHS of the above equation can be reduced to the following:

$$
\begin{aligned}
&\mathbb{E}_{(s,a,s')\sim\mu^\pi(s,a,s')}[r(s,s') - (\mathcal{B}^\pi Q - Q)(s,a)] \\
=&\mathbb{E}_{(s,a)\sim\mu^\pi(s,a)}\Big[\mathbb{E}_{s'\sim P(\cdot|s,a)}\big[r(s,s') - \big((\mathcal{B}^\pi Q - Q)(s,a)\big)\big]\Big] \\
=&\mathbb{E}_{(s,a)\sim\mu^\pi(s,a)}\Big[\mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,s')] + Q(s,a) - \mathbb{E}_{s'\sim P(\cdot|s,a)}[\mathcal{B}^\pi Q(s,a)]\Big] \\
=&\mathbb{E}_{(s,a)\sim\mu^\pi(s,a)}\Big[\mathbb{E}_{s'\sim P(\cdot|s,a)}\cancel{[r(s,s')]} + Q(s,a) - \mathbb{E}_{s'\sim P(\cdot|s,a),a'\sim\pi(\cdot|s')}[\cancel{r(s,s')} + \gamma Q(s',a')]\Big] \\
=&\mathbb{E}_{(s,a)\sim\mu^\pi(s,a)}\Big[Q(s,a) - \gamma\mathbb{E}_{s'\sim P(\cdot|s,a),a'\sim\pi(\cdot|s')}[Q(s',a')]\Big] \\
=&\underbrace{(1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim\mu_t^\pi(s),a\sim\pi(s)}[Q(s,a)]}_{\text{see Table 4.1}} - (1-\gamma)\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{s\sim\mu_t^\pi,a\sim\pi(\cdot|s),s'\sim P(\cdot|s,a),a'\sim\pi(\cdot|s')}[Q(s',a')]] \\
=&(1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim\mu_t^\pi,a\sim\pi(s)}[Q(s,a)] - (1-\gamma)\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{s\sim\mu_{t+1}^\pi,a\sim\pi(\cdot|s)}[Q(s,a)]] \\
=&(1-\gamma)\mathbb{E}_{s\sim p_0,a_0\sim\pi(\cdot|s_0)}[Q(s_0,a_0)].
\end{aligned}
$$

Therefore:

$$J_{\text{opolo}}(\pi, Q) = (1-\gamma)\mathbb{E}_{s\sim p_0,a_0\sim\pi(\cdot|s_0)}[Q(s_0,a_0)] + E_{(s,a)\sim\mu^R}[f_*((\mathcal{B}^\pi Q - Q)(s,a))].$$

□

A.0.7 Implementation Details of *OPOLO*

A.0.7.1 Practical Considerations for Algorithm Implementation

We provide some practical considerations to effectively implement our algorithm:

Initial state sampling: To increase the diversity of initial samples, we use state samples from an off-policy buffer and treat them as *virtual initial states*. A similar strategy is adopted by [88].

Constant shift on synthetic rewards: In practice, we adopt the same strategy of prior art [189] to use $r(s, s') = -\log(1-D(s, s'))$, instead of $\log(D)-\log(1-D)$ as the discriminator output. A fully optimized discriminator D^* satisfies $-\log(1 - D^*(s, s')) = \log(1 + \frac{\mu^E(s,s')}{\mu^R(s,s')})$, which corresponds to a constant shift on $\frac{\mu^E(s,s')}{\mu^R(s,s')}$ before the log term.

Q and π network update: We follow the advice of AlgeaDICE [122] by using a target Q network and policy gradient clipping. Especially, when taking the gradients of $J_{\text{opolo}}(\pi, Q, \alpha)$ *w.r.t.*Q, we use the value from a target Q network to calculate $\mathcal{B}^\pi Q(s, a)$ in order to stabilize training; on the other hand, since an optimal $x^*(s, a) = (\mathcal{B}^\pi Q^* - Q^*)(s, a) = \frac{\mu^\pi(s,a)}{\mu^R(s,a)}$ represents a density ratio and should always be non-negative, we clip $(\mathcal{B}^\pi Q - Q)(s, a)$ to above 0 when taking gradients *w.r.t.*π.

A.0.7.2 Hyper-parameters

Table A.4 lists the hyper-parameters for GAIL [68], GAIfO [171], BCO [169], DAC [87], and our proposed approach *OPOLO*. Specifically, for off-policy approaches, each self-generated interaction will be stored the replay buffer in a FIFO manner, and *update frequency* is the number of interactions sampled from the MDP after which the module is updated. Moreover, considering the different scales for the gradients of $J(\pi_\theta, Q_\phi)$ and $J_{\text{Reg}}(\pi_\theta)$ in Algorithm 4.1, we apply a coefficient λ for *OPOLO* to adjust the regularization strength when calculating the total policy loss:

$$\theta \leftarrow \theta + \alpha\big(J_{\triangledown\theta}(\pi_\theta, Q_\phi) + \lambda J_{\triangledown\theta} J_{\text{Reg}}(\pi_\theta)\big).$$

A.0.8 Challenges of DICE without Expert Actions

In this section, we analyze the principle of offline imitation learning using DICE [121, 201, 122] and the reason that impedes its direct application to an LfO setting.

Hyper-parameters	Value
Shared Parameters for Off-Policy Approaches	
Buffer size	10^7
Batch size	100
Learning rate	$3e^{-4}$
Discount factor γ	0.99
Network architecture	MLP [400, 300]
Q, π update frequency / gradient steps	$10^3/10^3$
D update frequency / gradient steps	500/10
Shared Parameters for On-Policy Approaches	
Batch size	2048
mini-Batch size	256
Learning rate	$3e^{-4}$
Discount factor γ	0.99
Network architecture	MLP [400, 300]
BCO	
P_I pre-train gradient steps	10^4
P_I update frequency / gradient steps	$10^3/100$
DAC	
Number of extra absorbing states	1
OPOLO	
P_I update frequency / gradient steps	500/50
P_I regularization coefficient λ	0.1

Table A.4: Hyper-parameters for Different Algorithms.

In a LfO setting where expert actions are unavailable, the learning objective is to minimize the discrepancy of *state-only* distributions induced by the agent and the expert. Without loss of generality, we consider an arbitrary f-divergence $\mathcal{D}_f$ as the discrepancy measure:

$$\begin{aligned} &\max_{\pi} -\mathcal{D}_f[\mu^\pi(s,s')||\mu^E(s,s')] \\ =&\max_{\pi} \min_{x:\mathcal{S}\times\mathcal{S}\to\mathbb{R}} \mathbb{E}_{\mu^\pi(s,s')}[-x(s,s')] + \mathbb{E}_{\mu^E(s,s')}[f^*(x(s,s'))], \end{aligned} \tag{A.3}$$

in which $f^*(x)$ is the conjugate of $f(x)$ for the f-divergence. To remove the on-policy dependence of $\mu^\pi(s,s')$, we follow the rationale of DICE and use a similar change-of-variable trick mentioned in Sec 4.3.2 to learn a value function $v(s,s')$:

$$v(s,s') := -x(s,s') + \gamma \mathbb{E}_{a'\sim\pi(.|s'),s''\sim P(.|s',a')}[v(s',s'')] = -x(s,s') + \mathcal{B}^\pi v(s,s').$$

This value function is a fixed point solution to an variant Bellman operator $\mathcal{B}^\pi$, which, however, is problematic in a model-free setting. To see this, we substitute $x(s, s')$ by $(\mathcal{B}^\pi v - v)(s, s')$ to transform Eq (A.3) into the following:

$$\begin{aligned}
&\max_\pi \min_{x:\mathcal{S}\times\mathcal{S}\to\mathbb{R}} \mathbb{E}_{\mu^\pi(s,s')}[-x(s,s')] + E_{\mu^E(s,s')}[f^*(x(s,s'))] \\
=&\max_\pi \min_{v:\mathcal{S}\times\mathcal{S}\to\mathbb{R}} (1-\gamma)\underbrace{\mathbb{E}_{s_0\sim p_0, s_1\sim P(\cdot|s_0,\pi(s_0))}}_{\text{term 1}}[v(s_0,s_1)] + \underbrace{\mathbb{E}_{\mu^E(s,s')}[f^*((\mathcal{B}^\pi v - v)(s,s'))]}_{\text{term 2}}.
\end{aligned}$$

where $\mathcal{B}^\pi v(s, s') = \gamma \mathbb{E}_{a'\sim\pi(.|s'),s''\sim P(.|s',a')}[v(s', s'')]$. Optimizing this objective is troublesome, in that the $\mathcal{B}^\pi v(s, s')$ in term 2 requires knowledge of $P(\cdot|s, \pi(s)),\ \forall s \sim \mu^E(s)$. In another word, for any state sampled from the *expert* distribution, we need to know what would be the *next* state if following policy π from this state. A similar issue is echoed in term 1, where s_1 is sampled from $P(\cdot|s_0, \pi(s_0))$. Consequently, directly applying DICE loses its advantage in a LfO setting, as it incurs a dependence on a *forward transition* model, which is costly to estimate and may counteract the efficiency brought by off-policy learning.

APPENDIX B

APPENDIX FOR *FEDGEN*

B.0.1 Notations and Preliminaries

Let $\mathcal{X} \subset \mathbb{R}^p$ be the *input* space, $\mathcal{Z} \subset \mathbb{R}^d$ be the *latent* feature space, and $\mathcal{Y} \subset \mathbb{R}$ be the output space. $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ denotes a ***representation function*** that maps inputs into features. $\mathcal{T}$ denotes a ***domain*** (or *task*), which consists of a data distribution $\mathcal{D}$ over $\mathcal{X}$ and a ground-truth *labeling* function $c^* : \mathcal{X} \to \mathcal{Y}$. Given a domain $\mathcal{T} := \langle \mathcal{D}, c^* \rangle$ and a representation function $\mathcal{R}$, we use $\tilde{\mathcal{D}}$ to denote the *induced image* of $\mathcal{D}$ under $\mathcal{R}$ [17], *s.t.* given a probability event $\mathcal{B}$,

$$\mathbb{E}_{z \sim \tilde{\mathcal{D}}}[\mathcal{B}(z)] = \mathbb{E}_{x \sim \mathcal{D}}[\mathcal{B}(\mathcal{R}(x))].$$

Accordingly, $\tilde{c}^*$ denotes the *induced* labeling function under $\mathcal{R}$:

$$\tilde{c}^*(z) := \mathbb{E}_{x \sim D}\left[c^*(x) | \mathcal{R}(x) = z\right].$$

Let $h : \mathcal{Z} \to \mathcal{Y}$ denote a ***hypothesis*** that maps features to predicted labels, and $\mathcal{H} \subseteq \{h : \mathcal{Z} \to \mathcal{Y}\}$ denote a hypothesis class. For our analysis, we assume the FL tasks are for binary classification, *i.e.* $\mathcal{Y} = \{0, 1\}$, and the loss function is 0-1 bounded, with $l(\hat{y}, y) = |\hat{y} - y|$. Same assumptions have been adopted by various prior art [17, 18, 16, 103, 17].

Given two distributions $\mathcal{D}$ and $\mathcal{D}'$, $d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}')$ is defined as the $\mathcal{H}$-divergence between $\mathcal{D}$ and $\mathcal{D}'$, *i.e.*:

$$d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') := 2 \sup_{\mathcal{A} \in \mathcal{A}_{\mathcal{H}}} |\Pr_{\mathcal{D}}(\mathcal{A}) - \Pr_{\mathcal{D}'}(\mathcal{A})|\},$$

where $\mathcal{A}_{\mathcal{H}}$ is a set of measurable subsets under $\mathcal{D}$ and $\mathcal{D}'$ for certain $h \in \mathcal{H}$. Moreover, $\mathcal{H} \triangle \mathcal{H}$ is defined as the symmetric difference hypothesis space [18], *i.e.*:

$$\mathcal{H} \triangle \mathcal{H} := \{h(z) \oplus h'(z), h, h' \in \mathcal{H}\}$$

where $\oplus$ denotes the XOR operator, so that $h(z) \oplus h'(z)$ indicates that h and h' disagrees with each other. Accordingly, $\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}$ is a set of measurable subsets for $\forall\, h(z)\oplus h'(z) \in \mathcal{H}\triangle\mathcal{H}$. Then $d_{\mathcal{H}\triangle\mathcal{H}}(\cdot,\cdot)$ is defined as the *distribution divergence* induced by the symmetric difference hypothesis space [18]:

$$d_{\mathcal{H}\triangle\mathcal{H}}(\mathcal{D},\mathcal{D}') := 2 \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} |\Pr_{\mathcal{D}}(\mathcal{A}) - \Pr_{\mathcal{D}'}(\mathcal{A})|\}.$$

