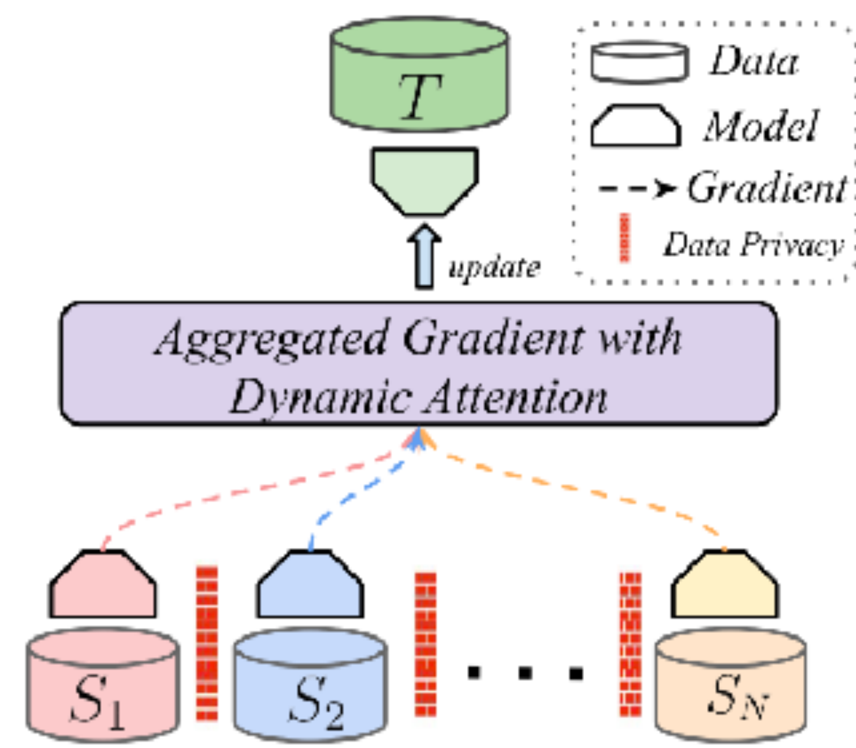
Federated Adversarial Domain Adaptation

Motivation

- FL challenges:
- Data stored locally
- Model parameters trained separately
- Knowledge is highly entangled
- Domain shift



Theory

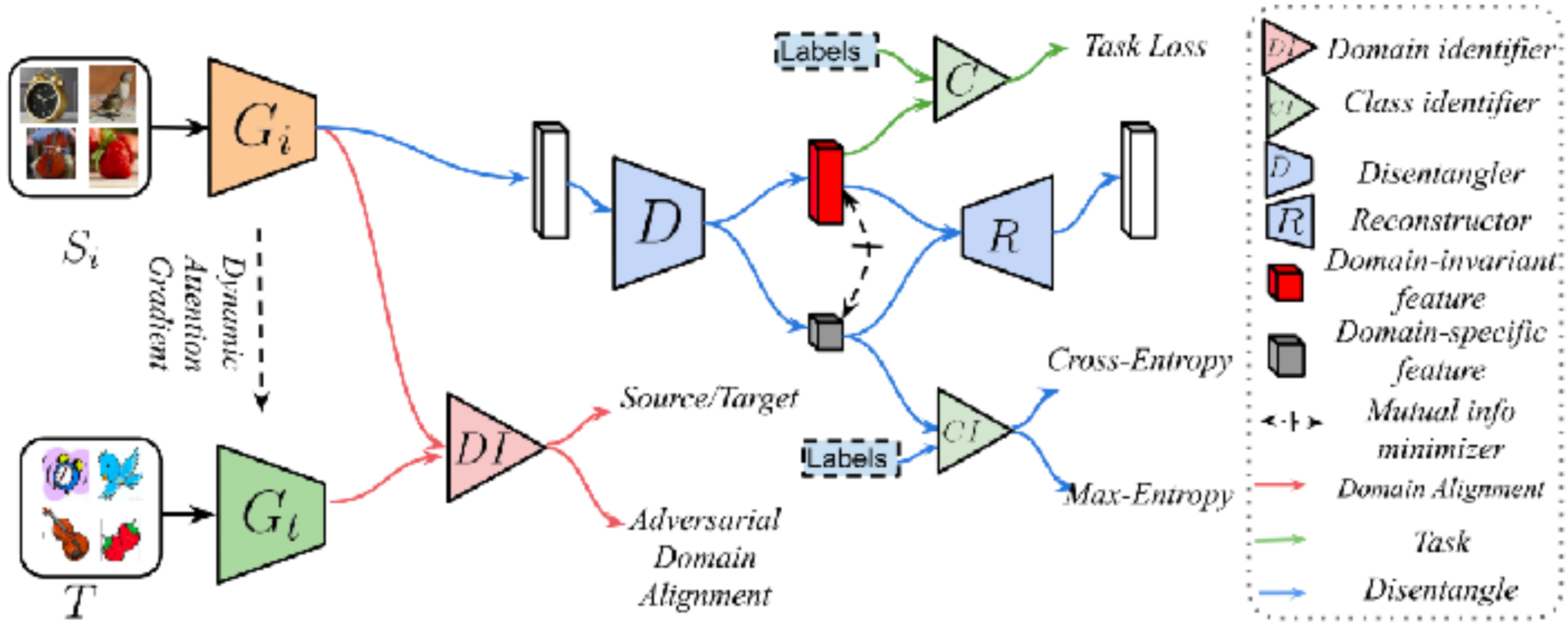
- Error bound single-source domain adaptation

$$\epsilon_T(h) \leq \widehat{\epsilon}_S(h) + \frac{1}{2}\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T) + 4\sqrt{\frac{2d\log(2m) + \log(4/\delta)}{m}} + \lambda$$

