Supplementary Material

Varun Khare[1]*, Divyat Mahajan[2*†], Homanga Bharadhwaj[3†], Vinay Kumar Verma[1], Piyush Rai[1]
[1]IIT Kanpur [2]Microsoft Research, India [3]University of Toronto
varunkhare1234@gmail.com, t-dimaha@microsoft.com, homanga@cs.toronto.edu
vkverma@iitk.ac.in , piyush@iitk.ac.in

1. Implementation Details

1.1. Generative Framework

In this section, we describe the architecture that yields our reported results in the ZSL setting wherein there are no images from seen classes in test samples. For the SUN dataset, both the networks used for modeling mean and co-variance have linear (1800 and 2048 nodes), batch normalization and Relu layers. Additionally, the co-variance is restricted to be in the range of 0.5 to 1.5 for numerical stability via sigmoid activation. Both the networks are trained with ADAM optimizer [3] using 0.001 and 0.1 as regularizer coefficients for means and covariance respectively.

For the AWA dataset, the generator networks have an architecture similar to SUN but consist of an additional dropout layer with probability 0.1. The parameters of the means are regularized with the coefficient 10^3 while the parameters of covariance are regularized with the coefficient 10^4.

For the CUB dataset, the generator networks have three linear layers (1200, 1800, 2048 nodes), 2 Relu, 2 batch normalization and 2 dropout layers. Their regularization coefficients are 0.01 and 0.1 for mean and covariance respectively. All the above networks are trained with a learning rate of 0.00001. The training of these networks was additionally regularized via early stopping

1.2. Adversarial Domain Adaption

As described above, ADA model comprises two discriminators ($D^{S,T}$), two classifiers ($C^{S,T}$) and two generators ($G^{S,T}$).

The discriminators $D^{S,T}$ are 5 layered neural networks comprising of 2 Linear layers of 1600 nodes and 1 node respectively, a single leaky Relu layer with a negative slope of 0.2 and batch normalization. The classifiers are single-layered networks with the number of nodes equal to the number of classes. $log(softmax(\mathbf{x}))$ is used as activation function for the classifiers $C^{S,T}$. The generators $G^{S,T}$ consist of three linear layers (1200,1200 and 2048 nodes), dropout layers, batch normalization and leaky Relu.

The overall objective is minimized using RMSprop ([2]) optimizer with a learning rate of 0.00001. A manual seed of 100 has been used for all the ADA experiments.

2. Training Procedures

For brevity, we provide the training algorithms for our base ZSL model and the ADA ZSL model. We use the following loss definition for adversarial training as described in the paper

$$\mathcal{L} = \mathcal{L}^T_{adv} + \mathcal{L}^S_{adv} + \chi\mathcal{L}_{cyc} + \xi\mathcal{L}^T_{clf} + \xi\mathcal{L}^S_{clf} \quad (1)$$

where $\mathcal{L}^{\{T,S\}}_{adv} = \{L_G + L_D\}^{\{T,S\}}$ with

$$L^T_G = \mathop{\mathbb{E}}_{c\sim p_c}[\beta\|G^T(x_{nc}) - x_{nc}\|_p - D^T_w \circ G^T(y_{nc})] \quad (2)$$

$$L^S_G = \mathop{\mathbb{E}}_{c\sim p_c}[\beta\|G^S(y_{nc}) - y_{nc}\|_p - D^S_w \circ G^S(x_{nc})] \quad (3)$$

$$L^T_D = \mathbb{E}_{c\sim p_c}[D^T_w \circ G^T(y_{nc})] - \mathbb{E}_{c\sim p_c}[D^T_w(x_{nc})] \quad (4)$$

$$L^S_D = \mathbb{E}_{c\sim p_c}[D^S_w \circ G^S(x_{nc})] - \mathbb{E}_{c\sim p_c}[D^S_w(y_{nc})] \quad (5)$$

Here, D_w is the Wasserstein loss [1] and c denotes the class label.