Specifically, let $\mathcal{D},\mathcal{D}'$ be two arbitrary distributions on the input space $\mathcal{X}$, and let $\tilde{\mathcal{D}},\tilde{\mathcal{D}}'$ be their induced images over $\mathcal{R}$. Then based on the definition of $d_{\mathcal{H}\triangle\mathcal{H}}(\cdot,\cdot)$, one can have:

$$\begin{aligned} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}},\tilde{\mathcal{D}}') &= 2 \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} |\mathbb{E}_{x\sim\mathcal{D}}\left[\Pr(\mathcal{A}(\mathcal{R}(x)))\right] - \mathbb{E}_{x\sim\mathcal{D}'}\left[\Pr(\mathcal{A}(\mathcal{R}(x)))\right]| \\ &= 2 \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} |\mathbb{E}_{z\sim\tilde{\mathcal{D}}}\left[\Pr(\mathcal{A}(x))\right] - \mathbb{E}_{z\sim\tilde{\mathcal{D}}'}\left[\Pr(\mathcal{A}(z))\right]| \\ &= 2 \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} |\Pr_{\tilde{\mathcal{D}}}(\mathcal{A}) - \Pr_{\tilde{\mathcal{D}}'}(\mathcal{A})|\}. \end{aligned}$$

B.0.2 Derivations of Remark 5

Remark. *Let $p(y)$ be the prior distribution of labels, and $r(z|y) : \mathcal{Y} \to \mathcal{Z}$ be the conditional distribution derived from generator $G_{\boldsymbol{w}}$. Then regulating a user model $\boldsymbol{\theta}_k$ using samples from $r(z|y)$ can minimize the conditional KL-divergence between two distributions, derived from the user and from the generator, respectively:*

$$\max_{\boldsymbol{\theta}_k}\ \mathbb{E}_{y\sim p(y),z\sim r(z|y)}\left[\log p(y|z;\boldsymbol{\theta}_k)\right] \equiv \min_{\boldsymbol{\theta}_k}\ \mathbb{D}_{\mathrm{KL}}[r(z|y)\|p(z|y;\boldsymbol{\theta}_k)],$$

Proof. Expanding the KL-divergence, we have

$$\begin{aligned} \because\ \mathbb{D}_{\mathrm{KL}}[r(z|y)\|p(z|y;\boldsymbol{\theta}_k)] &\equiv \mathbb{E}_{y\sim p(y)}\left[\mathbb{E}_{z\sim r(z|y)}\left[\log\frac{r(z|y)}{p(z|y;\boldsymbol{\theta})}\right]\right] \\ &= \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}\left[\log r(z|y)\right] - \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}\left[\log p(z|y;\boldsymbol{\theta})\right] \\ &= \underbrace{-\,H(r(z|y))}_{\text{constant w.r.t }\boldsymbol{\theta}_k} - \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log p(z|y;\boldsymbol{\theta})]. \end{aligned}$$

where $H(r(z|y))$ is constant w.r.t $\boldsymbol{\theta}_k$. Therefore when optimizing $\boldsymbol{\theta}_k$ we have:

$$
\begin{aligned}
&\min_{\boldsymbol{\theta}_k} \ \mathbb{D}_{\mathrm{KL}}[r(z|y)\|p(z|y;\boldsymbol{\theta}_k)] \\
\equiv &\min_{\boldsymbol{\theta}_k} \ -\mathbb{E}_{y\sim p(y),z\sim r(z|y)}\left[\log p(z|y;\boldsymbol{\theta}_k)\right] \\
\equiv &\max_{\boldsymbol{\theta}_k} \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log \frac{p(y|z;\boldsymbol{\theta}_k)p(z)}{p(y)}] \\
\equiv &\max_{\boldsymbol{\theta}_k} \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log p(y|z;\boldsymbol{\theta}_k) + \log p(z) - \log p(y)] \\
\equiv &\max_{\boldsymbol{\theta}_k} \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log p(y|z;\boldsymbol{\theta}_k)].
\end{aligned}
$$

where $H(r(z|y))$ denotes the entropy of the probability distribution $r(z|y)$ which is *not* optimizable w.r.t $\boldsymbol{\theta}_k$, and $p(z|y;\boldsymbol{\theta}_k) := \frac{p(y|z;\boldsymbol{\theta}_k)p(z)}{p(y)}$ is defined as the probability that the input representation to the predictor is z if it yields a label y. □

B.0.3 Derivations of Theorem 2

Before deriving Theorem 1, we first present an upper-bound for the generalization performance from prior art [17], which analyzes the role of a feature representation function in the context of *domain adaptation*:

Lemma 4. Generalization Bounds for Domain Adaptation [17, 18]:

Let $\mathcal{T}_S$ and $\mathcal{T}_T$ be the source and target domains, whose data distributions are $\mathcal{D}_S$ and $\mathcal{D}_T$. Let $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ be a feature representation function, and $\tilde{\mathcal{D}}_S, \tilde{\mathcal{D}}_T$ be the induced images of $\mathcal{D}_S$ and $\mathcal{D}_T$ over $\mathcal{R}$, respectively. Let $\mathcal{H}$ be a set of hypotheses with VC-dimension d. Then with probability at least $1-\delta$, $\forall\ h \in \mathcal{H}$:

$$
\mathcal{L}_{\mathcal{T}_T}(h) \le \hat{\mathcal{L}}_{\mathcal{T}_S}(h) + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4}{\delta}\right)} + d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_S, \tilde{\mathcal{D}}_T) + \lambda, \tag{B.1}
$$

where e is the base of the natural logarithm, $\hat{\mathcal{L}}_{\mathcal{T}_S}(\mathrm{h})$ is the empirical risk of the source domain given m observable samples, and $\lambda = \min_{h\in\mathcal{H}}\left(\mathcal{L}_{\mathcal{T}_T}(h) + \mathcal{L}_{\mathcal{T}_S}(h)\right)$ is the optimal risk on the two domains.

One insight from Lemma 4 is that a good representation function plays a tradeoff between minimizing the empirical risk ($\hat{\mathcal{L}}_{\mathcal{T}_S}(h)$) and the induced distributional discrepancy ($d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_S, \tilde{\mathcal{D}}_T)$). Based on Lemma 4, one can establish Theorem 1 as the following:

Theorem. (**Generalization Bounds for FL**) *Consider an FL system with K users. Let $\mathcal{T}_k = \langle \mathcal{D}_k, c^* \rangle$ and $\mathcal{T} = \langle \mathcal{D}, c^* \rangle$ be the k-th local domain and the global domain, respectively. Let $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ be a feature extraction function that is simultaneously shared among users. Let h_k denote the hypothesis learned on domain $\mathcal{T}_k$, and $h = \frac{1}{K}\sum_{k=1}^{K} h_k$ be the global ensemble of user predictors. Then with probability at least $1-\delta$:*

$$\mathcal{L}_{\mathcal{T}}(h) \leq \frac{1}{K}\sum_{k\in[K]} \hat{\mathcal{L}}_{\mathcal{T}_k}(h_k) + \frac{1}{K}\sum_{k\in[K]} (d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \lambda_k) + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4K}{\delta}\right)},$$

*where $\hat{\mathcal{L}}_{\mathcal{T}_k}(h_k)$ is the empirical risk of h_k, $\lambda_k := \min_h(\mathcal{L}_{\mathcal{T}_k}(h) + \mathcal{L}_{\mathcal{T}}(h))$ denotes an oracle performance on $\mathcal{T}_k$ and $\mathcal{T}$, and $\tilde{\mathcal{D}}_k$ and $\tilde{\mathcal{D}}$ is the **induced** image of $\mathcal{D}_k$ and $\mathcal{D}$ from $\mathcal{R}$, respectively, s.t. $\mathbb{E}_{z\sim\tilde{\mathcal{D}}_k}[\mathcal{B}(z)] = \mathbb{E}_{x\sim\mathcal{D}_k}[\mathcal{B}(\mathcal{R}(x))]$ given a probability event $\mathcal{B}$, and so for $\tilde{\mathcal{D}}$.*

Proof. By treating each one of the local domains $k \in [K]$ as the *source* and the global domain as the *target*, one can have that, $\forall\ \delta > 0$, with probability $1 - \frac{\delta}{K}$:

$$\mathcal{L}_{\mathcal{T}}(h_k) \leq \hat{\mathcal{L}}_{\mathcal{T}_k}(h_k) + d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \lambda_k + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4K}{\delta}\right)}.$$

Also, due to the convexity of risk function and Jesen inequality, one can have:

$$\mathcal{L}_{\mathcal{T}}(h) \equiv \mathcal{L}_{\mathcal{T}}\left(\frac{1}{K}\sum_{k\in[K]} h_k\right) \leq \frac{1}{K}\sum_{k\in[K]} \mathcal{L}_{\mathcal{T}}(h_k).$$

Therefore,

$$\begin{aligned}
&\Pr\left[\mathcal{L}_{\mathcal{T}}(h) > \frac{1}{K}\sum_{k\in[K]}\left(\hat{\mathcal{L}}_{\mathcal{T}_k}(h_k) + \sum_{k\in[K]}(d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \lambda_k) + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4K}{\delta}\right)}\right)\right] \\
\leq &\Pr[\frac{1}{K}\sum_{k\in[K]}\mathcal{L}_{\mathcal{T}}(h_k) > \frac{1}{K}\sum_{k\in[K]}(\hat{\mathcal{L}}_{\mathcal{T}_k}(h_k) + \sum_{k\in[K]}(d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \lambda_k) + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4K}{\delta}\right)})] \\
\leq &\Pr\left[\bigvee_{k\in[K]}\mathcal{L}_{\mathcal{T}}(h_k) > \hat{\mathcal{L}}_{\mathcal{T}_k}(h_k) + d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \lambda_k + \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4K}{\delta}\right)}\right] \\
\leq &\sum_{k\in[K]}\frac{\delta}{K} = \delta.
\end{aligned}$$

□

Theorem 1 shows that the performance of the aggregated hypothesis is upper-bounded by: 1) the local performance of each user hypothesis ($\hat{\mathcal{L}}_{\mathcal{T}_k}(h_k)$), 2) the dissimilarity between the global and local distributions over the feature space ($d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}})$), 3) the oracle performance (λ_k), and 4) the numerical constraints regarding the number of empirical samples m and the VC-dimension d.

B.0.4 Derivations of Corollary 1

Corollary. *Let $\mathcal{T}$, $\mathcal{T}_k$, $\mathcal{R}$ defined as in Theorem 1. $\mathcal{D}_{\mathrm{A}}$ denotes an **augmented** data distribution, and $\mathcal{D}'_k = \frac{1}{2}(\mathcal{D}_k + \mathcal{D}_{\mathrm{A}})$ is a **mixture** of distributions. Accordingly, $\tilde{\mathcal{D}}_{\mathrm{A}}$ and $\tilde{\mathcal{D}}'_k$ denote the **induced** image of $\mathcal{D}_{\mathrm{A}}$ and $\mathcal{D}'_k$ over $\mathcal{R}$, respectively. Let $\hat{\mathcal{D}}'_k = \hat{\mathcal{D}}_k \cup \hat{\mathcal{D}}_{\mathrm{A}}$ be an empirical dataset of $\mathcal{D}'_k$, with $|\hat{\mathcal{D}}_k|$=m, $|\hat{\mathcal{D}}'_k| = |\hat{\mathcal{D}}_k| + |\hat{\mathcal{D}}_{\mathrm{A}}| = m'$. Assume the discrepancy between $\tilde{\mathcal{D}}_{\mathrm{A}}$ and $\tilde{\mathcal{D}}$ is bounded, s.t $\exists\ \epsilon > 0, d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}}) \leq \epsilon$, then with probability $1 - \delta$:*

$$\mathcal{L}_{\mathcal{T}}(h) \leq \frac{1}{K}\sum_k \mathcal{L}_{\mathcal{T}'_k}(h_k) + \frac{1}{K}\sum_k (d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}})) + \frac{1}{K}\sum_k \lambda'_k + \sqrt{\frac{4}{m'}\left(d\log\frac{2em'}{d} + \log\frac{4K}{\delta}\right)}, \tag{B.2}$$

where $\mathcal{T}'_k = \{\mathcal{D}'_k, c^*\}$ is the updated local domain, $\lambda'_k = \min_h(\mathcal{L}_{\mathcal{T}'_k}(h) + \mathcal{L}_{\mathcal{T}}(h))$ denotes the oracle performance, and $d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}}) \leq d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}})$ when ϵ is small.

Proof. Equation B.2 can be directly derived by Theorem 1. We now focus on analyzing the relation between $d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}})$ and $d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}})$, which is the data dissimilarity ***before*** and ***after*** data augmentation using samples from distribution $\mathcal{D}_A$, respectively.

