Supplementary Material of Unsupervised Domain Adaptation via Regularized Conditional Alignment

Safa Cicek, Stefano Soatto
UCLA Vision Lab
University of California, Los Angeles, CA 90095
{safacicek,soatto}@ucla.edu

In Section 1, we provide proofs for the results given in Section 4 of the main paper. In Section 2, we describe the implementation details. In Section 3, we report and discuss the performance of the proposed method when one or more components in the loss are removed.

1. Analysis

First, we prove a simple lemma that will be handy in the proof of Proposition 1.

Lemma 1.

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{K} -\alpha[i]\log(\theta[i]) \text{ s.t. } 1 \geq \theta[i] \geq 0, \sum_{i=1}^{K} \theta[i] = 1, \alpha[i] > 0, \text{ for all } i. \text{ Then, } \theta^*[k] = \frac{\alpha[k]}{\sum_{i=1}^{K}\alpha[i]} \text{ for any } k.$$

Proof. Let us write the Lagrangian form excluding inequality constraints, $L(\theta,\lambda) = \sum_{i=1}^{K} -\alpha[i]\log(\theta[i]) + \lambda(\sum_{i=1}^{K}\theta[i]-1)$. $\nabla_{\theta[i]}L(\theta,\lambda) = -\frac{\alpha[i]}{\theta[i]} + \lambda = 0$ and $\theta[i] = \frac{\alpha[i]}{\lambda}$ for all i. $-\log$ is convex hence sum of them also convex and the stationary point is global minima. Then, the dual form becomes $g(\lambda) = \sum_{i=1}^{K} -\alpha[i]\log(\frac{\alpha[i]}{\lambda}) + \lambda(\sum_{i=1}^{K}\frac{\alpha[i]}{\lambda} - 1)$ then $\nabla_\lambda g(\lambda) = 0$ when $\lambda = \sum_{i=1}^{K}\alpha[i]$ and $\theta[k] = \frac{\alpha[k]}{\sum_{i=1}^{K}\alpha[i]}$. Note that the constraint $1 \geq \theta[i] \geq 0$ does not constrain the solution space as $\theta[i]$ has to be non-negative for $\log(\theta[i])$ to be defined and $\sum_{i=1}^{K}\theta[i] = 1$ enforces $1 \geq \theta[i]$. □

Proposition 1. *The optimal joint predictor h_j minimizing $L_{jsc}(h_j) + L_{jtc}(h_j)$ given in the Eq. 6,7 for any feature z with non-zero measure either on $g\#P_x^s(z)$ or $g\#P_x^t(z)$ is*

$$h_j(z)[i] = \frac{g\#P^s(z, y = e_i)}{g\#P_x^s(z) + g\#P_x^t(z)} \text{ and } h_j(z)[i+K] = \frac{g\#P^t(z, y = e_i)}{g\#P_x^s(z) + g\#P_x^t(z)} \text{ for } i \in \{1, ..., K\}$$

Proof.

$$\begin{aligned}
& L_{jsc}(h_j) + L_{jtc}(h_j) \\
&= E_{(x,y)\sim P^s}\ell_{CE}(h_j(g(x)), [y, \mathbf{0}]) + E_{(x,y)\sim P^t}\ell_{CE}(h_j(g(x)), [\mathbf{0}, y]) && (1)\\
&= \int_{(x,y)\sim P^s} P_x^s(x)\ell_{CE}(h_j(g(x)), [y, \mathbf{0}])dx + \int_{(x,y)\sim P^t} P_x^t(x)\ell_{CE}(h_j(g(x)), [\mathbf{0}, y])dx && (2)\\
&= \int_{z\sim g\#P_x^s}\int_{(x,y)\sim P^s s.t. z=g(x)} P_x^s(x)\ell_{CE}(h_j(z), [y, \mathbf{0}])dxdz \\
&\quad + \int_{z\sim g\#P_x^t}\int_{(x,y)\sim P^t s.t. z=g(x)} P_x^t(x)\ell_{CE}(h_j(z), [\mathbf{0}, y])dxdz && (3)\\
&= \int_{z\sim g\#P_x^s}\int_{(x,y)\sim P^s s.t. z=g(x)} P_x^s(x)\langle -\log h_j(z), [y, \mathbf{0}]\rangle dxdz
\end{aligned}$$