- Error bound unsupervised federated domain adaptation

$$\epsilon_T(h_T) \leq \underbrace{\widehat{\epsilon}_{\tilde{S}}(\sum_{i\in[N]} \alpha_i h_{S_i})}_{\text{error on source}} + \sum_{i\in[N]} \alpha_i\Big(\frac{1}{2}\underbrace{\widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{S_i}, \widehat{\mathcal{D}}_T)}_{(\mathcal{D}_{\tilde{S}_i}, \mathcal{D}_T)\text{ divergence}} + \lambda_i\Big) + \underbrace{4\sqrt{\frac{2d\log(2Nm) + \log(4/\delta)}{Nm}}}_{\text{VC-Dimension Constraint}}$$

Module



S_i
T
G_i
G_t
Dynamic Attention Gradient
D
R
C
CI
DI
Labels
Labels
Task Loss
Source/Target
Adversarial Domain Alignment
Cross-Entropy
Max-Entropy
Domain identifier
Class identifier
Disentangler
Reconstructor
Domain-invariant feature
Domain-specific feature
Mutual info minimizer
Domain Alignment
Task
Disentangle

Algorithm

Algorithm 1 Federated Adversarial Domain Adaptation

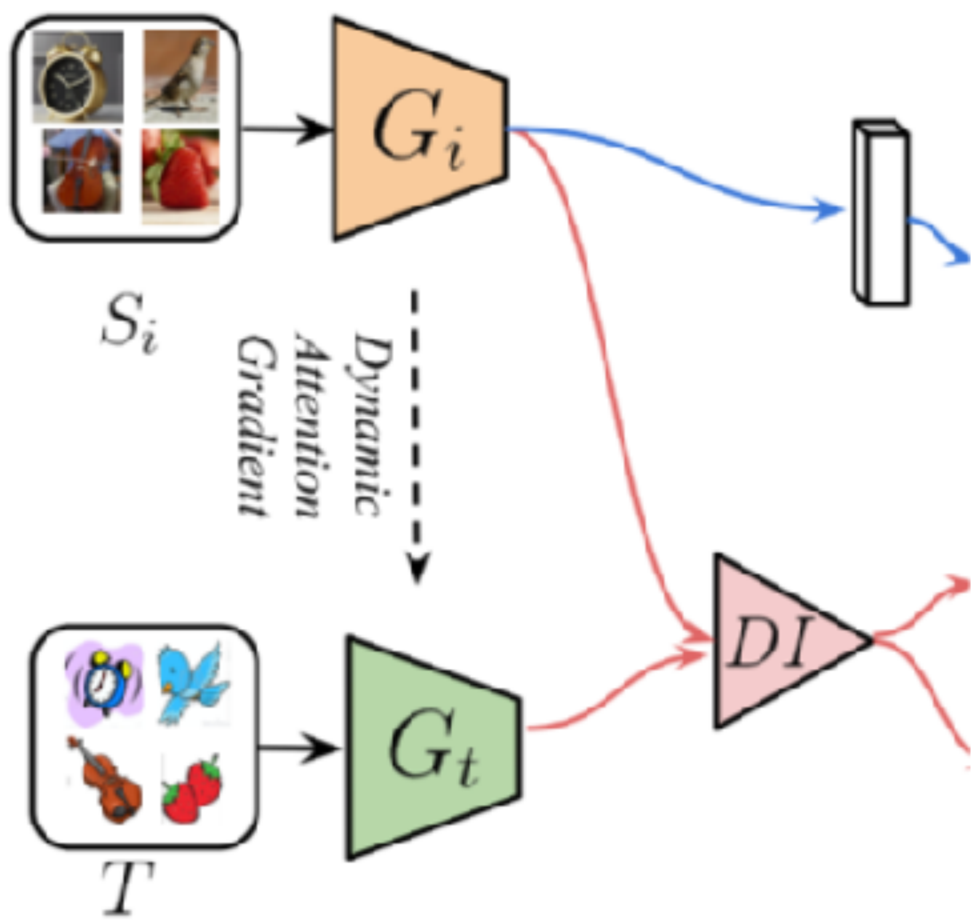
Input: N source domains $\mathcal{D}_S=\{\mathcal{D}_{S_i}\}_{i=1}^{N}$; a target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$; N feature extractors $\{\Theta^{G_1}, \Theta^{G_2}, ...\Theta^{G_N}\}$, N disentanglers $\{\Theta^{D_1}, \Theta^{D_2}, ...\Theta^{D_N}\}$, N classifiers $\{\Theta^{C_1}, \Theta^{C_2}, ...\Theta^{C_N}\}$, N class identifiers $\{\Theta^{CI_1}, \Theta^{CI_2}, ...\Theta^{CI_N}\}$, N mutual information estimators $\{\Theta^{M_1}, \Theta^{M_2}, ...\Theta^{M_N}\}$ trained on source domains. Target feature extractor Θ^{G_t}, classifier Θ^{C_t}. N domain identifiers $\{\Theta^{DI_1}, \Theta^{DI_2}, ..., \Theta^{DI_N}\}$

Output: well-trained target feature extractor $\hat{\Theta}^{G_t}$, target classifier $\hat{\Theta}^{C_t}$.

