Learning image classifiers from (limited) real and (abundant) synthetic data

Anonymous ECCV submission

No Institute Given

Abstract. While deep learning's biggest successes in computer vision rely on massive datasets consisting of labeled images, it's often costly or infeasible to acquire and annotate such voluminous data in practice. One promising solution is to train models on synthetic data, for which we know the true labels, and then deploy these models in real-world scenarios. Unfortunately, supervised learning techniques perform poorly when training and test distributions diverge. The subtle differences between real and synthetic data significantly degrade performance. To learn models without real-world labels, we propose a two-part solution: (i) we employ a synthetic renderer, capable of generating large amounts of realistically varying synthetic images; and (ii) we propose a domain adaptation strategy to bridge the gap between synthetic and real images. By mixing synthetic and real data in each minibatch during training, we improve the test accuracy for object classification tasks. Finally, we propose the Mixed-Reality Generative Adversarial Network (MrGAN) which iteratively maps between synthetic and real data via a multi-stage, iterative process. The result of the optimization is a shared space into which both real and synthetic images can be mapped. After training in the shared space, our models generalize better (from synthetic) to real data. We validate the advantages of using synthetic data and MrGANs on our CIFAR-based datasets for domain adaptation. Using both synthetic data and MrGANs, we achieve an improvement of 8.85% in test accuracy.

Keywords: renderer; synthetic data; domain adaptation; GANs; object classification

1 Introduction

Training state-of-the-art Computer Vision (CV) models remains remarkably data-intensive. the dramatic breakthrough results in object recognition [**?**] came on the back of ImageNet [**?**], an enormous dataset containing one million images, 1000 representatives each from 1000 categories. Creating datasets of this size requires both acquisition (of relevant images) and annotation (typically by humans). In general, annotation tends to be costly, and when data cannot be collected passively, acquisition itself might present a formidable expense. For a small research team or startup to create their own large datasets may be costly or infeasible. Even for large companies, data availability often comprises a formidable obstacle to deploying machine learning algorithms.

To combat deep learning's reliance on voluminous data, various techniques have been proposed. Some of the biggest successes owe to categories of transfer learning. In one common approach, called *fine-tuning*, researchers initialize a neural network for a new task B, with the pre-trained weights from a network trained on comparatively data-rich task A , exploiting the fact that many of the lower-level features are transferable across different CV tasks [**?**]. Still, fine-tuning approach may require many thousands of labeled examples on the target domain, and performance may depend strongly on precisely how similar the source and target domains are.

One ambitious approach to liberating researchers and practitioners from their dependence on annotated datasets is to exploit advances in graphics to render synthetic data for use in training machine learning models. Because we can access the underlying parameters driving the rendering engine, we know the correct label for each image. And given the power of today's graphics engines, we can generate a practically limitless number of images corresponding to each label. This increasingly popular and promising approach was recently adopted by [1], for eye gaze detection.

In this paper, we use a lightweight renderer that allows us to quickly render synthetic images with different latent variations. We show by experiments that augmenting real images with synthetic images during training help improve the test accuracy of the neural networks for object classification tasks. By adding more latent variations into the synthetic data and changing the mixing ratio between synthetic and real data in each minibatch, we can gain up to 4.81% and 13.23% improvement in test accuracy with and without data augmentation, respectively.

However, even with a good renderer, there is still a distribution gap between the rendered synthetic data and the real data. In order to "push" the synthetic data closer to the real data, we propose a new domain adaptation architecture, namely the Mixed-Reality Generative Adversarial Networks (MrGANs), that map synthetic images to the space of real images and vice versa. MrGANs is an extension of the CycleGANs [**?**] that iteratively maps between the two domains. We show that augmenting a small set of real images with refined synthetic images from the MrGANs improves the accuracy of the resulting predictive models.

In particular, we create a new dataset named CIFAR-Domain Adaptation (CIFAR-DA4). CIFAR-DA4 has 4 classes: car, bicycle, motorcycle, and train. Its training set contains synthetic images and real images, and its test set contains only real images. Real images in CIFAR-DA4 are from CIFAR100. We will describe CIFAR-DA4 and an extended version, CIFAR-DA9, in Sections 4.3 and 3.3, respectively. We show that the baseline training on real images only achieves 84.64% test accuracy on CIFAR-DA4. Augmenting real training images with synthetic images improves test set accuracy to 90.07%. Augmenting real training images with refined synthetic images from MrGANs further improves the test accuracy to 93.49%.

Our primary contributions are the following:

- Results demonstrating that augmenting real data with synthetic data can improve image classifiers
- Experiments suggesting that latent variations - such as different lighting conditions, textures, backgrounds - in synthetic data help the trained model generalizes better.
- We introduce MrGANs, a novel iterative domain adaptation approach that narrows down the gap between synthetic and real data and show that a classifier trained on real data augmented by refined synthetic data works better than the one trained on real data augmented by original synthetic data.

2 Related Work

2.1 Synthetic Renderer

Recently, graphic engines have been used in machine learning for various applications. Notable cases include training reinforcement learning algorithms for robotic without the high cost of interacting in real-world environments [**?**]. Powerful graphics engines, such as Unreal Engine, Unity, and Blender, have been used in conjunction with machine learning frameworks. For example, Unreal Engine supports embedded Torch scripts via UETorch [**?**]. However, these heavyweight graphics engines can be difficult to use. On the other hand, our synthetic renderer is lightweight and can be easily plugged into different machine learning frameworks.

2.2 Domain Adaptation

Domain adaptation is a classic area of machine learning. More theoretical works address the problem under strong assumptions such as covariate shift [**?**,**?**] or label shift [**?**]. Over the past few years, a flurry of papers have addressed the problem of domain adaptation with deep networks through a more empirical lens. These methods can be classified into two groups: representation-based and pixel-based methods.

