Correcting Nuisance Variation using Wasserstein Distance

Gil Tabak∗, Minjie Fan2, Samuel J. Yang3, Stephan Hoyer4, and Geoff Davis5

1Stanford University. Stanford, CA 94305
tabak.gil@gmail.com
2,3,4,5Google Inc. Mountain View, CA 94043
{mjfan, samuely, shoyer, geoffd}@google.com

November 6, 2017

Abstract

Profiling cellular phenotypes from microscopic imaging can provide meaningful biological information resulting from various factors affecting the cells. One motivating application is drug development: morphological cell features can be captured from images, from which similarities between different drugs applied at different dosages can be quantified. The general approach is to find a function mapping the images to an embedding space of manageable dimensionality whose geometry captures relevant features of the input images. An important known issue for such methods is separating relevant biological signal from nuisance variation. For example, the embedding vectors tend to be more correlated for cells that were cultured and imaged during the same week than for cells from a different week, despite having identical drug compounds applied in both cases. In this case, the particular batch a set of experiments were conducted constitutes the domain of the data; an ideal set of image embeddings should contain only the relevant biological information (e.g. drug effects). We develop a method for adjusting the image embeddings in order to ‘forget’ domain-specific information while preserving relevant biological information. To do this, we minimize a loss function based on the Wasserstein distance. We find for our transformed embeddings (1) the underlying geometric structure is preserved and (2) less domain-specific information is present.

1 Introduction

In the framework where our approach is applicable, there are some inputs (e.g. images) and a map \( F \) sending the inputs to vectors in a low-dimensional space which summarizes information about the inputs. \( F \) could either be engineered using specific image features, or learned (e.g. using deep neural networks). We will call these vectors ‘embeddings’ and the space to which they belong the ‘embedding space’. Each input may also have corresponding semantic labels and domains, and for inputs with each label and domain pair, \( F \) produces some distribution of embeddings. Semantically meaningful similarities between pairs of inputs can then be assessed by the distance between their corresponding embeddings, using some chosen distance metric. Ideally, the embedding distribution of a group of inputs depends only on their label, but often the domain can influence the embedding distribution as well. We wish to find an additional map to adjust the embeddings produced by \( F \) so that the distribution of adjusted embeddings for a given label is independent of the domain, while still preserving semantically meaningful distances between distributions of inputs with different labels.

The map \( F \) can be used for phenotypic profiling of cells. In this application, images of biological cells perturbed by one of several possible biological stimuli (e.g. various drug compounds at different doses, some of which may have unknown effects) are mapped to embeddings, which are used to reveal similarities among the applied perturbations.

There are a number of ways to extract embeddings from images of cells. One class of methods such as that used by Ljosa et al. (2013) relies on extracting specifically engineered features. In the recent work by Ando et al. (2017), a Deep Metric Network pre-trained on consumer photographic images (not microscope images of cells) described in Wang et al. (2014) was used to generate embedding vectors from cellular images, and it was shown that these clustered drug compounds by their mechanisms of action (MOA) more effectively. See Figure 1 for example images of the different MOAs.

∗Gil Tabak worked as an intern at Google during the summer of 2017
Currently one of the most important issues with using image embeddings to discriminate the effects of each treatment (i.e. a particular dose of a drug, the ‘label’ in the general problem described above) on morphological cell features is nuisance factors related to slight uncontrollable variations in each biological experiment. Many cell imaging experiments are organized into a number of batches of experiments occurring over time, each of which contains a number of sample plates (typically 3-6), each of which contains individual wells in which thousands of cells are grown and treatments are applied (typically around 96 wells per plate). For this application, the ‘domain’ is an instance of one of these hierarchical levels, and embeddings for cells with a given treatment tend to be closer to each other within the same domain than from a different one. For example, the experimentalist may apply slightly different concentrations or amounts of a drug compound in two wells in which the same treatment was anticipated. Another example is the location of a particular well within a plate or the order of the plate within a batch, which may influence the rate of evaporation, and hence, the appearance of the cells. Finally, ‘batch’ effects may result from differences in experiment conditions (temperature, humidity) from week to week; they are various instances of this hierarchical level that we will consider as ‘domains’ in this work.

