A EXPERIMENTAL SETUP

Optimizer and learning rate schedule. In all our experiments on datasets with natural distribution shift, we use Adam optimizer. When training on the labeled source domain we use a learning rate schedule with cosine decay and initial learning rate of 1e-4 and when adapting to the target domain we use a learning rate of 1e-5 with exponential decay rate of 0.9. For pretraining the models on ImageNet-1k, we use a batch size of 1024 and the learning rate schedule is linear warmup (for 5 epochs) + cosine decay with the base learning rate of 0.1.

For Perturbed Cifar10 experiments, we use SGD with momemtum. During the pretraining stage on the source domain, the learning rate schedule is cosine decay with an initial learning rate of 0.1. In all experiments we use L2 loss with the weight of 1e-5. During the adaptation phase we use a batch size of 256 and the learning rate is constant and set to 1e-3. For experiments on Perturbed Cifar10 with higher number of training steps (20000 steps), we use a lower learning rate of 1e-4 in the adaptation phase.

Neural network architectures. For experiments on dataset with natural shift, we use a ResNet101. For the experiments on Perturbed Cifar10, we use a WideResNet28-10 with a dropout rate of 0.3. For the virtual interpolations in GIFT: we use the first three layers (input, initial convolution layer, and the layer above it).

Training on source during the adaptation phase. We do not train the model on the source data during the adaptation phase. While in some cases this could result in a better performance on both the source and target domain, our assumption here is that there is no reason for the source and target data to be compatible, i.e. it is possible for the model to not be able to fit both distributions simultaneously.

Regularization During adaptation, we use the weight decay of 0.01. In addition to weight decay, we use another regularization term that encourages the model to stay close to its initial state. This regularization term is simply computed as the L2 distance of the parameters of the model and their value at its initial state. We set the weight for this factor to 0.001 in all adaptation experiments.

Backpropagating gradients through interpolations. We train the model with the representations of virtual examples that we create by interpolating the representations from real examples. During pretraining on the source domain we use manifold mixup regularization, where we interpolate between representations of labeled source examples. During the adaptation stage that is part of GIFT we interpolate between labeled source representations and unlabeled target representations. One important hyper-parameter related to this is whether in the backward pass we back-propagate the gradients all the way down to the input layer or stop the gradients at the layer in which the interpolated representations are computed. In our experiments, similar to Verma et al. (2019a) we allow the gradients to pass through the interpolated activations.

Computational Resources: We train our models on cloud TPU devices. Our estimate of the amount of compute used for the experiments of this paper is roughly about 2k TPU-core days.

B TRAINING ON LABELED SOURCE DOMAIN

We compare four different approaches for training the model on labeled source data.

Standard fine-tuning: Given labeled examples from a source domain, we employ a model that is pretrained on some large scale dataset, Imagenet-1k in our case, replace its head (projection layer) with a new head for the task at hand, and update all its parameters to fit the source domain data.

Mixup with convex combination interpolations: During fine-tuning on labeled source data, we apply mixup/manifold (Zhang et al., 2018a; Verma et al., 2019a) on the input and activation from the first layer of model, and to compute the interpolations we simply compute the convex combination of features for two randomly aligned examples.

Mixup with wasserstein interpolations: During fine-tuning on labeled source data, we apply a variant of mixup/manifold mixup where interpolations are computed using the closed form Monge Map for Gaussian Wasserstein distances. In our experiments we observed that in some cases, interpolating examples in this alternative way leads to better results compared to the convex interpolations used in Verma et al. (2019a).

Domain advarserial neural networks (DANN): Given labeled samples from a source domain and unlabeled samples from target domain, DANN (Ganin et al., 2016) learns domain invariant representations, while minimizing its error on labels source data. In our experiments the output of the prelogits layer is fed to the domain classifier, and the scheduling of the weight of the domain classification loss is the same as what is suggested in Ganin et al. (2016).

B.1 Manifold Mixup with Wasserstein Interpolation

During training on the labeled source dataset we use a variant of manifold mixup (Verma et al., 2019a) with an adapted strategy for interpolation. In manifold mixup (Verma et al., 2019a), representations of virtual examples are created by linearly interpolating representations of two randomly aligned examples (x_i, x_j) in a randomly selected layer L of the neural network M_θ. The labels for the interpolated examples, $\hat{y}_{ij}$, are computed by interpolating the labels of the aligned examples, (y_i, y_j), using the same interpolation coefficient, λ. This is summarized in equation 2.

$$\begin{aligned} \hat{z}_{ij} =& (1-\lambda)\mathcal{M}_\theta^{:L}(x_i) + \lambda \mathcal{M}_\theta^{:L}(x_j) \\ \hat{y}_{ij} =& (1-\lambda) y_i + \lambda y_j \end{aligned}. \tag{2}$$

Here M_θ^L denotes the part of the neural network that outputs the activations of layer L. The interpolated representations are then fed into the rest of the neural network at layer L, and together with the interpolated labels they serve as additional 'data' to which the model is fit.