Representation-Based Domain Adaptation: These methods try to map both source and target images into a representation space where the two are somehow matched The representation is typically produced by a hidden layer of a CNN. Some recent representation-based methods include the Correlation Alignment for Deep Domain Adaptation (DeepCORAL) network [2], the Domain Adaptation Network (DAN) [**?**], the Adversarial Discriminative Domain Adaptation (ADDA) network [**?**], and ReverseGrad [**?**]

Pixel-Based Domain Adaptation: The pixel-based methods convert images in the source domain into those in the target domain using, for example, the

Generative Adversarial Networks (GANs)[**?**]. Unlike the representation-based methods, pixel-based domain adaptation methods are independent of the task and the model (for example, the classifier in an object classification task or the detector in an object detection task). Relevant works in this direction include CycleGANs [**?**]. Additionally, SimGANs [1] apply a GAN-based approach to convert synthetic images to real images for domain adaptation between synthetic and real data. Both approaches require mapping form images to images, a fundamental ability demonstrated to be feasible with Pix2Pix-like architectures in [**?**].

Our MrGANs approach is an extension of the CycleGANs to convert between synthetic and real images. Unlike the CycleGANs and other approaches, we modify the images via two stages. Each stage is a CycleGAN. The second stage modifies the output of the first stagesin order to further close the gap between synthetic and real images. MrGANs is a pixel-based domain adaptation method and complimentary to the representation-based ones.

It is also worth noting that there are recent works combining both representation-based and pixel-based approaches such as the GraspGAN [**?**].

3 Synthetic Data

Synthetic images can be used as a substitute for real images when there is not enough of the later. However, it is not clear how to design synthetic datasets for improving learning or how to train a model with both modest amounts of real data and abundant synthetic data so that the advantages of using synthetic data are efficiently exploited. In this section, we first describe our lightweight renderer that we have developed for fast and flexible rendering of synthetic images. We then show that the accuracy of the trained models can be significantly improved by adding more latent variations into the synthetic datasets. When augmenting real data with synthetic data, we found that the ratio of real and synthetic images in each training minibatch plays an important role in determining the accuracy of the trained model.

3.1 A Lightweight Renderer for Machine Learning Applications

Our renderer is built on top of the Falcor rendering framework from NVIDIA [**?**]. It supports a variety of latent variations such as texture, lighting, camera views, and occlusion. Rendering process in the renderer can be parallelized across multiple GPUs, which enables it to take full advantage of cloud computational services. In addition, our renderer provides an extension package to render to NumPy arrays, which allows it to, potentially, interface well with many machine learning frameworks and libraries. Our renderer has been tested in Windows and Linux. Figure 1 shows a screenshot of the renderer's GUI. For large-scale rendering, our renderer also allows the users to automate the rendering process. Figure 2 shows sample images from the renderer with different latent variations.

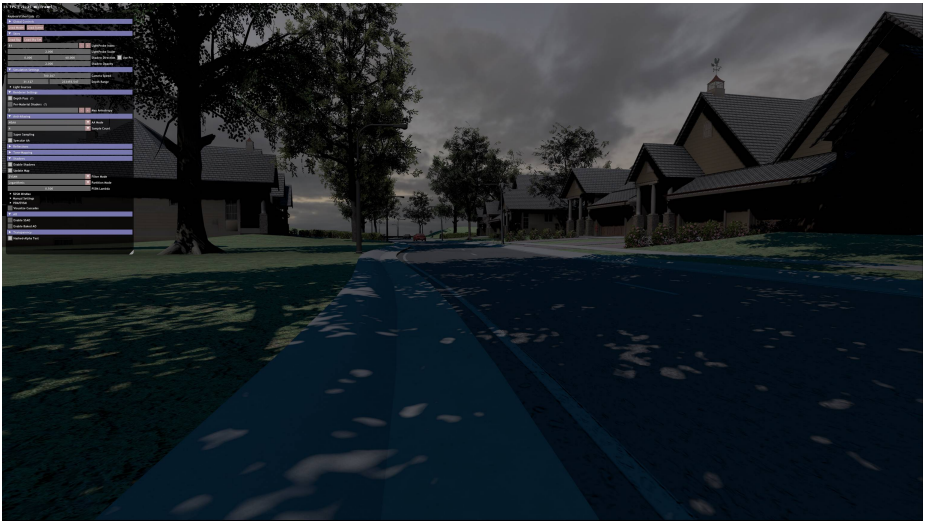

Fig. 1. Graphical user interface of of our renderer

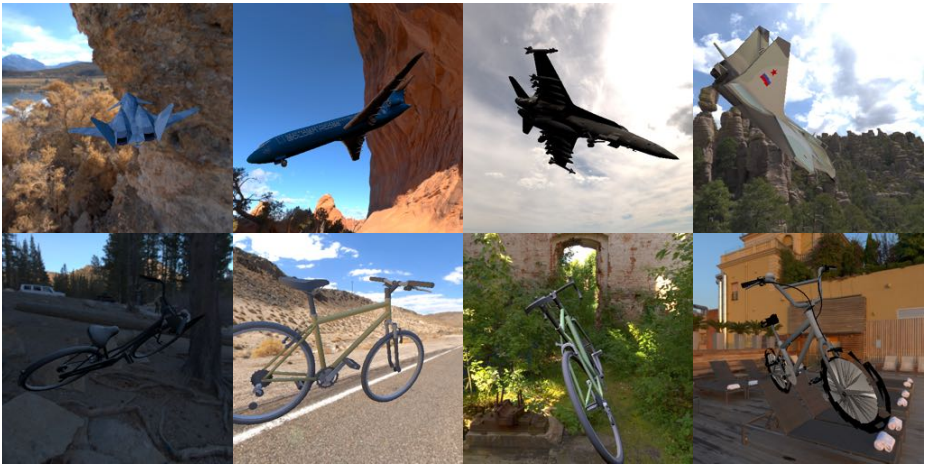

Fig. 2. Sample images rendered from our renderer with different latent variations. Top row: airplanes. Bottom row: bicycles

3.2 A Variety of Latent Variations in the Training Data is Necessary

Perceptual inference tasks such as object classification are complicated by a high amount of latent variations in the sensory input data. Those latent variations turn simple low-dimensional problems into much more challenging high-dimensional problems. For example, images of the same object but with different poses lie on a nonlinear and highly-curved manifold in a high-dimensional space,

making object classification tasks more difficult. A good machine learning model tries to capture the latent variations in the input and, during inference, marginalize out those task-irrelevant nuisances. To train such a model requires the training datasets to contain a variety of latent variations. Thus, it is expected that the synthetic training sets rendered from the renderer contains myriad different latent variations.

