PivotAlign: Improve Semi-Supervised Learning by Learning Intra-Class Heterogeneity and Aligning with Pivots Supplementary

Lingjie Yi[1], Tao Sun[1], Yikai Zhang[2], Songzhu Zheng[2], Weimin Lyu[1], Haibin Ling[1], Chao Chen[1]
[1] Stony Brook University, [2] Morgan Stanley
{chris.yi, weimin.lyu, chao.chen.1}@stonybrook.edu,
{tao, hling}@cs.stonybrook.edu,
{yikai.zhang, songzhu.zheng}@morganstanley.com

Supplementary

A Distance Among Pivots

In Sec.1, we mention that maintaining intra-class heterogeneity of data requires the prevention of pivots from shrinking together. Here, we visualize the distance matrix between pivots (Fig. 1) to confirm that our method does avoid the collapse of pivots throughout the training process.

It's also seen that the distribution of pivots also exhibits a hierarchical structure (also shown in Fig. 2e). So both inter-class and intra-class heterogeneity of data are well-maintained. These 30 pivots are learned with CIFAR-10 training images with 40 labeled data.

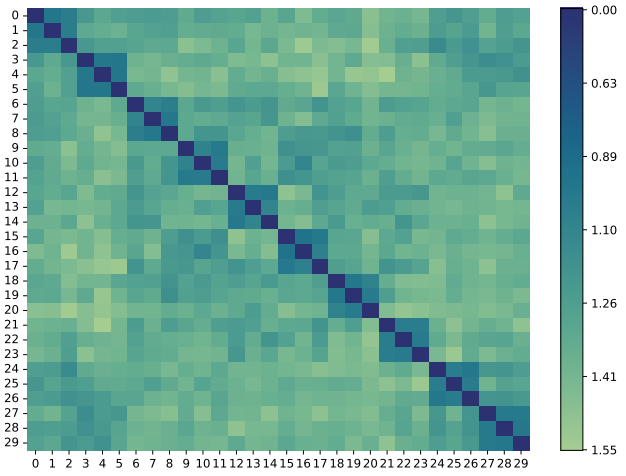


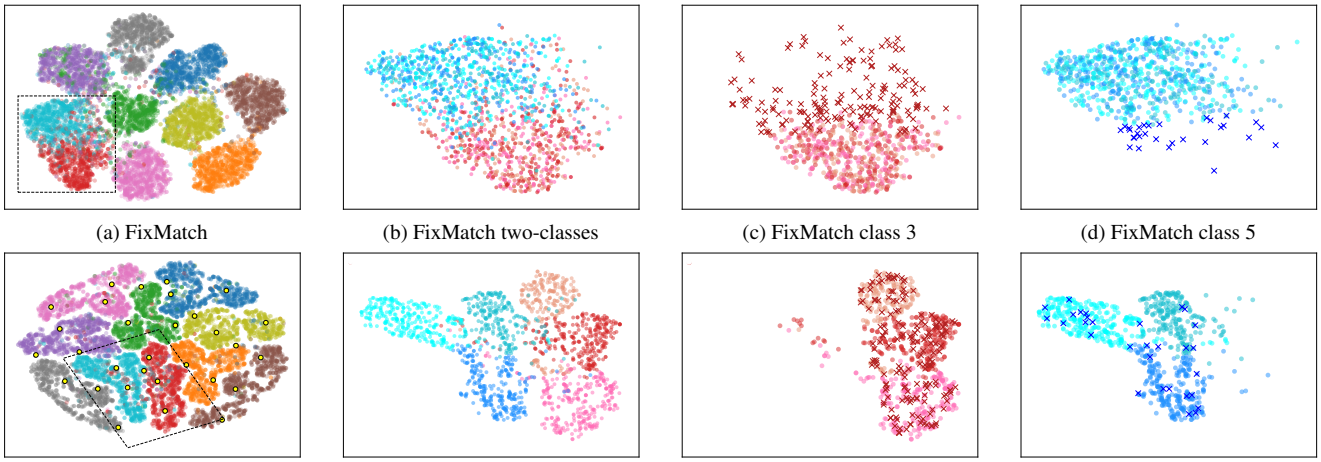
Figure 1. Distance Between 30 Learned Pivots

B Additional Visualization Results

In Sec.5.3, we have shown the detailed comparison between PivotAlign and a baseline model FixMatch through *t*-SNE visualization. Here, we provide the complete version of the visualization (Fig. 2). We also provide an additional comparison between PivotAlign and a self-supervised learning based baseline model, CoMatch (Fig. 3).

Similar to Fig. 2, Fig. 3a shows that CoMatch only learns inter-class heterogeneity of data. But, with the help of self-supervised learning, CoMatch learns more compact representations for each class and more separated decision boundaries, especially the boundary between class 3 and class 5 (Fig. 3b). Also, same with FixMatch, samples from the same sub-class spread arbitrarily within the class-level cluster while our model align samples together by sub-classes.

