SIMULATED+UNSUPERVISED LEARNING WITH ADAPTIVE DATA GENERATION AND BIDIRECTIONAL MAPPINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Collecting a large dataset with high quality annotations is expensive and time-consuming. Recently, Shrivastava et al. (2017) propose *Simulated+Unsupervised (S+U) learning*: It first learns a mapping from synthetic data to real data, translates a large amount of labeled synthetic data to the ones that resemble real data, and then trains a learning model on the translated data. While their algorithm is shown to achieve the state-of-the-art performance on the gaze estimation task, it may have a room for improvement, as it does not fully leverage flexibility of data simulation process and consider only the forward (synthetic to real) mapping. Inspired by these limitations, we propose a new S+U learning algorithm, which fully leverage the flexibility of data simulators and bidirectional mappings between synthetic data and real data. We show that our approach achieves the improved performance on the gaze estimation task, outperforming (Shrivastava et al., 2017).

1 INTRODUCTION

Collecting a large annotated dataset is usually a very expensive and time-consuming task, and sometimes it is even infeasible. Recently, researchers have proposed the use of synthetic datasets provided by simulators to address this challenge. Not only synthetic datasets can be easily annotated but one can also generate an arbitrarily large amount of synthetic data. In addition to that, recent advances in computer technologies enabled synthesis of high-quality data.

Specifically, a variety of computer vision applications have benefitted from advanced computer graphics technologies. For instance, Wood et al. (2016) manipulate a 3D game graphics engine, called *Unity*, to synthesize photo-realistic images of human eye regions. Then, using a million of synthetic images with labels, they achieve the state-of-the-art performance on the cross-domain appearance-based gaze estimation task (Sugano et al., 2014b). Another important application that is heavily benefitting from synthetic data is autonomous driving. A few recent works show that an infinite amount of realistic driving data can be collected from high-quality video games such as GTA V (Grand Theft Auto V) (Richter et al., 2016; Johnson-Roberson et al., 2016; Lee et al., 2017). Specifically, Richter et al. (2016) show that a semantic segmentation model trained only with synthetic data can even outperform the model trained with real data if the amount of synthetic data is large enough. In (Lee et al., 2017), the authors collect a synthetic dataset of vehicle accidents, which is hardly collectable from the real world, train an accident prediction model with the synthetic data, and then apply the trained model to a real accident dataset. As a result, they show that the model trained with large synthetic dataset can outperform the model trained within small real dataset. Further, researchers also propose the use of simulated environments for building algorithms for autonomous drones (Microsoft, 2017), autonomous truck driving (Im, 2017), etc.

While the use of synthetic datasets is increasingly popular, it is not clear how one should systematically address a machine learning problem when simulated data is given with unlabeled real data. Recently, (Shrivastava et al., 2017) propose *Simulated+Unsupervised (S+U) learning*, which is one the first methodologies that guides how one can use synthetic data to improve the performance of learning algorithms. They first learn a translation mapping from synthetic data to real data using a modified GAN (Generative Adversarial Networks) architecture (Goodfellow et al., 2014), map the synthetic data to the real-data domain, and then train a learning model with this mapped data.

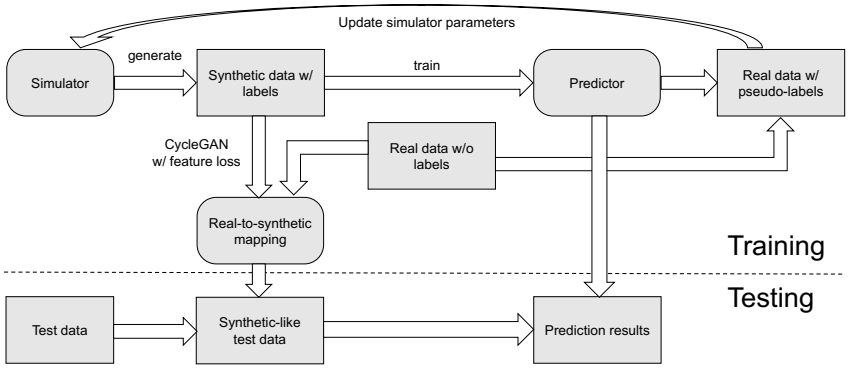


Figure 1: A block diagram of our S+U learning algorithm. The upper part of the diagram illustrates our training algorithm. First, we generate labeled synthetic data using a simulator, and then train a predictor on this data. Once the predictor is trained, we predict the labels of the unlabeled real data, which is available in the training phase. We call these predicted labels 'pseudo-labels'. Once the pseudo-labels are obtained, we compare the distribution of synthetic labels with that of pseudo-labels, and accordingly update the simulator parameters to reduce the gap. This procedure is repeated a few times until one achieves a small enough gap between the distributions. Once the synthetic dataset is finalized, we obtain a real-to-synthetic mapping between synthetic and real data. For testing, we first map test data to the synthetic domain, and then apply the predictor on it.

Using this methodology, they achieve the state-of-the-art performances for the cross-domain gaze estimation on the MPIIGaze dataset (Sugano et al., 2014b).