Model Initialization .

```
while not converged do
    for i do=1:N
        Sample mini-batch from from {(x_i^s, y_i^s)}_{i=1}^{n_s} and {x_j^t}_{j=1}^{n_t};
        Compute gradient with cross-entropy classification loss, update Θ^{G_i}, Θ^{C_i}.
        Domain Alignment:
        Update Θ^{DI_i}, {Θ^{G_i}, Θ^{G_t}} with Eq. 4 and Eq. 5 respectively to align the domain distribution;
        Domain Disentangle:
        update Θ^{G_i}, Θ^{D_i}, Θ^{C_i}, Θ^{CI_i} with Eq. 6
        update Θ^{D_i} and {Θ^{C_i}} with Eq. 7
        Mutual Information Minimization:
        Calculate mutual information between the disentangled feature pair (f_{di}, f_{ds}) with M_i;
        Update Θ^{D_i}, Θ^{M_i} by Eq.8;
    end for
    Dynamic weight:
    Calculate dynamic weight by Eq. 3
    Update Θ^{G_t}, Θ^{C_t} by aggregated {Θ^{G_1}, Θ^{G_2}, ..., Θ^{G_N}}, {Θ^{C_1}, Θ^{C_2}, ...Θ^{C_N}} respectively with the
    computed dynamic weight;
end while
return Θ^{G_t}, Θ^{C_t}
```

Federated Adversarial Alignment



$$\underset{\Theta^{DI_i}}{L_{adv_{DI_i}}}(\mathbf{X}^{S_i}, \mathbf{X}^T, G_i, G_t) = -\mathbb{E}_{\mathbf{x}^{s_i} \sim \mathbf{X}^{s_i}} \left[\log DI_i(G_i(\mathbf{x}^{s_i}))\right] - \mathbb{E}_{\mathbf{x}^t \sim \mathbf{X}^t}\left[\log(1 - DI_i(G_t(\mathbf{x}^t)))\right].$$

$$\underset{\Theta^{G_i}, \Theta^{G_t}}{L_{adv_G}}(\mathbf{X}^{S_i}, \mathbf{X}^T, DI_i) = -\mathbb{E}_{\mathbf{x}^{s_i} \sim \mathbf{X}^{s_i}}[\log DI_i(G_i(\mathbf{x}^{s_i}))] - \mathbb{E}_{\mathbf{x}^t \sim \mathbf{X}^t}[\log DI_i(G_t(\mathbf{x}^t))]$$

Algorithm

Algorithm 1 Federated Adversarial Domain Adaptation

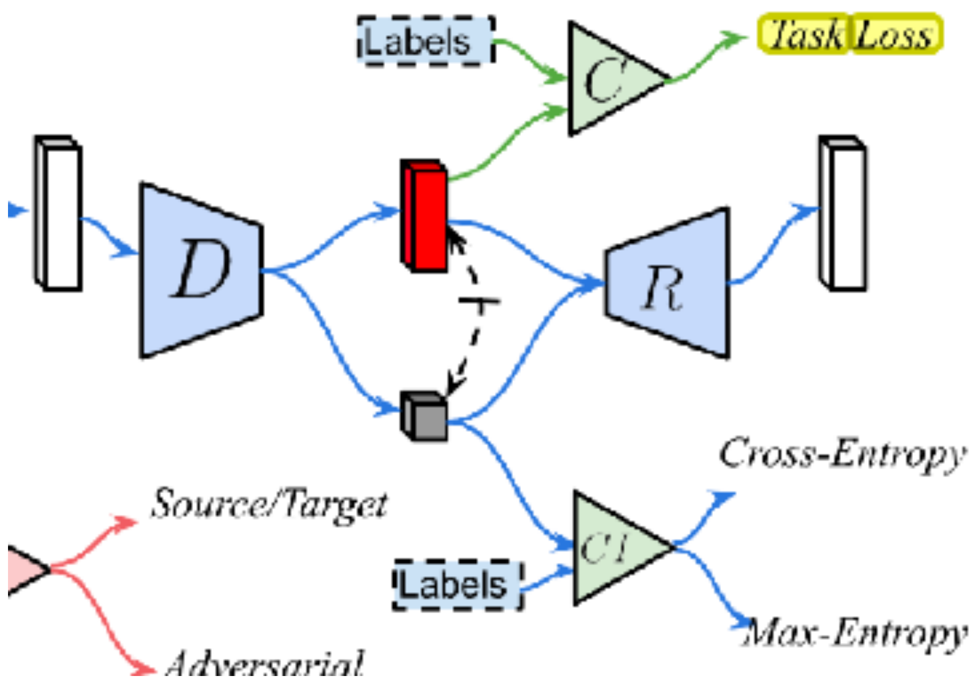
Input: N source domains $\mathcal{D}_S=\{\mathcal{D}_{S_i}\}_{i=1}^N$; a target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$; N feature extractors $\{\Theta^{G_1}, \Theta^{G_2}, ...\Theta^{G_N}\}$, N disentanglers $\{\Theta^{D_1}, \Theta^{D_2}, ...\Theta^{D_N}\}$, N classifiers $\{\Theta^{C_1}, \Theta^{C_2}, ...\Theta^{C_N}\}$, N class identifiers $\{\Theta^{CI_1}, \Theta^{CI_2}, ...\Theta^{CI_N}\}$, N mutual information estimators $\{\Theta^{M_1}, \Theta^{M_2}, ...\Theta^{M_N}\}$ trained on source domains. Target feature extractor Θ^{G_t}, classifier Θ^{C_t}. N domain identifiers $\{\Theta^{DI_1}, \Theta^{DI_2}, ..., \Theta^{DI_N}\}$

Output: well-trained target feature extractor $\hat{\Theta}^{G_t}$, target classifier $\hat{\Theta}^{C_t}$.

