Representation Learning via Adversarially-Contrastive Optimal Transport Supplementary Material

Anoop Cherian [1] **Shuchin Aeron** [2]

1. Proof of Theorem 1

Proof. Here we will prove a slightly general form of Theorem 1. We begin by noting that,

$$\mathcal{P}_k^2 \triangleq \max_{\mathbf{U}:\mathcal{S}(d,k)} \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \mathbb{E}_\pi \|\mathbf{U}^\top\mathbf{x} - \mathbf{U}^\top\mathbf{y}\|^2,$$
$$= \max_{\mathbf{U}:\mathcal{G}(d,k)} \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \mathbb{E}_\pi \|\mathbf{U}\mathbf{U}^\top\mathbf{x} - \mathbf{U}\mathbf{U}^\top\mathbf{y}\|^2$$

and

$$\mathcal{S}_k^2 \triangleq \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \max_{\mathbf{U}\in\mathcal{S}(d,k)} \mathbb{E}_\pi \|\mathbf{U}^\top\mathbf{x} - \mathbf{U}^\top\mathbf{y}\|^2.$$
$$= \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \max_{\mathbf{U}\in\mathcal{G}(d,k)} \mathbb{E}_\pi \|\mathbf{U}\mathbf{U}^\top\mathbf{x} - \mathbf{U}\mathbf{U}^\top\mathbf{y}\|^2.$$

Now,

$$\mathcal{C}_k^2 = \max_{\mathbf{U}\in\mathcal{G}(d,k)} \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \mathbb{E}_\pi \|\mathbf{U}\mathbf{U}^\top\mathbf{x} - \mathbf{y}\|^2$$
$$= \max_{\mathbf{U}\in\mathcal{G}(d,k)} \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \mathbb{E}_\pi \{\|\mathbf{U}\mathbf{U}^\top\mathbf{x} - \mathbf{U}\mathbf{U}^\top\mathbf{y}\|^2 + \|(\mathbf{I}_d - \mathbf{U}\mathbf{U}^\top)\mathbf{y}\|^2\}$$
$$\geq \max_{\mathbf{U}\in\mathcal{G}(d,k)} \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \mathbb{E}_\pi \|\mathbf{U}\mathbf{U}^\top\mathbf{x} - \mathbf{U}\mathbf{U}^\top\mathbf{y}\|^2 \quad (1)$$
$$= \mathcal{P}_k^2 \quad (2)$$

Now since $\max\min \leq \min\max$,

$$\max_{\mathbf{U}\in\mathcal{G}(d,k)} \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \mathbb{E}_\pi \|\mathbf{U}\mathbf{U}^\top\mathbf{x} - \mathbf{y}\|^2$$
$$\leq \min_{\pi\in\Pi(\mu_{\mathbf{X}},\nu_{\mathbf{Y}})} \max_{\mathbf{U}\in\mathcal{G}(d,k)} \{\mathbb{E}_\pi \|\mathbf{U}\mathbf{U}^\top\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top\mathbf{y}_j\|^2 + \mathbb{E}_\pi \|(\mathbf{I}_d - \mathbf{U}\mathbf{U}^\top)\mathbf{y}\|^2\}$$
$$\leq \mathcal{S}_k^2 + \max_{\mathbf{U}\in\mathcal{G}(d,k)} \mathbb{E}_{\nu_{\mathbf{Y}}} \|(\mathbf{I}_d - \mathbf{U}\mathbf{U}^\top)\mathbf{y}\|^2. \quad (3)$$

The term $\max_{\mathbf{U}\in\mathcal{G}(d,k)} \mathbb{E}_{\nu_{\mathbf{Y}}} \|(\mathbf{I}_d - \mathbf{U}\mathbf{U}^\top)\mathbf{y}\|^2 = \sum_{\ell=k+1}^{d} e_\ell(\mathbf{\Sigma}_{\mathbf{Y}})$ where $e_1, e_2, ..., e_d$ are the eigenvalues of the Gram matrix arranged in increasing order. □

[1]Mitsubishi Electric Research Labs, Cambridge, MA. [2]Tufts University, Medford, MA. Correspondence to: Anoop Cherian <cherian@merl.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

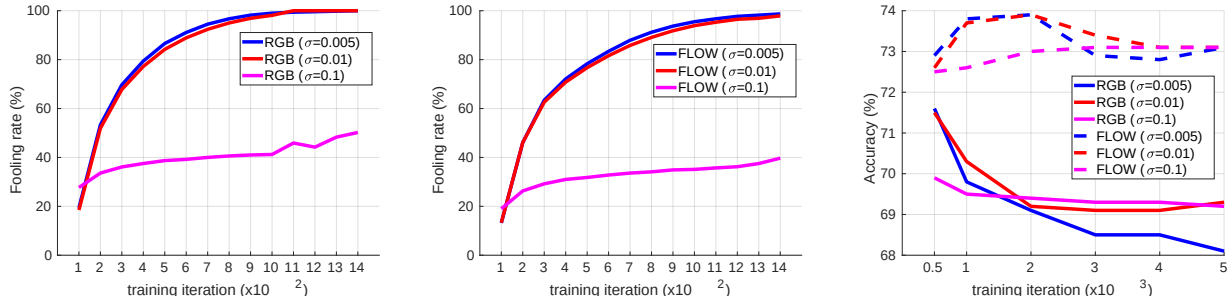
2. Additional Experiments

In this section, we detail our neural architectures in our COT framework and provide ablative studies of the various choices in our setup.

