Supplementary Material to

Batch Weight for Domain Adaptation with Mass Shift

Mikołaj Bińkowski, R Devon Hjelm, and Aaron Courville

Appendix A. One-sided batch weight

Assume that $\mathbb{P}_x$ and $\mathbb{Q}_y$ are *source* and *target* measures on domains $\mathcal{X}$ and $\mathcal{Y}$, respectively. We assume that correct domain transfer from $\mathcal{X}$ to $\mathcal{Y}$ can be represented by joint distribution $\mathbb{P}_{xy}$ such that the marginal $\mathbb{P}_y = \mathbb{E}_x \mathbb{P}_{xy}$ covers the target distribution

$$\text{supp}\mathbb{Q}_y \subset \text{supp}\mathbb{P}_y. \tag{7}$$

This assumption is much weaker than equality $\mathbb{Q}_y = \mathbb{P}_y$ which most domain transfer models implicitly assume.

We would like to learn to transfer (possibly non-deterministically) by training a generator function $G : \mathcal{X} \to \mathcal{Y}$ that mimics the conditional $\mathbb{P}_{y|x}$. Let $D \in \mathcal{L} = \text{Lip1}(\mathcal{Y}, \mathbb{R})$ be a Wasserstein discriminator (we will stick to the Wasserstein framework, however similar arguments can be derived in general for any divergence).

The Wasserstein GAN optimizes the following loss function,

$$\inf_G \sup_{D \in \mathcal{L}} \mathbb{E}_{X \sim \mathbb{P}_x}[D(G(X))] - \mathbb{E}_{Y \sim \mathbb{Q}_y}[D(Y)], \tag{8}$$

which is equivalent to,

$$\inf_G \sup_{D \in \mathcal{L}} \mathbb{E}_{Y \sim \mathbb{P}_y^G}[D(Y)] - \mathbb{E}_{Y \sim \mathbb{Q}_y}[D(Y)]), \tag{9}$$

where $\mathbb{P}_y^G = G\#\mathbb{P}_x$ is a push forward measure of $\mathbb{P}_x$ through G.

This optimization suffers from the problem of mode-mass imbalance, as in general we *do not* want $\mathbb{P}_y^G$ to match with $\mathbb{Q}_y$. However we do expect $\mathbb{P}_y^G$ to cover all the modes of $\mathbb{Q}_y$, as $\mathbb{P}_y$ does due to the assumption 7. If this is true, then the Radon-Nikodym derivative $\frac{d\mathbb{Q}_y}{d\mathbb{P}_y^G}$ exists and,

$$\begin{aligned}\mathbb{E}_{Y \sim \mathbb{Q}_y}[D(Y)] &= \mathbb{E}_{Y' \sim \mathbb{P}_y^G}\left[D(Y')\frac{d\mathbb{Q}_y}{d\mathbb{P}_y^G}(Y')\right] \\ &= \mathbb{E}_{X \sim \mathbb{P}_x}\left[D(G(X))\frac{d\mathbb{Q}_y}{d\mathbb{P}_y^G}(G(X))\right]. \end{aligned} \tag{10}$$

$\frac{d\mathbb{Q}_y}{d\mathbb{P}_y^G}(G(\cdot))$ is an unknown function, and therefore the last expression in the above equation cannot be obtained directly. However, we may try to estimate it using e.g. neural network W,

$$\inf_{W \in \mathcal{W}} \left(\mathbb{E}_{X \sim \mathbb{P}_x}[D(G(X) \cdot W(X)] - \mathbb{E}_{Y \sim \mathbb{Q}_y}[D(Y)]\right)^2, \tag{11}$$

where $\mathcal{W} = \{W : \mathbb{E}_{X \sim \mathbb{P}_x}[W(X)] = 1, W \geq 0\}$. Such constraint can easily be enforced by a softmax layer computed over samples in the batch.

