CNN-based Action Recognition and Supervised Domain Adaptation on 3D Body Skeletons via Kernel Feature Maps (Supplementary Material)

Yusuf Tas[1,2], Piotr Koniusz[1,2]
users.cecs.anu.edu.au/~koniusz

[1]Data61/CSIRO, Canberra, Australia
[2]Australian National University

1 Supervised Domain Adaptation [1]

For the full details of the *So-HoT* algorithm, please refer to paper [1]. Below, we review the core part of their algorithm for the reader's convenience. Suppose $\mathcal{I}_N$ and $\mathcal{I}_{N^*}$ are the indexes of N source and N^* target training data points. $\mathcal{I}_{N_c}$ and $\mathcal{I}_{N_c^*}$ are the class-specific indexes for $c\in\mathcal{I}_C$, where C is the number of classes. Furthermore, suppose we have feature vectors from an *FC* layer of the source network stream, one per an action sequence or image, and their associated labels. Such pairs are given by $\Lambda\equiv\{(\phi_n,y_n)\}_{n\in\mathcal{I}_N}$, where $\phi_n\in\mathbb{R}^d$ and $y_n\in\mathcal{I}_C$, $\forall n\in\mathcal{I}_N$. For the target data, by analogy, we define pairs $\Lambda^*\equiv\{(\phi^*_n,y^*_n)\}_{n\in\mathcal{I}^*_N}$, where $\phi^*\in\mathbb{R}^d$ and $y^*_n\in\mathcal{I}_C$, $\forall n\in\mathcal{I}^*_N$. Class-specific sets of feature vectors are given as $\Phi_c\equiv\{\phi^c_n\}_{n\in\mathcal{I}_{N_c}}$ and $\Phi^*_c\equiv\{\phi^{*c}_n\}_{n\in\mathcal{I}_{N^*_c}}$, $\forall c\in\mathcal{I}_C$. Then $\Phi\equiv(\Phi_1,...,\Phi_C)$ and $\Phi^*\equiv(\Phi^*_1,...,\Phi^*_C)$. The asterisk in superscript (*e.g.* ϕ^*) denotes variables related to the target network while the source-related variables have no asterisk. The *So-HoT* problem is posed as a trade-off between the classifier and alignment losses ℓ and $\hbar$. Figure 4 (the main submission) shows the setup we use. The loss $\hbar$ depends on two sets of variables $(\Phi_1,...,\Phi_C)$ and $(\Phi^*_1,...,\Phi^*_C)$ – one set per network stream. Feature vectors $\Phi(\Theta)$ and $\Phi^*(\Theta^*)$ depend on the parameters of the source and target network streams Θ and Θ^* that we optimize over. $\Sigma_c\equiv\Sigma(\Phi_c)$, $\Sigma^*_c\equiv\Sigma(\Phi^*_c)$, $\mu_c(\Phi)$ and $\mu^*_c(\Phi^*)$ denote the covariances and means, respectively, one covariance/mean pair per network stream per class. Specifically, we solve:

$$\mathop{\arg\min}_{\substack{W,W^*,\Theta,\Theta^*\\ \text{s. t. } ||\phi_n||_2^2\leq\tau,\\ ||\phi^*_{n'}||_2^2\leq\tau,\\ \forall n\in\mathcal{I}_N,n'\in\mathcal{I}^*_N}} \ell(W,\Lambda)+\ell(W^*,\Lambda^*)+\eta||W-W^*||_F^2 + \underbrace{\frac{\alpha_1}{C}\sum_{c\in\mathcal{I}_C}||\Sigma_c-\Sigma^*_c||_F^2+\frac{\alpha_2}{C}\sum_{c\in\mathcal{I}_C}||\mu_c-\mu^*_c||_2^2}_{\hbar(\Phi,\Phi^*)}. \tag{1}$$

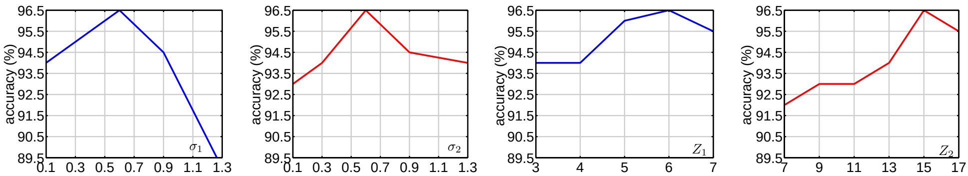
For ℓ, a generic Softmax loss is employed. For the source and target streams, the matrices $W,W^*\in\mathbb{R}^{d\times C}$ contain unnormalized probabilities. In Equation (1), separating the class-specific distributions is addressed by ℓ while attracting the within-class scatters of both network streams is handled by $\hbar$. Variable η controls the proximity between W and W^* which encourages the similarity between decision boundaries of classifiers. Coefficients α_1, α_2 control the degree of the cov. and mean alignment, τ controls the ℓ_2-norm of feature vectors.

2 Modifications to the So-HoT Approach

Algorithm 1 details how we perform domain adaptation. We enable the alignment loss $\hbar$ only if the source and target batches correspond to the same class. Otherwise, the alignment loss is disabled and the total loss uses only the classification log-losses ℓ_{src} and ℓ_{trg}. To generate the source and target batches that match w.r.t. the class label, we re-order source and target datasets class-by-class and thus each source/target batch contains only one class label at a time. Once all source and target datapoints with matching class labels are processed, remaining datapoints are processed next. Lastly, we refer readers interested in the details of the So-HoT algorithm and loss $\hbar$ to paper [1].

Algorithm 1 Batch generation + a single epoch of the training procedure on the source and target datasets.

$src\_data :=$ sort_by_class_label(src_data)
$target\_data :=$ sort_by_class_label($target\_data$)
C_s ▷ Number of the source classes
C_t ▷ Number of the target classes
$C_{s\cap t}$ ▷ Number of classes in common
procedure EPOCH($src\_data, target\_data, batch\_size$) ▷ Training (one epoch)
  for $i \leftarrow 1 : max(C_s, C_t)$ **do**
    if $i \leq C_s$ **then**
      $batch_s \leftarrow$ Choose($src\_data, i, batch\_size$) ▷ 'Choose' pre-fetches data of class i
    else
      $batch_s \leftarrow$ Choose($src\_data, rnd(), batch\_size$) ▷ 'Choose' pre-fetches data of random class
    if $i \leq C_t$ **then**
      $batch_t \leftarrow$ Choose($target\_data, i, batch\_size$)
    else
      $batch_t \leftarrow$ Choose($target\_data, rnd(), batch\_size$)
    if $i \leq C_{s\cap t}$ **then**
      $Loss \leftarrow \ell_{src} + \ell_{trg} + \hbar$
    else
      $Loss \leftarrow \ell_{src} + \ell_{trg}$
    Forward(net_data, $batch\_s$, $batch\_t$)
    Backward(net_data, $batch\_s$, $batch\_t$)
    Update(net_data, $batch\_s$, $batch\_t$)



(a) (b) (c) (d)

Figure 1: Sensitivity w.r.t. parameters σ_1 and σ_2 on UTK. Figures 1a, 1b, 1c and 1d show the accuracy w.r.t. σ_1 ($\sigma_2 = 0.6$), σ_2 ($\sigma_1 = 0.6$), Z_1 ($Z_2 = 15$) and Z_2 ($Z_1 = 5$), respectively.

References

[1] Piotr Koniusz, Yusuf Tas, and Fatih Porikli. Domain adaptation by mixture of alignments of second- or higher-order scatter tensors. *CVPR*, 2017.