Experiment Setup We have conducted a empirical study on the impact of latent variations in the training data on the test accuracy of a trained CNN, where test data are real images and the task is binary object classification: airplane vs. bicycle. We use AlexNet [**?**] as the classifier in our experiments. The latent variations we consider are: texture, lighting conditions, backgrounds, camera viewpoints, and 2D data augmentations (including random mirror, random crop, random contrast, random illumination, and random shearing). In particular, we have done the following experiments:

- Experiment 1: Synthetic training data does not have the latent variations of interest.
- Experiment 2: Synthetic training data contains different textures.
- Experiment 3: Synthetic training data contains different textures + random lighting conditions.
- Experiment 4: Synthetic training data contains different textures + random lighting conditions + random backgrounds.
- Experiment 5: Synthetic training data contains different textures + random lighting conditions + random backgrounds + random camera views.
- Experiment 6: Synthetic training data contains different textures + random lighting conditions + random backgrounds + random camera views + different 2D data augmentation.
- Experiment 7: Training data are real images.

In all experiments, the synthetic training data are rendered from 120 airplane models and 86 bicycle models, 1800 images per models. There are totally 6000 real images of airplanes and bicycles, half used for test data and half used for training data in Experiment 7. The synthetic 3D models and the real data are from the VisDA Challenge Datasets for Object Classification [3].

Results Table 1 shows that adding more latent variations into the synthetic training set indeed improves the test accuracy on real data. Interestingly, adding various backgrounds into the synthetic images helps achieving much better test accuracy (8% improvement). While comparing to the network trained on real images, the best test accuracy when training on synthetic images is still behind (88.2% vs. 93.5%). In the next section, we will show that when combining real and synthetic data during training, we achieve better test accuracies.

Table 1. Test accuracies on real images of an AlexNet trained on synthetic image datasets which contains different numbers of latent variations.

Texture	Random Lighting	Random Background	Random Camera Views	2D Data Augmentation	Test Accuracy
					70.3%
x					71.6%
x	x				75.0%
x	x	x			83.0%
x	x	x	x		86.9%
x	x	x	x	x	88.2%
Training on Real Images					93.5%

3.3 Mixing Ratio between Real and Synthetic Data in Each Minibatch Helps the Learning

In addition to the number of latent variations in the synthetic training data, how to combine synthetic and real data during training also plays an important role in improving the accuracy and generalizability of the trained classifier. In particular, we observe that by simply changing the mixing ratio between synthetic and real data in each minibatch during the network training using stochastic methods such as the Stochastic Gradient Descent [**?**] or ADAM [**?**] can help increase the classification accuracy on the real images in the test set. We describe the dataset, our experiments, and discuss the results below.

Dataset and Experiment Setup CIFAR-DOMAIN ADAPTATION 9 (CIFAR-DA9): We create the CIFAR-DA9 dataset for our experiments. Images in CIFAR-DA9 belong to one of the following nine classes: airplane, car, bicycle, bus, horse, motorcycle, person, train and truck. CIFAR-DA9 contains a training set, a validation set, and a test set. The training set are synthetic and real images while the test and validation sets contain only real images. Synthetic images rendered from our renderer using 3D models from the VisDA Challenge Datasets for Object Classification [3] . Same as experiments in Section 3.2, we render 1,800 images per model. Real images from the CIFAR100 [**?**] and CIFAR10 [**?**]. For each class, we use 500 real images for training, 50 real images for validation, and 50 real images for testing.

In our experiment, during training, we train on both synthetic and real training images but vary their mixing ratio λ = Real:Synthetic. We use ResNet as the classifier, and the network is trained using ADAM [**?**].

Results Table 2 shows the test accuracies on CIFAR-DA9 when $\lambda = 1:1$, $\lambda = 1:7$, and when $\lambda = 1:15$ for training with and without 2D augmentations (random crop and random flip). We also show the test accuracy when training on real images for reference. Using 2D augmentations during training, $\lambda = 1:1$ yield

Table 2. Test accuracy on the real images versus the mixing ratio between real and synthetic data in each minibatch for CIFAR-DA9 without 2D augmentation (Top) and with 2D augmentation (Bottom).

Training Setup	Mixing Ratio	Test Accuracies
No 2D Augmentation		
Real + Synthetic	1:15	**79.17**%
	1:7	77.29%
	1:1	74.44%
Real Only		65.94%
With 2D Augmentation		
Real + Synthetic	1:15	81.04%
	1:7	82.59%
	1:1	**84.71**%
Real Only		79.90%

the best test accuracy (84.71% vs. 79.90% when training on real images only). However, without 2D augmentation, $\lambda = 1 : 15$ yields the best test accuracy (79.17% vs. 65.94% when training on real images only).

We observe that when training with solely real images, the training is saturated quickly, and even with a decay learning rate (at epoch 89 and 139) the test accuracy is not improved much (see Appendix for the training curves). This is because of the small size of the training sets in the experiment (only 500 real images per class). In such cases, using synthetic data adds more advantage. These results show a potential of using synthetic data, together with real data, to train neural network models.

4 Domain Adaptation

Synthetic and real images are drawn from different distributions. Good synthetic datasets, when using in conjunction with real data, help learn a better model. However, even with the best available renderers, a distribution gap between synthetic and real images remains. This gap between training and test data is known as the covariate shift problem [**?**]. Our approach to overcome the covariate shift is to modify both synthetic and real data so that they are closer to each other. In particular, we convert synthetic data into real data and vice versus using a CycleGAN [**?**] (Stage 1). Since the distribution gap between synthetic and real data is usually large, we repeat the process using the outputs of Stage 1 together with the original synthetic and real data to further narrow down the distribution gap and reduce the covariate shift problem (Stage 2).