Problems 8 and 11 together motivate the following optimization criterion for Wasserstein batch-weighted domain transfer:

$$\inf_{G,W} \sup_D \left(\mathbb{E}_{X \sim \mathbb{P}_x}[D(G(X)) \cdot W(X)] - \mathbb{E}_{Y \sim \mathbb{Q}_y}[D(Y)]\right)^2 \tag{12}$$

Algorithm 2 One-sided Batch Weight

Given: $\mathbb{P}_x$ and $\mathbb{Q}_y$ - source and target distributions
Given: d - number of discriminator steps per generator step, N - total training steps, m - batch size
Initialize generator G, discriminator D and weighting W networks' parameters $\theta_G, \theta_D, \theta_W$.
for $k = 1$ **to** n **do**
  `# generator - weight step`
  Sample $x_1, \ldots, x_m \sim \mathbb{P}_x$ and $y_1, \ldots, y_m \sim \mathbb{Q}_y$.
  $w_1, \ldots, w_m \leftarrow softmax(W(x_1), \ldots, W(x_m))$
  $L^- \leftarrow \sum_{i=1}^m D(G(x_i)) \cdot w_i$
  $L^+ \leftarrow \sum_{i=1}^m D(y_i) \cdot \frac{1}{m}$
  $\theta_G \leftarrow \text{Adam}\left(\nabla_G L^-, \theta_G\right)$
  $\theta_W \leftarrow \text{Adam}\left(\nabla_W \left[(L^- - L^+)^2\right], \theta_W\right)$
  for $j = 1$ **to** d **do**
    Sample $x_1, \ldots, x_m \sim \mathbb{P}_x$ and $y_1, \ldots, y_m \sim \mathbb{Q}_y$.
    $w_1, \ldots, w_m \leftarrow softmax(W(x_1), \ldots, W(x_m))$
    $L \leftarrow \sum_{i=1}^m D(G(x_i)) \cdot w_i - \sum_{i=1}^m D(y_i) \cdot \frac{1}{m}$
    $\theta_D \leftarrow \text{Adam}\left(-\nabla_D L, \theta_D\right)$
  end for
end for

Although optimization of the Wasserstein objective is technically equivalent to optimization of its square, in practice it is more convenient to use standard WGAN loss. Therefore the training procedure optimizes slightly different losses for weighting and generator networks. The proposed procedure for batch-weighted domain transfer is shown in Algorithm 2.

A.1. Possible issues

Samples from modes in the source domain that are underrepresented in the target domain might be transferred poorly if too low weights are assigned to them by the weighting network. This problem essentially stems from the fact that we are weighting the *generated* samples, not the *target* ones, which biases the generator so that it values generated samples according to their frequency in target domain $\mathbb{Q}_y$, even though we care about quality of $\mathbb{P}_y^G$, which stems from $\mathbb{P}_x$.

Appendix B. Architecture details

Tables 3 and 2 present generator and joint-discriminator architectures used in experiments with 32x32 images.

image x	concat(x, y)	image y
4x4 conv(32) (x_1)	4x4 conv(64) (xy_1) concat(x_1, xy_1, y_1)	4x4 conv(32) $(y1)$
4x4 conv(64) (x_2)	4x4 conv(128) (xy_2) concat(x_2, xy_2, y_2)	4x4 conv(64) (xy_2)
	2 x ResBlock(128)	
4x4 conv(128) (x_3)	4x4 conv(256) (xy_3) concat(x_3, xy_3, y_3)	4x4 conv(128) (y_3)
	4x4 conv(256)	
	fc $1024 \to 256$	
	fc $256 \to 1$	

Table 2: Joint discriminator architecture. Each convolution has stride 2. Residual blocks [13] contain two 3x3 convolutions and skip connection.

image $x \in \mathbb{R}^{c \times 32 \times 32}$	noise $z \in \mathbb{R}^d$
KxK conv(64), stride s	repeat $(32/s \times 32/s)$ (z')
2 x ResBlock(64)	
1x1 conv(64), stride 1 (x')	
concat(x', z') 1x1 conv(64), stride 1	
2 x ResBlock(64)	
KxK transposed conv(c), stride s	

Table 3: Generator network architecture. c denotes number of channels (1 for greyscale, 3 for rgb), s stride and K kernel size. For MNIST - SR-MNIST task $s = 2, K = 4$, for MNIST - SVHN $s = 1, K = 1$. Residual blocks[13] contain two 3x3 convolutions and skip connection.

