Synthetic to Real Adaptation with Deep Generative Correlation Alignment Networks

Xingchao Peng
Boston University
xpeng@bu.edu

Kate Saenko
Boston University
saenko@bu.edu

Abstract

Synthetic images rendered from 3D CAD models have been used in the past to augment training data for object recognition algorithms. However, the generated images are non-photorealistic and do not match real image statistics. This leads to a large domain discrepancy, causing models trained on synthetic data to perform poorly on real domains. Recent work has shown the great potential of deep convolutional neural networks to generate realistic images, but has rarely addressed synthetic-to-real domain adaptation. Inspired by these ideas, we propose the Deep Generative Correlation Alignment Network (DGCAN) to synthesize training images using a novel domain adaption algorithm. DGCAN leverages the $\ell^2$ and the correlation alignment (CORAL) losses to minimize the domain discrepancy between generated and real images in deep feature space. The rendered results demonstrate that DGCAN can synthesize the object shape from 3D CAD models together with structured texture from a small amount of real background images. Experimentally, we show that training classifiers on the generated data can significantly boost performance when testing on the real image domain (PASCAL VOC 2007 and Office benchmark), improving upon several existing methods.

1. Introduction

Recent advances achieved by Deep Convolutional Neural Networks (DCNN) [9, 16, 37, 38, 34, 13] are unfortunately hampered by their dependence on massive amounts of training examples. Ad-hoc collection and annotation of training data for various computer vision applications is cumbersome and costly. In this paper, we aim to find an alternative method which can efficiently generate large-scale training data for diverse vision tasks at low cost.

3D CAD simulation is one promising solution to this problem [30, 17, 36, 37, 40]. Rendering images from freely available CAD models can potentially produce an infinite number of training examples from many viewpoints and for almost any object category. Previous work [30] utilized computer graphics (CG) technique to render 2D CAD-synthetic images and consequently train deep CNN-based classifiers on them. However, their CAD-synthetic images (Figure 1) are highly non-realistic due to the absence of natural object texture and background. More specifically, they exhibit the following problems: 1) large mismatch between foreground and background (e.g. cars floating above the road), 2) higher contrast between the object edges and the background (e.g. poorly-blended aeroplane in the sky), 3) non-photorealistic scenery. These problems inevitably lead to a very significant domain shift between CAD-synthetic and real images, and models trained on the synthetic domain have poor performance on real test images [29, 30].
with real images, such as content and style [8], similarity in feature space [18, 11, 45, 23, 26], etc. However, these approaches have several limitations for use in data augmentation. For example, Generative Adversarial Nets (GANs) [11] can theoretically generate images from a training distribution but cannot adapt between two distributions. Conditional GANs [14] can learn image-to-image translation but need paired training data that are very costly to obtain, i.e., CAD models and corresponding natural images. Coupled GANs [18] do not require paired data but handle only very limited domain shifts. Style transfer approaches [8, 7] can synthesize fascinating artistic images but cannot be directly applied to real world object recognition.

Inspired by these methods, in this work we propose the Deep Generative Correlation Alignment Network (DGCAN) to effectively synthesize novel images which can be directly used to train real-world image recognition systems. With DGCAN, we aim to bridge the gap between synthetic images generated by CAD simulation and the real image domain (Figure 1) using domain adaptation techniques. Domain adaptation algorithms aim to handle the domain discrepancy between training data (source domain) and testing data (target domain) [29]. Recent unsupervised algorithms align deep feature representations by adding auxiliary losses to the network that encourage the source data distribution to match the target. Matching can be done with losses such as maximum mean discrepancy [19] or adversarial losses [42, 6]. The recently proposed correlation alignment (CORAL) loss [38, 39] measures the difference between the correlation matrices of features in the two domains.

Our work is primarily motivated by [30, 8, 39]. As shown in Figure 1, we propose a network to generate images by satisfying two loss functions. We employ the CORAL loss for adaptation [38, 39], however, instead of tuning the network parameters to align features, we generate images whose feature correlations match the target real-image domain. Similar to neural style transfer [8], we generate novel images by matching the convolutional layer features with those of a content CAD model image and the style of a real image containing a background scene. However, unlike neural style, the goal is not to create an artistic effect but rather to adapt the synthetic training data to match the statistics of real images and thus improve generalization.

Our synthesized results reveal that DGCAN can satisfactorily blend the contour of specific objects (from CAD-synthetic images) with natural textures from real images. Although the generated images are not fully photorealistic, they appear to have more natural statistics to the deep network, improving its performance. Interestingly, by exhaustively exploring which are the right conv layers to apply the losses, we find DGCAN tends to generate more distinct object contours when the $\ell^2$ loss is applied to lower conv layers, and conversely, can synthesize more structured textures when applying CORAL loss to higher conv layers. This effect mainly comes from the increasing receptive field size and feature complexity along the DCNN’s processing hierarchy. As an application, we train classifiers on the DGCAN-rendered object images and test them on a real image benchmark from PASCAL VOC 2007 [4] and Office [31]. Extensive experiments on the PASCAL and Office dataset show that our approach yields a significant performance boost compared to the previous state-of-the-art methods [30, 38, 39, 5, 10, 19, 21, 8].