4.1 Cycle Generative Adversarial Networks

A Cycle Generative Adversarial Network (CycleGAN) is consisted of two generative adversarial networks (GANs) G and F. Each network takes images from

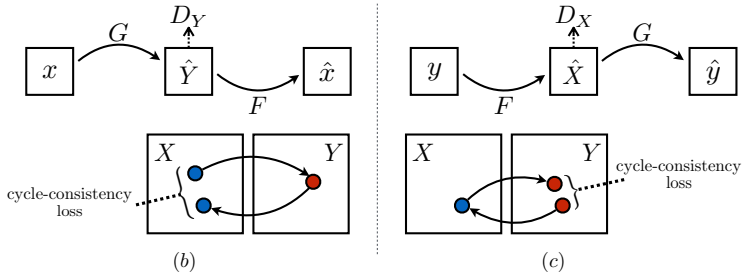

Fig. 3. CycleGANs. Source and target images are mapped toward each other via adversarial and cycle-consistency losses [**?**].

one domain as input and converts them into another domain via an adversarial loss, which distinguishes between converted images and images in the target domain. In particular, as illustrated in Figure 3, the network G maps images from domain X into domain Y, and the network F maps images from domain Y into domain X. In order to promote the consistency between the mappings and improve the quality of the generated images, cycle-consistency losses are introduced to enforce F to be the inverse mapping of G and vice versus.

4.2 MrGANs: The Mixed-Reality Generative Adversarial Networks for Iterative Domain Adaptation

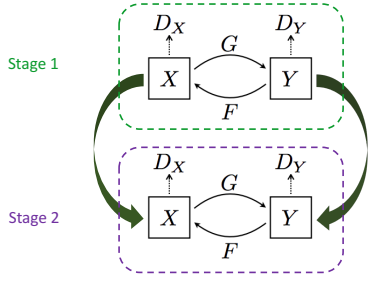

Fig. 4. Mixed-Reality Generative Adversarial Networks (MrGANs) that maps between synthetic and real images via a 2-stage process. The generated images from stage 1 are used as inputs for stage 2.

The Mixed-Reality Generative Adversarial Networks (MrGANs) is an extension of the CycleGANs, in which the mappings between domains are performed in two consecutive stages (see Figure 4). In this paper, we consider one domain to be synthetic images and another domain to be real images (see Figure 5A).

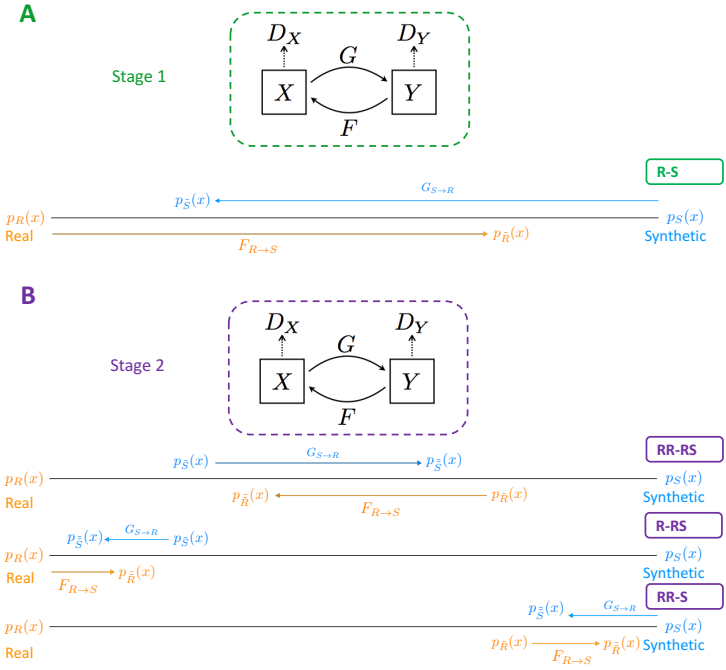

Fig. 5. The 2-stage process of the Mixed-Reality Generative Adversarial Networks (MrGANs). (A) Stage 1, synthetic images are mapped to real images and vice versus (R-S) via a CycleGANs [**?**]. (B) Stage 2, MrGANs do the following mappings: (RR-RS) mapping between refined real and refined synthetic images, (R-RS) mapping between original real and refined synthetic images, (RR-S) mapping between refined real and original synthetic images. All mappings are done via the CycleGANs

In Stage 1 of MrGANs, synthetic images are mapped into real images and vice versus. In Stage 2, MrGANs does one of the following (see Figure 5B):

- Map between the refined synthetic and refined real images from Stage 1.
- Map between the refined synthetic images from Stage 1 and the original real images.
- Map between the original synthetic images and the refined real images from Stage 1.

This extra mapping in Stage 2 allow the distribution gap between synthetic and real data to be narrowed down further. Even though we only present the two-stage MrGANs in this paper, MrGANs can be extended into more than two stages.

4.3 Experiment Setup

We conduct experiments on the airplane-bicycle dataset (see Section 3.2) and the CIFAR-DA4 dataset. We show the advantages of refined synthetic images from the MrGAN over the original synthetic images and the refined synthetic images from the CycleGAN when training with synthetic data only (for both the airplane-bicycle and CIFAR-DA4 datasets) and when training with real data augmented by synthetic data (for the CIFAR-DA4 dataset). We use an AlexNet and a ResNet as the classifier for the airplane-bicycle and the CIFAR-DA4 datasets, respectively, in our experiments. More details on the network architecture can be found in the Appendix.

CIFAR-DOMAIN ADAPTATION 4 (CIFAR-DA4): In our experiments, we use a smaller version of CIFAR-DA9 (see Section 3.3), namely CIFAR-DA4, which only contains 4 classes: car, bicycle, motorcycle, and train. Real images in CIFAR-DA4 are only from CIFAR100 [**?**].