Model Initialization .

```
while not converged do
    for i do=1:N
        Sample mini-batch from from {(x_i^s, y_i^s)}_{i=1}^{n_s} and {x_j^t}_{j=1}^{n_t};
        Compute gradient with cross-entropy classification loss, update Θ^{G_i}, Θ^{C_i}.
        Domain Alignment:
        Update Θ^{DI_i}, {Θ^{G_i}, Θ^{G_t}} with Eq. 4 and Eq. 5 respectively to align the domain distribution;
        Domain Disentangle:
        update Θ^{G_i}, Θ^{D_i}, Θ^{C_i}, Θ^{CI_i} with Eq. 6
        update Θ^{D_i} and {Θ^{C_i}} with Eq. 7
        Mutual Information Minimization:
        Calculate mutual information between the disentangled feature pair (f_{di}, f_{ds}) with M_i;
        Update Θ^{D_i}, Θ^{M_i} by Eq.8;
    end for
    Dynamic weight:
    Calculate dynamic weight by Eq. 3
    Update Θ^{G_t}, Θ^{C_t} by aggregated {Θ^{G_1}, Θ^{G_2}, ..., Θ^{G_N}}, {Θ^{C_1}, Θ^{C_2}, ...Θ^{C_N}} respectively with the
computed dynamic weight;
end while
return Θ^{G_t}, Θ^{C_t}
```

Representation Disentanglement



$$\underset{\Theta^{G_i},\Theta^{D_i},\Theta^{C_i},\Theta^{CI_i}}{L_{cross-entropy}} = -\mathbb{E}_{(\mathbf{x}^{s_i},\mathbf{y}^{s_i})\sim\widehat{\mathcal{D}}_{s_i}} \sum_{k=1}^{K} \mathbb{1}[k = \mathbf{y}^{s_i}] log(C_i(f_{di})) - \mathbb{E}_{(\mathbf{x}^{s_i},\mathbf{y}^{s_i})\sim\widehat{\mathcal{D}}_{s_i}} \sum_{k=1}^{K} \mathbb{1}[k = \mathbf{y}^{s_i}] log(CI_i(f_{ds}))$$

$$\underset{\Theta^{D_i},\Theta^{G_i}}{L_{ent}} = -\frac{1}{N_{s_i}} \sum_{j=1}^{N_{s_i}} \log CI_i(f_{ds}^{j}) = -\frac{1}{N_{s_i}} \sum_{j=1}^{N_{s_i}} \log CI_i(D_i(G_i(\mathbf{x}^{s_i})))$$

Algorithm

Algorithm 1 Federated Adversarial Domain Adaptation

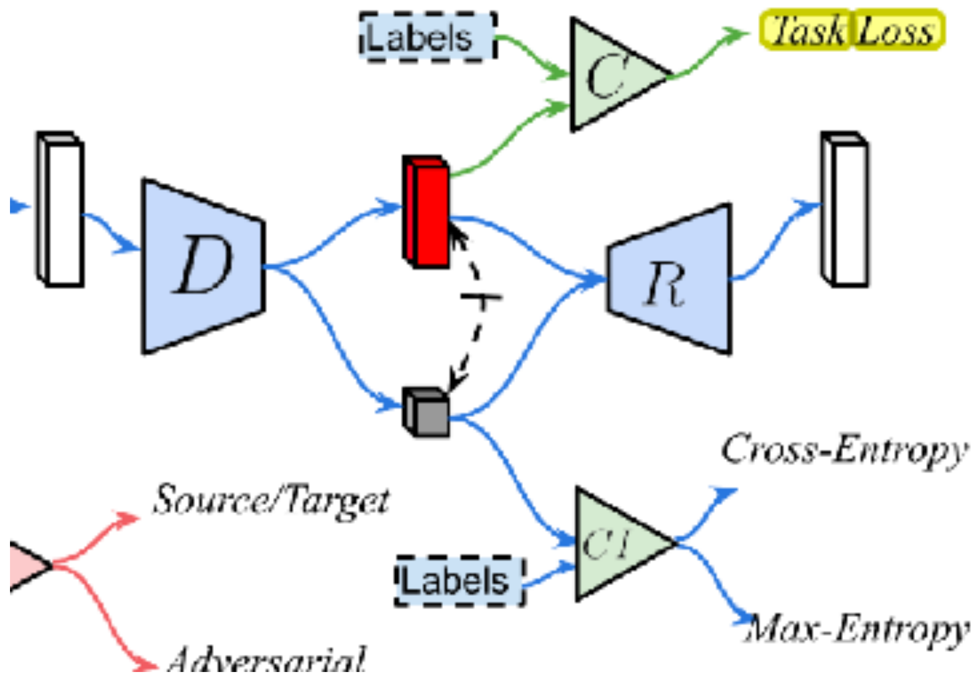
Input: N source domains $\mathcal{D}_S = \{\mathcal{D}_{S_i}\}_{i=1}^{N}$; a target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$; N feature extractors $\{\Theta^{G_1}, \Theta^{G_2}, ...\Theta^{G_N}\}$, N disentanglers $\{\Theta^{D_1}, \Theta^{D_2}, ...\Theta^{D_N}\}$, N classifiers $\{\Theta^{C_1}, \Theta^{C_2}, ...\Theta^{C_N}\}$, N class identifiers $\{\Theta^{CI_1}, \Theta^{CI_2}, ...\Theta^{CI_N}\}$, N mutual information estimators $\{\Theta^{M_1}, \Theta^{M_2}, ...\Theta^{M_N}\}$ trained on source domains. Target feature extractor Θ^{G_t}, classifier Θ^{C_t}. N domain identifiers $\{\Theta^{DI_1}, \Theta^{DI_2}, ..., \Theta^{DI_N}\}$