$$+\int_{z\sim g\#P_x^t}\int_{(x,y)\sim P^t s.t. z=g(x)} P_x^t(x)\langle -\log h_j(z), [\mathbf{0},\ y]\rangle dxdz \tag{4}$$

$$=\int_{z\sim g\#P_x^s}\langle -\log h_j(z), [\int_{(x,y)\sim P^s s.t. z=g(x)} P_x^s(x)ydx,\ \mathbf{0}]\rangle dz$$

$$+\int_{z\sim g\#P_x^t}\langle -\log h_j(z), [\mathbf{0},\ \int_{(x,y)\sim P^t s.t. z=g(x)} P_x^t(x)ydx]\rangle dz \tag{5}$$

$$=\int_{z\sim g\#P_x^s}\sum_{i=1}^{K} -\log h_j(z)[i]g\#P^s(z,y=e_i)dz+\int_{z\sim g\#P_x^t}\sum_{i=1}^{K} -\log h_j(z)[i+K]g\#P^t(z,y=e_i)dz \tag{6}$$

From Lemma 1, $h_j(z)[i]=\frac{g\#P^s(z,y=e_i)}{Z}$ and $h_j(z)[i+K]=\frac{g\#P^t(z,y=e_i)}{Z}$ for $i\in\{1,...,K\}$ where $Z=\sum_{i=1}^{K}(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i))=g\#P_x^s(z)+g\#P_x^t(z)$ for any z. Note that, we integrated losses over x only; instead of (x,y) as we are assuming that the ground-truth labeling function is deterministic. Moreover, in Lemma 1, we assumed $\alpha[i]>0$ while here $g\#P^s(z,y=e_i)$ and $g\#P^t(z,y=e_i)$ might be zero for some i. But since we are taking $\alpha[i]\log(\theta[i])$ as zero whenever $\alpha[i]=0$ for any value of $\theta[i]$, the result does not change. □

The following Lemma will be used in the proof of Theorem 1.

Lemma 2. $\min_{P,Q} L(P,Q)=E_{x\sim P}-\log\frac{Q(x)}{P(x)+Q(x)}+E_{x\sim Q}-\log\frac{P(x)}{P(x)+Q(x)}$ *is achieved only if* $P(x)=Q(x)$ *for all* x.

Proof.

$$L(P,Q) \tag{7}$$

$$=\int_x -P(x)\log(\frac{Q(x)}{P(x)+Q(x)})-Q(x)\log(\frac{P(x)}{P(x)+Q(x)})dx \tag{8}$$

$$=\int_x P(x)\log(\frac{P(x)+Q(x)}{Q(x)})+Q(x)\log(\frac{P(x)+Q(x)}{P(x)})dx \tag{9}$$

$$=\int_x P(x)\log(1+\frac{P(x)}{Q(x)})+Q(x)\log(1+\frac{Q(x)}{P(x)})dx \tag{10}$$

$$=\int_x \log(1+\frac{P(x)}{Q(x)})(1+\frac{P(x)}{Q(x)})Q(x)-\log(1+\frac{P(x)}{Q(x)})Q(x)+Q(x)\log(1+\frac{Q(x)}{P(x)})dx \tag{11}$$

$$=\int_x \Big(\log(1+\frac{P(x)}{Q(x)})(1+\frac{P(x)}{Q(x)})-\log(1+\frac{P(x)}{Q(x)})+\log(1+\frac{Q(x)}{P(x)})\Big)Q(x)dx \tag{12}$$

$$=\int_x \big(\log(1+\frac{P(x)}{Q(x)})(1+\frac{P(x)}{Q(x)})+\log(\frac{Q(x)}{P(x)})\big)Q(x)dx \tag{13}$$

$$=\log(4)-\int_x \log(2)\frac{P(x)+Q(x)}{Q(x)}Q(x)dx+\int_x \big(\log(1+\frac{P(x)}{Q(x)})(1+\frac{P(x)}{Q(x)})+\log(\frac{Q(x)}{P(x)})\big)Q(x)dx \tag{14}$$

$$=\log(4)+\int_x \big(\log(1+\frac{P(x)}{Q(x)})(1+\frac{P(x)}{Q(x)})-\log(\frac{P(x)}{Q(x)})-\log(2)(1+\frac{P(x)}{Q(x)})\big)Q(x)dx \tag{15}$$