Our approach addresses the issue of nuisance variation in embeddings by transforming the embedding space in a possibly domain-specific way in order to minimize the variation across domains for a given treatment. Our method makes few assumptions about the probability distributions of the embedding vectors and can be extended in a number of ways (see Section 4.1).

2 Method

2.1 Problem description

Denote the embedding vectors $x_{t,d,p}$ for $t \in T$, $d \in D$, and $p \in I_{t,d}$, where $T$ and $D$ are the treatment and domain labels respectively, and $I_{t,d}$ is the set of indices for embeddings belonging to treatment $t$ and domain $d$. Suppose that $x_{t,d,p}$ were sampled from a probability distribution $P_{t,d}$. Our goal is to ‘forget’ the nuisance variation in the embeddings, which we formalize in the following way. We wish to find maps $A_d$ transforming the embedding vectors such that the transformed marginals $\tilde{P}_{t,d}$ have the property that for each $t \in T$ and $d_i, d_j \in D$, $\tilde{P}_{t,d_i} \approx \tilde{P}_{t,d_j}$ (for some suitable metric between distributions). Intuitively, the transformations $A_d$ can be thought of as correcting a domain-specific perturbation. We do not have ‘source’ and ‘target’ distributions, and instead perturb all the embedding distributions simultaneously. The transformations $A_d$ should be small to avoid distorting the underlying geometry of the embedding space, and since we do not expect nuisance variation to be very large.

2.2 Wasserstein Distance

The 1-Wasserstein distance between two probability distributions $P_r$ and $P_g$ on a compact metric space $\chi$ with metric $\delta$ is given by

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y) \sim \gamma} \delta(x,y).$$

Here $\Pi(P_r, P_g)$ is the set of all joint distributions $\gamma(x,y)$ whose marginals are $P_r$ and $P_g$. This can be intuitively interpreted as the minimal cost of a transportation plan between the probability masses of $P_r$ and $P_g$. In our application, the metric space was $\mathbb{R}^n$ and $\delta$ was the Euclidean metric. We will refer to the 1-Wasserstein as the Wasserstein distance throughout.

We chose to use the 1-Wasserstein distance as a metric between probability distributions. This metric avoids problematic vanishing gradients during training that are known to occur when using metrics based on the KL-divergence, such as the cross entropy [Arjovsky et al. (2017)].

The Wasserstein distance does not have a closed form except for a few special cases, and must be approximated in some way. The Wasserstein distance is closely related to the maximum mean discrepancy (MMD) approximated in [Shaham et al. (2017)] using an empirical estimator based on the kernel method. This method requires selecting a kernel and relevant parameters. We choose instead to use a method based on the ideas in [Arjovsky et al. (2017)] and Gulrajani et al. (2017) to train a neural network to estimate the Wasserstein distance. To do this, first apply the Kantorovich-Rubinstein duality:

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} E_{x \sim P_r}[f(x)] - E_{x \sim P_g}[f(x)].$$

2
Here, $P_r$ and $P_g$ are two probability distributions. The function $f$ is in the space of Lipschitz functions with Lipschitz constant at most 1. To estimate the Wasserstein distance, a function $f$ can be optimized while keeping the norm of its gradient to be less than one. We will call $f$ the ‘Wasserstein function’ in this work.