The contributions of this paper can be summarized as follows. 1) We propose a network architecture DGCAN (Deep Generative Correlation Alignment Network) to synthesize the object contour from the CAD-synthetic domain with natural textures from the real image domain. 2) We explore the effect of applying the $\ell^2$ and CORAL losses to different layers and determine the optimal configuration to generate the most promising stimuli. 3) By bridging the CAD-synthetic and real domains, we propose an approach to train an effective classifier with minimal supervision, i.e., no labeled object images from the target real domain.

2. Related Work

We leverage 3D CAD simulation as our engine to generate large quantities of CAD-synthetic images with different object pose and shape from freely available CAD models, and design a novel domain adaptation algorithm to adapt the CAD-synthetic domain to the real domain by synthesizing images with DGCAN. Hence, our approach is related to CAD simulation, DCNN image synthesis and domain adaptation.

**CAD Simulation** CAD simulation has been extensively used by researchers since the early days of computer vision [27]. 3D CAD models have been utilized by vision researchers to generate stationary synthetic images with variable object poses, textures, and backgrounds [30]. Recent usage of CAD simulation has been extended to multiple vision tasks, e.g., object detection [30, 25], pose estimation [17, 36, 37, 40], robotic simulation [41]. However, it is widely recognized that CAD-synthetic images are too low-quality due to the absence of realistic backgrounds and texture. To mitigate this drawback, [30] proposes to directly add auxiliary texture and background to the rendered results, with the help of commercial software (e.g. AutoDesk 3ds MAX1). However, this method introduces new problems, such as unnatural positioning of objects (e.g. floating car above the road), high contrast between object boundaries and background, etc. Our approach tackles these problems by synthesizing novel stimuli with DGCAN and can

---

1http://www.autodesk.com/store/products/3ds-max
generate more natural looking images with well-blended foreground and background.

**DCNN Image Synthesis** Deep convolutional neural networks learn distributed, invariant and nonlinear feature representations from large-scale image repositories [1]. A class of image synthesis approaches have been proposed to visualize the learned kernels and feature representations. For example, [46] visualizes activations for each conv layer as it processes an image. [23] reconstructs images from a feature vector with the convolutional-deconvolutional paradigm. Generative adversarial networks (GANs) [11] and their variations [26, 28] aim to synthesize images to fit in the feature distribution of training data. However, training GANs is difficult and often leads to oscillatory behavior, and has not been shown to work for large domain shifts. [8] synthesizes novel stimuli by aligning the conv layer features and Gram Matrices of the features. In this way, the synthesized image simultaneously preserves the arrangement of a content image (often a normal photograph) and the colours and subtle local structures of a style image (often an artist’s work). Our approach synthesizes images by aligning the conv layer features to CAD-synthetic images (source domain) with a $\ell^2$ loss and aligning the correlation of conv layer features to realistic images (target domain) with a CORAL loss, and demonstrates improvement on an object classification task.

**Domain Adaptation** Domain shift results in a significant performance degradation when recognition systems are trained on one domain (source) and tested on another (target). Domain adaptation algorithms aim to bridge two domains, either by aligning the feature representations with a distribution distance metric [2], or by projecting the feature distribution of source domain and target domain to a common low-dimensional manifold [12, 10, 20]. Rather than only align the feature means, [38] proposes to minimize the difference between second-order statistics of the feature distributions measured by squared Frobenius norm. Adaptive DCNNs address the domain shift by adding one or multiple adaptation layer [43, 41, 19, 39], or a sub-network to confuse the domain discriminator [41, 18]. All the aforementioned methods follow the paradigm of aligning the source domain and target domain in feature space. In contrast, we take a generative approach to combine the statistics of target domain images with the content of source domain images.

### 3. Approach

Suppose we are given $n_s$ labeled source-domain CAD-synthetic (image, label) pairs $I_s = \{C^s_i, Y^s_i\}_{i=1}^{n_s}, C^s \in \mathbb{R}^{W \times H \times C}, Y^s_i \in \mathbb{R}$ and $n_t$ target-domain real examples $I_t = \{R^t_i\}_{i=1}^{n_t}, R^t \in \mathbb{R}^{W \times H \times C}$. We address the challenge of training an accurate object classifier when there is no annotated data in $I_t$ and classifier $f^s$ trained on $I_s$ can not be directly applied to $I_t$ due to the domain discrepancy. We assume that the target domain examples are composed of a small number of background images. Our aim is to synthesize a labeled intermediate dataset $I = \{S_i, Y_i\}_{i=1}^{n_t}, C^s \in \mathbb{R}^{W \times H \times C}, Y^s_i \in \mathbb{R}$, such that each $S_i \in I$ contains a similar object shape and contour with some $C^s \in I_s$ and similar local pattern, color, and subtle structure ("style") as illustrated in [8]) with some $R^t \in I_t$.