Based on the definition of $d_{\mathcal{H}\triangle\mathcal{H}}(\cdot,\cdot)$, one can derive that:

$$
\begin{aligned}
& d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}}) \\
=& 2 \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} \left| \mathbb{E}_{z\sim\tilde{\mathcal{D}}'_k} \left[\Pr(\mathcal{A}(z))\right] - \mathbb{E}_{z\sim\tilde{\mathcal{D}}} \left[\Pr(\mathcal{A}(z))\right] \right| \\
=& 2 \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} \left| \mathbb{E}_{z\sim\frac{1}{2}(\tilde{\mathcal{D}}_k+\tilde{\mathcal{D}}_A)} \left[\Pr(\mathcal{A}(z))\right] - \mathbb{E}_{z\sim\tilde{\mathcal{D}}} \left[\Pr(\mathcal{A}(z))\right] \right| \\
=& 2 \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} \left| \frac{1}{2}\mathbb{E}_{z\sim\tilde{\mathcal{D}}_K} \left[\Pr(\mathcal{A}(z))\right] + \frac{1}{2}\mathbb{E}_{z\sim\tilde{\mathcal{D}}_A} \left[\Pr(\mathcal{A}(z))\right] - \mathbb{E}_{z\sim\tilde{\mathcal{D}}} \left[\Pr(\mathcal{A}(z))\right] \right| \\
\leq & \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} \left| \mathbb{E}_{z\sim\tilde{\mathcal{D}}_K} \left[\Pr(\mathcal{A}(z))\right] - \mathbb{E}_{z\sim\tilde{\mathcal{D}}} \left[\Pr(\mathcal{A}(z))\right] \right| \\
& + \sup_{\mathcal{A}\in\mathcal{A}_{\mathcal{H}\triangle\mathcal{H}}} \left| \mathbb{E}_{z\sim\tilde{\mathcal{D}}_A} \left[\Pr(\mathcal{A}(z))\right] - \mathbb{E}_{z\sim\tilde{\mathcal{D}}} \left[\Pr(\mathcal{A}(z))\right] \right| \\
=& \frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}}).
\end{aligned}
$$

It is clear that $\frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}})$, which is bounded by ϵ, affects the dissimilarity between the *induced* image of local and the global distribution, therefore plays a key role in upper-bounding the global performance ($\mathcal{L}_{\mathcal{T}}(h)$ in Equation B.2). Next, we discuss different scenarios when FL can benefit from such augmented data, and when the quality of augmented distribution $\mathcal{D}_A$ can limit the generalization performance of the aggregated model.

$\mathcal{D}_A$ can benefit local users when ϵ is small: To see this, one can assume that:

$$
d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}}) = \epsilon \leq \min_k d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}),
$$

of which the intuition is that, after feature mapping, the discrepancy between the augmented distribution and the global distribution is smaller than the discrepancy between an individual user and the global. Based on this assumption, one can conclude that $\forall\ \mathcal{T}_k \in \boldsymbol{\mathcal{T}}$:

$$
\begin{aligned}
d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}}) =& \frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}}) \\
\leq & \frac{1}{2} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}) + \min_j d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_j, \tilde{\mathcal{D}}) \\
\leq & d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}),
\end{aligned}
$$

Therefore, a small $d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}})$ benefits local users w.r.t their generalization performance, by both reducing the data discrepancy and enriching the empirical samples, in that:

$$\mathcal{L}_{\mathcal{T}}(h_k) \leq \mathcal{L}_{\mathcal{T}'_k}(h_k) + \lambda'_k + \underbrace{\leq d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}})}_{\leq d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}})} + \underbrace{\sqrt{\frac{4}{m'}\left(d\log\frac{2em'}{d} + \log\frac{4}{\delta}\right)}}_{\leq\sqrt{\frac{4}{m}(d\log\frac{2em}{d}+\log\frac{4}{\delta})}} \quad (\text{ Lemma 4}).$$

$\mathcal{D}_A$ has positive effects on the generalization performance when ϵ is moderate: Instead, one might as well assume that

$$d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_A, \tilde{\mathcal{D}}) = \epsilon \leq \frac{1}{K}\sum_{k=1}^{K} d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}),$$

which implies that, after feature mapping over $\mathcal{R}$, the dissimilarity between $\mathcal{D}_A$ and the global distribution $\mathcal{D}$ is at least as small as the *average* dissimilarity between local users and the global. Based on this assumption, one can derive that:

$$\sum_k d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}'_k, \tilde{\mathcal{D}}) \leq \sum_k d_{\mathcal{H}\triangle\mathcal{H}}(\tilde{\mathcal{D}}_k, \tilde{\mathcal{D}}),\ \sqrt{\frac{4}{m'}\left(d\log\frac{2em'}{d} + \log\frac{4}{\delta}\right)} \leq \sqrt{\frac{4}{m}\left(d\log\frac{2em}{d} + \log\frac{4}{\delta}\right)},$$

which can still contribute to a tighter upper-bound for the global performance in Equation B.2, compared with not using the augmented data.

Conversely, when ϵ is over-large, which implies that $\mathcal{D}_A$ is not relevant to the original FL task, it may have negative impacts on the generalization performance. □

B.0.5 Practical Settings of *FedGen*

We first discuss some practical considerations for implementing our algorithm:

- **Weighting user models:** User models vary in their ability to predict certain labels over others due to their statistical heterogeneity. Therefore, we use the number of training labels available to users to summarize a weight matrix $\mathbf{\Lambda} = \{\lambda_k^c | c \in \mathcal{Y}, k \in \{1, 2, \cdots, K\}\}$, *s.t.* $\forall c, i, j,\ \frac{\lambda_i^c}{\lambda_j^c} = \frac{n_i^c}{n_j^c}$ indicates the ratio of training samples for label c between two users i and j, and $\sum_k \lambda_k^c = 1\ \forall c \in \mathcal{Y}$. We then apply this weight matrix

to adjust the generator objective as the following:

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}) := \mathbb{E}_{y\sim\hat{p}(y)}\mathbb{E}_{z\sim G_{\boldsymbol{w}}(z|y)}\left[\lambda_k^y l\left(\sigma\left(\frac{1}{K}\sum\nolimits_{k=1}^{K} g(z;\boldsymbol{\theta}_k^p)\right), y\right)\right].$$

We found that this weighted objective can further mitigate the impact of negative ensemble, especially when a teacher model is too weak to predict certain labels due to lacking training samples of that category.

- **Stochastic generative learning:** Built upon prior arts on generative learning [83], we use an auxiliary noise vector with dimension d_n to infer the desirable feature representation for a given label y, *s.t.* $z \sim G_{\boldsymbol{w}}(\cdot|y) \equiv G_{\boldsymbol{w}}(y, \epsilon|\epsilon \sim \mathcal{N}(0, I))$. To further increase the diversity of the generator output, we also leverage the idea of *diversity loss* from prior work [114] to train the generator model.

B.0.6 Prototype Results

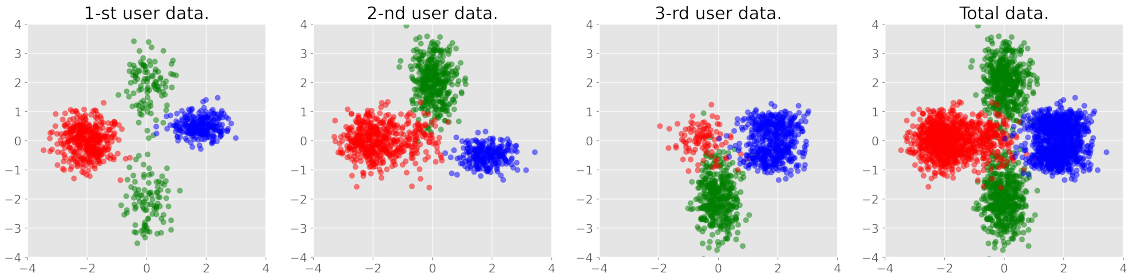
We adopt a one-round FL setting for the prototype experiment, for which the dataset distributions of local users, as well as their model decision boundaries *before* and *after* knowledge distillation, are illustrated in Figure B.1. Accuracy of user models on the global dataset is also summarized in Table B.1, from which one can observe that the generalization performance of user models have been notably improved by the distilled knowledge.

	User 1	User 2	User 3	Oracle
Before	97.1	81.3	81.2	98.4
After	98.6	98.3	98.2	

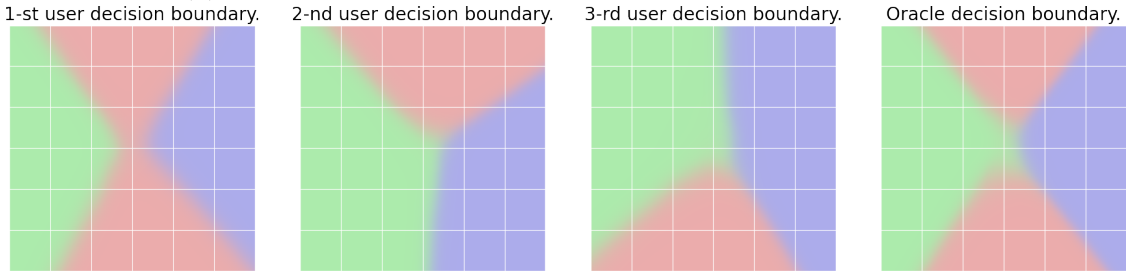
Table B.1: Accuracy (%) before and after KD.

B.0.7 Experimental Setup

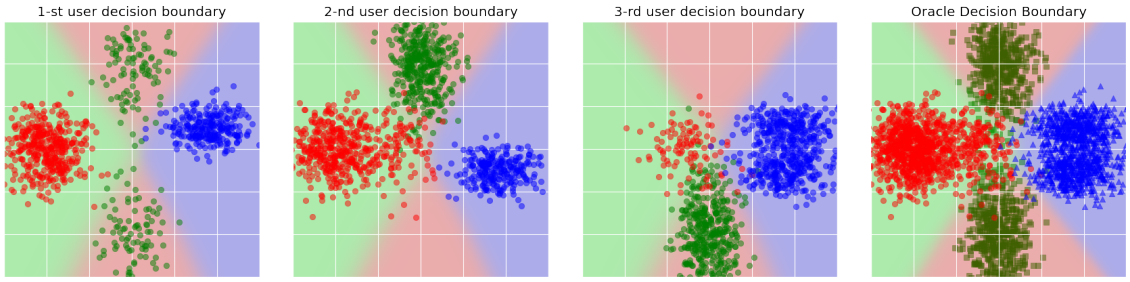
We provide the network architecture for the generator and the classifier in Table B.2 and Table B.3. For the generator $G_{\boldsymbol{w}}$, we adopt a two-MLP layer network. It takes a noise vector ϵ and a one-hot label vector y as the input, which, after a hidden layer with dimension d_h, outputs a feature representation with dimension d. For the classifier, we adopt a network



(a) Local user data distribution and total data distribution.



(b) User models generate biased decision boundaries **before** KD, provided with incomplete local data.



(c) Decision Boundaries of user models are improved **after** KD.

Figure B.1: Knowledge distillation process for the prototype experiment.

architecture with a CNN module followed by an MLP module. Hyperparameter settings for the experiments are provided in Table B.2 and B.3.

Dataset	Hyperparameter	Value
CelebA	d_n, d_h, d	32, 128, 32
Mnist& EMnist	d_n, d_h, d	32, 256, 32

Table B.2: Network architecture for the generator $G_{\boldsymbol{w}}$.

Dataset	Hyperparameter	Value
CelebA	CNN Module	[16, M, 32, M, 64]
	MLP Module	[784, 32]
Mnist & EMnist	CNN Module	[6, 16]
	MLP Module	[784, 32]

Table B.3: Network architecture for the classification model.

B.0.8 *FedGen* with Partial Parameter Sharing

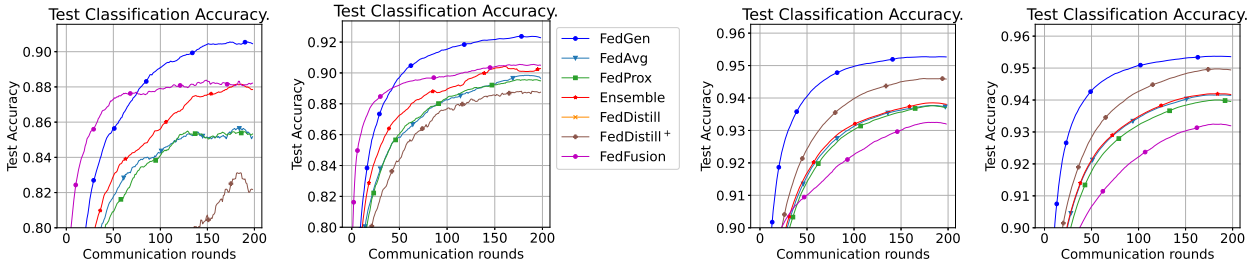
Algorithm B.1 summarizes a variant approach of FedGen for a specific FL setting, where only the last prediction layer is shared among users while keeping the feature extraction layers localized.