2.3 Network Architecture

In the current implementation, the domain-specific transformations $A_d$ map input embedding vectors to transformed embedding vectors of the same dimension. Each $A_d$ is implemented as an affine transformation $A_d(x) = M_d x + b_d$. This is motivated by assuming the perturbations applied to the embeddings can be sufficiently captured using a truncated Taylor expansion. Collectively denote the parameters for the transformations $A_d$ by $\theta_d$. If a particular treatment $t$ is replicated across two or more domains $d_1, d_2, ..., d_k$, the Wasserstein distance between the distributions of transformed vectors is estimated for all same-treatment domain pairs. Notice the parameters for estimating the Wasserstein distance for each $t$ and pair $d_i, d_j$ are different. Collectively denote all Wasserstein estimation parameters by $\theta_W$. We consider the loss function

$$L(\theta_T, \theta_W) = \sum_{t \in T} \sum_{d_i, d_j \in D, i \neq j} [W_{t,d_i,d_j}(\theta_T, \theta_W) - g_{t,d_i,d_j}(\theta_T, \theta_W)] + R(\theta_T).$$

The function $R(\theta_T)$ is a regularizer for the learned transformation whose purpose is to preserve the geometry of the original embeddings. For simplicity we did not include a regularization term, and instead rely on early stopping.

Using one of these methods is necessary since otherwise optimizing $L$ may result in embeddings which contain no treatment information (for example, if all embeddings are transformed to a single point).

$W_{t,d_i,d_j}$ above is used to approximate the Wasserstein distance between the transformed embeddings of domains $d_i$ and $d_j$ for treatment $t$. It is given by

$$W_{t,d_i,d_j}(\theta_T, \theta_W) = \frac{1}{N} \sum_{p \in I_{t,d_i}} f_{t,d_i,d_j}(A_d(x_{t,d_i,p}; \theta_T); \theta_W)$$

$$- \frac{1}{N} \sum_{q \in I_{t,d_j}} f_{t,d_i,d_j}(A_d(x_{t,d_j,q}; \theta_T); \theta_W).$$

Each Wasserstein function $f_{t,d_i,d_j}$ depends on the parameters $\theta_W$, while each transformation $A_d$ depends on the parameters $\theta_T$. We assume that $N = |I_{t,d_i}| = |I_{t,d_j}|$, where $| \cdot |$ indicates the size of a set. In practice, the sets $I_{t,d}$ are picked as minibatches for stochastic gradient descent. Each of the terms $g_{t,d_i,d_j}$ is a gradient penalty defined below in (eq. 7).

Each Wasserstein function should be Lipschitz with Lipschitz constant 1. For differentiable functions, this is equivalent to the norm being bounded by 1 everywhere. We use an approach based on Gulrajani et al. (2017) to impose a soft constraint on the norm of the gradient. Unlike Gulrajani et al. (2017), we impose the gradient penalty only if the gradient norm is greater than 1. Doing so worked better in practice for our application.

Since it is not possible to check the gradient everywhere, we use the same strategy as Gulrajani et al. (2017): choose the points $\epsilon A_d(x_{t,d_i,p_i}; \theta_T) + (1 - \epsilon) A_d(x_{t,d_j,q_i}; \theta_T)$ randomly, where $\epsilon \in U[0, 1]$ and $p_i$ and $q_i$ denote the $k_{th}$ element of $I_{t,d_i}$ and $I_{t,d_j}$, respectively. Denote this set of points by $J_{t,d_i,d_j}$.

Explicitly, we define each gradient penalty $g_{t,d_i,d_j}$ as

$$g_{t,d_i,d_j}(\theta_T, \theta_W) = \frac{1}{N} \sum_{z \in J_{t,d_i,d_j}} H_{t,d_i,d_j}(z; \theta_W),$$

where

$$H_{t,d_i,d_j}(z; \theta_W) = \begin{cases} \lambda (G_{t,d_i,d_j}(z; \theta_W) - 1)^2 & \text{if } G_{t,d_i,d_j}(z; \theta_W) > 1, \\ 0 & \text{otherwise}. \end{cases}$$

$$G_{t,d_i,d_j}(z; \theta_W) = \|\nabla_{\theta_W} f_{t,d_i,d_j}(z; \theta_W)\|_z.$$  

We found the value of $\lambda = 10$ used in Gulrajani et al. (2017) worked well in our application, and fixed it throughout.