Let $\phi(\beta):=\log(1+\beta)(1+\beta)-\log(\beta)-\log(2)(1+\beta)$. Then, $\nabla_\beta\phi(\beta)=1+\log(1+\beta)-\frac{1}{\beta}-\log(2)$ and $\nabla_\beta\nabla_\beta\phi(\beta)=\frac{1}{1+\beta}+\frac{1}{\beta^2}>0$ for $\beta>0$. Hence $\phi(\beta)$ is convex and we can apply Jensen,

$$L(P,Q)=\log(4)+\int_x \phi(\frac{P(x)}{Q(x)})Q(x)dx\geq\log(4)+\phi(\int_x\frac{P(x)}{Q(x)}Q(x)dx)=\log(4)+\phi(1)=\log(4) \tag{16}$$

Since ϕ is strictly convex, equality is satisfied only for constant argument i.e. when $\frac{P(x)}{Q(x)}=1$ which is also the global minima of $\phi(\beta)$ as $\nabla_\beta\phi(1)=0$. □

Theorem 1. *The objective* $L_{jsa}(g)+L_{jta}(g)$ *given in the Eq. 8,9 is minimized for the given optimal joint predictor if and only if* $g\#P^s(z|y=e_k)=g\#P^t(z|y=e_k)$ *and* $g\#P^s(z|y=e_k)>0\Rightarrow g\#P^s(z|y=e_i)=0$ *for* $i\neq k$ *for any* $y=e_k$ *and* z.

Proof. The objective for encoder is,

$$E_{(x,y)\sim P^s}\ell_{CE}((h_j(g(x)),[\mathbf{0},\,y])+E_{(x,y)\sim P^t}\ell_{CE}((h_j(g(x)),[y,\,\mathbf{0}]) \tag{17}$$

For samples with label e_k we want to minimize,

$$\begin{aligned}
&E_{(x,y)\sim P^s(x,y=e_k)}\ell_{CE}((h_j(g(x)),[\mathbf{0},\,y])+E_{(x,y)\sim P^t(x,y=e_k)}\ell_{CE}((h_j(g(x)),[y,\,\mathbf{0}]) &(18)\\
&=E_{(x,y)\sim P^s(x,y=e_k)}-\langle[\mathbf{0},\,y],\log(h_j(g(x))\rangle+E_{(x,y)\sim P^t(x,y=e_k)}-\langle[y,\,\mathbf{0}],\log(h_j(g(x))\rangle &(19)\\
&=-E_{(x,y)\sim P^s(x,y=e_k)}\log(h_j(g(x))[k+K]-E_{(x,y)\sim P^t(x,y=e_k)}\log(h_j(g(x))[k] &(20)
\end{aligned}$$

Given the classifier h_j is optimal, the above them becomes

$$\begin{aligned}
&-\int_{z\sim g\#P^s(z,y=e_k)} g\#P^s(z,y=e_k)\log\frac{g\#P^t(z,y=e_k)}{\sum_{i=1}^K(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i)}dz\\
&-\int_{z\sim g\#P^t(z,y=e_k)} g\#P^t(z,y=e_k)\log\frac{g\#P^s(z,y=e_k)}{\sum_{i=1}^K(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i))}dz &(21)\\
&=\int_{z\sim g\#P^s(z,y=e_k)} g\#P^s(z,y=e_k)\Big(-\log\frac{g\#P^t(z,y=e_k)}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}\\
&\qquad+\log\frac{\sum_{i=1}^K(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i))}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}\Big)dz\\
&+\int_{z\sim g\#P^t(z,y=e_k)} g\#P^t(z,y=e_k)\Big(-\log\frac{g\#P^s(z,y=e_k)}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}\\
&\qquad+\log\frac{\sum_{i=1}^K(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i))}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}\Big)dz &(22)
\end{aligned}$$

Let us write first and second terms in each integration separately:

$$\begin{aligned}
L_1(g\#P^s,g\#P^t)=&-\int_{z\sim g\#P^s(z,y=e_k)} g\#P^s(z,y=e_k)\log\frac{g\#P^t(z,y=e_k)}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}dz\\
&-\int_{z\sim g\#P^t(z,y=e_k)} g\#P^t(z,y=e_k)\log\frac{g\#P^s(z,y=e_k)}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}dz &(23)
\end{aligned}$$

$$\begin{aligned}
L_2(g\#P^s,g\#P^t)=&\int_{z\sim g\#P^s(z,y=e_k)} g\#P^s(z,y=e_k)\log\frac{\sum_{i=1}^K(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i))}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}dz\\
&+\int_{z\sim g\#P^t(z,y=e_k)} g\#P^t(z,y=e_k)\log\frac{\sum_{i=1}^K(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i))}{g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)}dz &(24)
\end{aligned}$$