Algorithm B.1: FedGen with Partial Parameter Sharing

Require: Tasks $\mathcal{T}_k, k \in \{1, \cdots, K\}$;
  Global predictor $\boldsymbol{\theta}^p$, local parameters $\{\boldsymbol{\theta}_k = [\boldsymbol{\theta}_k^f; \boldsymbol{\theta}_k^p]\}_{k=1}^K$;
  Generator parameter $\boldsymbol{w}$; $\hat{p}(y)$ uniformly initialized;
  Learning rate α, β, local steps T, batch size B, local label counter c_k.
repeat
  Server selects active users $\mathcal{A}$ uniformly at random, then broadcast $\boldsymbol{w}, \boldsymbol{\theta}^p$, $\hat{p}(y)$ to $\mathcal{A}$.
  for all user $k \in \mathcal{A}$ in parallel **do**
    $\boldsymbol{\theta}_k^p \leftarrow \boldsymbol{\theta}^p$,
    for $t = 1, \ldots, T$ **do**
      $\{x_i, y_i\}_{i=1}^B \sim \mathcal{T}_k$, $\{\hat{z}_i \sim G_{\boldsymbol{w}}(\cdot|\hat{y}_i), \hat{y}_i \sim \hat{p}(y)\}_{i=1}^B$.
      Update label counter c_k.
      $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - \beta \nabla_{\boldsymbol{\theta}_k} J(\boldsymbol{\theta}_k)$.
    User sends $\boldsymbol{\theta}_k^p$, c_k back to server.
  Server updates $\boldsymbol{\theta}^p \leftarrow \frac{1}{|\mathcal{A}|} \sum_{k \in \mathcal{A}} \boldsymbol{\theta}_k^p$, and $\hat{p}(y)$ based on $\{c_k\}_{k \in \mathcal{A}}$.
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w})$.
until training stop

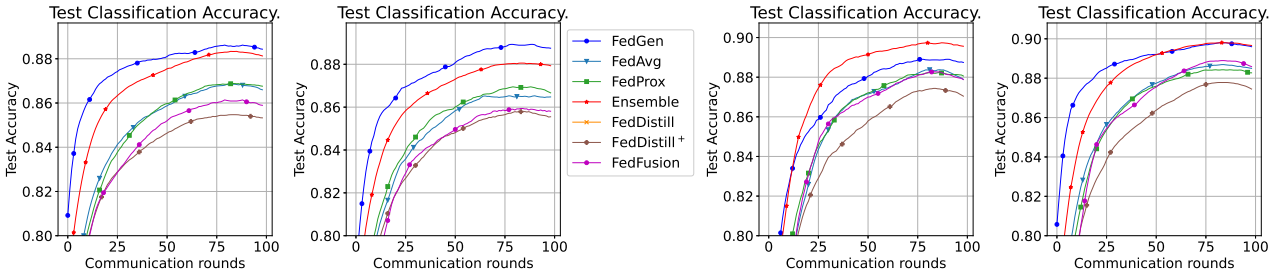
B.0.9 Extended Experimental Results

We elaborate the learning curves trained on the Mnist, CelebA, and EMnist dataset in Figure B.2, Figure B.3, and Figure B.4, respectively, with their performance summarized in Table B.4.



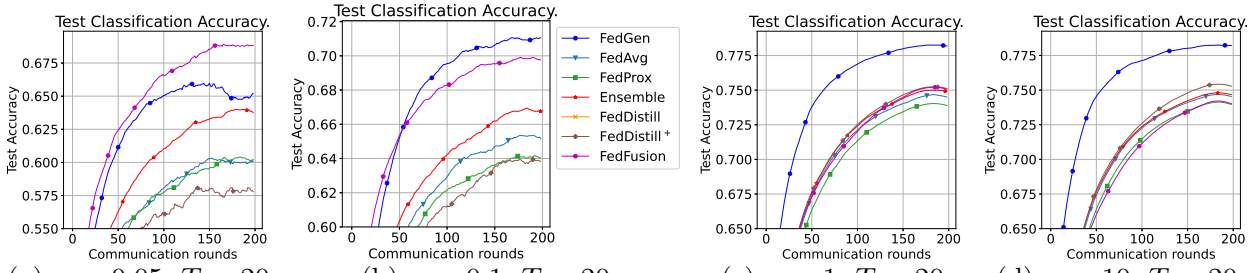
(a) $\alpha = 0.05$ (b) $\alpha = 0.1$ (c) $\alpha = 1$ (d) $\alpha = 10$

Figure B.2: Performance curves on MNIST dataset, where a smaller α denotes larger data heterogeneity.



(a) $r = 9/10$. (b) $r = 5/10$. (c) $r = 5/25$. (d) $r = 0.4$.

Figure B.3: Performance curves on CELEBA dataset.



(a) $\alpha = 0.05$, $T = 20$ (b) $\alpha = 0.1$, $T = 20$ (c) $\alpha = 1$, $T = 20$ (d) $\alpha = 10$, $T = 20$

(e) $\alpha = 0.05$, $T = 40$ (f) $\alpha = 0.1$, $T = 40$ (g) $\alpha = 1$, $T = 40$ (h) $\alpha = 10$, $T = 40$

Figure B.4: Performance curves on EMNIST dataset, under different kinds of data heterogeneity and communication frequencies.

Top-1 Test Accuracy.								
Dataset	Setting	FedAvg	FedProx	FedEnsemble	FedDistill	FedDistill$^+$	FedDFusion	FeDGen
Mnist	$\alpha = 0.05$	87.70±2.07	87.49±2.05	88.85±0.68	70.56±1.24	86.70±2.27	90.02±0.96	**91.30±0.74**
	$\alpha = 0.1$	90.16±0.59	90.10±0.39	90.78±0.39	64.11± 1.36	90.28±0.89	91.11±0.43	**93.03±0.32**
	$\alpha = 1$	93.84±0.25	93.83 ± 0.29	93.91±0.28	79.88±0.66	94.73±0.15	93.37±0.40	**95.52±0.07**
	$\alpha = 10$	94.23±0.13	94.06±0.10	94.25±0.11	89.21±0.26	95.04±0.21	93.36±0.45	**95.79±0.10**
CelebA	$r = 5/10$	87.48±0.39	87.67±0.39	88.48±0.23	76.68±1.23	86.37±0.41	87.01±1.00	**89.70±0.32**
	$r = 5/25$	89.13±0.25	88.84±0.19	**90.22±0.31**	74.99±1.57	88.05± 0.43	88.93±0.79	89.62±0.34
	$r = 10/25$	89.12±0.20	89.01±0.33	90.08±0.24	75.88±1.17	88.14±0.37	89.25±0.56	**90.29±0.47**
EMnist, T=20	$\alpha = 0.05$	62.25±2.82	61.93±2.31	64.99±0.35	60.49±1.27	61.56±2.15	**70.40±0.79**	68.53±1.17
	$\alpha = 0.1$	66.21±2.43	65.29±2.94	67.53±1.19	50.32±1.39	66.06±3.18	70.94±0.76	**72.15±0.21**
	$\alpha = 1$	74.83±0.99	74.12±0.88	75.12±1.07	46.19±0.70	75.41±1.05	75.43±0.37	**78.48±1.04**
	$\alpha = 10$	74.83± 0.69	74.24±0.81	74.90±0.80	54.77±0.33	75.55 ±0.94	74.36±0.40	**78.43±0.74**
EMnist, T=40	$\alpha = 0.05$	64.51±1.13	63.60±0.69	65.74±0.45	60.73±1.62	60.73±1.06	**70.46±1.16**	67.64 ±0.75
	$\alpha = 0.1$	67.71±1.31	66.79±0.77	68.96±0.66	49.54±1.18	67.01±0.38	**71.55±0.43**	70.90 ±0.49
	$\alpha = 1$	77.02±1.09	75.93 ±0.95	77.68±0.98	46.72±0.73	78.12±0.90	77.58±0.37	**78.92± 0.73**
	$\alpha = 10$	77.52±0.66	76.54±0.71	77.92±0.62	54.85±0.44	78.37±0.76	77.31±0.45	**79.29±0.53**

Table B.4: Performance overview under different data heterogeneity settings. For Mnist and EMnist, user data follows the Dirichlet distribution with hyperparameter α, with a smaller α indicating higher heterogeneity. For CelebA, r denotes the ratio between active users and total users. T denotes the local training steps (communication delay). All the above experiments use batch size B=32.

APPENDIX C

APPENDIX FOR *FEDRESCUE*

C.0.1 Case Study: Effects of Progressive Learning

For this prototype experiment, we apply the same network architecture used for learning DIGITSFIVE domains, which is presented in Section C.0.6.1. We illustrate the *progressive* and *overwriting* learning approaches in Figure C.1 and Figure C.2, respectively. For progressive learning, parameters in $\boldsymbol{\theta}_{\times 0.5}$ are updated first for learning the MNIST domain, which are then frozen when updating parameters in $\boldsymbol{\theta}_{\times 1.0} \backslash \boldsymbol{\theta}_{\times 0.5}$ for learning the SYNTHETIC domain. Contrarily, the overwriting approach updates the entire set of model parameters when learning on the SYNTHETIC domain. We set the learning rate to 0.01 and train 10 epochs for learning each domain, with a batch size set to be 32. For the 10% (5%) training data setting, we apply 743 (371) training samples for learning both MNIST and SYNTHETIC domains. Experimental results are averaged from 6 random seeds, with seed numbers set to be 1, 3, 5, 7, 9, 11, respectively.

As shown in Table C.1 and Table C.2, the overwriting learning procedure leads to drastic information forgetting on the previously learned domain (MNIST). On the contrary, evaluations on both the $\times 0.5$ model and $\times 1.0$ model indicate that a *progressive* learning scheme can adaptively build up knowledge on the new training data without undermining the preceding knowledge representations. In fact, the new collateral connections that are progressively learned can also improve the overall model performance on the MNIST domain. For instance, when training using 5% of domain data, model performance on the MNIST domain has been improved from 66.46% (on the $\times 0.5$ model) to 72.27% (on the $\times 1.0$ model), which verifies the advantage of our progressive model learning strategy.

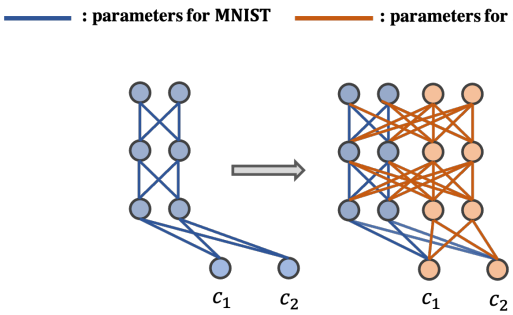


Figure C.1: Illustration of *progressive* learning.

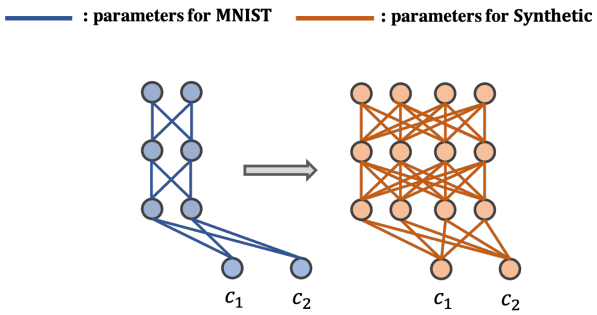


Figure C.2: Illustration of *overwriting* learning.

Accuracy (%) Evaluated on the × **1.0** Model		
Domain	Progressive	Overwriting
Given 10 % training data		
MNIST	**77.95±7.93**	38.17±28.19
SYNTHETIC	**69.48±1.93**	42.30±32.30
Given 5 % training data		
MNIST	**72.27±9.43**	19.60±17.53
SYNTHETIC	**52.43±5.88**	19.03±16.32

Table C.1: Progressive *vs.* overwriting learning.

Accuracy (%) of **MNIST** on the × **0.5** Model		
Train data	Progressive	Overwriting
Given 10 % training data		
10%	**74.99±11.26**	33.51±24.95
Given 5 % training data		
5%	**66.46±14.20.**	19.23±16.21

Table C.2: The *overwriting* learning approach leads to severe forgetting on the previously learned knowledge.

C.0.2 Details of Evaluated Related Work

One related approach to learn prunable models is the ***slimmable*** network proposed in [196], which works by sampling a set of pruning ratios $P = \{p_i | 0 < p_i \leq 1\}_{i=1}^{|P|}$ then performing a *batch gradient* update, as summarized in Algorithm C.1. During each learning iteration, a constant copy of the complete network $\bar{\boldsymbol{\theta}}$ is referred to as *teacher*, whose prediction distribution $p(\cdot|x;\bar{\boldsymbol{\theta}}) \propto f(x;\boldsymbol{\theta})$ is distilled into the submodel $\boldsymbol{\theta}_{\times p_i}$, *i.e.* the *student*. The gradients for both teacher and student models are later aggregated to update the network parameter.

Slimmable learning is proposed for non-FL settings, where sufficient data is accessible on a central machine. There are potential drawbacks of directly applying it to our FL setting. First is the error propagation issue: when the teacher model is underperforming *e.g.* given insufficient training or connection loss, its sub-optimality may be propagated back into the student. In contrast, we propose an objective (in Equation 6.1) that alleviates this issue by

introducing $\min_{\boldsymbol{\theta}_{\times p}} \mathcal{L}(f(\mathcal{X}; \boldsymbol{\theta}_{\times p}), \mathcal{Y})$, which allows submodels to receive label supervisions. Second, the gradients for different submodels may *interfere* with each other when being aggregated, which weakens the model's learning effect, as verified in Section 6.5.6.2.

Another compared related work is ***FjORD*** [69], which also learns prunable networks for a heterogeneous FL system. We summarize its local model updating procedure in Algorithm C.2. In practice, we also apply loss backpropagation through the teacher model (*i.e.* line 9 in Algorithm C.2) when optimizing towards the *FjORD* objectives, which is suggested in [69] to further improve their model performance. Note that *FjORD* does not involve progressive parameter updates, the effects of which will be elaborated more in Section C.0.4.1. In our experiments, we explore different choices of S and perform evaluations on *FjORD* and *FedSlim* using their optimal S accordingly.

