Supplementary Material

Generate To Adapt: Aligning Domains using Generative Adversarial Networks

1. Network Architectures and Hyperparameters

This section describes the details of the network architectures used in our experiments. A detailed description of all the architectures can be found in Fig. 1

Digits experiments For $SVHN \rightarrow MNIST$ experiment, we used $DigF1$, $DigC1$, $DigG$ and $DigD$ architectures mentioned in Fig. 1 as our F, C, G and D networks respectively. For all other digit experiments, we use $DigF2$, $DigC2$, $DigG$ and $DigD$. All models were trained from scratch and were initialized using random Gaussian noise with standard deviation 0.01. We used Adam solver with base learning rate of 0.0005 and momentum 0.8 to train our models. The cost coefficients α and β are set as 0.1 and 0.03 respectively based on validation splits. We resize all input images to 32×32 and scale their values to the range $[0, 1]$.

OFFICE experiments For OFFICE experiments, we used $OfcC$, OsG and OsD architectures mentioned in Fig. 1 as our C, G and D networks respectively. The F network is initialized with pretrained Resnet50 model trained on ImageNet, the last layer of which is removed and the resulting 2048 dimensional vector is used as the feature embedding. We use Adam solver for optimization with a base learning rate of 0.0004 and momentum 0.7 for all the experiments. The dimension of the random noise vector is set as 128 and the cost coefficient α and β are both set as 0.01.

Synthetic to Real experiments Similar to OFFICE experiments, we used $SynC$, OsG and OsD architectures mentioned in Fig. 1 as our C, G and D networks respectively. We remove the last layer of the pretrained VGG16 model trained on Imagenet, and initialize it as our F network. The resulting 4096 dimensional vector is used as the feature embedding. For all the experiments, we used the same hyperparameter settings as those used in the Office experiments.

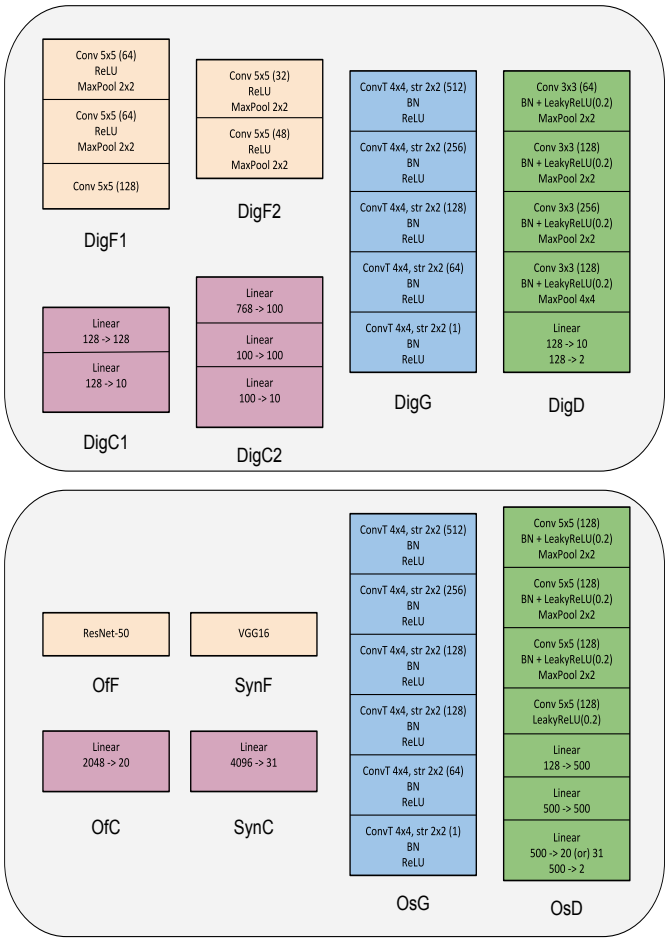

Figure 1: Network Architectures. Legend: BN - Batch Normalization, ConvT - Transposed convolution layer

2. Noise Analysis

As described in our approach in the main paper, the input to the generator network G is $x_g = [F(x), z, l]$, a concatenated version of the feature embedding, noise vector $z \in \mathbb{R}^d$ sampled from $\mathcal{N}(0, 1)$ and l, the one-hot encoding of the class label. In this section, we perform a study of how the dimensionality of the noise vector z affects the trans-