The most straightforward method for generating an image is employing pixel-level Euclidean loss $\ell^2_{\text{pixel}} = \|S - C^s\|^2 + \lambda \|S - R^t\|^2$ or $\ell^1_{\text{pixel}} = \|S - C^s\| + \lambda \|S - R^t\|$. However, the results of this approach are far from satisfactory. Traditional computer vision blending approaches, such as half-half alpha blending or pyramid blending lead to image artifacts that contribute to the domain shift. Instead, we propose to align the generated $S$ to $C^s$ and $R^t$ in the DCNN feature space.
3.1. Deep Convolutional Neural Network

We base our approach on the VGG-16 [34] network which consists of 13 convolutional layers (conv1
-conv5,3), 3 fully connected layers (fc6-fc8) and 5 pooling layers (pool1-pool5). The convolutional layers consist of a set of learnable kernels. Each kernel is convolved with the input volume to compute hidden activations during the forward pass and its parameters are updated through a back-propagation pass. The fully connected layers learn a non-linear transformation \( \mathcal{H}^l = \mathcal{A}^l(W^l\mathcal{H}^{l-1} + b^l) \), where \( \mathcal{H}^l \), \( \mathcal{A}^l \), \( W^l \), \( b^l \) are the hidden representations, activation function, weight and bias parameters of \( l^{th} \)-layer, respectively. \( \mathcal{A}^l \) is implemented as rectifier units (ReLu layer) \( \mathcal{A}^l(x) = \max(x, 0) \) or softmax units \( \mathcal{A}^l(x) = e^x / \sum_{i=1}^{N^l} e^x \).

Informed by previous studies [44] on feature transferability of DCNNs, we propose to apply a domain adaptation loss to convolutional layers when synthesizing \( \mathcal{I} \), for [44] reveals that convolutional layers can learn transferable representations. We denote \( \mathcal{H}^l(\cdot) \) as DCNN’s \( l^{th} \) layer’s representation matrix, \( \mathcal{H}_i^l(\cdot) \) as the \( i^{th} \) dimension of \( \mathcal{H}^l(\cdot) \) and \( \mathcal{H}_{ij}^l(\cdot) \) as the \( j^{th} \) value of \( \mathcal{H}_i^l(\cdot) \).

3.2. \( \ell^2 \) Loss In Feature Space

There are several ways to generate images. GANs [11] synthesize an image \( x \) from a given distribution \( p(x) \). [8] generate an image \( x \) from a random noise seed such that \( x \sim p(x|content, style) \). In our model, analogous to [8], we synthesize an image \( \mathcal{S} \) from \( \mathcal{C}^s \) with \( \mathcal{S} \sim p(\mathcal{S}|\mathcal{C}^s, \mathcal{R}^t) \).

The generation is guided by two losses, one to ensure the object contour stays the same, and one to ensure the image is realistic. For the former loss we propose to use the \( \ell^2 \) loss in feature space as follows

\[
\mathcal{L}_{\text{feat}}^{X_{l}} = \sum_{\mathcal{I} \in \mathcal{I}_s} \frac{\omega^l}{2\alpha^l} \sum_{i} \|\mathcal{H}_i^l(\mathcal{S}) - \mathcal{H}_i^l(\mathcal{C}^s)\|^2_2.
\]

where \( \mathcal{S} \in \mathcal{I}, \mathcal{C}^s \in \mathcal{I}_s; \omega^l \) is the loss weight of \( l^{th} \) layer in DCNN feature space; \( \mathcal{X}^l \) is the collection of convolutional layers which the \( \ell^2 \) loss is applied to; \( \alpha^l = C^l N^l \), where \( C^l \) is the channel number of \( l^{th} \) layer’s feature, and \( N^l \) is the length of feature in a single channel.

The derivative of this loss with respect to the activations in a particular layer \( l \) can be computed by:

\[
\frac{\partial \mathcal{L}_{\text{feat}}^{X_{l}}}{\partial \mathcal{H}_{ij}^l(\mathcal{S})} = \frac{\omega^l}{\alpha^l} (\mathcal{H}_{ij}^l(\mathcal{S}) - \mathcal{H}_{ij}^l(\mathcal{C}^s))
\]

This gradient can be back-propagated to update the pixels of the intermediate stimuli while synthesizing \( \mathcal{S} \).