Algorithm C.1: SLIMMABLE-TRAINING ([196])

1: **Inputs:** training dataset $D \subset \mathcal{X} \times \mathcal{Y}$; model with parameter set $\boldsymbol{\theta}$; pruning ratios $\mathcal{P}$, learning rate η, loss function $\mathcal{L}$, constant $S \leq |\mathcal{P}|$.
2: **repeat**
3: Sample batch $x, y \sim D$.
4: Sample widths $\hat{P} = [p_i | p_i \sim \mathcal{P}]_{i=1}^{S}$.
5: $\bar{\boldsymbol{\theta}} \leftarrow$ stop_gradient($\boldsymbol{\theta}$).
6: **for** $p_i \sim \hat{P}$ **do**
7: $g_i \leftarrow \nabla_{\boldsymbol{\theta}_{\times p_i}} KL[f(x; \bar{\boldsymbol{\theta}}) \| f(x; \boldsymbol{\theta}_{\times p_i})]$.
8: $g_{\boldsymbol{\theta}} \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(x; \boldsymbol{\theta}), y)$
9: Update parameter $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta * (g_{\boldsymbol{\theta}} + \sum_{i \in |S|} g_i)$
10: **until** training stop

C.0.3 Experiments

C.0.3.1 Treatments on Batch Normalization Layers

The BATCHNORM layers need to be carefully tackled for effectively training prunable models. Prior arts either make the BATCHNORM layers localized on user devices to improve personalized FL, such as proposed in FEDBN[100], or learn individual BATCHNORM modules for each possible submodel [196], which unnecessarily inreases the number of learnable

Algorithm C.2: Local Model Update for *FjORD*([69])

Inputs: training dataset $D \subset \mathcal{X} \times \mathcal{Y}$; model with parameter set $\boldsymbol{\theta}$; pruning ratios $\mathcal{P}$, learning rate η, loss function $\mathcal{L}$, constant $S \leq |\mathcal{P}|$.
repeat
  Sample batch $x, y \sim D$.
  Sample widths $\hat{P} = [p_i | p_i \sim \mathcal{P}]_{i=1}^{S}$. $\triangleright$ ($S = 1$ in [69])
  $\bar{\boldsymbol{\theta}} \leftarrow \text{stop\_gradient}(\boldsymbol{\theta})$.
  for $p_i \sim \hat{P}$ **do**
    $g_i \leftarrow \nabla_{\boldsymbol{\theta}_{\times p_i}} \left\{ \mathcal{L}(f(\mathcal{X}; \boldsymbol{\theta}_{\times p_i}), \mathcal{Y}) + \mathbb{D}_{\text{KL}}[f(x; \bar{\boldsymbol{\theta}}) \| f(x; \boldsymbol{\theta}_{\times p_i})] \right\}$.
  $g_{\boldsymbol{\theta}} \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(x; \boldsymbol{\theta}), y)$
  Update parameter $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta * (g_{\boldsymbol{\theta}} + \sum_{i \in |S|} g_i)$
until training stop

parameters. In our work, we apply a lightweight approach that still shares the trainable parameters in BatchNorm layers, while disabling tracking the running average and variance of training batches, which proves to be an effective scheme across different datasets [39]. For fair comparisons, we apply the same strategy on all baselines, including the *FedAvg*.

Table C.3 summarizes the impacts of different practices regarding BatchNorm layers on the *FedAvg* performance, which indicates that personalizing BN layers, as FedBN applies, might not be good practice for learning *i.i.d.* yet complicated domains such as CelebA. On the contrary, decoupling feature learning from relying on tracking the mean and variance of training data, proves to be effective on both *FedAvg* and our proposed self-distillation approach.

Effects of Different BatchNorm Layer Configurations.				
Algorithm	Personalized BN Layers	Tracking Training Status	Test Accuracy (%) (Given 100% training)	Test Accuracy (%) (Given 20% training)
FedAvg*	×	×	81.06±0.63	68.03±0.50
FedAvg	×	✓	79.73±0.22	68.14±0.47
FedBN	✓	✓	76.12±0.74	60.41±0.57

Table C.3: We adopted *FedAvg** as the baseline implementation in the main paper.

C.0.3.2 Overview of Communication Efficiency

We summarize the communication efficiency of evaluated algorithms on CelebA domain in Table C.4, where evaluation is performed under a heterogeneous FL system. Results demonstrate that *FedResCuE* requires the least communication rounds to reach the predefined accuracy.

Communication Efficiency on CelebA Dataset, *Cluster* Setting					
Accuracy	Model Size	*FedHetero*	*FjORD**	*FedSlim**	*FedResCuE*
	100 **% *training data,*** $0.1 \leq er \leq 0.2$***.***				
65%	$\boldsymbol{w}_{\times 1}$	-	-	-	**174.7±22.2**
60%	$\boldsymbol{w}_{\times 0.5}$	256.7±19.3	218.0±5.9	253.3±26.5	**124.7±11.1**
55%	$\boldsymbol{w}_{\times 0.25}$	222.7±6.8	165.3±5.2	190.7±23.8	**85.3±1.9**
	20 **% *training data,*** $er = 0$				
60%	$\boldsymbol{w}_{\times 1}$	-	244.0±33.5	188.7±133.4	**166.0±14.2**
55%	$\boldsymbol{w}_{\times 0.5}$	180.7±14.8	156.0±13.4	192.0±7.1	**96.0±2.8**
50%	$\boldsymbol{w}_{\times 0.25}$	163.3±11.5	122.7±4.1	168.7±5.2	**76.7±1.9**

Table C.4: Communication Efficiency Overview, where '-' indicates that predefined performance is not reached before training ends.

C.0.3.3 Performance with System Heterogeneity

In addition to the discussion in Section 6.5.2 of the main paper, we provide two more observations regarding system heterogeneity: 1) when FL is free from the risk of a connection interruption, a *uniform* setting in which the same model architecture is assigned to all the edge devices, is generally more beneficial than a *cluster* setting, where heterogeneous and smaller models are enabled. This result conforms to our perception that larger models can capture more representative feature maps for the learning domain, while smaller models sacrifice such information gain for computation and communication efficiency. 2) Contrarily, the diversity in model architecture makes FL resilient to connection loss to some extent. Specifically, performing parameter-wise averaging on heterogeneous models, just as *FedHetero* applies, can potentially make submodels within the global model function as well, in that the submodel parameters are contributed by smaller-capacity users. Therefore, it

can outperform *FedAvg* under unpredictable connection losses, although such an advantage is much less effective than *FedResCuE* with self-distillation learning. We provide a more comprehensive summary in Table C.5, with the accompanying learning curves illustrated in Section C.0.5.

Global Model Accuracy (%) Evaluated on CelebA.							
Training Data	User Capacity	Evaluated Model	*FedAvg*	*FedHetero*	*FjORD**	*FedSlim**	*FedResCuE*
100%	**uniform**	$\boldsymbol{w}_{\times 1}$	81.06±0.63	-	80.57±0.91	81.14±0.76	**81.39±0.20**
		$\boldsymbol{w}_{\times 0.5}$	49.46±1.10	-	77.59±0.31	77.47±0.75	**77.82±0.15**
		$\boldsymbol{w}_{\times 0.25}$	18.57±0.64	-	69.94±0.65	70.47±0.61	**71.19±0.19**
	cluster	$\boldsymbol{w}_{\times 1}$	-	76.80±0.53	75.71±0.47	77.49±0.40	**78.22±0.41**
		$\boldsymbol{w}_{\times 0.5}$	-	76.08±0.50	75.35±0.43	77.12±0.40	**77.58±0.33**
		$\boldsymbol{w}_{\times 0.25}$	-	68.56±0.51	70.98±0.75	73.22±0.34	**73.25±0.47**
20%	**uniform**	$\boldsymbol{w}_{\times 1}$	68.03±0.50	-	67.89±1.47	67.96±0.72	**71.27±0.27**
		$\boldsymbol{w}_{\times 0.5}$	36.56±1.74	-	65.13±1.70	64.80±1.19	**68.15±0.39**
		$\boldsymbol{w}_{\times 0.25}$	16.47±2.24	-	**61.38±1.69**	59.56±1.39	61.12±1.35
	cluster	$\boldsymbol{w}_{\times 1}$	-	59.38±0.41	62.43±1.65	59.53±0.86	**64.53±1.06**
		$\boldsymbol{w}_{\times 0.5}$	-	59.95±0.36	63.06±1.61	60.36±0.45	**66.37±0.78**
		$\boldsymbol{w}_{\times 0.25}$	-	55.41±0.39	61.86±1.21	58.31±0.23	**61.98±0.85**

Table C.5: FL Performance with *i.i.d.* user statistical distributions.

C.0.3.4 Performance Under Connection Loss

In our experiments, we simulate the unpredictable transmission scenarios with connection loss via a random variable er, which denotes the probability that the current *column* transmission is interrupted. For experiments on the CelebA domain, er is first *uniformly* sampled within an error bound (*e.g.* $er \in [0.1, 0.2]$). Next, we use er to determine whether the current column transmission will be interrupted, by comparing it against another uniformly-sampled random variable ϵ, which leads to interruption *iff* $\epsilon < er$. We perform such random sampling on er and ϵ for transmitting each column in a model, while a column for transmission is set to be $\times 0.125$ of the global model width.

Performance overview on the CelebA domain given different connection loss rates is presented in Table C.6. For experiments on the DigitsFive domains, we set constant er that depends on the domain type. Performance on all five domains is provided in Table C.7.

Note that we applied a small training dataset from DigitsFive to ensure the necessity of FL, so that *Local* learning without parameter sharing yields worse performance than FL.

Global Model Accuracy (%) Evaluated on CelebA Under Connection Loss.							
Connection Error	User Capacity	Evaluated Model	*FedAvg*	*FedHetero*	*FjORD*	*FedSlim*	***FedResCuE***
$er \leq 0.2$	uniform	$\boldsymbol{w}_{\times 1}$	50.36±2.17	-	61.79±1.62	57.31±1.27	**70.02±0.40**
		$\boldsymbol{w}_{\times 0.25}$	12.58±0.51	-	60.20±1.67	55.33±0.89	**67.40±0.84**
	cluster	$\boldsymbol{w}_{\times 1}$	-	60.92±1.33	64.52±0.60	62.35±1.76	**69.78±0.74**
		$\boldsymbol{w}_{\times 0.25}$	-	59.70±0.64	64.11±0.41	61.77±1.62	**68.83±1.00**
$er \leq 0.3$	uniform	$\boldsymbol{w}_{\times 1}$	41.79±4.33	-	54.91±1.07	48.46±0.73	**64.80±0.85**
		$\boldsymbol{w}_{\times 0.25}$	12.27±0.93	-	55.20±1.31	48.42±0.87	**64.10±0.26**
	cluster	$\boldsymbol{w}_{\times 1}$	-	51.70±1.14	57.60±2.25	56.28±1.61	**65.74±0.30**
		$\boldsymbol{w}_{\times 0.25}$	-	51.90±0.33	57.69±2.22	56.34±1.56	**65.18±0.57**

Table C.6: Performance under faulty connections, given 100% of training data.

Constant Error Rates *er* for DigitsFive					
Domain	SVHN	Synthetic	USPS	MNIST	MNIST-M
er	0.05	0.05	0.3	0.3	0.05
DigitsFive performance, trained using 5% of data.					
Domain	SVHN	Synthetic	USPS	MNIST	MNIST-M
	×1.0 Model.				
Local	45.88±0.92	62.04±1.16	89.95±1.93	85.84±3.35	61.99±1.79
FedAvg	**69.54±0.15**	80.17±0.24	94.38±0.55	93.61±0.38	75.22±0.87
FedSlim	66.77±0.61	78.95±0.82	94.00±0.41	93.80±0.53	73.95±1.02
FjORD	49.72±23.42	57.12±30.11	68.60±36.34	68.01±37.35	56.29±26.48
FedResCuE	69.32±0.78	**80.57±0.77**	**95.17±0.47**	**95.05±0.44**	**76.89±0.76**
	×0.25 Model.				
FedAvg	20.00±0.55	19.14±5.64	34.03±14.65	32.16±18.65	20.47±9.28
FedSlim	64.00±0.96	77.15±2.13	93.44±1.26	93.38±0.75	72.16±1.51
FjORD	48.11±22.35	54.58±31.15	68.51±36.28	65.79±39.13	52.75±29.16
FedResCuE	**67.11±0.77**	**79.06±0.41**	**94.55±0.37**	**94.64±0.28**	**75.64±0.37**

Table C.7: *FedResCuE* is the most robust algorithm given heterogeneous data and domain-dependent connection errors.

