Learning without limits

(Inspired by "Lifelong Domain Adaptation via Consolidated Internal Distribution" by Mohammad Rostami)

Group 69 : DCCLXXI

Indian Institute of Technology, Kanpur

November 26, 2024

Introduction

- Feature engineering using deep neural networks struggles with generalization to unseen data, especially when domain shifts occur between training and testing datasets.
- **Continual learning** (CL) aims to create models that can adapt to changing data distributions; learning new information without forgetting previous knowledge.
- **Unsupervised Domain Adaptation** (UDA) trains models for a target domain with unlabeled data by transferring knowledge from a related source domain with labeled data.
- **Catastrophic forgetting** refers to the loss of knowledge from previous tasks when learning a new task.

Background

Unsupervised Domain Adaptation (UDA) :

- Existing domain alignment methods use probability distance to measure distribution discrepancies and train an encoder to minimize cross-domain distance.
- Wasserstein Distance (WD) minimizes cross-domain distance due to its non-vanishing gradients, essential for gradient-based optimization. The Sliced Wasserstein Distance (SWD) improves computational efficiency with a closed-form solution based on empirical samples.

Continual Learning (CL) :

- **Model Regularization** consolidates crucial network weights to preserve past knowledge during updates.
- **Experience Replay** stores key data points in a memory buffer, replaying them alongside new task data.

What to solve?

- Standard classification problem on source domain S with a labeled training dataset $D_S = (X_0, Y_0)$, where X_0 is the feature matrix, and Y_0 is the corresponding label matrix involves training a deep neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$ by minimizing the loss function $L(\cdot)$, with optimal parameters: $\hat{\theta}_0 = \arg\min_\theta \sum_{i=1}^{N} L(f_\theta(x_i^0), y_i^0)$
- In this problem, we are presented with a sequence of target domains $\mathcal{T}_t$, where $t = 1, \ldots, T$, each having an unlabeled dataset $D_{\mathcal{T}}^t = (X_t)$. Each domain $\mathcal{T}_t$ follows its own unique input distribution $p_t(x)$, and these distributions differ: $\forall t_1, t_2 : p_{t_1} \neq p_{t_2}$.

Also need to tackle:

- Failure to apply traditional optimization techniques
- Non-stationary Distributions
- Catastrophic Forgetting
- Sequential Access Constraint

Solution Presented

The presented solution addresses domain shift and catastrophic forgetting by splitting the model f_θ into two components:

- **Deep Encoder** $\phi_v(\cdot) : X \to Z$: Maps input data to a latent space Z where source domain classes are well-separated.
- **Classifier** $h_w(\cdot) : Z \to Y$: Maps the latent embeddings Z to output predictions.

By stabilizing the distribution in Z, the solution minimizes the distance between source embeddings $\phi(p_0(x^0))$ and target embeddings $\phi(p_t(x^t))$ The encoder is trained to map the data into a multi-modal distribution $p_0^J(z)$ in the embedding space, modeled by a Gaussian Mixture Model (GMM) represented as: $p_0^J(z) = \sum_{j=1}^{k} \alpha_0^j N(z \mid \mu_0^j, \Sigma_0^j)$ Hence The MAP estimates are :

$$\hat{\alpha}_0^j = \frac{|S_0^j|}{N},\ \hat{\mu}_j = \frac{1}{|S_0^j|} \sum_{(x_0^i, y_0^i) \in S_0^j} \phi_v(x_0^i),\ \hat{\Sigma}_0^j = \frac{1}{|S_0^j|} \sum_{(x_0^i, y_0^i) \in S_0^j} (\phi_v(x_0^i) - \hat{\mu}_j)^T (\phi_v(x_0^i) - \hat{\mu}_j) \tag{1}$$

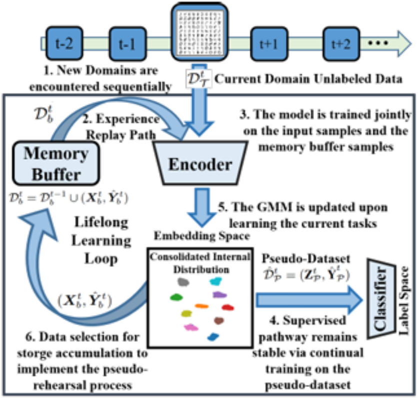
Solution Presented

The continual learning process aligns the target domain's distribution with the learned distribution in the embedding space by combining Pseudo-Generated Data and Filtered Input Data to align target distributions with the embedding space.
The optimization problem at time t is:

$$\min_{v,w} \sum_{i=1}^{N} L(h_w(z_i^p), \hat{y}_{p,t}^j) + \lambda D(\phi_v(p_t(X_t)), \hat{p}_t^J(Z_t^P))$$