3.3. Correlation Alignment (CORAL) Loss

To align the low-level texture statistics of the generated images to the real image domain, we propose to employ the CORAL loss. Correlation Alignment (CORAL) is first devised by [38] to match the second-order statistics of feature distributions in domain adaptation. It is derived by minimizing the domain discrepancy with squared Frobenius norm \( \min\|C_S - C_T\|^2_F \), where \( C_S, C_T \) are the covariance matrices of feature vectors from source domain and target domain, respectively. This problem is equivalent to solving \( A^* = \arg\min A^T C_S A - C_T^2 \).

According to [38], the final optimal solution for \( A \) is \( A^* = (U_S(\Sigma_S^{-\frac{1}{2}})\Sigma_T^{-\frac{1}{2}})(U_T^T\Sigma_T^{-\frac{1}{2}})^{-\frac{1}{2}}U_T^T\Sigma^{-\frac{1}{2}} \), where \( \Sigma_S, U \) are the singular values and singular vectors after conducting SVD on the covariance matrices and \( r = \min(r_{CS}, r_{CT}) \), where \( r \) denotes the matrix rank. One can refer to [38] for more details about the derivation process.

Inspired by [38], we define the CORAL loss \( \mathcal{L}_{\text{coral}}^{X_{c}} \) as

\[
\mathcal{L}_{\text{coral}}^{X_{c}} = \sum_{\mathcal{I} \in \mathcal{I}_c} \left( \frac{\omega^l}{4\alpha^l} \|\text{Cov}(\mathcal{H}_i^l(\mathcal{S})) - \text{Cov}(\mathcal{H}_i^l(\mathcal{R}^t))\|^2_F \right)
\]

where \( \mathcal{S} \in \mathcal{I}, \mathcal{R}^t \in \mathcal{I}_c; \omega^l \) is the CORAL loss weight of \( l^{th} \) layer; \( \mathcal{X}_c \) is the collection of convolutional layers that the CORAL loss is applied to; \( \text{Cov}(\cdot) \) is the covariance matrix of \( l^{th} \) layer’s activation; \( \|\cdot\|_F \) denotes the Frobenius distance.

Analogous to [39], the covariance matrices are given by:

\[
\text{Cov}(\mathcal{H}_i^l(M)) = \frac{1}{C^l}\{\mathcal{H}_i^l(M) \mathcal{H}_i^l(M)^T\} - \frac{1}{C^l}\{(1^T \mathcal{H}_i^l(M))(1^T \mathcal{H}_i^l(M))^T\}
\]

where \( M \in \{S, R^t\}, 1 \) is a column all-one vector, and \( C^l \) is the number of feature channels in \( l^{th} \) layer.

The derivative of CORAL loss with respect to particular layer \( l \) can be calculated with chain rule:

\[
\frac{\partial \mathcal{L}_{\text{coral}}^{X_{c}}}{\partial \mathcal{H}_{ij}^l(\mathcal{S})} = \frac{\omega^l}{C^l\alpha^l} \{\text{Cov}(\mathcal{S}) - \text{Cov}(\mathcal{R}^t)\}^T
\]

Our final method combines the loss functions defined by equation 1 and equation 3. We start from an image \( \mathcal{S} \in \mathcal{I}_s \) and pre-process it by adding a random perturbation \( \epsilon \), where \( \epsilon \sim \mathcal{N}(0, \Sigma) \). We then feed the image forward through DG-CAN and compute the \( \ell^2 \) loss with respect to \( \mathcal{S} \) and CORAL loss with respect to \( \mathcal{R}^t \). The back-propagated gradient thus guides the image synthesis process. Hence, the synthesized images are the output of the function:

\[
\mathcal{I} = \text{DG-CAN}(\mathcal{I}_s, \mathcal{L}_{\text{feat}} + \lambda \mathcal{L}_{\text{coral}}, \mathcal{I}_t, \mathcal{X}_f, \mathcal{X}_c, \lambda, \epsilon)
\]

\[
\lambda \mathcal{L}_{\text{feat}} + \lambda \mathcal{L}_{\text{coral}} \] denotes the total loss of DG-CAN, where \( \lambda \) is the trade-off weight between \( \ell^2 \) loss and the CORAL loss. The \( \lambda \) hyperparameter is set through cross validation.
Figure 3. Illustration of our synthesized results. We leverage DGCAN to synthesize novel images based on two inputs, i.e. source domain CAD-synthetic image $I^s$ and target domain real background image $R^t \in I_t$. (1) We exhaustively apply $L_{feat}$ and $L_{coral}$ to different $conv$ layers to find the best configuration. The results (left plot) demonstrate that DGCAN can generate more distinct object contours when applying $L_{feat}$ to lower conv layers and can synthesize more structured style texture when applying the $L_{coral}$ to higher conv layers. ($L_{1-5}^{[1]}$ means applying $L_{coral}$ to conv1,1, conv2,1, conv3,1, conv4,1, conv5,1 simultaneously) (2). We vary the trade-off parameter $\lambda$ in equation 6 from $10^{-5}$ to $10^{14}$ to learn the optimal value for $\lambda$. The results (right plot) show that the shape contour dominates the background texture when $\lambda$ is small.