Output: well-trained target feature extractor $\hat{\Theta}^{G_t}$, target classifier $\hat{\Theta}^{C_t}$.

Model Initialization .

```
while not converged do
    for i do=1:N
        Sample mini-batch from from {(x_i^s, y_i^s)}_{i=1}^{n_s} and {x_j^t}_{j=1}^{n_t};
        Compute gradient with cross-entropy classification loss, update Θ^{G_i}, Θ^{C_i}.
        Domain Alignment:
        Update Θ^{DI_i}, {Θ^{G_i}, Θ^{G_t}} with Eq. 4 and Eq. 5 respectively to align the domain distribution;
        Domain Disentangle:
        update Θ^{G_i}, Θ^{D_i}, Θ^{C_i}, Θ^{CI_i} with Eq. 6
        update Θ^{D_i} and {Θ^{C_i}} with Eq. 7
        Mutual Information Minimization:
        Calculate mutual information between the disentangled feature pair (f_{di}, f_{ds}) with M_i;
        Update Θ^{D_i}, Θ^{M_i} by Eq.8;
    end for
    Dynamic weight:
    Calculate dynamic weight by Eq. 3
    Update Θ^{G_t}, Θ^{C_t} by aggregated {Θ^{G_1}, Θ^{G_2}, ..., Θ^{G_N}}, {Θ^{C_1}, Θ^{C_2}, ...Θ^{C_N}} respectively with the computed dynamic weight;
end while
return Θ^{G_t}, Θ^{C_t}
```

Mutual Information



$$T_\theta\colon\ \widehat{I(\mathcal{P};\mathcal{Q})}_n = \sup_{\theta\in\Theta} \mathbb{E}_{\mathbb{P}^{(n)}_{\mathcal{P}\mathcal{Q}}}[T_\theta] - \log(\mathbb{E}_{\mathbb{P}^{(n)}_{\mathcal{P}}\otimes\widehat{\mathbb{P}}^{(n)}_{\mathcal{Q}}}[e^{T_\theta}])$$

$$I(\mathcal{P};\mathcal{Q}) = \int\int \mathbb{P}^n_{\mathcal{P}\mathcal{Q}}(p,q)\, T(p,q,\theta) - \log(\int\int \mathbb{P}^n_{\mathcal{P}}(p)\mathbb{P}^n_{\mathcal{Q}}(q)\, e^{T(p,q,\theta)})$$

$$I(\mathcal{P},\mathcal{Q}) = \frac{1}{n}\sum_{i=1}^{n} T(p,q,\theta) - \log(\frac{1}{n}\sum_{i=1}^{n} e^{T(p,q',\theta)})$$

Algorithm

Algorithm 1 Federated Adversarial Domain Adaptation

Input: N source domains $\mathcal{D}_S = \{\mathcal{D}_{S_i}\}_{i=1}^{N}$; a target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$; N feature extractors $\{\Theta^{G_1}, \Theta^{G_2}, ...\Theta^{G_N}\}$, N disentanglers $\{\Theta^{D_1}, \Theta^{D_2}, ...\Theta^{D_N}\}$, N classifiers $\{\Theta^{C_1}, \Theta^{C_2}, ...\Theta^{C_N}\}$, N class identifiers $\{\Theta^{CI_1}, \Theta^{CI_2}, ...\Theta^{CI_N}\}$, N mutual information estimators $\{\Theta^{M_1}, \Theta^{M_2}, ...\Theta^{M_N}\}$ trained on source domains. Target feature extractor Θ^{G_t}, classifier Θ^{C_t}. N domain identifiers $\{\Theta^{DI_1}, \Theta^{DI_2}, ..., \Theta^{DI_N}\}$

Output: well-trained target feature extractor $\hat{\Theta}^{G_t}$, target classifier $\hat{\Theta}^{C_t}$.

Model Initialization .