To prevent catastrophic forgetting, experience replay is introduced, storing and replaying a subset of past data. The updated objective function incorporates buffer samples:

$$\min_{v,w} \Bigg(\sum_{i=1}^{N} L(h_w(z_i^p), \hat{y}_{p,t}^j) + \sum_{i=1}^{N_b} L(h_w(\phi_v(x_i^b)), \hat{y}_i^b) \\ + \lambda D(\phi_v(p_t(X_t)), \hat{p}_t^J(Z_t^P)) + \lambda D(\phi_v(p_t(X_t^b)), \hat{p}_t^J(Z_t^P)) \Bigg) \tag{2}$$



Lifelong Domain Adaptation Agent

Algorithm 1 LDAuCID (λ, τ, N_b)

1: **Source Training**:
2: **Input:** source labeled dataset $\mathcal{D}_{\mathcal{S}} = (\boldsymbol{X}_0, \boldsymbol{Y}_0)$
3: $\hat{\theta}_0 = (\hat{\boldsymbol{w}}_0, \hat{\boldsymbol{v}}_0) = \arg\min_\theta \sum_i \mathcal{L}(f_\theta(\boldsymbol{x}_i^0), \boldsymbol{y}_i^0)$
4: **Internal Distribution Estimation:**
5: Use Eq. (2) and estimate $\alpha_j^0, \boldsymbol{\mu}_j^0$, and Σ_j^0
6: **Memory Buffer Initialization**
7: $\mathcal{D}_b^0 = (\boldsymbol{X}_b^0, \hat{\boldsymbol{Y}}_b^0)$
Pick the N_b/k samples with the least $d_{j,l}^t = \|\mu_j^t - \phi(\boldsymbol{x}_l^t)\|_2^2$, $\hat{\boldsymbol{y}}_b^{0,i} = \arg\max f_{\hat{\theta}^t}(\boldsymbol{x}_b^{0,N_b})$
8: **Continual Unsupervised Domain Adaptation**:
9: **for** $t = 1, \ldots, T$ **do**
10: **Input:** target unlabeled dataset $\mathcal{D}_{\mathcal{T}}^t = (\boldsymbol{X}_t)$
11: **Pseudo-Dataset Generation:**
12: $\hat{\mathcal{D}}_{\mathcal{P}}^t = (\boldsymbol{Z}_{\mathcal{P}}^t, \hat{\boldsymbol{Y}}_{\mathcal{P}}^t) =$
13: $([\boldsymbol{z}_1^{p,t}, \ldots, \boldsymbol{z}_N^{p,t}], [\hat{\boldsymbol{y}}_1^{p,t}, \ldots, \hat{\boldsymbol{y}}_N^{p,t}])$, where: $\boldsymbol{z}_i^{p,t} \sim \hat{p}_J^{t-1}(\boldsymbol{z}), 1 \le i \le N_p$ and $\hat{\boldsymbol{y}}_i^{p,t} = \arg\max_j \{h_{\hat{\boldsymbol{w}}_t}(\boldsymbol{z}_i^{p,t})\}$ if with confidence τ: $\max_j \{h_{\hat{\boldsymbol{w}}_t}(\boldsymbol{z}_i^{p,t})\} > \tau$
14: **for** $itr = 1, \ldots, ITR$ **do**
15: draw data batches from $\mathcal{D}_{\mathcal{T}}^t$ and $\hat{\mathcal{D}}_{\mathcal{P}}$
16: Update the model by solving Eq. (4)
17: **end for**
18: **Internal Distribution Estimate Update:**
19: Use Eq. (2) similar to step 5 above.
20: **Memory Buffer Update**
21: $\mathcal{D}_b^t = \mathcal{D}_b^{t-1} \cup (\boldsymbol{X}_b^t, \hat{\boldsymbol{Y}}_b^t)$, where $(\boldsymbol{X}_b^t, \hat{\boldsymbol{Y}}_b^t)$ is computed similar to step 7 above.
22: **end for**

Source: Images from "Lifelong Domain Adaptation via Consolidated Internal Distribution" (Rostami,2021)

Algorithm

Algorithm: LDAuCID

1. **Source Training**:
   Input the source labeled dataset $\mathcal{D}S = (X^0, Y^0)$.Initialize the model parameters:$\hat{\theta}^0 = (\hat{w}^0, \hat{v}^0) = \arg\min_\theta \sum_i L(f_\theta(x_i^0), y_i^0)$.
2. **Internal Distribution Estimation**:
   Use Eq. (1) to estimate α_j^0, μ_j^0, and Σ_j^0.
3. **Memory Buffer Initialization**:
   Initialize the memory buffer $\mathcal{D}_b^0 = (X_b^0, \hat{Y}_b^0)$ by selecting N_b/k samples with the least distances:
   $d_{j,l}^t = |\mu_j^t - \phi(x_l^t)|_2^2$
   where $\hat{y}_{i,b}^0 = \arg\max f_{\hat{\theta}_t}(x_{b,N_b}^0)$.