4. Experiments

Our experiments include two parts. First, we apply DGCAN to CAD-synthetic dataset provided by [30] to synthesize well-blended results. Second, we train deep classifiers on the DGCAN-synthetic images and perform classification.

4.1. Synthesize Images with DGCAN

We choose CAD-synthetic data provided by [30] as $I^s$ and realistic images downloaded by Google search engine as $I_t$. Datasets (both CAD-synthetic and DGCAN-synthetic), codes and experimental configurations will be made available publicly.

CAD-Synthetic Dataset CAD-synthetic dataset [30] is generated from 3D CAD models for zero-shot or few-shot learning task. The dataset has 6 configurations (i.e. RR-RR, W-RR, W-UG, RR-UG, RG-UG, RG-RR)\(^2\). The process of rendering the dataset (we refer the reader to [30] for more details) can be concluded as follows: (1) collect 3D-CAD models from large-scale on-line repositories (Google Sketchup, Stanford 3D ShapeNet\(^3\)), (2) select image cues (background, texture, pose, etc.), (3) render synthetic images with AutoDesk 3ds Max. Bounding box annotations are produced automatically while generating the whole dataset. In our experiments, we only adopt images with white background because other subsets are severely confronted with the issues described in section 1.

Parameter tuning The prototype of DGCAN is VGG-16 [34] with 13 convolutional layers and 3 fully connected layers. We implement it with Caffe [15] deep learning framework. To gather the optimal configuration for $X_f$, $X_e$, and $\lambda$, we exhaustively apply $L_{feat}$, $L_{coral}$ to different $conv$ layers and vary $\lambda$ from $10^{-5}$ to $10^{14}$.

Results and Analysis A representative subset of rendered results for exploring $X_f$, $X_e$ and $\lambda$ are shown in Figure 3. The left plot shows the effect of different configuration of $X_f$ and $X_e$. The results explicitly verify when $L_{feat}$ is applied to lower conv layers, DGCAN can generate more distinct contour of the object from CAD-synthetic data and when $L_{coral}$ is applied to higher conv layers, DGCAN can generate more structured texture. Empirical evidence [8] exhibits this effect mainly benefits from two factors. The first reason is the increasing receptive field size along the processing hierarchy, given the receptive field sizes of VGG-16’s conv1,2, conv2,2, conv3,2, conv4,2 conv5,2 are 5, 14, 32, 76 and 164, respectively. The second factor is the increasing feature complexity. A batch of previous literature [45, 35] have demonstrated that low-layer DCNN features are more related to the local patterns (edges, pixels, etc.), while the high-layer features are more pertinent to the overview scenery of the image.

To find the optimal trade off ratio $\lambda$, we synthesize images with $\lambda$ ranging from $10^{-5}$ to $10^{14}$. The right plot in Figure 3 reveals when $\lambda (L_{coral} \text{ to } L_{feat} \text{ ratio})$ is small, the object contour information will dominate the background texture cues. On the contrary, when $\lambda$ is increased to $10^{5}$, the contour of the object gradually fades away and more structured textures from realistic images are synthesized. In figure 5, we show the effect of $\lambda$ on how $L_{feat}$ and $L_{coral}$
will change with respect to the training iterations (the losses are computed on 128 images). Figure 5 reveals when $\lambda$ is larger, the CORAL loss starts from a bigger value and drops drastically within $0 \sim 100$ iterations. Hence, it will take a larger proportion in the total loss, compared to the scenario when $\lambda = 10^3$, which explains the contour of the “aeroplane” (in Figure 3) fades away gradually when $\lambda$ is larger than $10^6$.

Figure 5. Effect of $\lambda$ on the losses. With a larger $\lambda$, $L_{\text{coral}}$ will start from a larger value and will dominate the total loss. Dashed lines show the standard deviation.

We randomly select some rendered results from three categories (“aeroplane”, “potted plant”, “sofa”) and show them in the left plot of Figure 4. The images are generated with the configuration $X^f = \text{conv}_3^2$, $X^c = \text{conv}[1 - 5]_1$, ($\omega_c^{1-5} = 0.2$) and $\lambda = 10^3$. The results demonstrate that DGCAN-synthetic imagery preserves clear object contour from CAD-synthetic images and synthesizes textures from realistic domain.

We further utilize the DCNN visualization tool provided by [24] to demonstrate that our DGCAN-synthetic images are more realistic than CAD-synthetic images in DCNN’s perspective. [24] provides a effective tool to visualize what DCNN has learned by reconstructing an image from its representation. We compare three kinds of data, i.e. DGCAN-synthetic bird images, corresponding CAD-synthetic bird images and natural images of bird. We select VGG-16 [34] model pre-trained on realistic image repository (ImageNet [3]) as the inspector. As showed in the right plot of Figure 4, the results indicate our synthetic images have a significant better interpretation of bird than CAD-synthetic images.