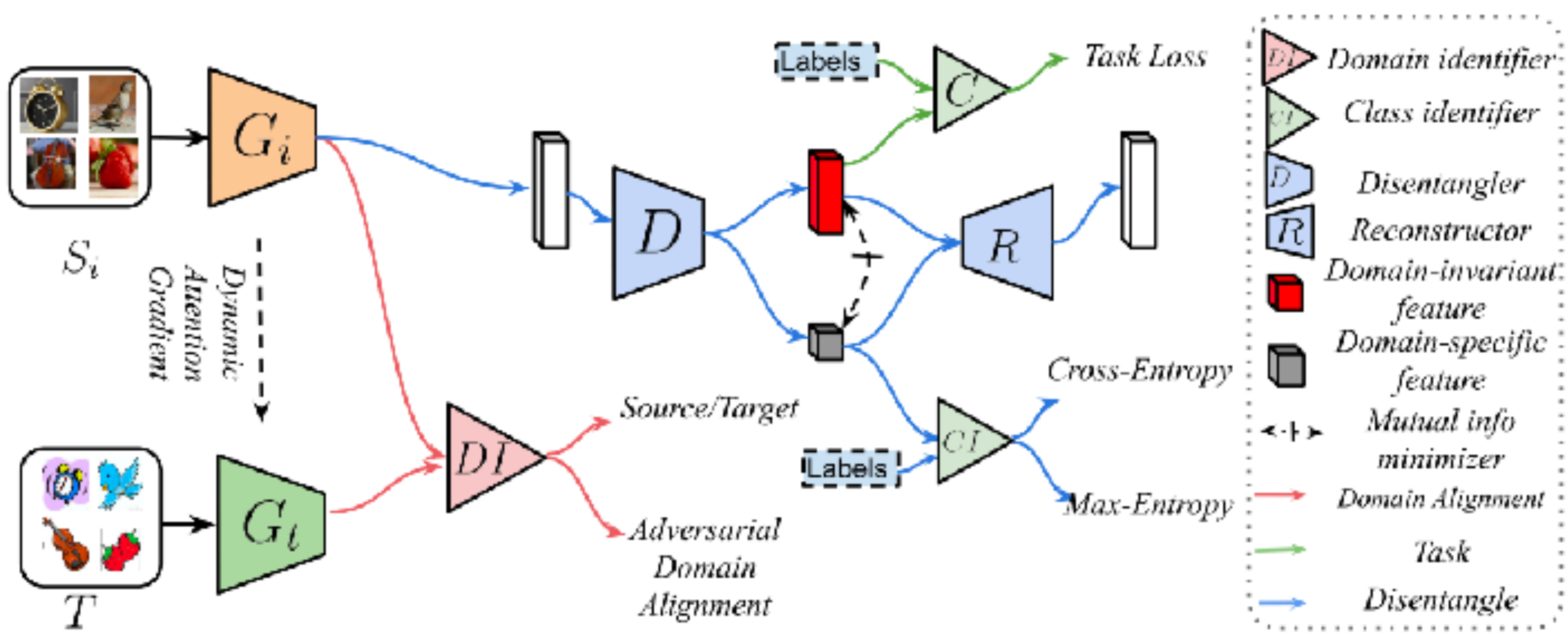
while not converged **do**
    for i **do**=1:N
        Sample mini-batch from from $\{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and $\{\mathbf{x}_j^t\}_{j=1}^{n_t}$;
        Compute gradient with cross-entropy classification loss, update Θ^{G_i}, Θ^{C_i}.
        Domain Alignment:
        Update Θ^{DI_i}, $\{\Theta^{G_i}, \Theta^{G_t}\}$ with Eq. 4 and Eq. 5 respectively to align the domain distribution;
        Domain Disentangle:
        update $\Theta^{G_i}, \Theta^{D_i}, \Theta^{C_i}, \Theta^{CI_i}$ with Eq. 6
        update Θ^{D_i} and $\{\Theta^{C_i}\}$ with Eq. 7
        Mutual Information Minimization:
        Calculate mutual information between the disentangled feature pair (f_{di}, f_{ds}) with M_i;
        Update Θ^{D_i}, Θ^{M_i} by Eq.8;
    end for
    Dynamic weight:
    Calculate dynamic weight by Eq. 3
    Update $\Theta^{G_t}, \Theta^{C_t}$ by aggregated $\{\Theta^{G_1}, \Theta^{G_2}, ..., \Theta^{G_N}\}$, $\{\Theta^{C_1}, \Theta^{C_2}, ...\Theta^{C_N}\}$ respectively with the computed dynamic weight;
end while
return $\Theta^{G_t}, \Theta^{C_t}$

Dynamic Attention

- Gap statistics: $I = \sum_{r=1}^{k} \frac{1}{2n_r} \sum_{i,j \in C_r} ||f_i^t - f_j^t||_2$
- Gap statistics gain: $I_i^{gain} = I_i^{p-1} - I_i^p$
- Mask on the gradients: $Softmax(I_1^{gain}, I_2^{gain}, ..., I_N^{gain})$

Conclusion

$$\epsilon_T(h_T) \leq \underbrace{\hat{\epsilon}_{\tilde{S}}(\sum_{i\in[N]} \alpha_i h_{S_i})}_{\text{error on source}} + \sum_{i\in[N]} \alpha_i \Big(\frac{1}{2} \underbrace{\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\hat{\mathcal{D}}_{S_i}, \hat{\mathcal{D}}_T)}_{(\mathcal{D}_{S_i}, \mathcal{D}_T)\text{ divergence}} + \lambda_i \Big) + \underbrace{4\sqrt{\frac{2d\log(2Nm) + \log(4/\delta)}{Nm}}}_{\text{VC-Dimension Constraint}}$$



Improving Federated Learning Personalization via MAML

Motivation

- Similarity between FL and MAML
- Finetune improve global model and personalized model
- Fedavg model is easier to personalized than traditional model

Introduction

- interpret existing FL algorithm in the light of existing MAML algorithms
- novel modification of FedAvg
- demonstrate that FedAvg optimize for personalized performance, as opposed to quality of the global model.