To approximate the Wasserstein distance we must maximize over $\theta_W$. Thus our objective is to find

$$\hat{\theta}_T, \hat{\theta}_W = \arg\min_{\theta_W} \arg\max_{\theta_W} L(\theta_T, \theta_W).$$
We used the approach of Ganin & Lempitsky (2015) to transform our minimax problem to a minimization problem by adding a ‘gradient reversal’ between the transformed embeddings and the approximated Wasserstein distances. The gradient reversal is the identity in the forward direction, but negates the gradients used for backpropagation.

3 Empirical Results

The embeddings we used as inputs were generated using the method described in Ando et al. (2017), summarized in Section 3.1.

3.1 Dataset and Preprocessing

We used the images from the BBBC021 dataset available from the Broad Bioimage Benchmark Collection (Ljosa et al., 2012). This image dataset corresponds to cells prepared on 55 plates across 10 separate batches, and imaged in 3 color channels (i.e., stains): for a population of control cells, a compound (DMSO) with no anticipated drug effect was applied, while various other drug compounds were applied to the remaining cells. We computed the corresponding embeddings for each cell image using the method in Ando et al. (2017), summarized as follows. For a 128 by 128 pixel crop around each cell for each of the 3 color channels, a Deep Metric Network generated a 64-dimensional embedding vector. The 3 vectors corresponding to the 3 color channels were concatenated, forming a 192-dimensional embedding for each cell image. Using the embedding vectors for all cells, a Typical Variation Normalization (TVN) was applied in which the controls are whitened. Specifically, in the principal component analysis (PCA) basis of only control cells, an affine transformation is found so that the controls have zero mean and unit covariance. The same transformation is then applied to the embeddings of all cells.

We used the same subset of treatments (concentration of a particular compound) evaluated in Ljosa et al. (2013) and Ando et al. (2017). This subset has 103 treatments from 38 compounds, each belonging to one of 12 known mechanisms of action (MOA) groups. Sample cell images from the 12 MOA groups are shown in Figure 1.

Figure 1: A flowchart describing the procedure we use to generate and remove nuisance variation from image embeddings. The embedding generation is described in Section 3.1 is characterized by $F$, which maps each 128 by 128 color image into a 192-dimensional embedding vector. The nuisance variation removal by our method is denoted by WDN (Wasserstein Distance Network). The 12 images on the right side show representative images of cells treated with drug compounds with one of the 12 known mechanisms of action (MOA), from the BBBC021 dataset (Ljosa et al. 2012).
3.2 Metrics

We use two metrics to evaluate the performance of our algorithm, which serve complementary purposes.

3.2.1 k-Nearest Neighbor Mechanism of Action Assignment

Each compound applied to the BBBC021 dataset has a known MOA. A desirable property of embedding vectors is that compounds with the same MOA should group closely in the embedding space. This property can be assessed in the following way using the ground truth MOA labels for each treatment.

First, compute the mean \( m_X \) of the embeddings for each treatment \( X \) in each domain. Find the nearest \( k \) neighbors \( n_{X,1}, n_{X,2}, ..., n_{X,k} \) of \( m_X \) either (i) not belonging to the same compound or (ii) not belonging to the same compound or batch (domain), and compute the portion of them having the same MOA as \( m_X \). Our metric is defined as the average of this quantity across all treatment instances \( X \) in all domains. If nuisance variation is corrected by transforming the embeddings, we may expect this metric to increase. The reason for excluding same-domain nearest neighbors is to avoid the in-domain correlations from interfering with the metric.

The distance used for these k-NN MOA assignment metrics was the cosine distance, in order to compare with Ando et al. (2017). Our k-NN metrics are generalizations of the 1-NN metrics used in Ljosa et al. (2013) and Ando et al. (2017).

3.2.2 Domain Classification Accuracy per Treatment

Another metric measures how well domain-specific nuisance information has been ‘forgotten’. To do this, for each treatment we train a classifier to predict for each embedding the batch (domain) from the set of possible batches (domains) for that treatment. We evaluated both a linear classifier (i.e. logistic regression) and a random forest with 3-fold cross validation. If nuisance variation is being corrected, the batch (domain) classification accuracy should decrease significantly. Because only the control (DMSO) compound had replicates across experiment batches in our dataset, we trained and evaluated these two batch classifiers on this compound only.