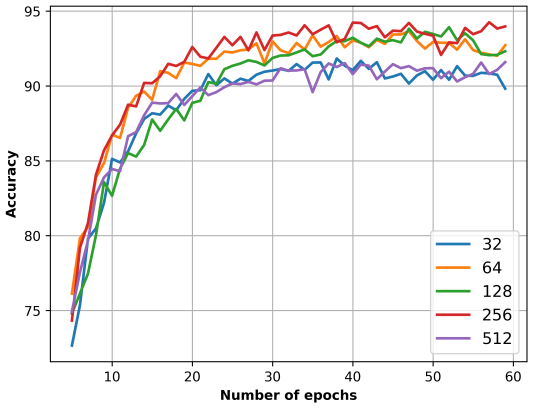

Figure 2: Effect of the noise dimension on classification accuracy for the transfer task SVHN → MNIST

fer accuracy. In figure 2, the transfer accuracy for the task SVHN → MNIST is plotted against the number of training epochs. The dimensionality d is varied over the set: $\{32, 64, 128, 256, 512\}$. The following observations can be made: (1) The approach is not overly sensitive to d, given that all values obtain an average performance of 90.5% or more. (2) The values of dimensionality that is too low (32) or too high (512) result in slightly suboptimal performance.

3. Generation visualization

In Fig. 3, we show some sample images generated by the G network in two experimental settings - $SVHN \rightarrow MNIST$ and Office $A \rightarrow W$. The top set of images show the generations when the input to the system are the samples taken from the source dataset, while the bottom set are the generations when inputs are the images from the target dataset. We make the following observations: (1) The quality of image generation is better in the digits experiments compared to the Office experiments (2) The generator is able to produce source-like images for both the source and target inputs in a class-consistent manner (3) There is mode collapse in the generations produced in the Office experiments.

The difficulty of GANs in generating realistic images in the Office and Synthetic to real datasets makes it significantly hard for the methods that use cross-domain image generation as a data augmentation step. Since we rely on the image generation as a mode for deriving rich gradients to the feature extraction network, our method works well even in the presence of severe mode collapse and poor generation quality.

4. Synthetic to Real adaptation with ResNet

This experiment is an extension to the Synthetic to Real experiments in the main paper. Instead of initializing F

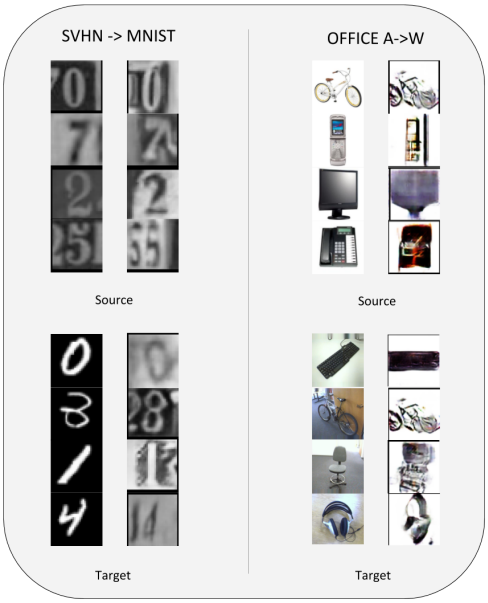

Figure 3: Example of images sampled from G after training. In each set, the images on the left indicate the source images and the images on the right indicate the generated images

Table 1: Accuracy (mean ± std%) values over five independent runs on the Synthetic to real dataset. The best numbers are indicated in **bold**.

Method	CAD → PASCAL
ResNet50 - Source only	30.2 ± 0.6
RevGrad	41.7 ± 1.3
Ours	**46.5** ± 0.9

network with the pretrained VGG16 model, we initialize it with pretrained Resnet-50 model trained on ImageNet as done in the OFFICE experiments. The results of the experiments are presented in Table. 1. We observe that the model trained only on source domain achieves 30.2% performance, which is 7.9% less than the VGG16 baseline performance mentioned in the main paper. However, our method achieves a performance of 46.5% (which is 16.3% above the baseline) and outperforms other compared approaches.