The experiment setups are same as before. As mentioned above, here, we consider two settings:

1) Train on synthetic data or refined synthetic data only. The test accuracies are computed using real or refined real images.

2) Augment real data with synthetic data or with refined synthetic data to form a hybrid training set. When testing, we again test on real or refined real images.

4.4 Results

Train on (Refined) Synthetic Data Only The top half Table 3 shows the test accuracies on the CIFAR-DA4 dataset while training using (refined) synthetic data only. The test accuracy when training with the refined synthetic data from MrGANs is the highest. In this case, we map refined synthetic images from Stage 1 of MrGANs to refined real images from Stage 1 of MrGANs and test the model with refined real data from the Stage 1 of MrGANs. In the two other cases (CycleGANs and no modification), we test the model with original real data since it yields better results.

We also conduct the same experiments using the airplane-bicycle dataset in Section 3.2. Figure 7 shows the test accuracies with different refined datasets (both real and synthetic) from MrGANs. Again, the model trained and tested on refined data from MrGANs yields highest test accuracy. Samples of modified images from the two training stages of MrGANs for the airplane-bicycle dataset are shown in Figure 6.

Train on Augmented Data When concatenating with real data during training, refined synthetic data from MrGANs also show an advantage over the original synthetic data and refined synthetic data from the CycleGANs. We again demonstrate this advantage empirically via an object classification task on CIFAR-DA4. In all experiments, we cross-validate the mixing ration between

synthetic and real data in each minibatch. We present the results in Table 3. The test accuracies during the training are shown in Figure 8. More results can be found in the Appendix.

Table 3. Test accuracy on the real images versus the mixing ratio between real and synthetic data in each minibatch for CIFAR-DA4 without 2D augmentation (Top) and with 2D augmentation (Bottom).

Training Setup	Test Accuracies
Synthetic Data Only	
Refined synthetic images from MrGANs	**63.55**%
Refined synthetic images from CycleGANs	61.02%
Synthetic images without modification	57.52%
Real + Synthetic Data	
Real + Refined synthetic images from MrGANs	**93.49**%
Real + Refined synthetic images from CycleGANs	89.73%
Real + Synthetic images without modification	90.07%
Real data Only	84.64%

The training with refined synthetic images from MrGANs yields an improvement in test accuracies on real data over both training with the original synthetic images (around 2.42% better) and training with the refined ones from CycleGANs (around 3.76% better). Notice that in all cases here, we train the networks with both real and (refined) synthetic images while testing on real images.

5 Conclusions

In this paper, we have used a lightweight renderer to generate synthetic images for object classification tasks. We have shown that the accuracy of neural networks models can be improved by training with real data augmented by synthetic data for object classification tasks. Significant improvement can be achieved by adding more latent variations into the synthetic and choosing the right mixing ratio between synthetic and real data in each minibatch during training. Using refined synthetic images from MrGANs further improves the accuracy of the neural network model.

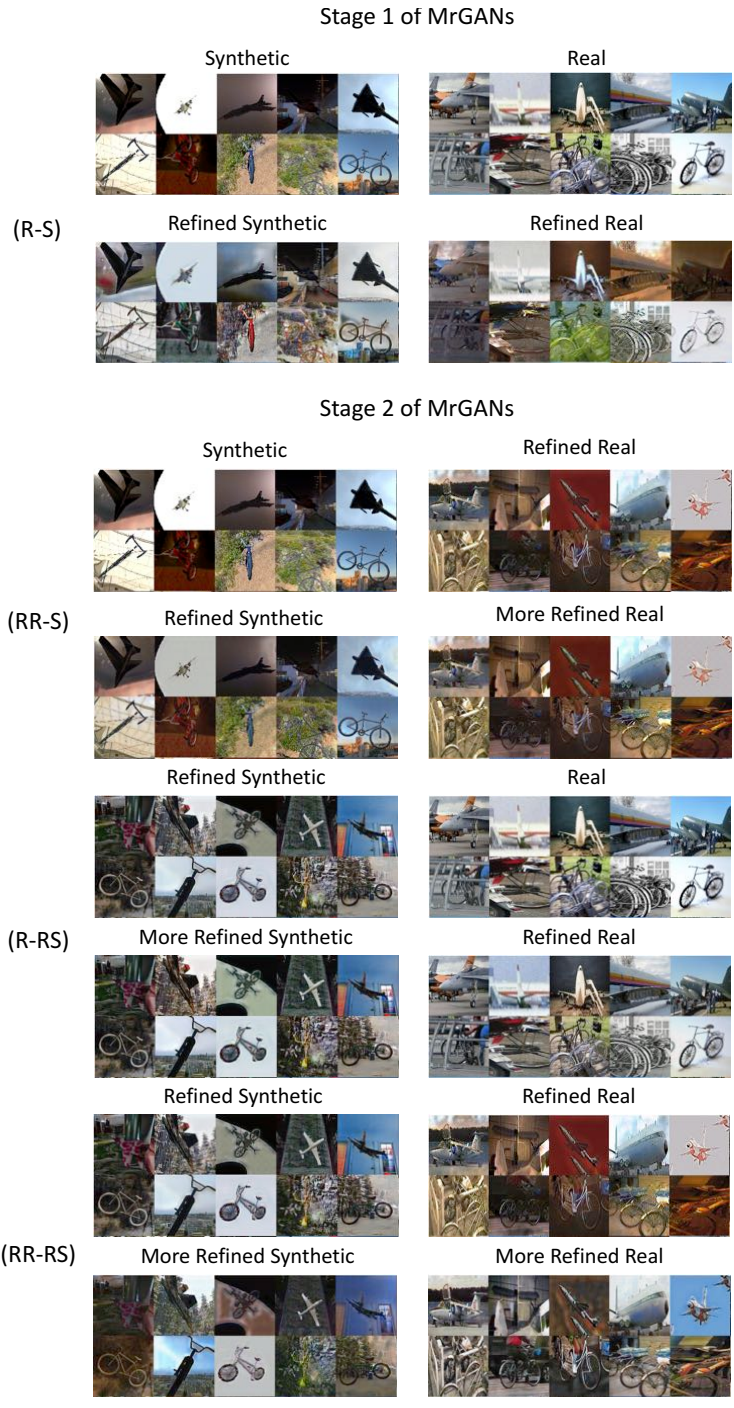

Fig. 6. Modified images from two stages of the MrGAN trained on airplane and bicycle images.