Algorithm

Algorithm: LDAuCID (contd.)

4. **Continual Unsupervised Domain Adaptation**
   For each time step $t = 1, \ldots, T$:
   - Input the target unlabeled dataset $\mathcal{D}_T^t = (X^t)$.
   - **Pseudo-Dataset Generation**:
     Generate the pseudo-dataset $\hat{\mathcal{D}}_P^t = (Z_P^t, \hat{Y}_P^t)$ as follows:
     $Z_P^t = [z_1^{p,t}, \ldots, z_N^{p,t}], \quad Y_P^t = [\hat{y}_1^{p,t}, \ldots, \hat{y}_N^{p,t}]$
     where $z_i^{p,t} \sim \hat{p}_J^{t-1}(z), \quad 1 \leq i \leq N_p$
     and $\hat{y}^{p,t}i = \arg\max_j h_{\hat{w}_t}(z^{p,ti})$ if the confidence threshold τ is satisfied:
     $\max_j h_{\hat{w}_t}(z_i^{p,t}) > \tau$.

Algorithm

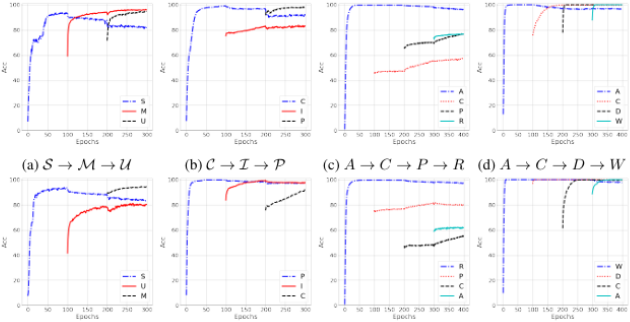
Algorithm: LDAuCID (contd.)

4. **Continual Unsupervised Domain Adaptation**
   For each time step $t = 1, \ldots, T$:
   - **For** each iteration $\text{itr} = 1, \ldots, \text{ITR}$:
     - Draw data batches from $\mathcal{D}_T^t$ and $\hat{\mathcal{D}}_P^t$.
     - Update the model by solving Eq. (2).
   - **Internal Distribution Estimate Update**: Update internal distribution parameters using Eq. (1) as in Step 2.
   - **Memory Buffer Update**:
     - Update the memory buffer: $\mathcal{D}_b^t = \mathcal{D}_b^{t-1} \cup (X_b^t, \hat{Y}_b^t)$ where X_b^t and $\hat{Y}_b^t$ are selected as in Step 2.

Datasets: ImageCLEF-DA, Office-Home, Office-Caltech, Digit Recognition. **Methodology:** VGG16 (digit tasks), ResNet-50 (ImageCLEF-DA, Office-Home), Decaf6 (Office-Caltech).

Key Findings:

- **Learning Curves:** Performance drops initially, improves as model adapts.



(a) $\mathcal{S} \rightarrow \mathcal{M} \rightarrow \mathcal{U}$ (b) $\mathcal{C} \rightarrow \mathcal{I} \rightarrow \mathcal{P}$ (c) $A \rightarrow C \rightarrow P \rightarrow R$ (d) $A \rightarrow C \rightarrow D \rightarrow W$

(e) $\mathcal{S} \rightarrow \mathcal{U} \rightarrow \mathcal{M}$ (f) $\mathcal{P} \rightarrow \mathcal{I} \rightarrow \mathcal{C}$ (g) $R \rightarrow P \rightarrow C \rightarrow A$ (h) $W \rightarrow D \rightarrow C \rightarrow A$

Source:"Lifelong Domain Adaptation via Consolidated Internal Distribution" (Rostami,2021)

Results & Inference (Continued)

Key Findings (cont.):

- **Catastrophic Forgetting:** Mitigated by LDAuCID, though more pronounced in challenging tasks (e.g., SVHN).
- **Dataset Performance:**
  - ImageCLEF-DA: Significant improvement due to balanced data.
  - Office-Home: Strong performance; CDAN outperforms.
- **Buffer/Hyperparameters:** Larger memory buffer (Nb) improves performance; $\tau \approx 1$ reduces label pollution.
- **Imbalanced Data:** LDAuCID performs well, though with reduced performance on imbalanced data.

Conclusion

- **LDAuCID** integrates **UDA** and **CL** to address domain shift and catastrophic forgetting by aligning internal data distributions in the embedding space.
- **Outperforms traditional UDA methods**, particularly in moderate domain shifts, and mitigates catastrophic forgetting effectively.
- **Performance** can be improved with techniques like **class-conditional alignment** for large domain gaps (e.g., Office-Home dataset).
- Uses **experience replay** and a **memory buffer** to retain acquired knowledge, enhancing continual learning, especially with imbalanced data or domain shifts.