3.3 Procedure

3.3.1 Training and Evaluation Datasets

To avoid overfitting the parameters for embedding transformations \( A_d \), we split the dataset into a training and an evaluation set by compound, for all compounds excluding the control (DMSO). Because the k-NN MOA metrics require at least two compounds for each MOA in the train and evaluation sets, we use the following method to split the compounds. Iterate over all MOAs in random order; for each MOA, if the number of compounds is four or more, randomly split them into two sets and assign the compounds in these sets to the train and evaluation sets, respectively. If the number of compounds for an MOA is three or less, we assign all compounds of that MOA to either the train or evaluation set (whichever has fewer).

The training set is used to determine the early stopping step during model training, and is chosen to maximize the performance over a selected metric. In the results presented, this is done by finding the first maximum of the k-NN MOA metric (not same compound or batch, with \( k=4 \)) as a function of steps for the training set (see Figures 3a and 3b).

3.3.2 Training Procedure

The embedding transformations \( A_d(x) = M_d x + b_d \) were initialized to \( M_d = I, b_d = 0 \), since we wish for the learned transformations to be not too far from the identity transformation.

To approximate each of the Wasserstein functions \( f_{t,d,i,j} \) in (eq. 4), we used a network consisting of softplus layer followed by a scalar-valued affine transformation. The softplus loss was chosen because the Wasserstein distance estimates it produces are less noisy than other kinds of losses and it avoids the issue of all neurons becoming deactivated (which can occur for example when using relu activations).

The dimension of the softplus layer used to approximate each Wasserstein function was 2. Optimization was done using stochastic gradient instead of the sums in (eq. 4). The minibatch size was fixed throughout for simplicity. In the results presented, we used a batch size of 50. Optimization for both classes of parameters \( \theta_T \) and \( \theta_W \) was done using separate RMSProp optimizers. Prior to training \( \theta_T \), we used a ‘pre-training’ period of 20000 steps to obtain a good approximation for the Wasserstein distances. After this, we alternated between adjusting \( \theta_T \) for 40 timesteps and optimizing over \( \theta_W \) for a single timestep.
3.4 Results

The procedure of splitting a dataset into a train and an evaluation sets and determining the optimal embedding transform defined by an early stopping point described in Section 3.3 was repeated 20 times. The figures show the results for one of these 20 training and evaluation splits, while the results across all 20 repeats are summarized in both tables.

We see in Figure 2 that the estimated Wasserstein distance converges as the transformation parameters $\theta_T$ are trained. The k-NN MOA assignment metrics as a function of training step are shown for the training and evaluation sets in Figures 3a and 3b respectively. We did not include a regularizer in this experiment and rely on early stopping. The vertical lines in Figures 3a and 3b indicate the early stopping point. Early stopping was based on where the training set k-NN MOA (not same compound or batch, $k=4$) assignment metric was maximized (see Section 3.2.1). If the metric plateaued, we used the first time when its maximum value was reached.

Figure 4 shows the first two principal components of our transformed embeddings compared to the original TVN embeddings (see Section 3.1) and the embeddings generated by the CORAL method introduced in Sun et al. (2017) and applied by Ando et al. (2017). CORAL applies a domain-specific affine transformation to the embeddings represented as the rows of a matrix $X_d$ from domain $d$ in the following way. On the negative controls only, the covariance matrix across the entire experiment $C$ as well as the covariance $C_d$ in each domain $d$ are computed. Notice that since TVN had already been applied (see Section 3.1), $C = I$. Then, all embedding coordinates in domain $d$ are aligned by matching the covariance structures. Alignment is done by computing the new embeddings $X_d^{\text{aligned}} = X_d R_{d}^{-1/2} R^{1/2}$. Here $R_d = C_d + \lambda I$ and $R = C + \lambda I$ are regularized covariance matrices, with the regularizer $\lambda = 1$, which is the same as that in Ando et al. (2017).