Even though the work of (Shrivastava et al., 2017) presents an interesting solution to deal with simulated data, their solution has some room for improvements for the following reasons. First, their approach assumes a fixed synthetic data, and does not leverage the flexibility of data simulation process. Since the data simulator can be freely manipulated, one may hope for further performance improvements. In addition to that, their approach makes a use of the forward (synthetic to real) mapping only, while recent works have shown the efficacy of bidirectional mappings between two domains (Zhu et al., 2017; Kim et al., 2017). Inspired by these limitations, we propose a new S+U learning framework, which fully leverages both the flexibility of data simulators and bidirectional mappings between synthetic data and real data.

In this work, we propose a new S+U learning framework, consisting of three stages. The first stage is where we fully leverage the flexibility of data simulators. Here, we first predict the labels of the unlabeled real data, which is available in the training phase. We then update simulation parameters so that the synthetic label distribution matches the distribution of the predicted labels, and this procedure is repeated a few times. The second stage learns a bidirectional cross-domain mapping between the synthetic data and the real data, using the cyclic image-to-image translation frameworks (Zhu et al., 2017; Kim et al., 2017). To better preserve labels while translating images, we make a few modifications to the existing framework, observing a significant performance improvement. In the last stage, we translate test (real) data to the synthetic domain, and then apply to them an inference model trained on the synthetic images. Note that our approach does not require additional training of the inference model. Our approach is superior if the backward mapping from real data to synthetic data is highly accurate and the inference model is well trained on the synthetic data. We show that our approach achieves the improved performance on the gaze estimation task, outperforming the state of the art of (Shrivastava et al., 2017). Our contributions can be summarized as follows:

Contributions

1. A new end-to-end S+U learning algorithm is proposed (Sec. 2).
2. We develop a simple method of exploiting the flexibility of simulators (Sec. 2.1).
3. We propose a novel prediction method that first translates real images to synthetic images and then applies a prediction algorithm trained on synthetic images (Sec. 2.3).
4. Using the proposed methodology, we outperform the state-of-the-art performance on the gaze estimation task (Sec. 3).

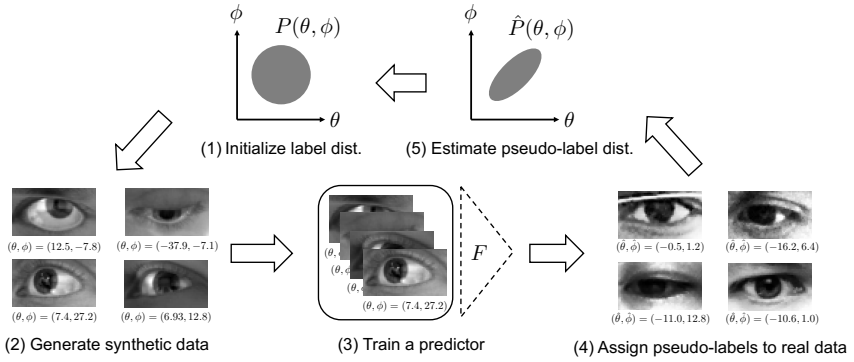


Figure 2: An overview of our adaptive data generation process. For the ease of explanation, consider the eye gaze estimation problem where the goal is to estimate a 2-dimensional gaze vector (θ, ϕ) given an image of the eye region. In the beginning, the simulator arbitrarily initializes the label distribution, say $P(\theta, \phi)$. It then draws random labels according to the label distribution, and generates corresponding images according to some rendering rules. Using this synthetic dataset, it trains a predictor, and then predicts the gaze vectors of each real image (initially unlabeled), i.e., annotates each image with a *pseudo-label*, using the trained predictor. The last stage estimates the pseudo-label distribution $\hat{P}(\theta, \phi)$, which is used as the initial distribution of the subsequent iteration.

2 Our Approach

2.1 Adaptive data generation based on pseudo-labels

Let us first consider how one would build a simulator that can generate an arbitrary amount of synthetic data. For ease of explanation, imagine a simulator that generates synthetic images for the digit classification problem. A typical process of building a simulator is as follows. First, one chooses a distribution of labels, arbitrarily or possibly aided with the prior knowledge on the target dataset. Then, the simulator specifies a distribution of images, conditioned on each of the labels. For instance, given the label "0", a simulator might first draw a perfect ellipsoid and then add some pixel noise to reflect the diversity of images.

As illustrated above, a simulator is usually fully specified with the label distribution and the data (image) distribution conditioned on each label. A naïve choice of taking arbitrary distributions may result in a distributional gap between the synthetic dataset and real dataset, i.e., sample selection bias (Huang et al., 2007). Fortunately, for most existing simulators, it is easy to adjust the label distribution. For instance, *UnityEyes* is a high-resolution 3D simulator that can render realistic human eye regions (Wood et al., 2016). The underlying label distribution can be adjusted by controlling 8 parameters of the label distribution. See Sec. 3 for more details. On the other hand, the way an image is rendered given a label is is not modifiable.

Motivated by this observation, we propose a simple iterative algorithm that can be used to find a good set of distribution parameters for the data generator. We note that the approach of (Shrivastava et al., 2017) does not adjust the data generation process, and stick with an arbitrary distribution. Our algorithm is based on the novel use of *pseudo-labels* (Lee, 2013). Pseudo-label is originally proposed to tackle semi-supervised learning problems in which a large number of unlabeled data is provided with a small number of labeled data: Once a coarse predictor is trained with the small labeled dataset, one can assign pseudo-labels to the large unlabeled dataset using the prediction results of the coarse predictor.