C.0.4 Modeling of Connection Uncertainty

In our experiments, we assume that an error rate *er* is related to each model column transmission between the server and the edge device. This setting is built upon the mechanism of

downstream wireless connections. Specifically, we present a fine-grained formulation of the connection error rates in a wireless communication scenario:

Lossy Wireless Connections: Following the wireless model of [141], we can leverage LTE-A [78], which is a representative model of 4G network, for the wireless links between the edge server (ES) and edge devices for FL, considering the orthogonal frequency division multiple access (OFDMA) scheme. The parameter $d_{es,k}$ denotes the distance between the edge server and the k^{th} edge device while the path loss between them can be characterized by $d_{es,k}^{-\sigma}$ and the white Gaussian noise power N_0, where σ is the path loss exponent. The wireless channel is modeled as a frequency-flat block-fading Rayleigh fading channel, with the uplink channel fading coefficient h [176]. The uplink data rate of the k^{th} edge device is defined as:

$$R_k = B_k log_2 \left(1 + \frac{P_t d_{es,k}^{-\sigma} |h^2|}{N_0 + I}\right). \tag{C.1}$$

In the equation above, B_k denotes the channel bandwidth, P_t represents the transmission power of the ES. I is the inter-cell interference. As R_k fluctuates based on changing wireless network conditions, we can define a minimum uplink rate R_{min}, such that any rate lower than R_{min} will lead to timeouts and packet loss. As a result, **the probability that a packet gets lost over the edge network** shall be derived as the following:

$$\begin{aligned} Pr\{R_k < R_{min}\} &= Pr\left\{B_k log_2\left(1 + \frac{P_t d_{es,k}^{-\sigma} |h^2|}{N_0 + I}\right) < R_{min}\right\} \\ &= Pr\left\{d^{\sigma} > \frac{P_t |h^2|}{(N_0 + I)\, 2^{\frac{R_{min}}{B_k}} - N_0 - I}\right\} \\ &= Pr\left\{d > \left(\frac{P_t |h^2|}{(N_0 + I)\, 2^{\frac{R_{min}}{B_k}} - N_0 - I}\right)^{\frac{1}{\sigma}}\right\}. \end{aligned} \tag{C.2}$$

Suppose in a given LTE-A network, B_k, N_0, I, P_t, σ, and h are constants. If the distances between the edge devices and the ES, *i.e.*, $d_{es,k}$, follow the *Poisson* distribution, then the probability of a packet to be lost during transmission can be derived by Equation (C.2).

Macro-Level Simulation of Connection Uncertainty: When conducting experiments for unstable network connection scenarios, we use er, *i.e.* the connection loss rate for

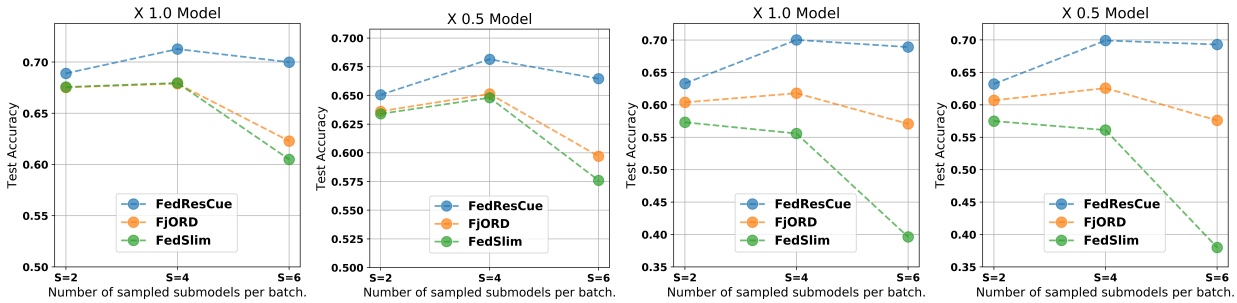
each *column*, as a macro-level modeling to approximate $Pr\{R_k < R_{min}\}$ in Equation (C.2). This type of modeling is grounded in that an upstream *column* and a downstream *packet* are logically related to each other. Depending on the size limit of a transmission packet and the granularity of our model decomposition, a column could be further decomposed into one or multiple packets during wireless transmission.

C.0.4.1 Ablation Study

Impacts of Sampling Frequency: We explored the effects of different sampling frequency S on model learning, where S is the number of submodels sampled per batch update (*e.g.*$|\hat{P}| = S$ in Algorithm 6.1). $S = 0$ would reduce all algorithms to regular *FedAvg* without self distillation. An overlarge S, on the other hand, may bring extra computation workload to edge devices. In our experiments, the sampling granularity is set to be $\times 0.05$ of the largest model width, and the smallest model width is set to be $\times 0.1$. Hence there are 19 eligible pruning ratios, *i.e.* $|\mathcal{P}| = 19$.

As shown in Figure C.3, *FedResCuE* can benefit from finer-grained submodel sampling and maintains a robust performance across different choices of S. On the contrary, the performance of *FjORD* is only slightly improved by increasing S from 2 to 4, whose performance drops notably when S becomes overlarge, especially given insufficient training data. *FedSlim* is the most sensitive to the choice of S, which learns prunable submodels purely by knowledge distillation and batch-gradient updates. Its performance degrades drastically with an increasing S and becomes highly unstable under connection interruptions. We ascribe the robustness of *FedResCuE* over others to its progressive learning scheme, rather than a batch-gradient update strategy as *FedSlim* and *FjORD* applies.

Effects of Knowledge Distillation: We compare *FedResCuE* against its ablated variant named *FedSeq* which does not require minimizing the KL-divergence between the largest model and the sampled submodel. We provide the model learning process of *FedSeq* in Algorithm C.3. Table C.8 summarizes their asymptotic performance under different FL



(a) 20% training data, ×1.0 model. (b) 20% training data, ×0.5 model. (c) $0.1 \leq er \leq 0.2$, ×1.0 model. (d) $0.1 \leq er \leq 0.2$, ×0.5 model.

Figure C.3: Impacts of the submodel sampling frequency.

settings, which demonstrates that *FedResCuE* is more beneficial to smaller submodels. We present the corresponding evaluation curves in Figure C.4.

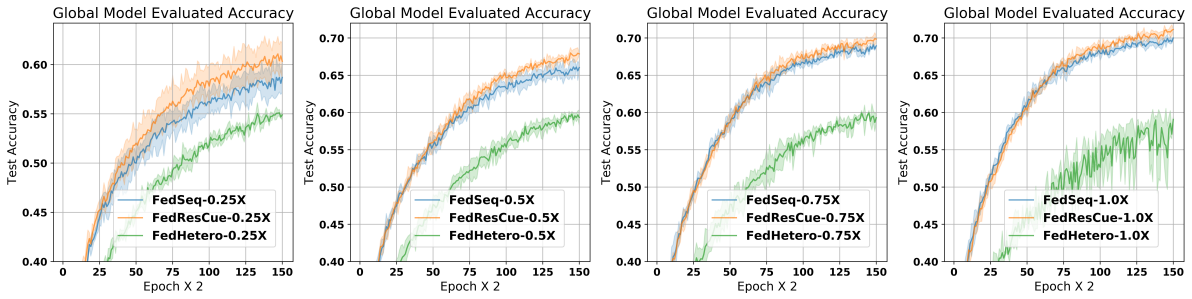
Algorithm C.3: Local Model Update for *FedSeq*

Inputs: Training dataset $D \subset \mathcal{X} \times \mathcal{Y}$; model with parameter set $\boldsymbol{\theta} \in \Theta$; pruning ratios $\mathcal{P}$, learning rate η, loss function $\mathcal{L}$, constant $S \leq |\mathcal{P}|$.
repeat
  Sample batch of $x, y \sim D$.
  Sample *ordered* ratios $\hat{P} = [p_i | p_i \in \mathcal{P}, p_i < p_{i+1}\ \forall i < S, p_S = 1]_{i=1}^{S}$
  for $p_i \sim \hat{P}$ **do**
    $g_i \leftarrow \nabla_{\{\boldsymbol{\theta}_{\times p_i} \setminus \boldsymbol{\theta}_{\times p_{i-1}}\}} \mathcal{L}(f(\mathcal{X}; \boldsymbol{\theta}_{\times p_i}), \mathcal{Y})$
    $\boldsymbol{\theta}_{\times p_i} \leftarrow \boldsymbol{\theta}_{\times p_i} - \eta * g_i$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta * \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(x; \boldsymbol{\theta}), y)$.
until training stop
Return $\boldsymbol{\theta}$

Comparing *FedResCuE* and *FedSeq*, on CelebA dataset				
Algorithm	$\boldsymbol{w}_{\times 0.25}$	$\boldsymbol{w}_{\times 0.5}$	$\boldsymbol{w}_{\times 0.75}$	$\boldsymbol{w}_{\times 1.0}$
	Given 20 % training data			
FedResCuE	61.12±1.35	68.15±0.39	69.98±0.41	71.27±0.27
FedSeq	58.96±1.24	66.39±0.60	69.18±0.28	70.13±0.16
	$0.1 \leq er \leq 0.2$			
FedResCuE	67.40±0.84.	69.91±0.42.	70.23±0.42.	70.02±0.40
FedSeq	66.47±0.36	69.45±0.41	69.70±0.37	69.67±0.32

Table C.8: Performance with and without knowledge-distillation.

Effects of Progressive Learning: One contributing factor to *FedResCuE*'s superior performance is its progressive learning strategy, which adaptively learns gradients by fixing the



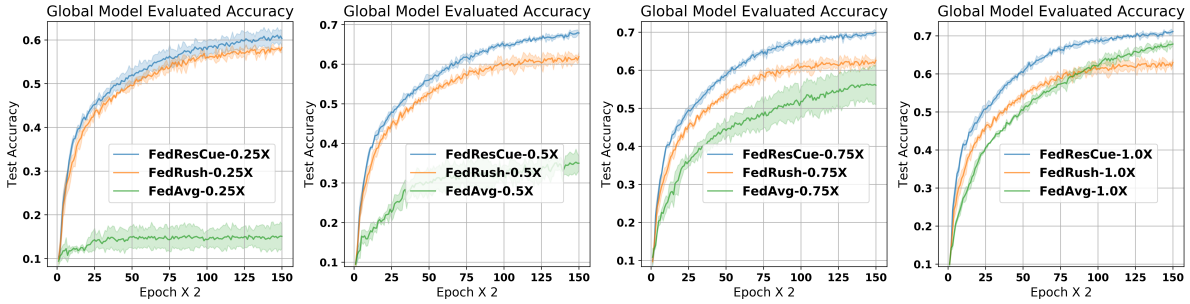
(a) $\times_{0.25}$ model. (b) $\times_{0.5}$ model. (c) $\times_{0.75}$ model. (d) $\times_{1.0}$ model.

Figure C.4: Compared with *FedSeq*, knowledge-distillation strategy in *FedResCuE* can benefit smaller submodels.

parameters of previously learned submodels. To validate the efficacy of progressive learning, we compared *FedResCuE* against a variant called *FedRush*.The detailed model learning process of *FedRush* is provided in Algorithm C.4. Learning curves of these two algorithms are illustrated in Figure C.5, which demonstrate that the progressive parameter update, as *FedResCuE* adopts, is necessary to derive reliable models, especially given insufficient training data or connection interruptions. *FedRush*, on the other hand, undermines the knowledge learned by smaller submodels by overwriting parameters during the model update, which could otherwise serve as a good initialization for the subsequent submodel to build up representative domain knowledge.

Algorithm C.4: Local Model Update for *FedRush*

Inputs: training dataset $D \subset \mathcal{X} \times \mathcal{Y}$; model with parameter set $\boldsymbol{\theta}$; pruning ratios $\mathcal{P}$, learning rate η, loss function $\mathcal{L}$, constant $S \leq |\mathcal{P}|$.
repeat
  Sample batch $x, y \sim D$.
  Sample *ordered* ratios $\hat{P} = [p_i | p_i \in \mathcal{P}, p_i < p_{i+1}\ \forall i < S, p_S = 1]_{i=1}^{S}$
  $\bar{\boldsymbol{\theta}} \leftarrow \text{stop\_gradient}(\boldsymbol{\theta})$, $\boldsymbol{\theta}_{\times 0} = \emptyset$.
  for $p_i \sim \hat{P}$ **do**
    $g_i \leftarrow \nabla_{\boldsymbol{\theta}_{\times p_i}} \left\{ \mathcal{L}(f(\mathcal{X}; \boldsymbol{\theta}_{\times p_i}), \mathcal{Y}) + \mathbb{D}_{\text{KL}}[f(x; \bar{\boldsymbol{\theta}}) \| f(x; \boldsymbol{\theta}_{\times p_i})] \right\}$.
    $\boldsymbol{\theta}_{\times p_i} \leftarrow \boldsymbol{\theta}_{\times p_i} - \eta * g_i$. ▷ (Overwrite previously learned $\boldsymbol{\theta}_{p_{i-1}}$.)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta * \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(x; \boldsymbol{\theta}), y)$
until training stop



(a) $\times_{0.25}$ model. (b) $\times_{0.5}$ model. (c) $\times_{0.75}$ model. (d) $\times_{1.0}$ model.

Figure C.5: Compared to *FedRush*, *FedResCuE* maintains a high performance for the $\times 1.0$ model.