B.1. Weighting network

We considered three different ways of modelling weight network W. Given batches of pairs of real and generated samples $(\mathbf{x}, G_{yx}(\mathbf{x})), \mathbf{x} \sim \mathbb{P}_x^n$ and $(G_{xy}(\mathbf{y}), \mathbf{y}), \mathbf{y} \sim \mathbb{Q}_y^n$ we may get the weights $w_{\mathbf{x}}, w_{\mathbf{y}}$ using each of the following architectures

1. (W concatenates two arguments)

$$W : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$
$$w_{\mathbf{x}} = \sigma(W(\mathbf{x}, G_{xy}(\mathbf{x}))), \quad w_{\mathbf{y}} = \sigma(-W(G_{yx}(\mathbf{y}), \mathbf{y})).$$

2. (W takes one argument)

$$W : \mathcal{X} \to \mathbb{R},$$
$$w_{\mathbf{x}} = \sigma(W(\mathbf{x})), \quad w_{\mathbf{y}} = \sigma(-W(G_{yx}(\mathbf{y}))).$$

3. (composite)

$$W_x : \mathcal{X} \to \mathbb{R}, \quad W_y : \mathcal{Y} \to \mathbb{R},$$
$$w_{\mathbf{x}} = \tfrac{1}{2}\left(\sigma(W_x(\mathbf{x})) + \sigma(-W_y(G_{xy}(\mathbf{x})))\right),$$
$$w_{\mathbf{y}} = \tfrac{1}{2}\left(\sigma(-W_x(G_{yx}(\mathbf{y}))) + \sigma(W_y(\mathbf{y}))\right).$$

The weight network(s) W (W_x, W_y) were the same as DCGAN discriminator [29] with four convolutional layers and 64 features in the first layer.