If there is a solution which is global minima of both L_1,L_2 then it is also the global minima of the overall term L_1+L_2. L_2 has its minimum at $\sum_{i=1 s.t. i\neq k}^K(g\#P^s(z,y=e_i)+g\#P^t(z,y=e_i))=0$ whenever $g\#P^s(z,y=e_k)+g\#P^t(z,y=e_k)>0$. From Lemma 2, $L_1(g\#P^s,g\#P^t)$ achieves its minimum only when $g\#P^s(z,y=e_k)=g\#P^t(z,y=e_k)$ for any z. Intersection of two minimas gives the desired solution.

□

2. Implementation Details

Batchsize of 64 is used for both the source and the target samples during training. Batchsize of 100 is used at inference time. The networks used in the experiments are given in Table 2. Instance norm is only used in MNIST $\leftrightarrow$ SVHN experiments. In all experiments, networks are trained for $60,000$ iterations. This is less than $80,000+80,000=160,000$ iterations that SOA methods VADA+DIRT-T and Co-DA+DIRT-T are trained for. Weight decay of 10^{-4} is used. In CIFAR $\leftrightarrow$ STL and SYN-DIGITS $\rightarrow$ SVHN, as an optimizer we use SGD with the initial learning rate of 0.1. Learning rate is decreased to

Source dataset Target dataset	MNIST SVHN	SVHN MNIST	CIFAR STL	STL CIFAR	SYN-DIGITS SVHN	MNIST MNIST-M
λ_t	0.1	0.1	0.1	0.1	0.1	0.1
λ_{tvat}	10.0	10.0	10.0	10.0	10.0	10.0
λ_{jtc}	10.0	1.0	1.0	1.0	10.0	10.0
λ_{jta}	1.0	0.1	0.1	0.1	0.1	1.0
λ_{svat}	0.0	0.0	0.0	0.0	1.0	0.0
λ_{jsc}	1.0	1.0	1.0	1.0	1.0	1.0
λ_{jsa}	1.0	0.1	1.0	1.0	1.0	1.0
ϵ_x	4.0	4.0	2.0	1.0	1.0	0.5

Table 1. **Hyperparameters.** Hyper-parameters used in the proposed method for each task.

Encoder
3×3 convolution, 64 lReLU
3×3 convolution, 64 lReLU
3×3 convolution, 64 lReLU
2×2 max-pool, stride 2, dropout with probability 0.5
3×3 convolution, 64 lReLU
3×3 convolution, 64 lReLU
3×3 convolution, 64 lReLU
2×2 max-pool, stride 2, dropout with probability 0.5

Class predictor
3×3 convolution, 64 lReLU
1×1 convolution, 64 lReLU
1×1 convolution, 64 lReLU
Global average pooling, $6 \rightarrow 1$
Fully connected layer: $128 \rightarrow$ K
Softmax

Joint predictor
3×3 convolution, 64 lReLU
1×1 convolution, 64 lReLU
1×1 convolution, 64 lReLU
Global average pooling, $6 \rightarrow 1$
Fully connected layer: $128 \rightarrow 2$K
Softmax

Encoder
3×3 convolution, 128 lReLU
3×3 convolution, 128 lReLU
3×3 convolution, 128 lReLU
2×2 max-pool, stride 2, dropout with probability 0.5
3×3 convolution, 256 lReLU
3×3 convolution, 256 lReLU
3×3 convolution, 256 lReLU
2×2 max-pool, stride 2, dropout with probability 0.5
3×3 convolution, 512 lReLU
1×1 convolution, 256 lReLU
1×1 convolution, 128 lReLU
Global average pooling, $6 \rightarrow 1$

Class predictor
Fully connected layer: $128 \rightarrow$ K
Softmax

Joint predictor
Fully connected layer: $128 \rightarrow 2$K
Softmax

Table 2. **Left.** The network used in the tasks involving MNIST dataset (i.e. MNIST $\leftrightarrow$ SVHN and MNIST $\rightarrow$ MNIST-M), *small-net* from [3, 2]. **Right.** The network used in the rest of the classification tasks (i.e. STL $\leftrightarrow$ CIFAR, SYN-DIGITS $\rightarrow$ SVHN), *conv-large* from [1]. Slope of each leaky RELU (lReLU) layer is 0.1. Each conv is followed by a batch norm layer.