C.0.5 Overview of Model Learning Performance

Performance with Stable Network Connections:

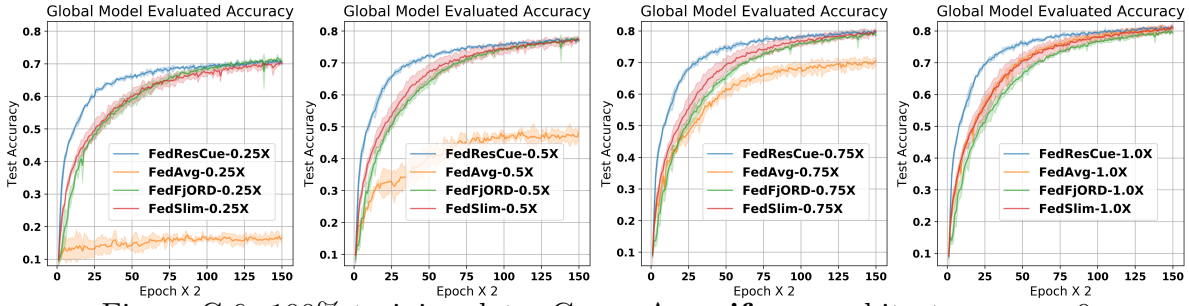
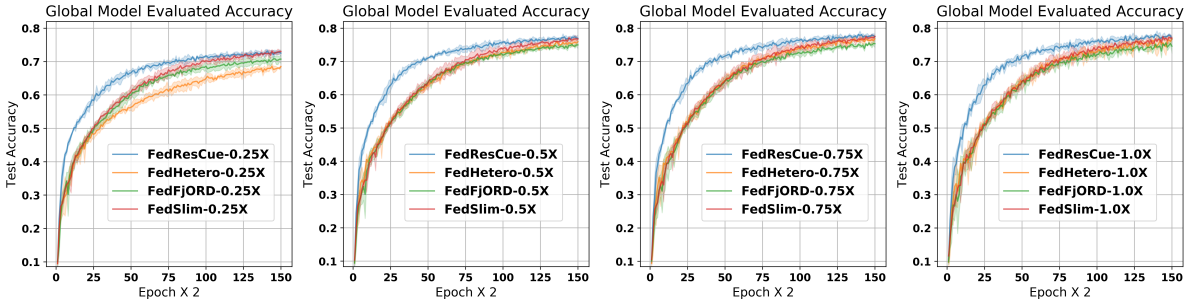


Figure C.6: 100% training data, CELEBA, **uniform** architecture, $er = 0$.



(a) $\times 0.25$ model (b) $\times 0.5$ model (c) $\times 0.75$ model (d) $\times 1.0$ model

Figure C.7: 100% training data, CELEBA, cluster architecture, $er = 0$.

Performance Given Insufficient Training Data:

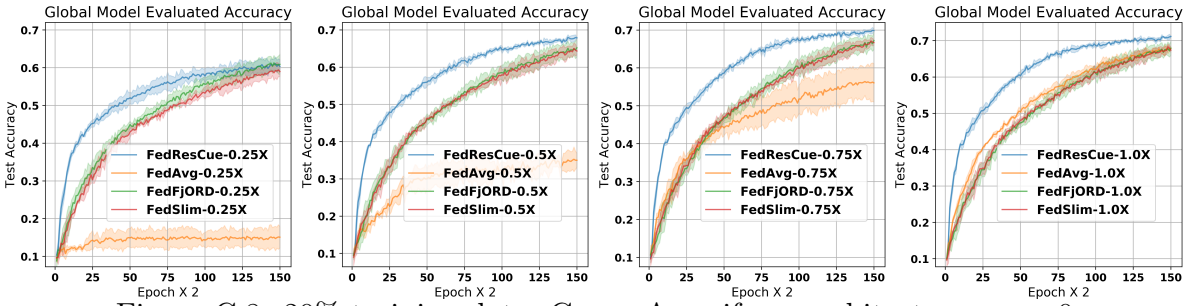
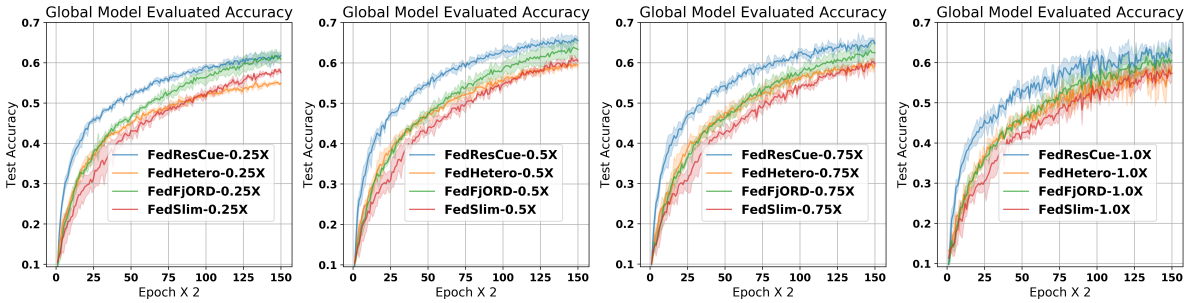


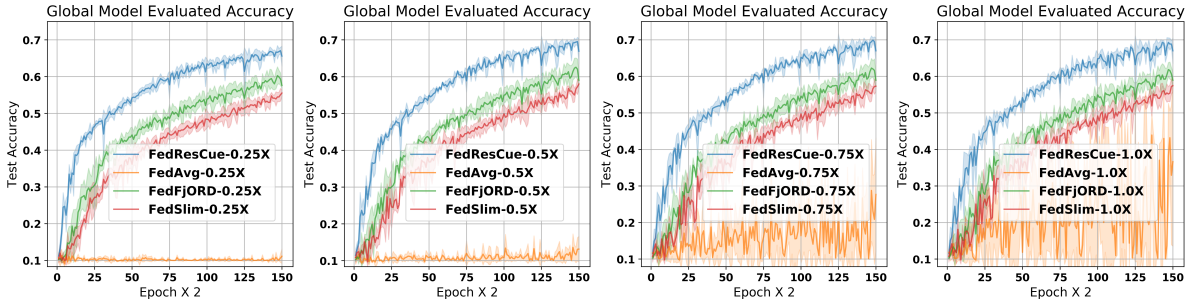
Figure C.8: 20% training data, CelebA, uniform architecture, $er = 0$.



(a) $\times 0.25$ model (b) $\times 0.5$ model (c) $\times 0.75$ model (d) $\times 1.0$ model

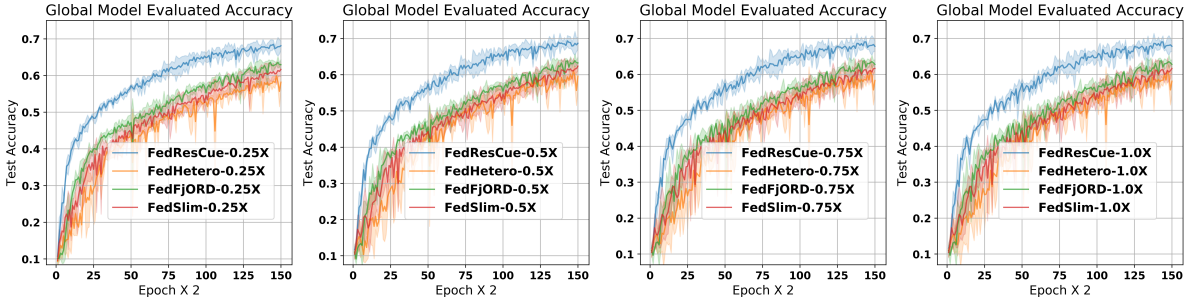
Figure C.9: 20% training data, CelebA, cluster architecture, $er = 0$.

Performance Under Connection Loss:



(a) ×0.25 model (b) ×0.5 model (c) ×0.75 model (d) ×1.0 model

Figure C.10: 100% training data, CelebA, uniform architecture, $0.1 \leq er \leq 0.2$.



(a) ×0.25 model (b) ×0.5 model (c) ×0.75 model (d) ×1.0 model

Figure C.11: 100% training data, CelebA, cluster architecture, $0.1 \leq er \leq 0.2$.

C.0.6 Experiment Configurations

C.0.6.1 Model Architecture

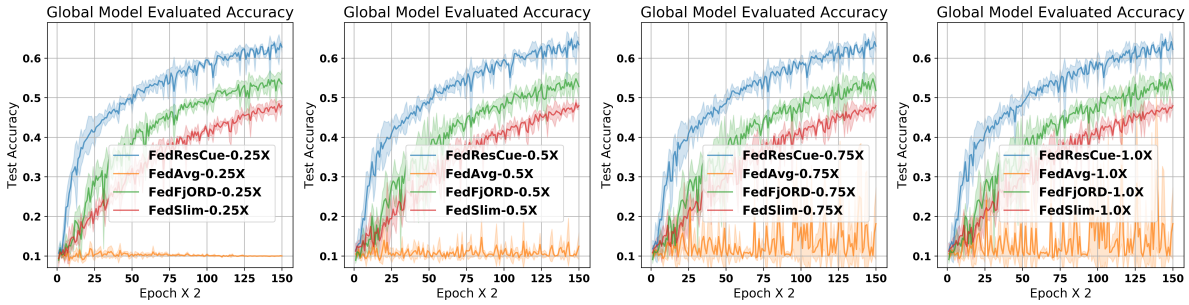
For the CelebA domain, we build a ResNet neural network [65] using 4 *residual blocks*, while a residual block maintains 1) a convolution module that consits of 3 Conv2d layers, each followed by a BatchNorm2d layer and a ReLU activation layer; and 2) a *shortpath* module to be in parallel with the convolution module. The ResNet model contains 8,036,426 trainable parameters in total, with a model size of 33.09 MB. For the DigitsFive domains, we build a neural network consiting of 3 Conv2d layers, each followed by a BatchNorm layer, and 3 Linear layers. It contains 14,214,090 trainable parameters in total, with a model size of 55.30 MB.

Network Architecture for Learning **CelebA**.		
Layer	Output Shape	# of Parameters
Conv2d-1	$64 \times 14 \times 14$	9,408
MaxPool2d-4	$64 \times 7 \times 7$	0
Conv2d-5	$64 \times 7 \times 7$	4,096
Conv2d-8	$64 \times 7 \times 7$	36,864
Conv2d-11	$256 \times 7 \times 7$	16,384
Conv2d-13	$256 \times 7 \times 7$	16,384
Conv2d-17	$128 \times 7 \times 7$	32,768
Conv2d-20	$128 \times 4 \times 4$	147,456
Conv2d-23	$512 \times 4 \times 4$	65,536
Conv2d-25	$512 \times 4 \times 4$	131, 072
Conv2d-29	$256 \times 4 \times 4$	131,072
Conv2d-32	$256 \times 2 \times 2$	589,824
Conv2d-35	$1024 \times 2 \times 2$	262,144
Conv2d-37	$1024 \times 2 \times 2$	524,288
Conv2d-41	$512 \times 2, 2$	524,288
Conv2d-44	$512 \times 1 \times 1$	2,359,296
Conv2d-47	$2048 \times 1 \times 1$	1,048,576
Conv2d-49	$2048 \times 1 \times 1$	2,097,152
AvgPool2d-53	$2048 \times 1 \times 1$	0
Linear-54	10	20,490

Table C.9: ResNet Architecture for Learning CelebA, omitting BatchNorm and ReLU layers.

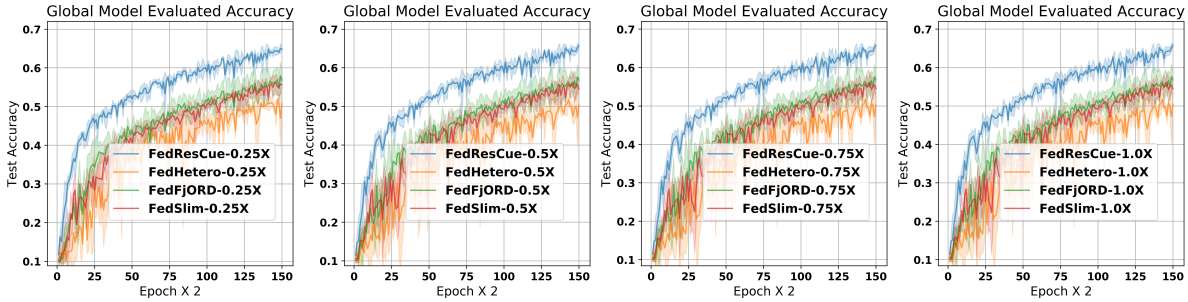
Network Architecture for Learning **DigitsFive**.		
Layer	Output Shape	# of Parameters
Conv2d-1	$64 \times 28 \times 28$	4,864
BatchNorm2d-2	$64 \times 28 \times 28$	128
Conv2d-3	$64 \times 14 \times 14$	102,464
BatchNorm2d-4	$64 \times 14 \times 14$	128
Conv2d-5	$128 \times 7 \times 7$	204,928
BatchNorm2d-6	$128 \times 7 \times 7$	256
Linear-7	2048	12,847,104
Linear-8	512	1,049,088
Linear-9	10	5,130

Table C.10: Model Architecture for Learning DigitsFive.



(a) ×0.25 model (b) ×0.5 model (c) ×0.75 model (d) ×1.0 model

Figure C.12: 100% training data, CelebA, uniform architecture, $0.2 \leq er \leq 0.3$.



(a) ×0.25 model (b) ×0.5 model (c) ×0.75 model (d) ×1.0 model

Figure C.13: 100% training data, CelebA, cluster architecture, $0.2 \leq er \leq 0.3$.