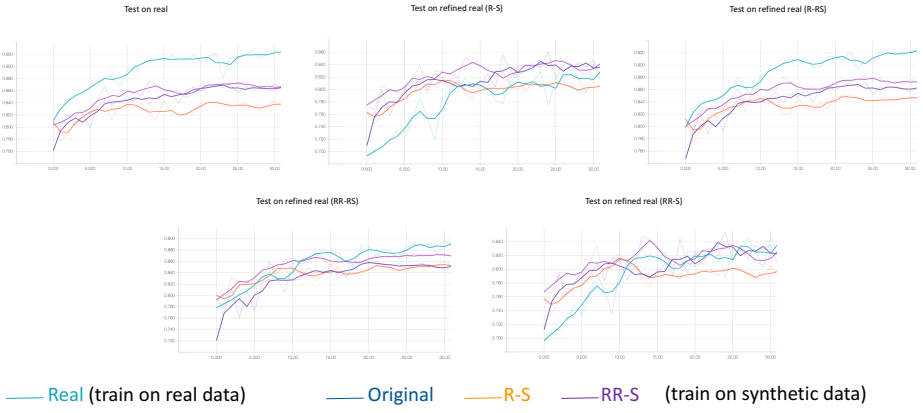

Fig. 7. Test accuracies on classifying airplane and bicycle.

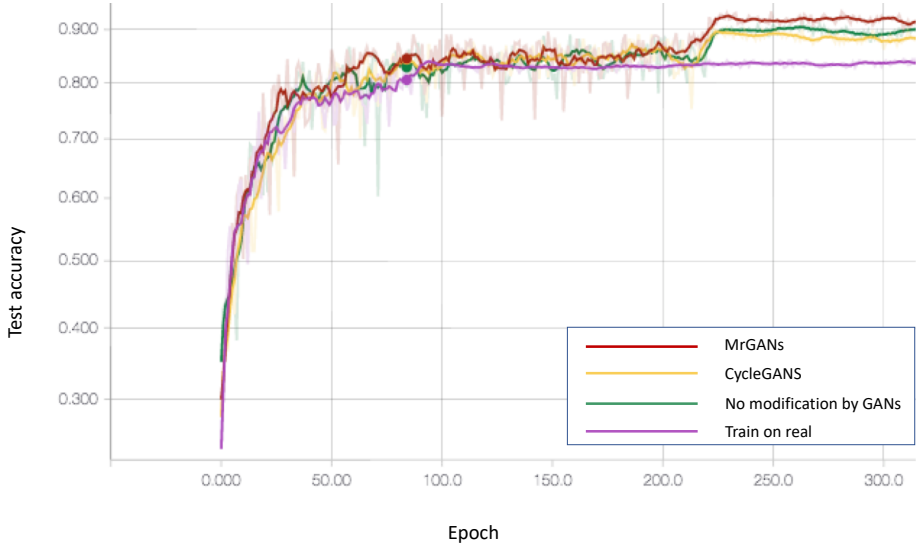

Fig. 8. Test accuracies on CIFAR-DA4 when augmenting the real training images with (refined) synthetic data.

Appendix: Learning image classifiers from (limited) real and (abundant) synthetic data

Anonymous ECCV submission

No Institute Given

1 Results for Airplane vs Bicycle with Various Mixing Ratio

See Table 1 below

Table 1. Test accuracies on the real images versus the mixing ratio between real and synthetic data in each minibatch for the airplane-bicycle classification task without 2D augmentation (Top) and with 2D augmentation (Bottom).

Training Setup	Mixing Ratio	Test Accuracies
No 2D Augmentation		
Real + Synthetic	1:15	**93.71**%
	1:7	93.25%
	1:1	94.17%
Real Only		93.5%
With 2D Augmentation		
Real + Synthetic	1:15	96.04%
	1:7	**96.22**%
	1:1	96.07%
Real Only		95.04%

2 Architecture and Training Details for Classifying Airplane and Bicycle in Section-3.2

2.1 Architecture

In Section 3.2, we train an AlexNet [**?**] on synthetic data only. Since we are doing binary classification, airplane vs. bicycle, at the end of the network, we use a 2-way softmax classifier. Let Ck-f-s-p denote a Convolution-ReLU layer with k filters of size f $\times$ f, stride s, and padding p. M denotes a MaxPooling layer of size 3 x 3 and stride 2. Dk denotes a Fully-Connected layer with k filters. Dropout(0.5) denotes a Dropout layer with the dropout rate 0.5. The architecture of our network is as follows:

C64-11-4-2, M, C192-5-1-2, M, C384-3-1-1, C256-3-1-1, C256-3-1-1, M, D4096, Dropout(0.5), D4096, Dropout(0.5), 2-way Softmax.

2.2 Training Details

We initialize our network using the Glorot initialization [**?**]. We use ADAM [**?**] with learning rate 0.0001 to train our network in 250 epochs. After epoch 50, we decay the learning rate to zero linearly. We use the batchsize 128.

3 Architecture and Training Details for Object Classification with CIFAR-DA9 in Section 3.3

3.1 Architecture

In Section 3.3, we train a ResNet [**?**] on real data augmented by synthetic data. Let Ck-f-s-p denote a Convolution layer with k filters of size f $\times$ f, stride s, and padding p. BN denotes batchnorm, ReLU denotes the layer which applies the activation ReLU, and AvgPooling denotes a AvgPooling layer. Let Rk-s denote a ResNet building block with k filters and stride s. Rk-s is the same as the ResNet building block without the bottleneck described in [**?**]. Let Bk-r-s denote a block containing r Rk-s residual blocks. Bk-r-s consists of Rk-s, Rk-1, Rk-1.