Datasets and Features: As noted in the main paper, we use two datasets, namely (i) the JHMDB dataset, and (ii) the HMDB dataset. For the former, we explore our scheme using two types of features: (i) vgg-16 features, and (ii) I3D features. The vgg-16 features are 4096 dimensional each for every frame in the sequence. That is, we have feature matrices of size $4096 \times n$ and $4096 \times n-1$ for the RGB and optical flow respectively, where n denotes the number of frames in the sequence. As for the I3D features, they are 1024 dimensional each and are extracted from the average pooling layer (after the "max_5c" layer) of the Inception V3 network (Carreira & Zisserman, 2017). These features are produced from short clips, in which the I3D network takes clips consisting of 8 consecutive video frames, and produces one 1024 dimensional feature for that short clip. We use a sliding window with a temporal stride of 2 frames to generate our feature matrix for the two streams. Thus, in our setup, for a sequence with n frames, we will have feature matrices of size $1024 \times \lfloor \frac{n}{2} \rfloor$ and $1024 \times \lfloor \frac{n-1}{2} \rfloor$ for the RGB and flow streams respectively. Note that the features (from either network) are the outputs of ReLU activations and thus are all non-negative. We also normalize these features to have unit-norm.

Baseline Networks and Training: As alluded to in the main paper, we have not trained the baseline networks ourselves as our goal is to demonstrate the advantages of adversarially constrastive optimal transport on features extracted from off-the-shelf neural models. To this end, for the vgg-16 features on the JHMDB dataset, we directly use the features provided to us by the authors of (Cherian et al., 2017). As is mentioned in that paper, these features were infact produced using a network that was fine-tuned on the JHMDB dataset. For the I3D features, we used a ImageNet+Kinetics pretrained I3D network implemented in PyTorch from a public git-hub repo[1] to extract the features as described above.

[1]https://github.com/piergiaj/pytorch-i3d



(a) RGB (b) FLOW (c) ACOT Accuracy

Figure 1. Fooling rates for RGB stream (Figure 1(a)) and FLOW stream (Figure 1(b)) using I3D network on the HMDB dataset against the number of WGAN training iterations. We plot for three different variances of the Normal distribution, i.e., $\sigma = 0.005, 0.01, 0.1$. Note that standard deviation of the features is about 0.008. As we see from the two plots, with a lower $\sigma = 0.005, 0.01$, the WGAN learns to generate adverarial pertubations with 100% fooling rate in about 1000 iterations, however with a larger $\sigma = 0.1$, the network could achieve about 50% fooling rate on average. On the right 1(c), we plot the validation accuracy (of ACOT) against the respective training iterations on the left. For RGB, higher-fooling rates seem to affect performances, however, the effect is reversed on the FLOW stream. This is perhaps because the RGB stream of I3D does not capture any useful temporal cues.

2.1. Neural architectures

Apart from the baseline feature-generating neural networks as described above, our framework has two other neural sub-modules, namely (i) the Wasserstein GAN (WGAN) framework for generating the adversarial samples, and (ii) the classifier to ensure the samples are adversarial.

Generator and Discriminator: Our generator g has the following neural composition:

$$g := [\mathrm{FCN}(d,d), \mathrm{ReLU}(), \mathrm{FCN}(d,d), \mathrm{ReLU}(), \mathrm{FCN}(d,d)]$$

where d is the input feature dimensionality (4096 for vgg-16 and 1024 for I3D), where this input is a noise sample from a multivariate normal distribution. Our discriminator has a similar structure, except that the final layer uses $\mathrm{FCN}(d,1)$.

Classifier: As our representations for the sequences are linear subspaces, we decided to have the adversarial classifier also be limited in capacity, and thus we used a linear classifier for ζ in (10). Specifically, our classifier consists of a single $\mathrm{FCN}(d,c)$, where c denotes the number of data classes. We attempted adding more layers and non-linearities to this classifier, however we found that such attempts made it difficult for the generator to learn the perturbations, and also the learned perturbations were difficult to be separated using the linear subspaces U in our ACOT scheme.

2.2. Adversarial Training

We used RMSprop for training our models. We used a learning rate of $1e-4$ for the generator and discriminator, and for the classifier. We trained the classifier for 500 iterations and it achieves roughly 80% accuracy on the input features (on the training set). More training resulted in overfitting, and thus posed difficulties when training the subsequent adversarial network. For WGAN, we adapted the public implementation from the authors of (Arjovsky et al., 2017). This code uses 5 discriminative updates for every generator updates, which we also found to be useful in our setup. We measured the quality of the generated perturbations via the fooling rates on the positive samples. Specifically, the generated random perturbations are added to the original data samples (positives), passed through a $\mathrm{ReLU}()$, and then normalized to unit norm (note that all our data is unit-normalized) to produce the negative samples. Thus, if c is the correct class label that a classifier ζ produces on an input $\mathbf{x}$, then $\mathbf{y} = \frac{\mathrm{ReLU}(\mathbf{x}+g(\mathbf{z}))}{\|\mathrm{ReLU}(\mathbf{x}+g(\mathbf{z}))\|}$, where $\mathbf{z} \sim \mathcal{N}(\overline{\mathbf{X}}, \sigma^2 I)$ is classified as $\bar{c}$ by ζ, where $\bar{c}$ means the class c has the lowest likelihood of being predicted, i.e., $c = \mathrm{softmin}(\zeta(\mathbf{y}))$. We define fooling rate as the performance of the generator to produce a $\mathbf{y}$ that fools ζ as described. Figure 1 show the trend in training the WGAN for various choices of σ and its impact on the ACOT performance. Please see the text accompanying Figure 1 for the empirical analysis. Going by that analysis, we use $\sigma = 0.01$ in our experiments.

References

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

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

Cherian, A., Fernando, B., Harandi, M., and Gould, S. Generalized rank pooling for activity recognition. In *CVPR*, 2017.