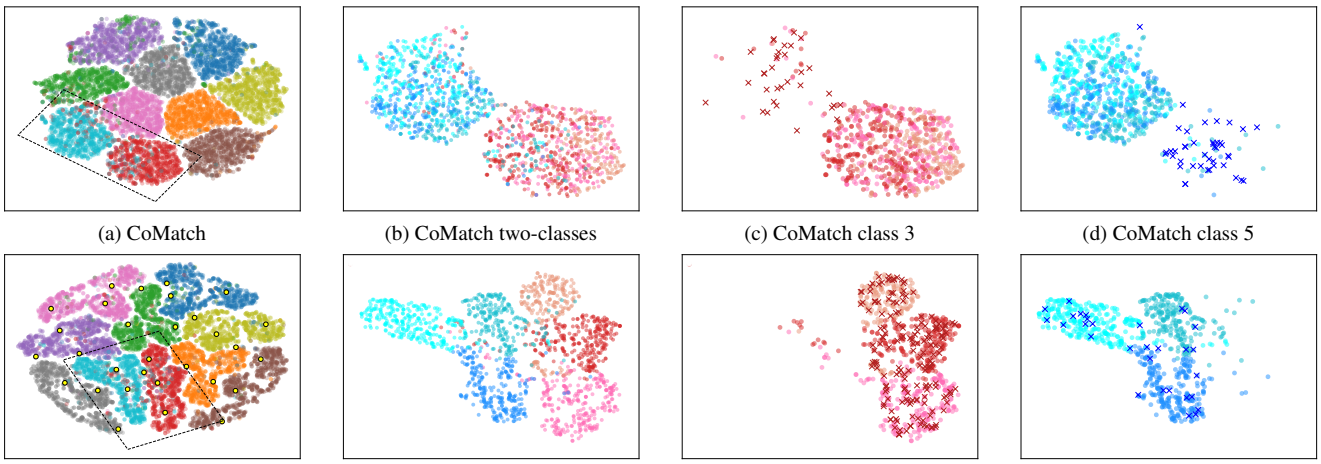
In Fig. 3, we provide further details about samples from class 3 (Fig. 3c vs Fig. 3g) and class 5 (Fig. 3d vs Fig. 3h) that are mis-classified by CoMatch but corrected by PivotAlign. We can see that, via learning intra-class heterogeneity of data, many samples that are difficult for baseline models to learn are correctly grouped with more related neighboring points by our model.



(a) FixMatch (b) FixMatch two-classes (c) FixMatch class 3 (d) FixMatch class 5

(e) PivotAlign (f) PivotAlign two-classes (g) PivotAlign class 3 (h) PivotAlign class 5

Figure 2. *t*-SNE visualization of CIFAR-10 training set. (a) and (e) show representations learned by FixMatch [2] and PivotAlign respectively. Each color denotes one of the 10 distinct classes. Yellow dots denote learned pivots. Dashed boxes denote partial-view to be zoomed-in. (b) and (f) are zoom-in-views of class 3 and class 5. These two classes are colored in blue and red in (a) and (f), but colored by six sub-clusters (from light blue to dark blue, from light red to dark red) identified by PivotAlign. It can be seen that samples with similar intra-class semantic information are dispersed randomly within the class-level cluster in (b) but are grouped together in (f). (c) and (g) are further isolated views of class 3. '×' denotes samples mislabeled by FixMatch but corrected by PivotAlign. (d) and (h) are isolated views of class 5.



(a) CoMatch (b) CoMatch two-classes (c) CoMatch class 3 (d) CoMatch class 5

(e) PivotAlign (f) PivotAlign two-classes (g) PivotAlign class 3 (h) PivotAlign class 5

Figure 3. *t*-SNE visualization of CIFAR-10 training set. (a) and (e) show representations learned by CoMatch [1] and PivotAlign respectively. Each color denotes one of the 10 distinct classes. Yellow dots denote learned pivots. Dashed boxes denote partial-view to be zoomed-in. (b) and (f) are zoom-in-views of class 3 and class 5. These two classes are colored in blue and red in (a) and (f), but colored by six sub-clusters (from light blue to dark blue, from light red to dark red) identified by PivotAlign. It can be seen that samples with similar intra-class semantic information are dispersed randomly within the class-level cluster in (b) but are grouped together in (f). (c) and (g) are further isolated views of class 3. '×' denotes samples mislabeled by CoMatch but corrected by PivotAlign. (d) and (h) are isolated views of class 5.

C Pseudo-code

In this section, we present pseudo-code of our model. In Algorithm 1, we show the training pipeline of PivotAlign within one mini-batch.