0.01 at iteration $40,000$. Momentum of SGD is 0.9. In MNIST $\leftrightarrow$ SVHN and MNIST $\rightarrow$ MNIST-M, Adam optimizer with the fixed learning rate 0.001 is used. Momentum is chosen to be 0.5.

We fix $\lambda_t = 0.1$ and $\lambda_{jsc} = 1.0$. We searched rest of the parameters over $\lambda_{tvat} \in \{1.0, 10.0\}$, $\lambda_{jtc} \in \{1.0, 10.0\}$, $\lambda_{jta} \in \{0.1, 1.0\}$, $\lambda_{svat} = \{0.0, 1.0\}$, $\lambda_{jsa} \in \{0.1, 1.0\}$. We also searched for the upper bound of the adversarial perturbation in VAT, $\epsilon_x \in \{0.1, 0.5, 1.0, 2.0, 4.0, 8.0\}$. Optimal hyperparameters are given in Table 1 for each task. Only for MNIST $\rightarrow$ SVHN, class predictor performs poorly in the early epochs. So, we apply curriculum learning within $60,000$ iterations. In the first $4,000$ iterations, only λ_{jsc} and λ_{jsa} are non-zero i.e. losses only depending on the labeled-source data are minimized. After $4,000$ iterations, SSL regularizations are started to be applied: λ_t, λ_{svat} and λ_{tvat} are also set to non-zero. After $8,000$ iterations, losses depending on the pseudo-labels are activated by assigning all hyperparameters to their optimal values given in Table 1.

Source dataset Target dataset	MNIST SVHN	SVHN MNIST	CIFAR STL	STL CIFAR	SYN-DIGITS SVHN	MNIST MNIST-M
Without VAT	60.65	98.79	81.59	70.20	93.15	98.45
Without EntMin and VAT	62.95	88.33	80.97	71.62	92.10	97.74
Without source alignment	75.78	88.91	81.11	74.80	95.72	99.25
Without target alignment	71.59	98.89	80.90	74.87	95.48	99.20
Without source and target alignment	60.07	98.83	80.20	73.52	94.94	99.08
Source-only (baseline)	44.21	70.58	79.41	65.44	85.83	70.28
The proposed loss with the class predictor	89.19	99.33	81.65	77.76	96.22	99.47
The proposed loss with the joint predictor	87.88	99.16	81.19	77.62	95.97	99.40

Table 3. **Ablations.** Performance of the proposed method when one or two terms in the loss function are removed (first five rows). We also report performance of the source-only baseline (6th row) and model optimizing the original loss (7th row) as a reference. In the last row, we report the performance of the joint predictor.

3. Ablations

In Table 3, we report the performance of the proposed method by removing one or more components from the original loss function. We report the results by removing VAT regularizations ($\lambda_{svat} = \lambda_{tvat} = 0$), VAT and entropy minimization, the source-alignment loss ($\lambda_{jsa} = 0$), the target-alignment loss ($\lambda_{jta} = 0$) and both alignment losses ($\lambda_{jsa} = \lambda_{jta} = 0$). Removing any of these components degraded the performance in all the tasks. All results are still better than the source-only model.

Removing both entropy-minimization and VAT losses makes the performance worse than only removing VAT losses in all the tasks except STL→CIFAR and MNIST→SVHN. In these tasks, the entropy-minimization loss only helped when it combined with VAT losses. This is expected as the entropy minimization without VAT regularizations can easily lead to trivial, degenerate solutions by encouraging to cluster samples from different classes. Removing the source and the target-alignment losses together degraded the performance compared to removing either of them except SVHN→MNIST. Applying the target-alignment loss without the source-alignment loss might have a detrimental effect as former one relies on the *noisy* pseudo-labels.

We also report the best performances with the joint-predictor for completeness. The joint predictor achieves very close performance to the class-predictor but it is slightly worse than the class-predictor. We believe this is because the joint-predictor is trained for the harder task of domain and class learning while only latter one is needed at the test time. That is why we choose to use the class-predictor for inference.

References

[1] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. 2018. 4

[2] Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, Bill Freeman, and Gregory Wornell. Co-regularized alignment for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 9366–9377, 2018. 4

[3] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018. 4