We now describe our algorithm, visualized in Fig. 2. It first starts with an arbitrary label distribution (and hence an arbitrary data distribution), and then trains a predictor model using the generated data. Once the predictor is trained, it computes the empirical distribution of the pseudo-labels of the unlabeled real dataset. Finally, it finds the new label distribution for the simulator that minimizes

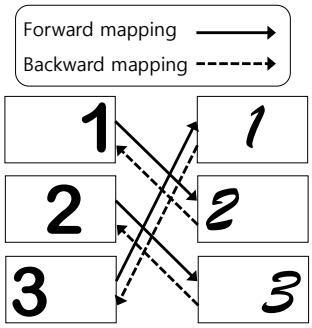


Figure 3: A toy example where perfect cycle-consistency is achieved but labels are not preserved across domains.

the distance from it to the pseudo-label distribution in some metric, and then accordingly updates the simulator parameters.[1] This procedure can be repeated a few times.

2.2 Label-preserving image-to-image translation

Recently, a GAN-based bidirectional image-to-image translation is proposed (Zhu et al., 2017; Kim et al., 2017). In this work, we slightly modify the CycleGAN framework of Zhu et al. (2017) to better preserve labels when translating images. We begin with briefly describing the original CycleGAN framework. Denote by $\mathcal{X}$ the synthetic dataset and by $\mathcal{Y}$ the real dataset. Define the GAN loss as:

$$\mathcal{L}_{\text{GAN}}(G, D, \mathcal{X}, \mathcal{Y}) := \mathbb{E}_{y\sim p_{\mathcal{Y}}(y)}[\log D(y)] + \mathbb{E}_{x\sim p_{\mathcal{X}}(x)}[\log(1 - D(G(x))], \tag{1}$$

where we abuse the notation by denoting by $\mathbb{E}_{x\sim p_{\mathcal{X}}}$ the empirical average with respect to a given dataset X. Using this notation, the forward GAN loss and the backward GAN loss can be written as:

$$\mathcal{L}_{\text{GAN, forward}} := \mathcal{L}_{\text{GAN}}(G_{\mathcal{X}\to\mathcal{Y}}, D_{\mathcal{Y}}, \mathcal{X}, \mathcal{Y}), \tag{2}$$

$$\mathcal{L}_{\text{GAN, backward}} := \mathcal{L}_{\text{GAN}}(G_{\mathcal{Y}\to\mathcal{X}}, D_{\mathcal{X}}, \mathcal{Y}, \mathcal{X}). \tag{3}$$

Here, $G_{\mathcal{X}\to\mathcal{Y}}$ is the mapping from synthetic images to real images, and $D_{\mathcal{Y}}$ is the discriminator that classifies the real images and the outputs of $G_{\mathcal{X}\to\mathcal{Y}}(\cdot)$; Similarly, $G_{\mathcal{Y}\to\mathcal{X}}$ and $D_{\mathcal{X}}$ are defined for the opposite mapping. Further, the CycleGAN framework has an additional loss term, called the cycle-consistency loss:

$$\mathcal{L}_{\text{cyc}}(G_{\mathcal{X}\to\mathcal{Y}}, G_{\mathcal{Y}\to\mathcal{X}}, \mathcal{X}, \mathcal{Y}) := \mathbb{E}_{x\sim p_{\mathcal{X}}(x)}[\|G_{\mathcal{Y}\to X}(G_{\mathcal{X}\to\mathcal{Y}}(x)) - x\|] \tag{4}$$

$$+ \mathbb{E}_{y\sim p_{\mathcal{Y}}(y)}[\|G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y)) - y\|], \tag{5}$$

where $\|\cdot\|$ can be an arbitrary norm, e.g., ℓ_1 norm. This loss essentially imposes a restriction on the forward/backward mappings so that $G_{\mathcal{Y}\to\mathcal{X}}(G_{\mathcal{X}\to\mathcal{Y}}(x)) \simeq x$, and similarly, $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y)) \simeq y$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Combining these terms and omitting the function arguments for simplicity, the overall loss function is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{GAN, forward}} + \mathcal{L}_{\text{GAN, backward}} + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}}, \tag{6}$$

where λ_{cyc} is the regularization parameter. Then, one can apply the standard training methodology of GANs to find G's that minimize the loss function and D's that maximize it.

While the CycleGAN framework has been shown useful in practice, it still has a room for improvements. Indeed, it is not very clear how the cycle-consistency loss helps preserve labels while translating data. See Fig. 3 for a toy example where the goal is to learn an image-to-image translation rule between digit images in Arial font ($\mathcal{X}$) and those in Script font ($\mathcal{Y}$), where the label is the

[1] In our experiments with the UnityEye simulator, we simply employ the moment matching algorithm since the simulator only allows us to specify the first and second moments of the label distribution. However, in general, one may choose an arbitrary metric and employ some optimization algorithm for finding a new label distribution.

number present in the image. Consider the $G_{\mathcal{X}\to\mathcal{Y}}$ and $G_{\mathcal{Y}\to\mathcal{X}}$ shown in the figure. One can easily observe that these mappings result in perfect cycle-consistency, i.e., $\mathcal{L}_{\text{cyc}} = 0$. Further, assuming that $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$ are perfect, $\mathcal{L}_{\text{GAN, forward}} = \mathcal{L}_{\text{GAN, backward}} = 0$. Thus, the labels are not preserved at all even though the zero loss is achieved.

