Unsupervised Domain Adaptation
Two domain adaptation models that perform explicit class alignment via cross-domain class prototypes on top of typical feature alignment

Moving Semantic Transfer Network (MSTN) (Xie et al., ICML 2018)
- The paper proposes a new module that pseudo-labels batches of unlabeled target examples and then aligns class prototypes of labeled source and newly pseudo-labeled target examples during each iteration

Progressive Progressive Feature Alignment (PFAN) (Chen et al., CVPR 2019), an incremental improvement upon the MSTN model
- The model separately pseudo-label all unlabeled examples based on encoded source prototypes and selectively collect “easy” pseudo-labeled ones to form smaller pseudo-labeled target dataset
- Then, the model performs class prototypes alignment

Overall, both models do not provide good performance compared to the implicit joint domain-class alignment (JDCA), but these papers present good ideas on how to solve the class alignment problem (the idea of pseudo-labeling has been applied to many other topics)
Moving Semantic Transfer Network (MSTN): Class Alignment Objective

- New module: for each iteration with batches of labeled source and unlabeled target examples
  - pseudo-label the batch of unlabeled target examples
  - compute the local class prototypes of both labeled source and pseudo-labeled target batches
  - update the global class prototypes based on exponential moving average
  - (new objective) minimize the distance between the cross-domain global class prototypes

**Algorithm**: MSTN’s procedure to compute the additional objective of class alignment

```plaintext
Require: labeled source examples \( \mathcal{D}^s \), and unlabeled target examples \( \mathcal{D}^t \)

1: for epoch in num epochs do
2: initialize global class prototypes of source and target domains \( c_{k(I)}^s \) and \( c_{k(I)}^t \) to be 0
3: for iter i in num iters do
4: get source and target batches of examples \( B_i^s \in \mathcal{D}^s \) and \( B_i^t \in \mathcal{D}^t \)
5: pseudo-label the batch of unlabeled target examples
   \( \hat{B}_i^t = \{(x_i^t, \hat{y}_i^t)\} \)
   \( \hat{y}_i^t = \arg\max_k h(g(x_i^t)) \)
6: compute local (batch) class prototypes of both domains
   \( c_{k(i)} = \frac{1}{|B_i|} \sum_{(x_i, y_i) \in B_i} g(x_i) \)
7: update global class prototypes of both domains
   \( c_{k(I)} = \alpha c_{k(I)} + (1 - \alpha) c_{k(i)} \)
8: additional objective: aligning global class prototypes of both domains
   \( L_{ca} = \sum_k^K \psi(c_{k(I)}^s, \hat{c}_{k(I)}^t) \)
9: end for
10: end for
```
Moving Semantic Transfer Network (MSTN): Notes

- We should use moving average global prototypes instead of local batch prototypes
  - it is possible that some classes are missing in the current batch since the batch is randomly selected
  - if the batch size is small, even one false pseudo-labeled example will lead to the large deviation between pseudo-labeled prototypes and true prototypes
Progressive Feature Alignment (PFAN): Class Alignment

- **Easy-to-Hard Transfer Strategy (EHTS)**
  - pseudo-label all target examples based on the encoded source prototypes and select most “easy” pseudo-labeled ones to form a pseudo-labeled dataset

- **Adaptive Prototype Alignment (APA)**
  - perform class alignment by computing and aligning the class prototypes of labeled source and pseudo-labeled target examples

---

**Algorithm**: PFAN’s procedure to compute the additional objective of class alignment

**Require**: labeled source examples $D^s$, and unlabeled target examples $D^t$

1. for epoch in num epochs do
2.   → EHTS
3.   compute the class prototypes of source domain
4.   $c^s_k = \frac{1}{|D^s|} \sum_{(x_i^s, y_i^s) \in D^s_k} g(a_i^s)$
5.    pseudo-label all unlabeled target examples, and select only “easy” examples that have similarity score above a threshold
6.    $\tilde{D}^t = \{(a_i^t, \tilde{y}_i^t)\}$
7.    $\tilde{y}_i^t = \arg\max_k \psi(g(a_i^t), c_k^s)$
8.    and $\max_k \psi(g(a_i^t), c_k^s) > \tau$
9. for iter $i$ in num iters do
10.   get source and pseudo-labeled target batches of examples $B_i^s \in D^s$ and $B_i^t \in \tilde{D}^t$
11.   compute local (batch) class prototypes of both domains
12. $c_k(i) = \frac{1}{|B_i|} \sum_{(x_i, y_i) \in B_i} g(x_i)$
13.   update global class prototypes of both domains
14. $\bar{c}_k(i) = \frac{1}{i} \sum_{j=1}^i c_k(j)$
15. $\rho_i = \psi(\bar{c}_k(i), c_k(i))$
16. $c_k(i) = \rho_i^2 c_k(i) + (1 - \rho_i^2)\bar{c}_k(i)$
17. additional objective: aligning global class prototypes of both domains
18. $L_{ga} = \sum_k \psi(c_k(i), \bar{c}_k(i))$
19. end for
20. end for
Progressive Feature Alignment (PFAN): EHTS

- EHTS: pseudo-label all target examples and select most “easy” pseudo-labeled ones to form a pseudo-labeled dataset
  - compute source prototypes \( c^S_k \) for every classes
    \[
    c^S_k = \frac{1}{N^S_k} \sum_{(x^s_i, y^s_i) \in D^S_k} g(x^s_i)
    \]
  - cluster and pseudo-label all unlabeled target examples to the corresponding source prototypes using a similarity metric
    \[
    \hat{y}_i = \arg\max_k \psi(g(x^T_i), c^S_k)
    \]
  - rank and select only “easy” pseudo-labeled examples that have similarity scores above an annealing threshold \( \tau \)
    - after each training epoch, since the model is better at performing representations and likely gathers more ”easy” examples
    - in contrast, we want to keep the selection ratio constant, so we gradually increase the threshold to harder the selection
Progressive Feature Alignment (PFAN): APA

APA: perform class alignment by computing and aligning the class prototypes of labeled source and pseudo-labeled target examples

- initialize global prototypes $c_k^{s(I)}$ and $c_k^{t(I)}$ of all examples from $D^s$ and $\hat{D}^t$ respectively
- at each iter $i$, compute local prototypes $c_k^{s(i)}$ and $c_k^{t(i)}$ of the batches $B_i^s$ and $\hat{B}_i^t$ respectively, then update the global prototypes as follows:

$$c_k(i) = \frac{1}{i} \sum_{j=1}^{i} c_k(j)$$

$$\rho_i = \psi(\bar{c}_k(i), c_k(I))$$

$$c_k(I) = \rho^2_i \bar{c}_k(i) + (1 - \rho^2_i) c_k(I)$$

- (new objective) minimize the distance between cross-domain global class prototypes

$$L_{apa}(c_k^{s(I)}, c_k^{t(I)}) = \| c_k^{s(I)} - c_k^{t(I)} \|^2$$
In order to prevent models biased over (over-reliance on) source classification, we suggest to gradually reduce and remove the convergence of the source classification by adding a controllable temperature variable into the last softmax output function

$$\hat{q}_i = \frac{\exp z_i/T}{\sum_j \exp z_j/T}$$
Thank you!