Algorithm 1 Pseudocode of PivotAlign

```
# f: backbone encoder
# g: projection MLP
# l: prediction MLP
# pivots: feature embeddings of pivots
# p_labels: labels of pivots

# MB contrains
# z_m: feature embeddings in MB
# pseudo_labels_m: pseudo_labels in MB
# mask_m: confident samples mask in MB

for x, y, u in loader: # load a minibatch
   x = aug_w(x) # random augmentation
   u1, u2 = aug_w(x), aug_s(x)

   px = l(f(x))
   h1, h2 = f(u1), f(u2) # encoding, n-by-d
   p1, p2 = l(h1), l(h2) # predictions, n-by-d
   z1, z2 = g(h1), g(h2) # projections, n-by-d

   z1 = normalize(z1, dim=1)
   z2 = normalize(z2, dim=1)

   with torch.no_grad():
      # calculate pseudo labels
      probs = DA(softmax(p1, dim=-1))
      ## Eq.14 and Eq.15
      clusters = get_cluster_label(
         z1, pivots, p_labels, MB
      )
      ## Eq.16
      pred = alpha*probs + (1-alpha)*clusters
      pseudo_labels, mask = get_label_mask(pred)

      # pivot assignments
      ## Eq.7, Eq.8 and Eq.9
      assign, assign_m = get_pivot_assignment(
         pivots, p_labels,
         z1, pseudo_labels, mask, MB
      )

      # weight for contrastive loss
      ## Eq.13
      weight_c = get_contrast_weight(
         assign, pseudo_labels, mask,
         assign_m, p_labels, MB
      )

   # calculate loss
   ## Eq.1
   lx = loss_x(px, y)
   ## Eq.2
   lu = loss_u(p2, pseudo_labels, mask)
   ## Eq.11
   lp = loss_p(pivots, z1, assign, mask)
   ## Eq.12
   lc = loss_c(z2, z_m, weight_c)
   ## Eq.17
   loss = calc_loss(lx, lu, lp, lc)

   loss.backward() # back-propagate
   update(f, l, g, pivots) # SGD update
   update_MB(z1, pseudo_labels, mask) # Update MB
```

For key functions mentioned in Algorithm 1, we provide an additional explanation about how we generate pivot assignments in Algorithm 2. We also provide pseudo-code about how we generate cluster labels in Algorithm 3.

Algorithm 2 Pivot Assignments Generation

```
# z: feature embeddings of input
# mask: confident samples mask

def get_pivot_assignment(
      pivots, p_labels,
      z, pseudo_labels, mask,
      MB
):
   # key numbers
   num_pivots = pivots.shape[0]
   num_conf = torch.where(mask)[0].shape[0]

   # feature
   z_batch = z[mask]
   z_bank = z_m[mask_m]
   z_sample = torch.cat([z_batch, z_bank], 0)

   # pseudo label
   pl_batch = pseudo_labels[mask]
   pl_bank = pseudo_labels_m[mask_m]
   pl_sample = torch.cat([pl_batch, pl_bank],0)

   # affinity
   label_mask = pseudo_labels.reshape(-1, 1).eq(
      p_labels.reshape(1, -1)
   )
   affinity = torch.mm(z_sample, pivots.t())
   affinity = affinity * label_mask.float()

   # marginal distribution
   ## column dist
   c = torch.ones(num_pivots) / num_pivots
   ## row dist
   count = Counter(pseudo_labels)
   count = [count[i] for i in range(num_classes)]
   prob = np.array(
      [1 / (num_classes * i) for i in count]
   )
   r = prob[pseudo_labels]

   # OT clustering
   assign_all = sinkhorn(
      -affinity_sample, r, c, epsilon
   )
   assign_all *= label_mask.float()
   assign_all /= assign_all.sum(1, keepdim=True)

   assign = assign_all[:num_conf]
   assign_m = assign_all[num_conf:]

   return assign, assign_m
```

Algorithm 3 Cluster Labels Generation

```
def get_cluster_label(z, pivots, p_labels, MB):
   # key numbers
   num_feats = z.shape[0]

   # feature
   z_sample = torch.cat([z, z_m], 0)

   # aggregated prob
   sim = torch.mm(z_sample, pivots.t())
   weight = torch.softmax(sim / temperature, dim=1)
   prob_agg = torch.mm(
      weight, F.one_hot(pivots_label, num_classes)
   )

   # OT clustering
   cluster = sinkhorn(
      -prob_agg, epsilon=epsilon, exp=False
   )
   cluster = cluster[:num_feats]

   return cluster
```

References

[1] Junnan Li, Caiming Xiong, and Steven C.H. Hoi. CoMatch: Semi-supervised Learning with Contrastive Graph Regularization. In *ICCV*, 2021. 2

[2] Kihyuk Sohn, David Berthelot, Chun Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. FixMatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020. 2