The architecture of our network is as follows:
C16-3-1-1, B16-3-1, B32-3-2, B64-3-2, AvgPooling, 9-way Softmax.

3.2 Training Details

We initialize our network using the Glorot initialization [**?**]. When training with real data augmented by synthetic data, we use SGD [**?**] with learning rate 0.1 to train our network in 500 epochs. After epoch 221 and 348, we divide the learning rate by 10. When training with real data only, we use SGD [**?**] with learning rate 0.1 to train our network in 500 epochs. After epoch 89 and 139, we divide the learning rate by 10. We also use weight decay 0.0001 in both cases. We use the batchsize 128.

4 Test Accuracy over Epochs for CIFAR-DA9 Object Classification Trainings in Section 3.3

See Figure 1 for the test accuracy vs. epochs curves for CIFAR-DA9 trainings.

5 Aligning Labels during the Training of MrGANs

During the training of MrGANs, we align real and synthetic images using the labels. In particular, in each iteration, we use pairs of real and synthetic images from the same class to train MrGANs, e.g. use real airplane images to refine synthetic airplane images and vice versa, use real motorcycle images to refine synthetic motorcycle images and vice versa.

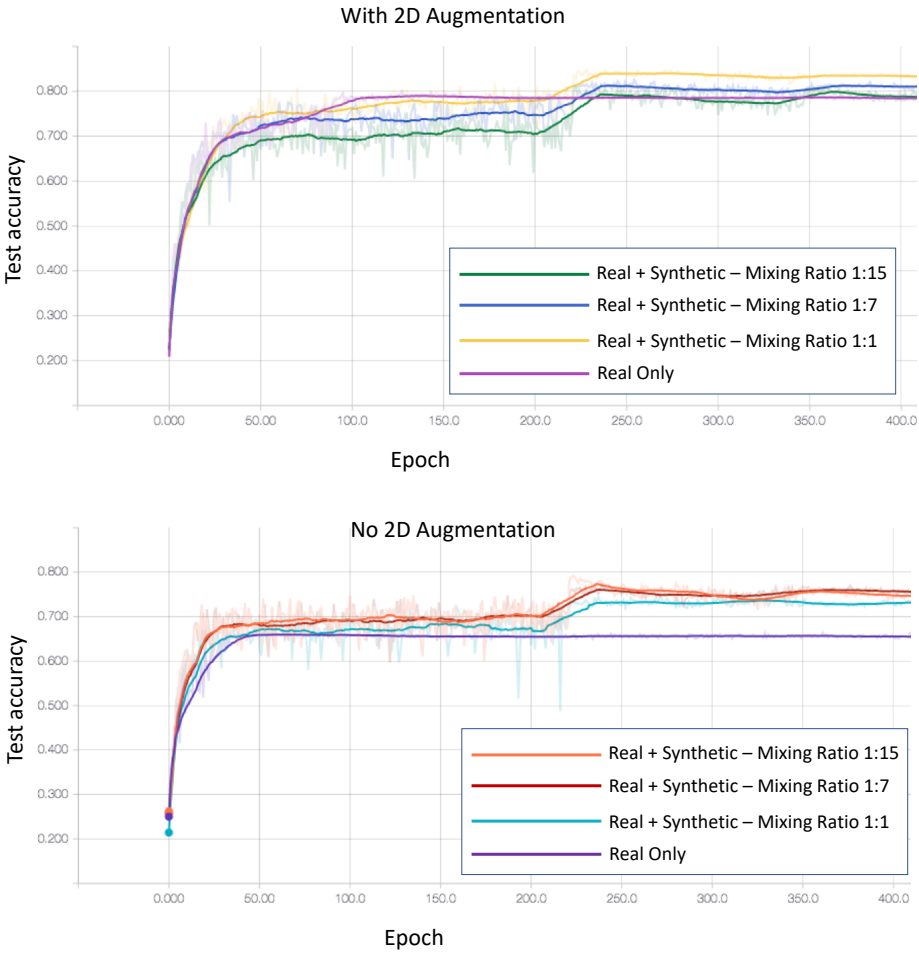

Fig. 1. Test accuracies on the real images versus the mixing ratio between real and synthetic data in each minibatch for CIFAR-DA9 classification task with 2D augmentation (Top) and without 2D augmentation (Bottom).

6 MrGANs Architecture and Training Details for Domain Adaptation with the Airplane-Bicycle Dataset in Section 4.3

6.1 Architecture

The building blocks of MrGANs are the CycleGANs. For training airplane vs. bicycle, we use the same CycleGAN with 9 blocks as in [**?**]. Using the same notations defined in [**?**], our generator consists of:

c7s1-32, d64, d128, R128, R128, R128, R128, R128, R128, R128, R128, R128, u64, u32, c7s1-3,

and our discriminator consists of:

C64, C128, C256, C512.

Same as the CycleGANs in [**?**], we use the least-square losses [**?**] (as the adversarial losses) (L_{GAN}) and the cycle-consistency losses (L_{Cycle}) to train the networks.

$L = L_{GAN} + 10L_{Cycle}$

6.2 Training Details

We initialize the scaling correction terms in Batchnorm and the weights using Gaussian distributions of zero mean and variance 0.02. We initialize other parameters by 0. We use ADAM [**?**] with learning rate 0.0002 for the generators and 0.00001 for the discriminator. We train for 40,000 iterations. After iteration 20,000, we decay the learning rate to zero linearly. We use the batchsize 32 (note that the image size is 256 x 256 x 3). During each training iteration, we update the generator 5 times and the discriminator 1 time. Also, while training the discriminator, we randomly flip the labels (source vs target domain) with the probability 0.9. In addition, we use label smoothing [**?**]. For each incoming image to the discriminator, if the image is from the target domain, then replace the label with a random number between 0.7 and 1.2, and if the image is a sample from the source domain, the replace the label with 0.0 and 0.3.