In our experiments we take a different approach to interpolation. Inspired by the style transfer method discussed in Mroueh (2020), we use interpolations based on the Wasserstein distance between two Gaussian distributions that are fit to representations of two input images. i.e., the spatial features in the representations of two images are the datapoints for two datasets. We estimate the empirical mean and diagonal covariance matrices for these datapoints and use the closed form optimal transport map between two Gaussian distributions to interpolate the spatial features between two representations. More precisely, given two images x_i and x_j, we compute representations $z_i = M_\theta^L(x_i)$ and $z_j = M_\theta^L(x_j)$, where z_i and z_j are three-dimensional tensors with a width W^L, height H^L and channel size C^L. Each spatial feature vector of size C^L within z_i and z_j is considered one datapoint. We compute the average feature vectors within z_i and z_j, denoted by μ_i and μ_j respectively, as well as the variances σ_i^2 and σ_j^2. Note that we are approximating the empirical covariance matrices with diagonal matrices with the variances σ_i^2 and σ_j^2 on the diagonals. Given these quantities, we can compute the closed form Monge map between two Gaussian distributions with diagonal covariance matrices as

$$T_{z_i \to z_j}(z) = \mu_j + \operatorname{diag}\left(\frac{\sigma_j}{\sigma_i}\right)(z - \mu_i). \tag{3}$$

Here z is understood to be a feature map of the same size as z_i and z_j. Interpolated representations are then computed with

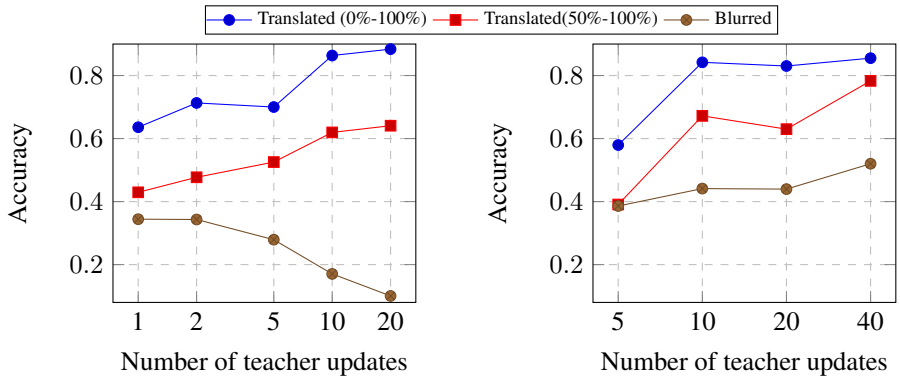
$$\hat{z}_{ij} = (1-\lambda) z_i + \lambda T_{z_i \to z_j}(z_i). \tag{4}$$

Similar to its use in style transfer (Mroueh, 2020), we assume this transformation does not change the content of the representation, and we therefore do not apply an interpolation scheme to the labels y_i and y_j of datapoints x_i and x_j. Instead, we use the label $\hat{y}_{ij} = y_i$ for the virtual representation $\hat{z}_{ij}$.

In Verma et al. (2019a) the interpolation coefficient λ is sampled from a Beta distribution $\mathrm{Beta}(\alpha, \beta)$, where α and β are hyperparameters. In our experiments we set both α and β to 1, so that we are effectively sampling λ uniformly from the interval $[0, 1)$.

C Effect of Number of Teacher Updates

To better distinguish the effect of the number of teacher updates from the number of training steps between each two consecutive teacher updates, in Figure 4 we plot the accuracy as a function of



(a) iterative self-training (b) GIFT

Figure 4: Effect of the number of teacher updates on the accuracy when the number of training steps before each teacher update is fixed and set to 100, for different perturbations of CIFAR10. For Translated (0%-100%) CIFAR10 and Translated (50%-100%) CIFAR10, we see an increasing trend in accuracy as we increase the number of teacher updates for both iterative self-training and GIFT. Whereas for Blurred CIFAR10, the accuracy decreases for iterative self-training.

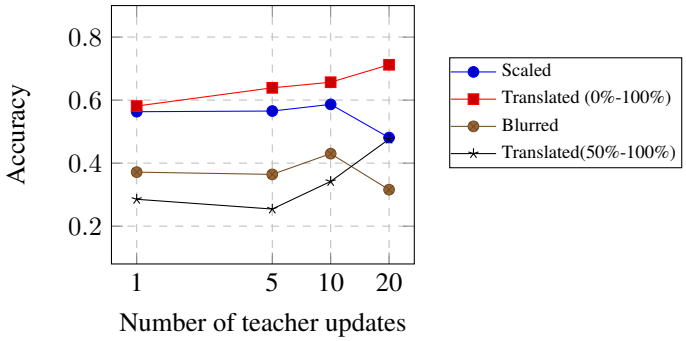


Figure 5: Effect of number of self training iterations on accuracy when all interpolations are represented to the model at the same time with the total number of training steps of 1000 for different perturbations of CIFAR10. Similar to iterative self-training, the performance improves by increasing the number of self training iterations up to a threshold. Beyond the threshold, the performance deteriorates.

the number of teacher updates when the number of training steps is fixed, i.e., the total number of training steps increases as we increase the number of teacher updates. For GIFT, we observe an increasing trend in the accuracy as the number of teacher update increases on all the benchmarks. However, for iterative self-training we only see a benefit in increasing the number of teacher updates for datasets with a range of perturbations in the target domain such as the Translated CIFAR10 datasets, as opposed to Blurred CIFAR10.

Additionally, Figure 5 shows the effect of the number of teacher updates when the total number training steps is 1000 for a non gradual version of GIFT, where all the interpolations are presented to the model simultaneously. We observe that compared to GIFT, where the value of the interpolation coefficient λ is gradually increased, having more teacher updates is much less beneficial.