C.0.6.2 Optimizer Implementation for Progressive Learning

In practice, we customzie the default SGD optimizer implemented in Pytorch to realize progressive gradient updates. A trainable neural layer consits of parameter tensors, each of which can be treated as a weight matrix. When calculating gradients for the neural layer, we specify the *columns* to be updated, which corresponds to a set of indices for elements in the weight matrix. When applying the gradients, we *mask* out the gradients of parameters that were not included in the specified columns.

C.0.6.3 Hyper-parameter Configurations

We summarize in Table C.11 the hyper-parameters used in our experiments.

Hyper-parameter Configurations		
Domain	Hyper-parameter	Value
Shared	Optimizer	SGD
	learning rate	0.1
	Momentum	0.9
	Nesterov	TRUE
	Weight decay	10^{-4}
	Track training in BatchNorm	FALSE
	Share BatchNorm	TRUE
	Data category	10
	# of active users	5
	Random seeds for training	3, 5, 7
	Batch Size	32
CELEBA	Training Epoch	300
	# of total users	20
	Used training data	100%, 20%
	Column Granularity for $\mathcal{P}$	$\times 0.05$
DIGITSFIVE	Training Epoch	100
	# of total users	10
	# users per domain	2
	Used training data	5%
	Column Granularity for $\mathcal{P}$	$\times 0.125$

Table C.11: Configurations of Hyper-parameters.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H Brendan Mcmahan. cpsgd: Communication-efficient and differentially-private distributed sgd. *Advances in Neural Information Processing Systems*, 2018.

[3] Divyansh Aggarwal, Jiayu Zhou, and Anil K Jain. Fedface: Collaborative learning of face recognition model. *Proceedings of International Joint Conference on Biometrics*, 2021.

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 2017.

[5] Mohammad Mohammadi Amiri, Deniz Gunduz, Sanjeev R Kulkarni, and H Vincent Poor. Federated learning with quantized global model updates. *arXiv preprint arXiv:2006.10672*, 2020.

[6] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

[7] Haitham Bou Ammar and Matthew E. Taylor. Reinforcement learning transfer via common subspaces. In *Proceedings of the 11th International Conference on Adaptive and Learning Agents*, ALA'11, pages 21–36, Berlin, Heidelberg, 2012. Springer-Verlag.

[8] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[10] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2930–2941, 2018.

[11] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27:2654–2662, 2014.

[12] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[13] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016.

[14] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[15] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[16] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[17] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[18] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136, 2008.

[19] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128, 2006.

[20] Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

[21] Daniel S Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. Safe imitation learning via fast bayesian reward inference from preferences. *ICML*, 2020.

[22] Daniel S Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. *arXiv preprint arXiv:1904.06387*, 2019.

[23] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, pages 330–359, 2020.

[24] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

[25] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[26] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[27] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *ICLR*, 2019.

[28] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2020.

[29] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[30] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *ICLR*, 2021.

[31] Mingzhe Chen, Zhaohui Yang, Walid Saad, Changchuan Yin, H Vincent Poor, and Shuguang Cui. Performance optimization of federated learning over wireless networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[32] Mingzhe Chen, Zhaohui Yang, Walid Saad, Changchuan Yin, H Vincent Poor, and Shuguang Cui. A joint learning and communications framework for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, 20(1):269–283, 2020.

[33] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.

[34] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[35] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[36] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[38] Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 433–440. IFAAMAS, 2012.

[39] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

[40] Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[41] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

[42] Zhaoyang Du, Celimuge Wu, Tsutomu Yoshinaga, Kok-Lim Alvin Yau, Yusheng Ji, and Jie Li. Federated learning for vehicular internet of things: Recent advances and open issues. *IEEE Open Journal of the Computer Society*, 1:45–61, 2020.

[43] John C Duchi, Michael I Jordan, and Martin J Wainwright. Privacy aware learning. *Journal of the ACM (JACM)*, 61(6):1–57, 2014.

[44] Ashley D Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L Isbell. Imitating latent policies from observation. *ICML*, 2019.

[45] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33, 2020.

[46] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[47] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[48] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[49] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *ICLR*, 2017.

[50] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[51] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2):123–156, 2012.

[52] Tanmay Gangwani and Jian Peng. State-only imitation with transition dynamics mismatch. *ICLR*, 2020.

[53] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[54] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[55] Seyed Kamyar Seyed Ghasemipour, Shane Gu, and Richard Zemel. Understanding the relation between maximum-entropy inverse reinforcement learning and behaviour cloning. *ICLR 2019 Workshop*, 2019.

[56] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[57] Xinran Gu, Kaixuan Huang, Jingzhao Zhang, and Longbo Huang. Fast federated learning in the presence of arbitrary device unavailability. *35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[58] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[59] Yijie Guo, Junhyuk Oh, Satinder Singh, and Honglak Lee. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950*, 2018.

[60] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017.

[61] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[62] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*, 2016.

[63] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[64] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33, 2020.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[66] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[67] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[68] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[69] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos I Venieris, and Nicholas D Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *35th Conference on Neural Information Processing Systems (NeurIPS).*, 2021.

[70] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[71] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[72] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[73] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed sgd with sketching. *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.

[74] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[75] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[76] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271, 2007.

[77] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[78] Sravanthi Kanchi, Shubhrika Sandilya, Deesha Bhosale, Adwait Pitkar, and Mayur Gondhalekar. Overview of lte-a technology. In *2013 IEEE global high tech congress on electronics*, pages 195–200. IEEE, 2013.

[79] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2474–2483, 2018.

[80] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[81] Latif U Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 2021.

[82] Rinat Khoussainov, Andreas Heß, and Nicholas Kushmerick. Ensembles of biased classifiers. In *Proceedings of the 22nd international conference on Machine learning*, pages 425–432, 2005.

[83] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[84] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *arXiv preprint arXiv:2002.00444*, 2020.

[85] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[86] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[87] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *ICLR*, 2019.

[88] Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *ICLR*, 2020.

[89] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[90] Andras Kupcsik, David Hsu, and Wee Sun Lee. Learning dynamic robot-to-human object handover from human feedback. In *Robotics research*, pages 161–176. Springer, 2018.

[91] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[92] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[93] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[94] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[95] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[96] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[97] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[98] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[99] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *ICLR*, 2020.

[100] Xiaoxiao et al. Li. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

[101] Xuejun Liao, Ya Xue, and Lawrence Carin. Logistic regression with an auxiliary data source. In *Proceedings of the 22nd international conference on Machine learning*, pages 505–512, 2005.

[102] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

[103] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *arXiv preprint arXiv:2006.07242*, 2020.

[104] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[105] Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *ICLR*, 2019.

[106] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *AAAI*, 2020.

[107] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018.

[108] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *ICLR*, 2019.

[109] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[110] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

[111] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *ICLR*, 2018.

[112] Xiaojian Ma, Mingxuan Jing, Wenbing Huang, Fuchun Sun, Chao Yang, Bin Fang, and Huaping Liu. Reinforcement learning from imperfect demonstrations under soft expert guidance.

[113] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[114] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1429–1437, 2019.

[115] Roger McFarlane. A survey of exploration strategies in reinforcement learning. *McGill University, http://www. cs. mcgill. ca/cs526/roger. pdf, accessed: April*, 2018.

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

[117] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems*, pages 9551–9561, 2019.

[118] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 593–600, 2005.

[119] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[120] Jaemin Na, Heechul Jung, Hyung Jin Chang, and Wonjun Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1094–1103, 2021.

[121] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pages 2315–2325, 2019.

[122] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[123] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[124] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[125] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[126] Khoa Nguyen, Steve Drew, Changcheng Huang, and Jiayu Zhou. Edgepv: collaborative edge computing framework for task offloading. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[127] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[128] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[129] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[130] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. *arXiv preprint arXiv:1806.05635*, 2018.

[131] Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions by integrating human demonstrations and preferences. *arXiv preprint arXiv:1906.08928*, 2019.

[132] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[133] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[134] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[135] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[136] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *ICLR*, 2019.

[137] Huy X Pham, Hung M La, David Feil-Seifer, and Luan V Nguyen. Autonomous uav navigation using reinforcement learning. *arXiv preprint arXiv:1801.05086*, 2018.

[138] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[139] Zhaonan Qu, Kaixiang Lin, Jayant Kalagnanam, Zhaojian Li, Jiayu Zhou, and Zhengyuan Zhou. Federated learning's blessing: Fedavg has linear speedup. *arXiv preprint arXiv:2007.05690*, 2020.

[140] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11893–11902, 2020.

[141] Salman Raza, Wei Liu, Manzoor Ahmed, Muhammad Rizwan Anwar, Muhammad Ayzed Mirza, Qibo Sun, and Shangguang Wang. An efficient task offloading scheme in vehicular edge computing. *Journal of Cloud Computing*, 9:1–14, 2020.

[142] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

[143] Amirhossein Reisizadeh, Isidoros Tziotis, Hamed Hassani, Aryan Mokhtari, and Ramtin Pedarsani. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. *arXiv preprint arXiv:2012.14453*, 2020.

[144] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

[145] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[146] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[147] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

[148] Fumihiro Sasaki, Tetsuya Yohira, and Atsuo Kawaguchi. Sample efficient imitation learning for continuous control. *ICLR*, 2019.

[149] Fumihiro Sasaki, Tetsuya Yohira, and Atsuo Kawaguchi. Sample efficient imitation learning for continuous control. *ICLR*, 2019.

[150] Felix et al. Sattler. Fedaux: Leveraging unlabeled auxiliary data in federated learning. *arXiv preprint arXiv:2102.02514*, 2021.

[151] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[152] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *ICLR*, 2018.

[153] Hyowoon Seo, Jihong Park, Seungeun Oh, Mehdi Bennis, and Seong-Lyun Kim. Federated knowledge distillation. *arXiv preprint arXiv:2011.02367*, 2020.

[154] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):1–12, 2020.

[155] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

[156] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[157] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *ICLR*, 2017.

[158] Benyuan Sun, Hongxing Huo, Yi Yang, and Bo Bai. Partialfed: Cross-domain personalized federated learning via partial initialization. *Advances in Neural Information Processing Systems*, 34, 2021.

[159] Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. *arXiv preprint arXiv:2009.05537*, 2020.

[160] Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *ICLR*, 2018.

[161] Wen Sun, Anirudh Vemula, Byron Boots, and J Andrew Bagnell. Provably efficient imitation learning from observation alone. *arXiv preprint arXiv:1905.10948*, 2019.

[162] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3309–3318. JMLR. org, 2017.

[163] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[164] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. pages 45–47. MIT press, 2018.

[165] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pages 1449–1456, 2008.

[166] Akihito Taya, Takayuki Nishio, Masahiro Morikura, and Koji Yamamoto. Decentralized and model-free federated learning: Consensus-based distillation in function space. *arXiv preprint arXiv:2104.00352*, 2021.

[167] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[168] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

[169] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 2018.

[170] Faraz Torabi, Garrett Warnell, and Peter Stone. Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2229–2231. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[171] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *ICML Workshop on Imitation, Intent, and Interaction (I3)*, 2019.

[172] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *IJICAI*, 2019.

[173] Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.

[174] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[175] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[176] Hansong Wang, Xi Li, Hong Ji, and Heli Zhang. Federated offloading scheme to minimize latency in mec-enabled vehicular networks. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2018.

[177] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *arXiv preprint arXiv:2007.05084*, 2020.

[178] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *ICLR*, 2020.

[179] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[180] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[181] Paul Weng. Markov decision processes with ordinal rewards: Reference point-based preferences. In *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

[182] Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 792–799, 2003.

[183] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, 18(1):4945–4990, 2017.

[184] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[185] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. *ICML*, 2019.

[186] Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

[187] Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269*, 2021.

[188] Yikai Yan, Chaoyue Niu, Yucheng Ding, Zhenzhe Zheng, Fan Wu, Guihai Chen, Shaojie Tang, and Zhihua Wu. Distributed non-convex optimization with sublinear speedup under intermittent client availability. *arXiv preprint arXiv:2002.07399*, 2020.

[189] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In *Advances in Neural Information Processing Systems*, pages 239–249, 2019.

[190] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018.

[191] Haiyan Yin and Sinno Jialin Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[192] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. Gat: Generative adversarial training for adversarial example detection and robust classification. In *International Conference on Learning Representations*, 2019.

[193] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In *Advances in Neural Information Processing Systems*, pages 2705–2714, 2019.

[194] Tehrim et al. Yoon. Fedmix: Approximation of mixup under mean augmented federated learning. *ICLR*, 2021.

[195] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1803–1811, 2019.

[196] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[197] Xiao-Tong Yuan and Ping Li. On convergence of distributed approximate newton methods: Globalization, sharper bounds and beyond. *arXiv preprint arXiv:1908.02246*, 2019.

[198] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.

[199] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[200] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019.

[201] Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. *International Conference on Learning Representations (ICLR)*, 2020.

[202] Xiaoqin Zhang, Yunfei Li, Huimin Ma, and Xiong Luo. Pretrain soft q-learning with imperfect demonstrations. *arXiv preprint arXiv:1905.03501*, 2019.

[203] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. *Advances in neural information processing systems*, 31:8559–8570, 2018.

[204] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[205] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. *International Conference on Machine Learning*, 2021.

[206] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

[207] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[208] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. 2010.

[209] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.