Before the training of each CycleGAN in MrGAN, we pretrain the CycleGAN's generator for 1,000 iteration and the CycleGAN's discriminator for 500 iterations. We use L1 loss between the original images and the refined images (L_{L1}) to pretrain the generator. We use the adversarial loss to pretrain the discriminator.

7 MrGANs Architecture and Training Details for Domain Adaptation with the CIFAR-DA4 Dataset in Section 4.3

The architecture and training details for domain adaptation with the CIFAR-DA4 dataset are similar to those with the Airplane-Bicycle dataset in Appendix 6. There are only a few differences as follows:

- The discriminator consists of: C64, C128.
- In addition to the adversarial loss (L_{GAN}) and the cycle-consistency loss (L_{Cycle}), we also apply a L1 loss between the original images and the refined images (L_{L1})
  $L = L_{GAN} + 10L_{Cycle} + 0.1L_{L1}$
- We use the batchsize 144.

.

Rebuttal

We thank the reviewers for their valuable comments.

The novelty of MrGANs (Reviewer 1, Reviewer 3): The key idea of MrGANs is to stack multiple GANs for domain adoptation. We show that the gap between real and synthetic data can be further narrowed down via this multi-stage domain adaptation approach. Our method can be applied on most GAN architectures, and we use the CycleGANs as the baseline only to demonstrate our method. While this approach seems to be simple, there are multiple options at the second stage after training the first GAN. As illustrated in Figure 5B, there are three possible mappings at the second stage: (1) mapping between refined real and refined synthetic images, (2) mapping between original real and refined synthetic images, and (3) mapping between refined real and original synthetic images. We study all of these possible mappings in our experiments and show the domain adaptation and classification results for the task of classifying airplane vs. bicycle in Figure 6 and 7, respectively. We consider this study non-trivial and worth discussing.

CIFAR-DA9 dataset is very simple and not the standard in the domain adaptation literature (Reviewer 1, Reviewer 2): We agree with the reviewer that the CIFAR-DA9 is simple, and we need to test our method on more challenging datasets with more classes and higher resolution images like those using images from ImageNet or MS COCO as the target domain, e.g. the target datasets used in the VISDA challenge [3]. However, we would just like to make a small correction here. The virtual 3D models in CIFAR10-DA9 are not very close to the real data. While we try to select high-quality models to create CIFAR10-DA9, there are still notable differences between the synthetic images and the real ones. In fact, in our opinion, the quality of the 3D object models is the main challenge in using synthetic images for machine learning tasks and doing domain adaptation between synthetic and real images. As a result, we consider CIFAR10-DA9, as well as CIFAR10-DA4, to be reasonable benchmarks to test our method.

Comparison with other virtual image generators (Reviewer 2): We do not mean to claim the renderer as one of the paper's contributions. Our renderer is only a prototype at the time we submitted the paper, and we discuss the renderer in the paper only to be clear on how we generate the synthetic data. However, compared to other existing virtual image generators, we plan to make our renderer more lightweight and easier to plug into popular deep learning frameworks for different tasks including object classification, active learning, and reinforcement learning.

Relation to Balanced Gradient Contribution (Reviewer 2): Our mixing real and virtual samples method is similar to the Balanced Gradient Contribution in [4, 5]. Slightly different from [4, 5], we study the method for classification using both synthetic and real images instead of semantic segmentation.

Comparison with the state of art of domain adaptation for classification (Reviewer 2, Reviewer 3): We agree with the reviewer that we need to compare MrGANs to other state-of-the-art domain adaptation methods for object classification on standard benchmarks. However, we would like to mention

that most of the state-of-the-art domain adaptation methods for classification involve either converting images from the source domain into the target domain using the GANs (pixel-based methods, e.g. [1]) or matching the representations learned from source and target images (representation-based methods, e.g. [2]) or the combination of both. As mentioned in the **The novelty of MrGANs** section above, MrGANs can be applied on pixel-based methods by stacking the GAN baseline architectures. The contribution of our paper is to show that this stacking approach improves the classification results when training with both revised synthetic and real images. Additionally, in the Related Work section, we discuss that the representation-based domain adaptation methods are complimentary to pixel-based methods including MrGANs. To our understanding, the combination of MrGANs and these representation-based methods may produce even better outcomes, and we will explore this combination in our future work.

Why is it going to work well when CycleGANs are stacked? (Reviewer 3): We agree with the reviewer that the paper does not have a rigorous proof of why stacking multiple GAN baseline architectures will help improve the performance. However, such rigorous proof is outside the scope of this paper. Instead, we provide a brief and intuitive explanation of why our approach should work at the beginning of Section 4. The intuition is that unlike other domain adaptation tasks, the distribution gap between synthetic and real images is usually large. As a result, it is difficult to convert synthetic images to real images in one step. Using the divide and conquer approach, we split this domain adaptation task into multiple stages. At each stage, we adapt the synthetic images one step closer to the real ones. We validate our stacking approach by experimental results. As the reviewers mentioned, we need experiments on more challenging datasets and more analysis of the experimental results to support our approach.

Why does the accuracy of CycleGAN drop in the last experiment? (Reviewer 3): This is showing the advantage of MrGANs. The contribution of the paper is not to show that CycleGANs can be used to do domain adaptation for classification. We would like to show that CycleGANs and other GAN architectures are not adequate to accomplish this task and by stacking them to form MrGANs, we gain improvement in the classification accuracy when training with both revised synthetic and real images.

References

1. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from simulated and unsupervised images through adversarial training. In: CVPR. (2017)
2. Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: ECCV. (2016)
3. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: The visual domain adaptation challenge. (2017)
4. Ros, G., Stent, S., Alcantarilla, P.F., Watanabe, T.: Training constrained deconvolutional networks for road scene semantic segmentation. arXiv preprint arXiv:1604.01545 (2016)
5. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: CVPR. (2016)