$$\mathcal{L}_{\text{cyc}}(G^T, G^S) = \mathbb{E}_{c\sim p_c}[\|G^S \circ G^T(y_{nc}) - x_{nc}\|_p] + \mathbb{E}_{c\sim p_c}[\|G^T \circ G^S(x_{nc}) - y_{nc}\|_p]. \quad (6)$$

Here, $||\cdot||_p$ denotes the L_p norm.

$$L^T_{clf} = \mathbb{E}_{c\sim p_c}[L(C^T_{clf} \circ G^T(y_{nc}), Y^T)] + \mathbb{E}_{c\sim p_c}[L(C^T_{clf}(x_{nc}), \bar{Y}^U)] \quad (7)$$

*VK and DM contributed equally
†DM and HB contributed while being part of IIT Kanpur

$$L^S_{clf} = \mathbb{E}_{c\sim p_c}[L(C^S_{clf} \circ G^S(x_{nc}), Y^S)] + \mathbb{E}_{c\sim p_c}[L(C^S_{clf}(y_{nc}), \bar{Y}^U)] \quad (8)$$

Algorithm 1 SUPERFICIAL TRAINING SCHEME

Train ResNet101 on Imagenet
Randomly initialize model parameters
Initialize dataset $\mathcal{D}_{ada} \leftarrow \emptyset$
Train the generative model to describe data from class c by $p(\mathbf{x}|c, \Theta)$ $\forall c \in [1, ...C]$
$Source \longleftarrow$ data sampled from generative model
$Target \longleftarrow$ unlabelled test data
Initialize weights of $G^T : S \longrightarrow T$,$G^S : T \longrightarrow S$, D^S, D^T
Augment class labels to G^T and G^S
for epoch $i = 1 : K$ **do**
    Randomly sample $o^s_i \sim Source$, $o^t_i \sim Target$
    Perform ADA

Algorithm 2 All the notations have same meaning as that in the running text. *Params*() return the parameters of the model in (.)

Input: Maximum iterations N_{step}, batch size n, the iteration number of discriminator in a loop n_d, RMSprop hyperparameters α, class attributes $\{a_c\}_{c=1}^{S+U}$, ADAM hyperparameters $\alpha_2, \beta_1, \beta_2$
// *Pre-training*
for $c = 1...S + U$ **do**
    $\zeta_{\mathbf{c}} = f_{\mathbf{\Theta}}(\mathbf{a_c})$
for iter $= 1, ..., N_{step}$ **do**
    Sample minibatch of examples $x_1, x_2, ...x_n$
    $L = \sum_{c=1}^{S} \sum_{n:y_n=c} (x_n - f_\mu(a_c))^T [f_\Sigma(a_c)] (x_n - f_\mu(a_c))$
    $\mathbf{\Theta} \leftarrow \text{Adam}(\nabla_{\mathbf{\Theta}} L, \mathbf{\Theta}, \alpha_2, \beta_1, \beta_2)$
// *Adversarial Domain Adaptation*
Initialize $\{G, D, C\}^{T,S}$
$\lambda_d = Params(D^T, D^S, C^T, C^S)$
$\lambda_g = Params(G^T, G^S)$
for iter $= 1, ..., N_{step}$ **do**
    Sample minibatch $x_1, x_2, ...x_n$ from test data for training G^T
    Sample minibatch $x'_1, x'_2, ...x'_n$ from class conditional distributions for training G^S
    Compute the overall loss $\mathcal{L}$ using Eq.1
    $\lambda_g \leftarrow \text{RMSprop}(\nabla_{\lambda_g} \mathcal{L}, \lambda_g, \alpha)$
    for $t = 1$, ..., n_d **do**
        Sample minibatch of examples x
        Compute the overall loss $\mathcal{L}$ using Eq.1
        $\zeta_d \leftarrow \text{RMSprop}(\nabla_{\lambda_d} \mathcal{L}, \lambda_d, \alpha)$

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14:8, 2012.

[3] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.