In general, if the number of samples in each domain is n, there are at least $n!$ pairs of mappings that attains zero loss without perfect cycle-consistency. Indeed, while researchers have shown that the cycle-consistency loss helps maintain the label information in a variety of scenarios (Zhu et al., 2017; Kim et al., 2017), in our own experiments, we observe that it easily fails to maintain label information unless a significant amount of effort is put to fine-tune the training methodology. In Sec. B, we show some other cases in which the cycle-consistency only is not sufficient to preserve label information.

In this work, to better preserve labels, we add an additional loss term, which we call *the feature-consistency loss*. The feature-consistency loss term is originally proposed for learning a style-transfer function (Shrivastava et al., 2017). The key idea is simple: Given a feature extractor $F(\cdot)$, one simply regularizes the difference between the features of input images and those of output images. In the previous example, assume that we are given with a perfect digit recognizer F. With the F-consistency loss term added to the objective function, the mappings shown in Fig. 3 will incur non-zero loss while the correct bidirectional mapping, which maintains digits within images, will incur zero loss.

Leveraging this idea, we propose to use the bidirectional feature-consistency loss, defined as

$$\mathcal{L}_{\text{feature}}(F, G_{\mathcal{X}\to\mathcal{Y}}, G_{\mathcal{Y}\to\mathcal{X}}, \mathcal{X}, \mathcal{Y}) := \mathbb{E}_{x\sim p_{\mathcal{X}}(x)}[\|F(G_{\mathcal{X}\to\mathcal{Y}}(x)) - F(x)\|] \tag{7}$$

$$+ \mathbb{E}_{y\sim p_{\mathcal{Y}}(y)}[\|F(G_{\mathcal{Y}\to\mathcal{X}}(y)) - F(y)\|], \tag{8}$$

where $\|\cdot\|$ is an appropriate norm of the feature space. That is, we simply add the feature-consistency loss of the forward mapping and that of the backward mapping.

In many practical applications, two domains share the high-level structure and differ in detailed expressions, and hence one may set the identity mapping as the feature extractor. (Note that the example shown in Fig. 3 is not the case.) In such cases, the feature-consistency loss term reduces to the norm of the difference between the input and output images.

Incorporating the bidirectional feature-consistency term, we have the following total loss function:

$$\mathcal{L} = \mathcal{L}_{\text{GAN, forward}} + \mathcal{L}_{\text{GAN, backward}} + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}} + \lambda_{\text{feature}}\mathcal{L}_{\text{feature}}. \tag{9}$$

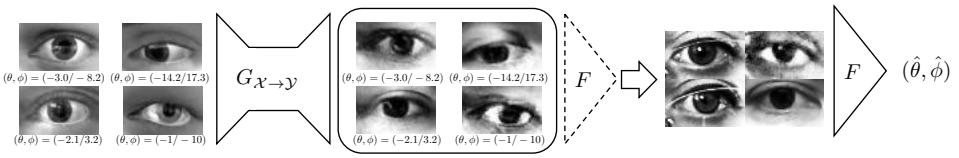
Then, using the typical training methods for GANs, we obtain the bidirectional mapping G's.

2.3 Prediction with backward translation

After the translation mappings are obtained, there are two possible approaches that one can take. The first approach, proposed by Shrivastava et al. (2017), is to translate all the labeled synthetic data using the forward mapping $G_{\mathcal{X}\to\mathcal{Y}}$. The authors of (Shrivastava et al., 2017) show that a model trained on this real-like (labeled) dataset significantly outperforms the previous state of the arts. See Fig. 4a for visual illustration.

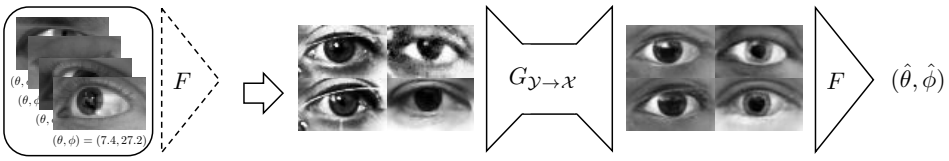
The second approach, which we propose in this work, relies on the backward translation, i.e., $G_{\mathcal{Y}\to\mathcal{X}}$. As described in the previous section, leveraging simulators, one can generate an arbitrarily large amount of synthetic data, and then train an efficient model on it. Hence, assuming perfect backward translation, one can first translate real data to synthetic data, and then apply the model (trained on synthetic data) to the mapped data. See Fig. 4b for visual illustration.

Indeed, in most practically relevant cases, the amount of available synthetic data is much larger than that of real data, and hence one can easily train a highly accurate predictor at least in the synthetic domain (but not necessarily in the real domain). Another advantage of this approach is that one does not have to train a new model when a new real dataset arrives, while the first approach needs to retrain the prediction model for each real dataset. We remark that this approach requires an additional computational cost for backward translation during prediction. However, the additional computational cost is usually not prohibitive compared to the complexity of the prediction model.