Table 1 shows the k-NN MOA assignment metric (not same compound or batch) of our transformed embeddings compared to the original embeddings as well as the standard error across the 20 runs. We also included the values of this metric for CORAL. We did not find a statistically significant difference between the different methods for this metric. This shows that the geometry of the embeddings is roughly preserved.

Finally, Table 2 compares the average batch classification accuracy for a linear classifier and a random forest classifier for the original TVN embeddings, our embeddings using the Wasserstein distance network (WDN), CORAL embeddings, and for reference, a trivial transformation for which all embeddings are set to zero. For each run, given a classifier and a transformed set of embeddings, we compute the mean accuracy for that classifier using 3-fold cross validation. The average and standard error are then reported across the 20 trials. We see that the batch classification accuracy for the embeddings using our method is substantially smaller than that using TVN or CORAL, indicating good batch forgetfulness. The larger standard error for WDN appears to have been from trials where our simple early stopping criteria was suboptimal, stopping too soon, a limitation we discuss in Section 4.1.

![Figure 2](image-url)

**Figure 2:** The Wasserstein loss function and the gradient penalty terms as a function of the number of steps trained on one of the 20 random train split subset of embeddings from images of cells from the BBBC021 image dataset, after the Wasserstein parameters have been pre-trained for 20000 steps.
(a) k-NN MOA assignment metrics for training set of compounds. The (not same compound or batch, k=4) metric was used to pick the stopping point.

(b) k-NN MOA assignment metrics for evaluation set of compounds.

Figure 3: In the current proposed implementation, the lack of regularization requires choosing an optimal early stopping point, at which most relevant biological information is still preserved while batch nuisance variation has been reduced. The k-NN MOA assignment metrics as a function of training steps for a particular training (Figure 3a) and evaluation (Figure 3b) sets reveal our strategy for selecting this early stopping point. The vertical line indicates where the stopping point was chosen based on the training set. For each 3a and 3b, the top plot shows the ‘not same compound’ metric, and the bottom shows the ‘not same compound or batch metric’ (see Section 3.2.1).

4 Conclusion

We have shown how a neural network can be used to transform embedding vectors to ‘forget’ specifically chosen domain information as indicated by our proposed domain classification metric. The transformed embeddings still preserve the underlying geometry of the space as indicated by the k-NN MOA assignment metrics. Our approach uses the Wasserstein distance and can in principle handle fairly general distributions of embeddings (as long as the neural network used to approximate the Wasserstein function is general enough). Importantly, we do not have to assume that the distributions are Gaussian.

The framework itself is quite general and extendible (see Section 4.1). Unlike methods that use only the controls for adjusting the embeddings, our method can also utilize information from replicates of a treatment across different
Figure 4: A comparison of the first two principal components for embeddings after only preprocessing (TVN) only (first column), embeddings transformed using the CORAL method used in [Ando et al, 2017] (second column), and embeddings transformed using our Wasserstein distance network method (WDN, third column), which illustrates the reduction of batch nuisance variation. Each row corresponds to a random subset of embeddings for cells treated with compounds with a particular MOA, and the colors correspond to specific batches. The first row is specifically the controls (DMSO). The other rows are randomly selected MOAs of the 12 total. Our method is designed to align compounds (in this case using only the control embeddings) across various batches while not distorting the geometry of the embedding space.
Table 1: K-NN MOA assignment (not same compound or batch) evaluation dataset accuracy evaluated on 20 random train-evaluation splits shows no statistical difference among the three methods, suggesting WDN preserves as much relevant biological information as before (TVN only) and compared to CORAL.

| k-NN | TVN only | WDN | CORAL |
|------|----------|-----|-------|
| 1    | 86.8% ± 6.7% | 87.7% ± 6.2% | 87.3% ± 6.8% |
| 2    | 87.7% ± 6.1% | 88.7% ± 5.5% | 88.6% ± 5.8% |
| 3    | 90.1% ± 5.1% | 91.0% ± 4.5% | 90.9% ± 4.9% |
| 4    | 90.2% ± 4.4% | 91.1% ± 4.1% | 90.9% ± 4.4% |

