Image Domain Adaptation with Cyclic Generative Adversarial Networks

Yusuf Dalva, Ayhan Okuyan
Computer Engineering
Bilkent University
Ankara, Turkey
{name}.{surname}@bilkent.edu.tr

I. Introduction

Image to image translation has been an interesting and challenging task where no perfect solution still exists for various domains. In recent studies, Generative Adversarial Networks (GANs) are applied to this task [1]. GANs are known for their success in generating samples that are similar to a given data distribution, where generated samples cannot be so easily distinguished from the images that originally belong to the given distribution. In this study, we applied a popular method named CycleGAN [2] to perform image to image translations with different variations of GANs. Our task was to learn a two-way translation from unpaired data that is able to convert images from real-life domain to cartoon domain and visa versa. This can be considered as domain adaptation as we assume that the cartoon images and real images are sampled from different distributions. The details of the GAN variations applied and evaluation metrics together with the results are given in this report.

II. Dataset

For this project, we have used an unpaired dataset of face and scenery real/cartoon images [1]. Furthermore, the selected learning objective was to transform face images, and we only used them and disregarded the scenery images. The face images contained two separate sources for images, and we have used the images provided by PA Works, which corresponded to 5000 images. For the cartoon face images, we have selected to adopt the Shinkai style, which contains face images framed from the works of Makoto Shinkai, also corresponding to 5000 images. Each image in the dataset is a 3-channel RGB image with a resolution of 256x256.

As we are training, we have noticed that the training became intractable in terms of time, the train of one network corresponded to 155 hours ($\approx$ 6 days) on GPU. Hence, we have further narrowed down each training dataset to 1000 randomly selected images and used another 1000 randomly selected images for test purposes.

III. Methods

Since Generative Adversarial Networks are widely applied for sample generation, image domain translation was considered as a natural use case of GANs. Here in our task, we wanted to learn a translation that can successfully translate real-world images to cartoon-like images. Here we considered different architectures that are proposed for image translation tasks and adopted CycleGAN as our learning method [2].

[1]Dataset taken from https://github.com/zhen8838/AnimeStylized

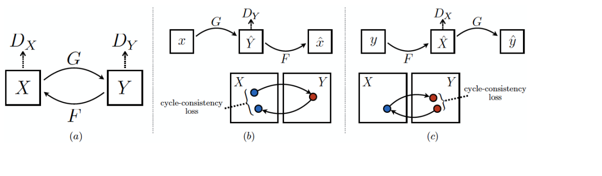

Fig. 1. Visual illustration of cyclic translations and cycle-consistency loss [2]. Showing two way translations with domains X and Y (a) and Visual interpretation of cycle-consistency loss for translations $X \rightarrow Y \rightarrow X$ (b) and $Y \rightarrow X \rightarrow Y$ (c)

A. Using CycleGAN Architecture for image translation

Considering the problem of learning a domain translation subjecting different image domains, the main consideration is the loss of information during the translation process. To make the loss function consider this criterion, CycleGAN architecture is proposed in [2]. This work introduces an additional loss term that is called cycle-consistency loss, which measures the difference between the original image and the translated image using two-way transformations. This framework aims to learn a two-way transformation which consists of the mapping $A \rightarrow B$ and $B \rightarrow A$, where B and A are the domains that are translated to one another. This cycle-consistency loss component is given as follows where G performs translation $A \rightarrow B$ and H performs translation $B \rightarrow A$.

$$L_{CYCLE} = E_{x \sim p_A(x)}[||H(G(x)) - x||_1] + E_{y \sim p_B(y)}[||G(H(y)) - y||_1] \tag{1}$$

For the equation given for L_{CYCLE}, the notation p_A shows the probability distribution for domain A where x is sampled and p_B shows the probability distribution of domain B where y is sampled for calculating the loss value. The scheme illustrating the cycle-consistency loss taken from the original CycleGAN paper is given in Fig. 1. With the effect of cycle consistency which is established by the loss term L_{CYCLE}, the image features are needed to be preserved to perform a successful cyclic translation between domains.
As another contribution that comes from CycleGAN to our study, another loss term has been introduced to preserve pixel intensities as much as possible while performing the desired

translation. To achieve this, identity loss has been introduced. The formulation of this loss component is formulated as follows.