(1) Training a predictor using the forward translated images (2) Predict using the original test images

(a) The training/test methodology proposed in (Shrivastava et al., 2017).



(1) Training a predictor (2) Predict using the backward translated images

(b) Our training/test methodology.

Figure 4: Comparison of our methodology with that of (Shrivastava et al., 2017).

3 EXPERIMENTS

3.1 CROSS-DOMAIN APPEARANCE-BASED GAZE ESTIMATION

In this section, we apply our methodology to tackle the cross-domain appearance-based gaze estimation problem, and evaluate its performance on the the MPIIGaze dataset (Zhang et al., 2015). The goal of this problem is to estimate the gaze vector given an image of a human eye region using the data collected from a different domain.

To generate a synthetic dataset of labeled human eye images, we employ *UnityEyes*, a high-resolution 3D simulator that renders realistic human eye regions (Wood et al., 2016). For each image, UnityEyes draws the pitch and yaw of the eyeball and camera, uniformly at random, and then renders the corresponding eye region image. More specifically, the random distributions are specified by 8 input parameters $(\theta_p, \theta_y, \phi_p, \phi_y, \delta\theta_p, \delta\theta_y, \delta\phi_p, \delta\phi_y)$: The first two are the expected values of eyeball pitch and yaw, and the following two are the expected values of camera pitch and yaw; And the other four are the half-widths of the distributions. As a default setting, UnityEyes uses the following distribution parameters: $(\theta_p, \theta_y, \phi_p, \phi_y, \delta\theta_p, \delta\theta_y, \delta\phi_p, \delta\phi_y) = (0^\circ, 0^\circ, 0^\circ, 0^\circ, 30^\circ, 30^\circ, 20^\circ, 40^\circ)$. More details on how UnityEyes render eye regions can be found in (Wood et al., 2016).

We first evaluate the performance of our data generation algorithm based on pseudo-labels. Using the 380k images generated with UnityEyes with the default parameters, we train a simple gaze estimation network, with which we annotate the real dataset with pseudo-labels. We then obtain the horizontal and vertical statistics of the gaze vectors, and accordingly adjust the UnityEyes parameters to make the means and variances coincide. Since there are many free parameters, we reduce the number of free parameters by considering the following specific classes of distributions: $\theta_p = \phi_p, \theta_y = \phi_y, \delta\theta_p = \delta\phi_p, \delta\phi_p = \delta\phi_y$. We adapt the data distribution for 4 times. We run the data adaption algorithm for 4 iterations, and report the Hellinger distances between the synthetic label distribution of each iteration and the label distribution of MPIIGaze dataset in Table 1.

Table 1: Experimental results of the adaptive data generation algorithm. Here, ℓ denotes the number of iterations. The first two rows are Hellinger distances between the synthetic label (Θ, Φ) and the MPIIGaze label (Θ^0, Φ^0) distributions, and the last row is the average angle error of the prediction model trained on the corresponding synthetic data.

	$\ell = 0$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
$H(P_\Theta, P_{\Theta^0})$	0.641	0.400	**0.163**	0.231	0.813
$H(P_\Phi, P_{\Phi^0})$	0.586	0.231	**0.153**	0.344	0.764
Mean angle error ($^\circ$)	12.34	9.14	8.57	**8.40**	9.20

Table 2: Comparison of the state of the arts on the MPIIGaze dataset. The error is the mean angle error in degrees. The parameter α denotes the learning rate used for training the predictor. In the second and third columns, 'R' denotes 'Real', 'S' denotes 'Synthetic', 'RS' denotes 'Refined Synthetic', and 'RR' denotes 'Refined Real'. Our approach with some hyperparameters achieves the state-of-the-art performance.

Method	**Trained on**	**Tested with**	**Error**
Support Vector Regression (SVR) (Schneider et al., 2014)	R	R	16.5
Adaptive Linear Regression (ALR) (Lu et al., 2014)	R	R	16.4
kNN w/ UT Multiview (Zhang et al., 2015)	R	R	16.2
Random Forest (RF) (Sugano et al., 2014a)	R	R	15.4
CNN w/ UT Multiview (Zhang et al., 2015)	R	R	13.9
CNN w/ UnityEyes (Shrivastava et al., 2017)	S	R	11.2
kNN w/ UnityEyes (Wood et al., 2016)	S	R	9.9
CNN w/ UnityEyes (Shrivastava et al., 2017)	RS	R	7.8
Ours ($\ell = 2, \lambda_{\text{feature}} = 0, \alpha = 10^{-6}$)	S	RR	7.73
Ours ($\ell = 2, \lambda_{\text{feature}} = 0.5, \alpha = 10^{-6}$)	S	RR	**7.60**

Specifically, we represent the unit gaze vector by a pair of pitch and yaw, i.e., (θ, ϕ), and then measure the Hellinger distance for each coordinate.[2] We observe that after the second iteration of the algorithm, the Hellinger distributions are minimized. Note that in this experiment, our algorithm attempts at matching the first two moments of the distributions, which does not necessarily reduce the Hellinger distance to the target distribution. This may explain why the Hellinger distance increases after the second iteration in our experiment. In appendix, we show how the synthetic label distribution evolves as the algorithm proceeds.