Table 2: Batch (domain) classification accuracy evaluation dataset accuracy evaluated on 20 random train-evaluation splits illustrates WDN removes more batch nuisance variation.

| 3-fold linear regression | TVN only | WDN | CORAL | Trivial transformation |
|--------------------------|----------|-----|-------|------------------------|
| 3-fold random forest     | 46.4% ± 0.4% | 39.4% ± 5.8% | 48.1% ± 0.3% | 16.6% |

domains. However, the dataset used did not have treatment replicates across batches, so we only relied on aligning based on the controls. Thus we implicitly assume that the transformation for the controls matches that of the various compounds. We expect our method to be more useful in the context of experiments where many replicates are present, so that they can all be aligned simultaneously. We expect transformations learned for such experiments to have better generalizability since it would be using available knowledge from a greater portion of the embedding space.

Our approach requires a free parameter either for regularization or early stopping, which we address by cross validation across compounds. We discuss potential future directions below, as well as other limiting issues.

4.1 Future Work

A type of regularization instead of early stopping could be achieved by confining a distance of the transformation away from the identity transformation. This might be a better approach than early stopping since the size of the perturbation is actively kept checked.

Another possible improvement is to use the Cramer distance instead of the Wasserstein distance direction, as in Bellemare et al. (2017). This could reduce the number of steps required to adjust the Wasserstein distance approximation for each step of training the embedding transformation.

Another issue is how to weigh the various Wasserstein distances against each other. This might improve the results if there are many more points from some distributions than others (which happens in the real data). Further, it is unclear how a regularization term should be weighed against the Wasserstein loss terms.

Another extension may involve applying our method hierarchically on the various domains of the experiment. However, this would require replicates on multiple hierarchical levels.

The Wasserstein functions were also approximated with very simple nonlinear functions, and it is possible better results would be obtained using more sophisticated functions capturing the Wasserstein distance and its gradients more accurately. Similarly, The transformations $A_d$ could be generalized from affine to a more general class of functions. As in Shaham et al. (2017), we expect residual networks would make natural candidates for these transformations.

Since the k-NN MOA assignment metric is based on the cosine distance, it is possible better results could be obtained by modifying the metric used to compute the Wasserstein distance accordingly, e.g. finding an optimal transportation plan only in non-radial directions.

Another possibility is to fine-tune the Deep Metric Network used to generate the embeddings instead of training a separate network on its outputs (or perhaps several such networks for the separate image stains used).
Acknowledgments

We would like to thank Mike Ando, Marc Coram, Marc Berndl, Subhashini Venugopalan, Arunachalam Narayanaswamy, Yaroslav Ganin, Luke Metz, Eric Christiansen, Philip Nelson, and Patrick Riley for useful discussions and suggestions.
References

D Michael Ando, Cory McLean, and Marc Berndl. Improving phenotypic measurements in high-content imaging screens. *bioRxiv*, pp. 161422, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Marc G Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pp. 1180–1189, 2015.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature methods*, 9(7):637–637, 2012.

Vebjorn Ljosa, Peter D Caie, Rob Ter Horst, Katherine L Sokolnicki, Emma L Jenkins, Sandeep Daya, Mark E Roberts, Thouis R Jones, Shantanu Singh, Auguste Genovesio, et al. Comparison of methods for image-based profiling of cellular morphological responses to small-molecule treatment. *Journal of biomolecular screening*, 18(10):1321–1329, 2013.

Uri Shaham, Kelly P Stanton, Jun Zhao, Huamin Li, Khadir Raddassi, Ruth Montgomery, and Yuval Kluger. Removal of batch effects using distribution-matching residual networks. *Bioinformatics*, pp. btix196, 2017.

Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pp. 153–171. Springer, 2017.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1386–1393, 2014.