4.2. Deep Classifiers Trained On DGCAN-Synthetic Domain

Image classification is a fundamental task in computer vision. Given the in-domain training images with detailed annotations are expensive to get, we train deep classifiers on the DGCAN-synthetic images, which are visually similar with real domain images. We demonstrate the effectiveness of our model over previous methods [16, 39, 38, 10, 19, 21, 8, 5] on two benchmarks: PASCAL VOC 2007 [4] and Office [31] dataset.

4.2.1 Experiments on PASCAL VOC 2007 benchmark

Train/Test Set Acquisition As a training set, we generate 1082 images with DGCAN from the W-UG subset of CAD-synthetic dataset [30]. These images are equally distributed into 20 PASCAL categories. For evaluation, we crop 14976 patches from 4952 images in the test subset of PASCAL VOC 2007 dataset [4]. The patches are cropped by the given bounding boxes and each patch contains only one object.

Experimental Setup We evaluate the effectiveness of our approach with three different DCNN architectures, i.e. “AlexNet” [16], “VGG-16” [34] and “ResNet-50” (residual net with 50 layers) [13]. In the training process, the networks are initialized with the parameters pre-trained on ImageNet [3]. We replace the last output layer with a 20-way classifier and randomly initialize it with $\mathcal{N}(0, 0.01)$. We use mini-batch stochastic gradient descent (SGD) with a momentum of 0.9 to finetune all the layers. The base learning rate is $10^{-3}$ and the weight decay is $5 \times 10^{-4}$. Specifically, we set dropout ratios for $fc6$ and $fc7$ of “AlexNet” to
SA-pool5 [5]  
DCORAL-Alex [39]  
CORAL-Alex [38]  
DCORAL-VGG [39]  
GFK-fc7 [10]  
DCORAL-Res [38]  
DGCAN-VGG  
DGCAN-Res  

Table 1. Results on PASCAL 2007. We perform our experiments on three different DCNN architectures, i.e. “AlexNet” [16], “VGG-16” [34] and “ResNet-50” [13]. The results clearly demonstrate the superiority of our model over CAD-synthetic method [30] and several state-of-the-art domain adaptation models [38, 39, 5, 10, 19, 21].

Figure 6. Confusion Matrices and t-SNE plots of fc7 feature. (1). The confusion matrices on the left show models trained on DGCAN-synthetic dataset (right subplot) pose a different error mode from those trained on CAD-synthetic dataset (left subplot). (2) The t-SNE plots on the right shows the embedded fc7 features of realistic and synthetic images are better aligned after applying our model to CAD-synthetic images. (Best viewed in color!)

0.5. We report the results when the training iteration reaches 40k.

Baselines We compare our approach with CAD-synthetic method [30], style transfer [8] and domain adaptation algorithms [38, 39, 5, 10, 19, 21]. For equal comparison, we take the same 1082 W-UG images which we utilized to generate our DGCAN-synthetic dataset as the source domain for domain adaptation algorithms. For style-transfer method [8], we use the same CAD-synthetic (content) images and realistic images (style) to generate new dataset.

Within the domain adaptation methods, CORAL [38] aligns the feature distribution of source domain \((P(x, y))\) to target domain \((P^t(x^t, y^t))\). DCORAL (Deep CORAL) [39] incorporates CORAL as a loss layer to DCNN. SA (Subspace Alignment) [5] proposes a mapping function to align the subspace of source domain with the target domain. The subspace is described by the eigenvectors of features. GFK (Geodesic Flow Kernel) [10] models domain discrepancy by integrating numerous subspaces which characterize changes in geometric and statistical properties. Based on these subspaces, geodesic curve is constructed, geodesic flow kernel is computed and a kernel-based classifier is trained. DAN (Deep Adaptation Network) [19] and RTN (Residual Transfer Network) [21] train deep models with Maximum Mean Discrepancy [33] loss to align the feature distribution of two domains.

Results and Analysis The results demonstrating our approach outperforms competing methods are presented in table 1. After applying our approach to CAD-synthetic data, the performances rise from 18.48% to 27.46% with “AlexNet” [16], from 10.3% to 22.92% with “VGG-16” [34] and from 13.13% to 20.59% with “ResNet-50” [13]. Additionally, Table 1 shows our approach gains a clear advantage over the state-of-the-art domain adaptation algorithms [38, 39, 5, 10, 19, 21] and style-transfer baseline [8]. This empirical evidence reveals that aligning the covariance matrix works better than aligning the Gram matrix in the synthetic-to-real domain adaptation scenario.