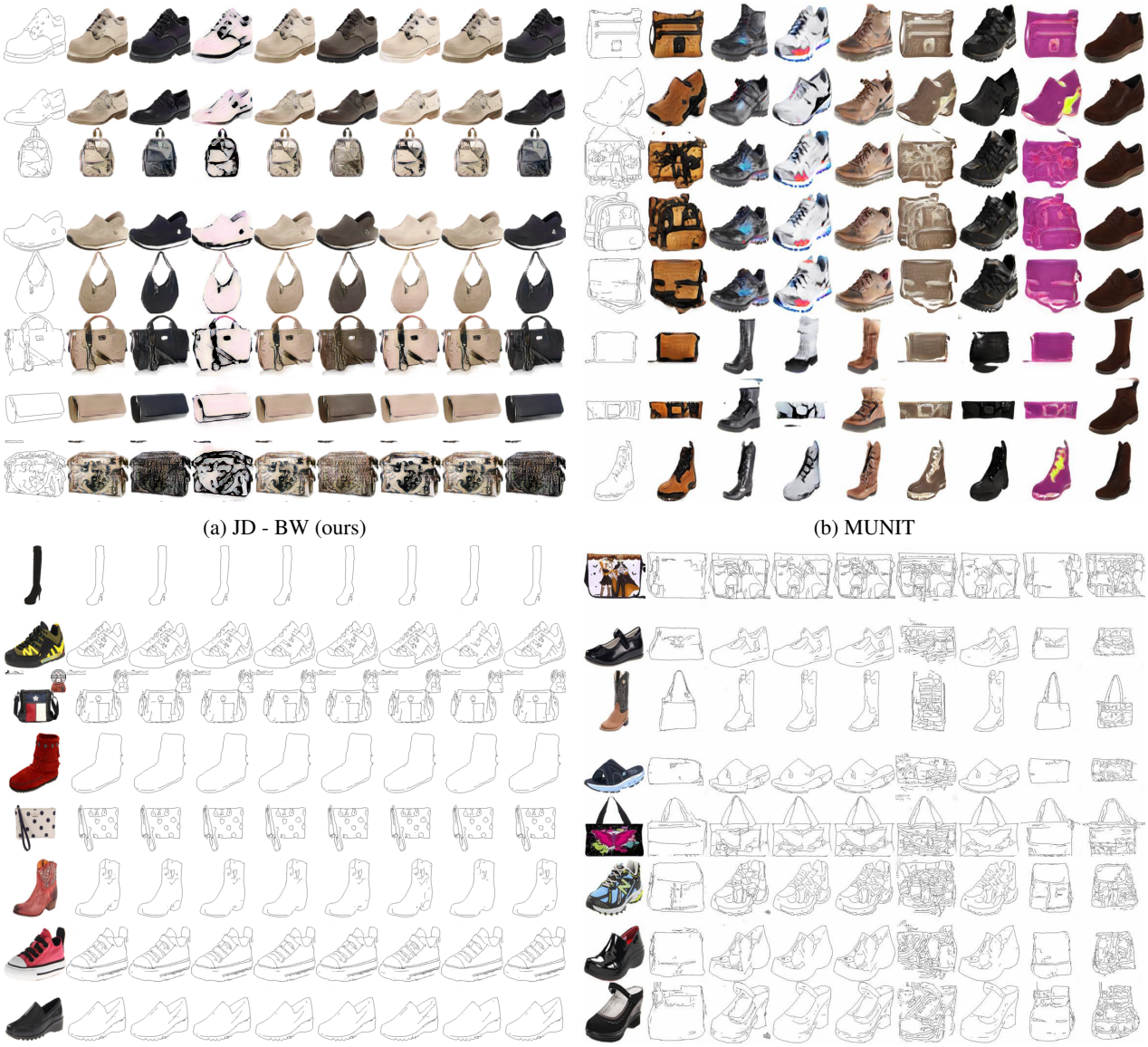
We found the last (composite) architecture to be the most stable one. The first approach, although the most natural, most probably suffers from the fact that it takes longer for joint samples to look similar to each other than it does for the marginals.

Appendix C. Role of the noise term

We have observed two types of failures made by MUNIT-trained models in the presence of mode-mass imbalance, both related to what these models encode in the noise term.

In the first one, the class/mode is kept in a transferred sample depending on the noise term. This has been observed in Edges to Shoes&Bags task, see Figure 7b and 7d. In multimodal domain transfer, noise term should only encode the features which are not present in the source domain. In this task, however, the MUNIT-trained model encoded the conditional mode: some noise values caused the generator 'forget' the source image and generate one over-represented in the target domain (regardless of the source image mode/class).

The second issue is the amount of the source image features retained in the transferred one. In CelebA to Portrait transfer, MUNIT tends to keep very few features of the source image (e.g. position of eyes and and nose) and model all high-level features using noise term. As shown in Figure 8b, samples obtained with the same source image are less similar to each other than those generated using the same noise term. With JD-BW model (Fig. 8a) it is the opposite: generators retain much more features of the source image, while the noise terms determines only the style of the generated portraits.



(a) JD - BW (ours)

(b) MUNIT

(c) JD - BW (ours)

(d) MUNIT

Figure 7: Edges to Shoes&Bags transfer with fixed noise values. In each picture first column represents original images; other columns present transfer with noise term fixed per column and source picture fixed per row. In MUNIT, some noise terms (e.g. columns 3, 4, 5 in (b); 2, 6, 8 and 9 in (d)) 'deactivate' source images to produce images from over-represented class in the target domain.

(a) JD - BW (ours)

(b) MUNIT

Figure 8: CelebA to Portrait transfer with fixed noise values. In both pictures the first columns represent original images; other columns present transfer with noise term fixed per column and source CelebA photo fixed per row. The proposed JD-BW model retains much more facial features, using noise term to encode mostly the portrait style. MUNIT, on the other hand, infers most of the features from noise term, keeping only the position from the source photo.