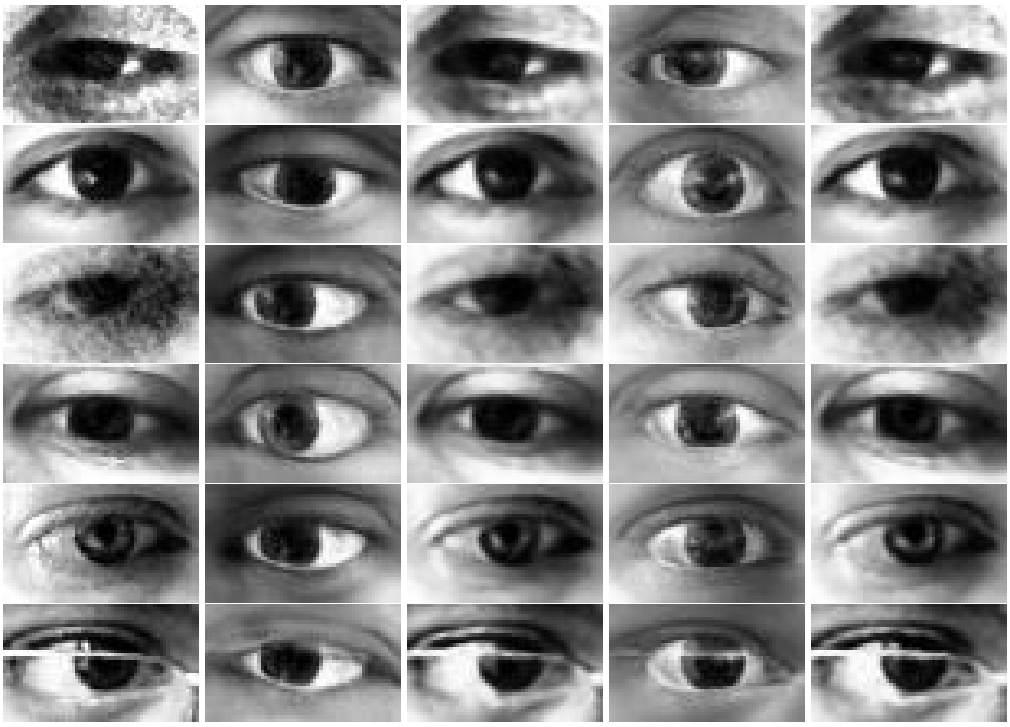
In the rest of this section, we report the results based on the output of the second iteration. (In practice, one typically has access to a validation dataset, which can help decide the right number of iterations.) Using the newly generated synthetic dataset together with the unlabeled real dataset, we then train a cyclic image-to-image translation mapping with the new loss function (9). We use the ℓ_1 norm for the feature-consistency term. For the regularization parameters, we test $\lambda_{\text{cyc}} = 10$ and $\lambda_{\text{feature}} \in \{0, 0.5\}$.

Shown in Fig. 5 are some examples of the translation results. Here, we compare the CycleGAN trained only with the cycle-consistency loss and that trained with both the cycle-consistency loss and the feature-consistency loss. The first column shows the input real images, the next two columns are the translated images using the output of the former approach, and the last two columns are the outputs of the latter approach. It is clear that the the learned mappings satisfy near-perfect cycle-consistency by comparing y's with $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$'s. However, the gaze vector is not preserved when real images are translated to synthetic images. On the other hand, when both consistencies are enforced, the gaze vector does not alter, maintaining the label information, showing the necessity of the feature-consistency loss term.

For each configuration of the parameters and the dataset, we measure the test accuracy by predicting gaze vectors with backward-translated images. That is, $F(G_{Y\to X}(\cdot))$ is used to estimate the gaze vector given a real image.

Summarized in Table 2 are the experimental results along with the state of the arts reported in the literature. Note that out approach outperforms the state-of-the-art performance of Shrivastava et al. (2017). Observe that our algorithm is trained with 380k images while the results of Shrivastava et al. (2017) is trained with 1200k synthetic images, demonstrating the efficacy of our adaptive data generation algorithm.

[2]The Hellinger distance between two probability distributions $P = (p_1, p_2, \cdots, p_k)$ and $Q = (q_1, q_2, \cdots, q_k)$ is defined as $H(P, Q) := \frac{1}{2}\sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2}$.



(a) (Input images) y (b) (Cycle only) $G_{\mathcal{Y}\to\mathcal{X}}(y)$ (c) (Cycle only) $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$ (d) (Cycle&feature) $G_{\mathcal{Y}\to\mathcal{X}}(y)$ (e) (Cycle&feature) $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$

Figure 5: Some examples of the translation results (real $\to$ synthetic $\to$ real). The first column is the original real image, say y. The second and third columns are $G_{\mathcal{Y}\to\mathcal{X}}(y)$ and $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$, where G's are trained with the cyclic consistency term. The last two columns are $G_{\mathcal{Y}\to\mathcal{X}}(y)$ and $G_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{Y}\to\mathcal{X}}(y))$, where G's are trained with both the cyclic consistency term and the feature consistency term. Both approaches achieve nearly perfect cycle-consistency; However, only the latter approach maintains the structure of the images when translated to the other domain.