We visualize how the confusion mode and the feature embedding have changed after applying our model, as
Table 2. Results on Office Dataset. We apply DGCAN to 775 CAD-synthetic images and train “AlexNet” classifiers [16]. The test images come from Amazon domain of Office Dataset [32]. The results clearly shows our approach outperforms the competing baselines [38, 39, 5, 10, 19, 21]. The suffix “-web” and “-CAD” represent the methods are trained on webcam domain and CAD-synthetic domain, respectively.

| Method            | hp | bk | bh | bc | ca | dc | dl | dp | lc | lp | mp | mt | ms | mg | ps | pe | pr | pj | ps | rc | sc | sp | st | td | tc | all  |
|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| AlexNet-web [16]  | 76 | 96 | 90 | 88 | 33 | 86 | 82 | 59 | 3 | 51 | 60 | 70 | 76 | 14 | 14 | 60 | 86 | 32 | 32 | 35 | 22 | 22 | 44 | 49 | 72 | 30 | 32 | 36 |
| SA-web [5]        | 77 | 96 | 87 | 27 | 34 | 77 | 84 | 35 | 5 | 40 | 63 | 71 | 73 | 8 | 29 | 47 | 77 | 33 | 32 | 32 | 45 | 32 | 32 | 32 | 44 | 49 | 72 | 30 | 32 | 36 |
| GFK-web [10]      | 10 | 94 | 85 | 17 | 11 | 73 | 76 | 26 | 15 | 10 | 65 | 72 | 79 | 9 | 5 | 21 | 44 | 27 | 27 | 46 | 23 | 32 | 13 | 21 | 13 | 24 | 48 | 37 | 31 | 10 |
| CORAL-web [38]    | 82 | 95 | 93 | 38 | 39 | 78 | 88 | 45 | 12 | 35 | 74 | 79 | 73 | 10 | 36 | 51 | 77 | 33 | 44 | 44 | 37 | 60 | 62 | 17 | 27 | 45 | 42 | 17 | 24 |
| DAN-web [19]      | 87 | 96 | 82 | 23 | 58 | 87 | 90 | 35 | 2 | 36 | 78 | 85 | 74 | 9 | 57 | 89 | 83 | 68 | 43 | 48 | 35 | 52 | 44 | 17 | 37 | 43 | 42 | 18 | 19 |
| AlexNet-CAD [16]  | 61 | 94 | 15 | 30 | 47 | 78 | 91 | 49 | 16 | 7 | 57 | 88 | 72 | 26 | 70 | 71 | 49 | 73 | 32 | 20 | 53 | 93 | 55 | 12 | 3 | 16 | 15 | 6 | 2 | 15 |
| SA-CAD [5]        | 78 | 94 | 83 | 52 | 42 | 84 | 85 | 38 | 5 | 46 | 65 | 78 | 62 | 10 | 41 | 49 | 72 | 30 | 32 | 32 | 45 | 40 | 50 | 60 | 16 | 25 | 34 | 40 | 10 | 6 |
| GFK-CAD [10]      | 60 | 96 | 78 | 17 | 19 | 69 | 84 | 22 | 18 | 16 | 49 | 56 | 68 | 12 | 1 | 23 | 35 | 22 | 20 | 45 | 30 | 12 | 16 | 14 | 24 | 35 | 42 | 8 | 6 |
| CORAL-CAD [38]    | 60 | 98 | 92 | 63 | 39 | 79 | 90 | 42 | 15 | 36 | 80 | 80 | 74 | 12 | 37 | 55 | 63 | 33 | 32 | 47 | 46 | 58 | 63 | 10 | 19 | 30 | 39 | 44 | 20 |
| DAN-CAD [19]      | 90 | 96 | 62 | 62 | 35 | 76 | 86 | 57 | 26 | 24 | 66 | 82 | 72 | 31 | 70 | 72 | 66 | 67 | 46 | 30 | 26 | 52 | 63 | 11 | 13 | 16 | 32 | 38 | 18 |
| DGCAN             | 91 | 98 | 36 | 72 | 39 | 89 | 91 | 47 | 20 | 56 | 66 | 85 | 67 | 23 | 59 | 67 | 54 | 72 | 30 | 18 | 52 | 76 | 60 | 2 | 17 | 27 | 42 | 23 | 22 |
| DGCAN+DAN         | 90 | 96 | 35 | 67 | 50 | 80 | 85 | 59 | 27 | 48 | 77 | 78 | 54 | 74 | 32 | 74 | 73 | 64 | 80 | 53 | 22 | 47 | 67 | 4 | 21 | 39 | 33 | 37 | 29 |

The results demonstrate our approach outperforms baselines shown in Table 2. The overall classification accuracy of our model is 49.91% versus 44.69% for classifier trained on CAD-synthetic domain directly. The table also shows that DGCAN beats other baselines [5, 10, 38, 19] and classifiers trained on real images (Webcam domain). The results show the potential of our approach in dataset bias scenarios.

We further show our model can convert a coarse source domain to a delicate one. When applying DAN [19] to the newly generated dataset, we boost the performance from 49.63% (49.27%) for DAN trained on real (CAD-synthetic) domain to 51.93%.