INTERPRETING FEDAVG AS A META LEARNING ALGORITHM

Algorithm 1 Connects FL and MAML (left), Reptile Batch Version(middle), and FedAvg (right).

OuterLoop/Server learning rate α
InnerLoop/Client learning rate β
Initial model parameters θ
while not done **do**
  Sample batch of tasks/clients $\{T_i\}$
  for Sampled task/client T_i **do**
    if FL **then**
      $g_i, w_i = ClientUpdate(\theta, T_i, \beta)$
    else if MAML **then**
      $g_i = InnerLoop(\theta, T_i, \beta)$
    end if
  end for
  if FL **then**
    $\theta = ServerUpdate(\theta, \{g_i, w_i\}, \alpha)$
  else if MAML **then**
    $\theta = OuterLoop(\theta, \{g_i\}, \alpha)$
  end if
end while

Require: : Reptile Step K.
  function $InnerLoop(\theta, T_i, \beta)$
    Sample K-shot data $D_{i,k}$ from T_i.
    $\theta_i = \theta$
    for each local step i from 1 to K **do**
      $\theta_i = \theta_i - \beta \nabla_\theta L(\theta_i, D_{i,k})$
    end for
    Return $g_i = \theta_i - \theta$
  end function
Require: : Meta Batch Size M.
  function $OuterLoop(\theta, \{g_i\}, \alpha)$
    $\theta = \theta + \alpha \frac{1}{M} \sum_{i=1}^{M} g_i$
    Return θ
  end function

Require: FedAvg Local Epoch E.
  function $ClientUpdate(\theta, T_i, \beta)$
    Split local dataset into batches B
    $\theta_i = \theta$
    for each local epoch i from 1 to E **do**
      for batch $b \in B$ **do**
        $\theta_i = \theta_i - \beta \nabla_\theta L(\theta_i, b)$
      end for
    end for
    Return $g_i = \theta_i - \theta$
  end function
Require: Clients per training round M.
  function $ServerUpdate(\theta, \{g_i, w_i\}, \alpha)$
    $\theta = \theta + \alpha \sum_{i=1}^{M} w_i g_i / \sum_{i=1}^{M} w_i$
    Return θ
  end function

INTERPRETING FEDAVG AS A META LEARNING ALGORITHM

$$g_{FedSGD} = \frac{-\beta}{T} \sum_{i=1}^{T} \frac{\partial L_i(\theta)}{\partial \theta} = \frac{1}{T} \sum_{i=1}^{T} g_1^i.$$

$$\theta_K^i = U_K^i(\theta) = \theta - \beta \textstyle\sum_{j=1}^{K} g_j^i = \theta - \beta \sum_{j=1}^{K} \frac{\partial L_i(\theta_j)}{\partial \theta}$$

$$\frac{\partial U_K^i(\theta)}{\partial \theta} = I - \beta \frac{\partial \sum_{j=1}^{K} g_j^i}{\partial \theta} = I - \beta \sum_{j=1}^{K} \frac{\partial^2 L_i(\theta_j)}{\partial \theta^2}.$$

$$g_{MAML} = \frac{\partial L_{MAML}}{\partial \theta} = \frac{1}{T} \sum_{i=1}^{T} \frac{\partial L_i(U_K^i(\theta))}{\partial \theta} = \frac{1}{T} \sum_{i=1}^{T} L_i'(U_K^i(\theta))(I - \beta \sum_{j=1}^{K} \frac{\partial^2 L_i(\theta_j)}{\partial \theta^2})$$

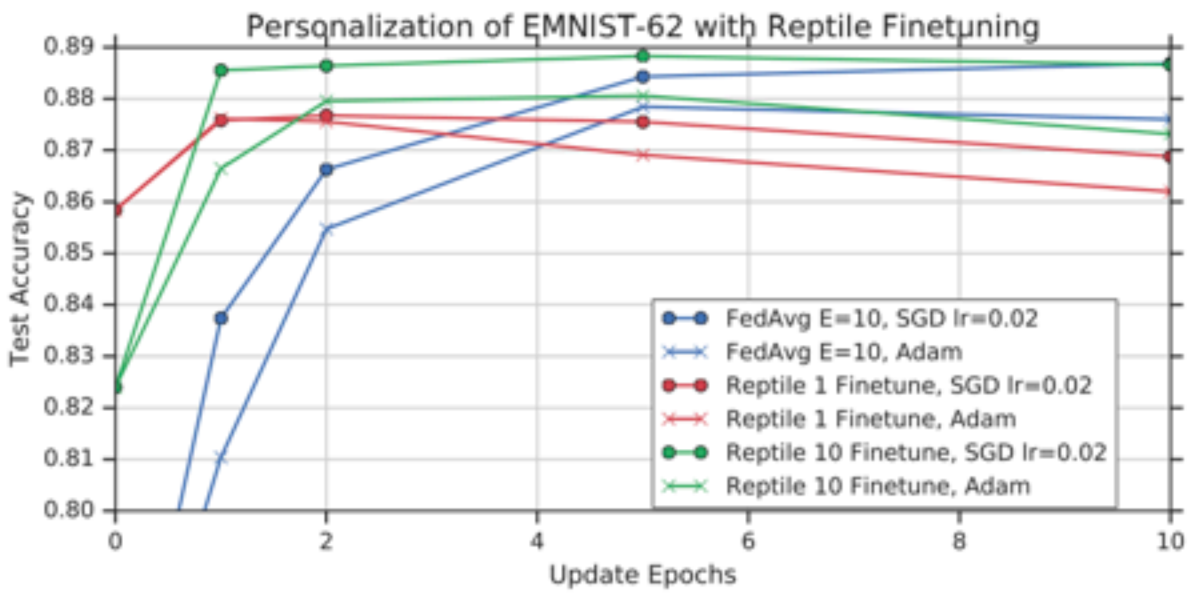
$$g_{FOMAML}(K) = \frac{1}{T} \sum_{i=1}^{T} L_i'(U_K^i(\theta)) I = \frac{1}{T} \sum_{i=1}^{T} L_i'(\theta_K^i) = \frac{1}{T} \sum_{i=1}^{T} g_{K+1}^i$$