3.2 Implementation details

For our experiments, we slightly modify the CycleGAN framework of Zhu et al. (2017). The generator network, G, is modified to take input image of 36×60, and the rest of the G is identical to that of the original CycleGAN architecture with 6 blocks. The discriminator network, D is identical to that of the original CycleGAN framework. For detailed information, we refer the readers to (Zhu et al., 2017). The CycleGAN was trained with batch size of 64 and learning of 2×10^{-4}.

The eye gaze prediction network is designed based on the architecture proposed in (Shrivastava et al., 2017). The input is a 36×60 gray-scale image that is passed through 5 convolutional layers followed by 3 fully connected layers, the last one encoding the 3-dimensional (normalized) gaze vector: (1) Conv3x3, 32 feature maps, (2) Conv3x3, 32 feature maps, (3) Conv3x3, 64 feature maps, (4) MaxPool3x3, stride $= 2$, (5) Conv3x3, 80 feature maps, (6) Conv3x3, 192 feature maps, (7) MaxPool2x2, stride $= 2$, (8) FC9600, (9) FC1000, (10) FC3, (11) ℓ_2 normalization (12) ℓ_2 loss. The predictor network is trained with batches of size 512, until the validation error converges.

4 Conclusion

In this work, we propose a new S+U learning algorithm, which fully exploits the flexibility of simulators and the recent advances in learning bidirectional mappings, and show that it outperforms the state-of-the-art performance on the gaze estimation task.

We conclude the paper by enumerating a few open problems. In our experiments, we arbitrarily choose the feature for the consistency term. We, however, observe that the choice of feature mapping

significantly affects the performance of the algorithm. Thus, one needs to study how the optimal feature mapping can be designed for different tasks.

In this work, we have separated learning of cross-domain translation mappings from learning of a prediction algorithm. One interesting open question is whether one can jointly train these two components and potentially outperform the current separation-based approach.

Another interesting direction is building a differentiable data generator (Graves et al., 2014; Gaunt et al., 2017; Feser et al., 2016). It is well known that neural networks with external memory resources is a differentiable Turing Machine or differentiable Von Neumann architecture (Graves et al., 2014). Further, researchers have proposed the use of differentiable programming language with neural networks Gaunt et al. (2017); Feser et al. (2016). Indeed, the adaptive data generation algorithm proposed in this work can be viewed as an extremely limited way of adjusting the way synthetic data is generated. If the data generator can be written in a differentiable language, one could possibly jointly optimize the synthetic data generator together with the other components such as translation networks and prediction networks, potentially achieving an improved performance.

REFERENCES

John K. Feser, Marc Brockschmidt, Alexander L. Gaunt, and Daniel Tarlow. Neural functional programming. *CoRR*, abs/1611.01988, 2016.

Alexander L. Gaunt, Marc Brockschmidt, Nate Kushman, and Daniel Tarlow. Differentiable programs with neural libraries. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1213–1222, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3354–3361. IEEE, 2012.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pp. 601–608, 2007.

Gyuri Im. Europilot: A toolkit for controlling Euro Truck Simulator 2 with python to develop self-driving algorithms. `https://github.com/marshq/europilot`, 2017. Accessed: 2017-10-27.

Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:1610.01983*, 2016.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1857–1865, 2017.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

Kangwook Lee, Hoon Kim, and Changho Suh. Crash to not crash: Playing video games to predict vehicle collisions. In *ICML Workshop on Machine Learning for Autonomous Vehicles*, 2017. URL https://openreview.net/forum?id=r1GXtBEf-&noteId=r1GXtBEf-.

Feng Lu, Yusuke Sugano, Takahiro Okabe, and Yoichi Sato. Adaptive linear regression for appearance-based gaze estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10):20332046, 2014.

Microsoft. AirSim: Open source simulator based on Unreal Engine for autonomous vehicles from Microsoft AI & Research. https://github.com/Microsoft/AirSim, 2017. Accessed: 2017-10-27.

Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pp. 102–118. Springer, 2016.

Timo Schneider, Boris Schauerte, and Rainer Stiefelhagen. Manifold alignment for person independent appearance-based gaze estimation. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 11671172. IEEE, 2014.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. Learning-by-synthesis for appearance-based 3d gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 18211828, 2014a.

Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. Learning-by-synthesis for appearance-based 3d gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1821–1828, 2014b.