5. Conclusion

In-domain fully annotated training data is always expensive to get. Generating large-scale training examples from 3D CAD models is an intriguing option. While the domain discrepancy between CAD-synthetic images and realistic images severely undermines the performance of recognition in real world applications.

In this work, we have proposed and implemented Deep CORAL Net to bridge CAD-synthetic domain with realistic domain by generating more photo-realistic images. We demonstrated that leveraging $\ell^2$ loss to minimize the domain shift in feature space and applying second-order CORAL loss to diminish the domain discrepancy in Frobenius space are effective in synthesizing novel images. We empirically and experimentally show DGCAN-synthetic images are more photo-realistic than CAD-synthetic ones. An extensive evaluation on standard benchmarks demonstrates the feasibility and effectiveness of the proposed approach against previous methods. We believe our model can be generalized to other generic tasks such as pose estimation, saliency detection and robotic grasping.

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. IEEE transactions
on pattern analysis and machine intelligence, 35(8):1798–1828, 2013. 3

[2] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics, 22(14):e49–e57, 2006. 3

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009. 6

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007. 2, 5, 6

[5] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In Proc. ICCV, 2013. 2, 6, 7, 8

[6] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. arXiv preprint arXiv:1409.7495, 2014. 2

[7] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897, 2016. 2

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576, 2015. 2, 3, 4, 5, 6, 7

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524, 2013. 1

[10] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2066–2073. IEEE, 2012. 2, 3, 6, 7, 8

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014. 2, 3, 4

[12] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In Proc. ICCV, 2011. 3

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015. 1, 6, 7

[14] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. CoRR, abs/1611.07004, 2016. 2

[15] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013. 5

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012. 1, 6, 7, 8

[17] J. Liebelt and C. Schmid. Multi-view object class detection with a 3d geometric model. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1688–1695. IEEE, 2010. 1, 2

[18] M. Liu and O. Tuzel. Coupled generative adversarial networks. CoRR, abs/1606.07536, 2016. 2, 3

[19] M. Long and J. Wang. Learning transferable features with deep adaptation networks. CoRR, abs/1502.02791, 2015. 2, 3, 6, 7, 8

[20] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer joint matching for unsupervised domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1410–1417, 2014. 3

[21] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In Advances in Neural Information Processing Systems, pages 136–144, 2016. 2, 6, 7, 8

[22] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(Nov):2579–2605, 2008. 8

[23] A. Mahendran and A. Vedaldi. Understanding Deep Image Representations by Inverting Them. ArXiv e-prints, Nov. 2014. 2, 3

[24] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. International Journal of Computer Vision, pages 1–23, 2016. 6

[25] F. Massa, B. C. Russell, and M. Aubry. Deep exemplar 2d-3d detection by adapting from real to rendered views. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6024–6033, 2016. 2

[26] M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014. 2, 3

[27] R. Nevatia and T. O. Binford. Description and recognition of curved objects. Artificial Intelligence, 8(1):77 – 98, 1977. 2

[28] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. arXiv preprint arXiv:1605.09304, 2016. 3

[29] S. J. Pan and Q. Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2010. 1, 2

[30] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. In Proceedings of the IEEE International Conference on Computer Vision, pages 1278–1286, 2015. 1, 2, 3, 5, 6, 7, 8

[31] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In Computer Vision–ECCV 2010, pages 213–226. Springer, 2010. 2, 5, 6

[32] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In Proc. ECCV, 2010. 8

[33] D. Sejdinovic, B. Sriperumbudur, A. Gretton, K. Fukumizu, et al. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. The Annals of Statistics, 41(5):2263–2291, 2013. 7

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. 1, 3, 4, 5, 6, 7

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 5
[36] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3d cad data. In BMVC, volume 2, page 5, 2010. 1, 2

[37] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In Proceedings of the IEEE International Conference on Computer Vision, pages 2686–2694, 2015. 1, 2

[38] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. arXiv preprint arXiv:1511.05547, 2015. 1, 2, 3, 4, 6, 7, 8

[39] B. Sun and K. Saenko. Deep CORAL: correlation alignment for deep domain adaptation. CoRR, abs/1607.01719, 2016. 2, 3, 4, 6, 7, 8

[40] M. Sun, H. Su, S. Savarese, and L. Fei-Fei. A multi-view probabilistic model for 3d object classes. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1247–1254. IEEE, 2009. 1, 2

[41] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell. Towards adapting deep visuo-motor representations from simulated to real environments. arXiv preprint arXiv:1511.07111, 2015. 2, 3

[42] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In The IEEE International Conference on Computer Vision (ICCV), December 2015. 2

[43] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.3474, 2014. 3

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3320–3328. 2014. 4

[45] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014, pages 818–833. Springer, 2014. 2, 5

[46] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In Computer vision–ECCV 2014, pages 818–833. Springer, 2014. 3