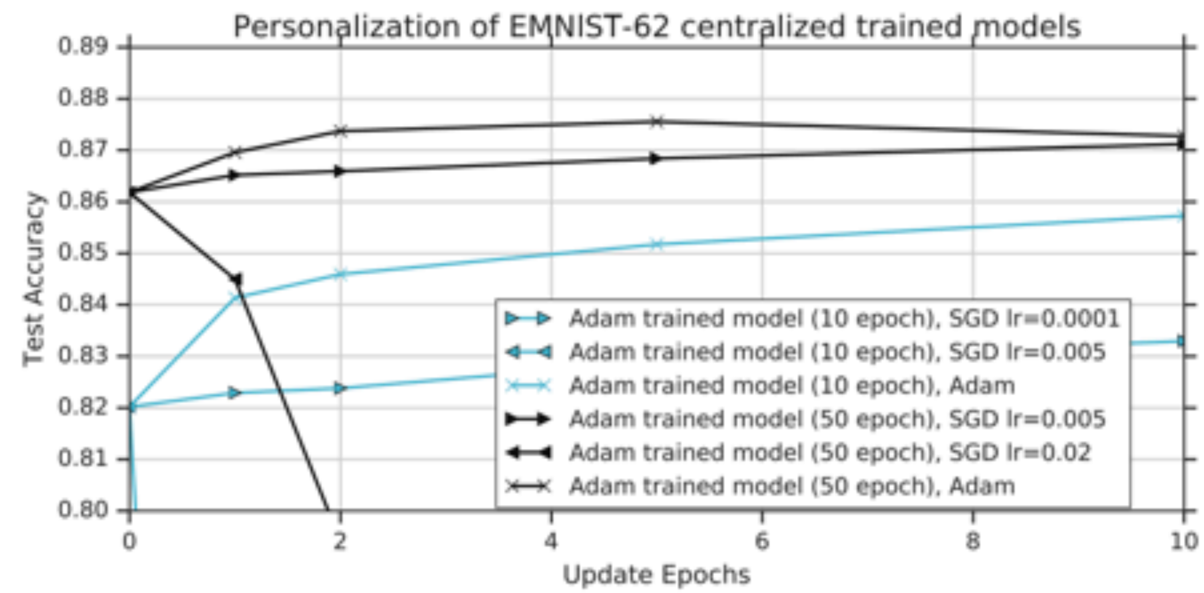
$$g_{FedAvg} = \frac{1}{T} \sum_{i=1}^{T} \sum_{j=1}^{K} g_j^i = \frac{1}{T} \sum_{i=1}^{T} g_1^i + \sum_{j=1}^{K-1} \frac{1}{T} \sum_{i=1}^{T} g_{j+1}^i = g_{FedSGD} + \sum_{j=1}^{K-1} g_{FOMAML}(j)$$

Personalized FedAvg

Algorithm 2 Personalized FedAvg

1: Run $FedAvg(E)$ with momentum SGD as server optimizer and a relatively larger E.
2: Switch to $Reptile(K)$ with Adam as server optimizer to fine-tune the initial model.
3: Conduct personalization with the same client optimizer used during training.





	Initial Acc	Personalized Acc
Reptile(1) Finetuned test clients	0.8320 (0.0133)	0.8764 (0.0017)
Reptile(1) Finetuned train clients	0.8577 (0.0019)	0.8927 (0.0015)
Reptile(10) Finetuned test clients	0.8116 (0.0148)	0.8858 (0.0014)
Reptile(10) Finetuned train clients	0.8612 (0.0020)	0.9028(0.0009)

Table 2: Test performance on clients seen and unseen during FL-training

Challenges for FL

- adaptation to the statistical heterogeneity
- optimize the personalized performance and global model
- Influence on capacity to personalize
- Solely optimizing have negative impact

Challenges for MAML

- consider the performance of the initial model
- Importance of fast convergence
- Datasets with a natural user/client structure being established for FL

Future Work

- How does the training algorithm impact personalization ability of the trained model?
- Is there something we can measure that will predict the adaptability of the model?
- Is it something we can directly optimize for, potentially leading to novel optimization methods?