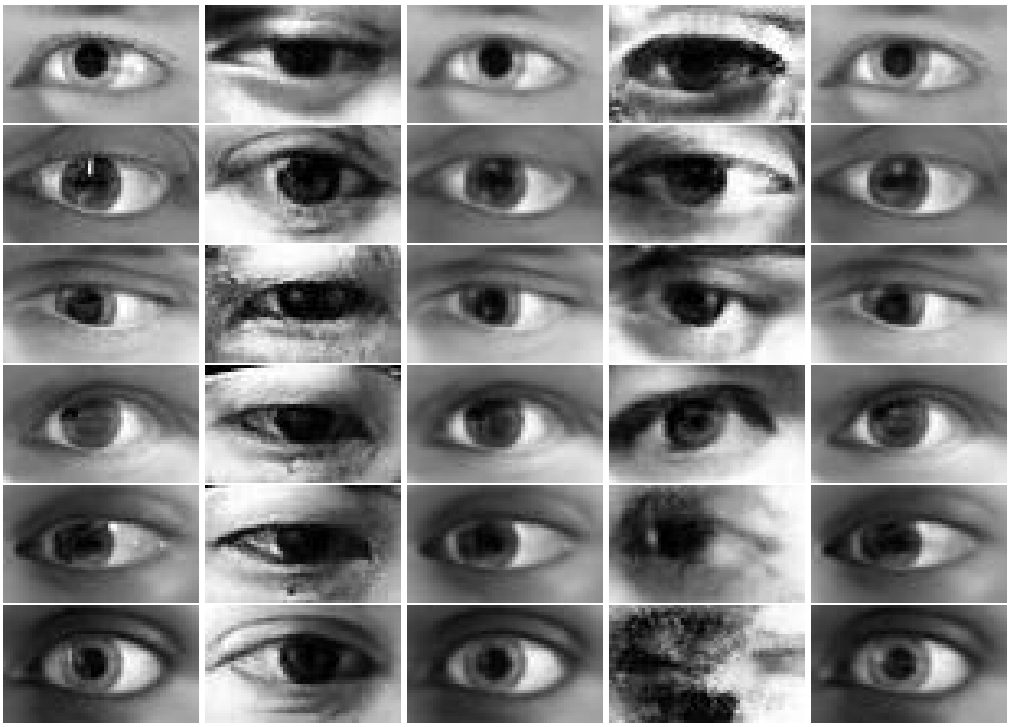
Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, pp. 131–138, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4125-7. doi: 10.1145/2857491.2857492.

Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Appearance-based gaze estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 45114520, 2015.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

A ADDITIONAL EXPERIMENTAL RESULTS ON EYE GAZE ESTIMATION

In this section, we provide additional qualitative results on the eye gaze estimation task. See Fig. 6.



(a) (Input images) x (b) (Cycle only) $G_{\mathcal{X}\to\mathcal{Y}}(x)$ (c) (Cycle only) $G_{\mathcal{Y}\to\mathcal{X}}(G_{\mathcal{X}\to\mathcal{Y}}(x))$ (d) (Cycle&feature) $G_{\mathcal{X}\to\mathcal{Y}}(x)$ (e) (Cycle&feature) $G_{\mathcal{Y}\to\mathcal{X}}(G_{\mathcal{X}\to\mathcal{Y}}(x))$

Figure 6: Some examples of the translation results (synthetic $\to$ real $\to$ synthetic). The first column is the original synthetic image, say x. The second and third columns are $G_{\mathcal{X}\to\mathcal{Y}}(x)$ and $G_{\mathcal{Y}\to\mathcal{X}}(G_{\mathcal{X}\to\mathcal{Y}}(x))$, where G's are trained with the cyclic consistency term. The last two columns are $G_{\mathcal{X}\to\mathcal{Y}}(x)$ and $G_{\mathcal{Y}\to\mathcal{X}}(G_{\mathcal{X}\to\mathcal{Y}}(x))$, where G's are trained with both the cyclic consistency term and the feature consistency term.

B ADDITIONAL EXPERIMENTAL RESULTS ON GTA DATA

We learn a bidirectional mapping between the driving images collected from the video game GTA V and the KITTI driving dataset (Geiger et al., 2012). Here, we use the last fully connected layer of `resnet18` as the feature mapping (He et al., 2016). Shown in Fig. 7 and Fig. 8 are some qualitative mapping results, proving the efficacy of the feature-consistency loss.

Figure 7: Some examples of the translation results (real $\rightarrow$ synthetic $\rightarrow$ real). The first column is the original real image, say y. The second and third columns are $G_{\mathcal{Y}\rightarrow\mathcal{X}}(y)$ and $G_{\mathcal{X}\rightarrow\mathcal{Y}}(G_{\mathcal{Y}\rightarrow\mathcal{X}}(y))$, where G's are trained with the cyclic consistency term. The last two columns are $G_{\mathcal{Y}\rightarrow\mathcal{X}}(y)$ and $G_{\mathcal{X}\rightarrow\mathcal{Y}}(G_{\mathcal{Y}\rightarrow\mathcal{X}}(y))$, where G's are trained with both the cyclic consistency term and the feature consistency term. Both approaches achieve nearly perfect cycle-consistency; However, only the latter approach maintains the structure of the images when translated to the other domain.

Figure 8: Some examples of the translation results (synthetic $\rightarrow$ real $\rightarrow$ synthetic). The first column is the original synthetic image, say x. The second and third columns are $G_{\mathcal{X}\rightarrow\mathcal{Y}}(x)$ and $G_{\mathcal{Y}\rightarrow\mathcal{X}}(G_{\mathcal{X}\rightarrow\mathcal{Y}}(x))$, where G's are trained with the cyclic consistency term. The last two columns are $G_{\mathcal{X}\rightarrow\mathcal{Y}}(x)$ and $G_{\mathcal{Y}\rightarrow\mathcal{X}}(G_{\mathcal{X}\rightarrow\mathcal{Y}}(x))$, where G's are trained with both the cyclic consistency term and the feature consistency term.