$$L_{IDENTITY} = E_{x \sim p_A(x)}[||G(x) - x||_1] + E_{y \sim p_B(y)}[||H(y) - y||_1] \tag{2}$$

In the formulation of $L_{IDENTITY}$ the same formulation principles are followed with L_{CYCLE}. By applying an L1 distance between the generated sample and the inputted sample to the generator the pixel-wise differences are tried to be stabilized while changing the distribution that generator represents.

Overall, to apply the CycleGAN framework towards our aim, which is the task of learning the two-way translation between cartoon images and real-world images, the initial step was introducing two generator-discriminator pairs as two GANs bounded via a common loss function. For the translation $A \rightarrow B$ the generator-discriminator pair is shown as $G(x) - D_1(y)$ which is $H(x) - D_2(y)$ for the translation $B \rightarrow A$. For the common loss function for these two pairs, the terms $L_{IDENTITY}$ and L_{CYCLE} in addition to adversarial loss introduced by GAN framework [1]. The adversarial loss term is determined by the GAN variant used and explained in detail in the following section.

B. GAN Variations Used

As mentioned by several studies, the main concern regarding successful GAN training is maintaining stable and continuous learning. In the literature, there are different approaches that modify the loss function to achieve the trainability and stability in training GANs [3], [4], **wggan**. To find a suitable model for our task, which is learning the image domain translation between real-world images and cartoon-like images, three different approaches are followed to form the adversarial loss term (L_{ADV}) in the loss function used to train the model with CycleGAN framework.

*1) **Vanilla GAN**:* As stated by the original GAN paper [1], GANs use an adversarial loss term to learn a generator and a discriminator simultaneously. In literature this initial model is mentioned as vanilla GAN, stating that it is the purest form of generative adversarial networks. This architecture adopts binary cross-entropy as a loss function to distinguish between the images from the original data distribution and the generated images. The adversarial loss component mentioned in [1] is formulated as follows.

$$L_{ADV} = E_{x \sim p_{real}(x)}[log(D(x))] + E_{z \sim p_{gen}(z)}[log(1 - D(G(z)))] \tag{3}$$

In the formulation above the probability distributions p_{real} and p_{gen} show the distributions of images from the real world for the target domain and images from the initial domain. Here an image sample x is sampled from p_{real} whereas z is sampled from p_{gen}. The implicit function D shows the discriminator learned, whereas G shows the generator learned. This form of the adversarial loss is considered as the most naive approach that adapts the idea from performing binary classification and adapts it to generative domain.

*2) **Least Squares GAN (LSGAN)**:* Despite the success of GANs as a form of a generative model, models with complex properties cannot be trained with standard Vanilla GAN adversarial loss. As mentioned by [3], [5] [4], vanishing gradients is a major problem encountered while training GANs with the adversarial loss given in the original GAN paper. The main reason of the problem is the nature of the binary cross-entropy (BCE) loss used in vanilla GAN. As mentioned in [4], BCE loss tends to give very small results as the predicted values for the discriminator output tend to be larger. Considering this fact, the distinction made by the adversarial loss between the generated images that are close to the decision boundary and the ones that only lie on the correct side of the boundary is not that accurate. This fact leads to vanishing gradients and unsuccessful learning at the end. In order to be able to make the mentioned distinction more accurately, the least-squares error metric is adapted to the GAN framework in [4]. The formulation of the adversarial loss component modeling the LSGAN model is given below. Here the adversarial loss component for the generator and the discriminator are separated, but still dependent on each other to maintain adversarial feedback. Loss component for the generator is shown as $L_{ADV,G}$ and for discriminator, it is shown as $L_{ADV,D}$.

$$L_{ADV,D} = \frac{1}{2}E_{x \sim p_{real}(x)}[(D(x) - 1)^2] + \frac{1}{2}E_{z \sim p_{gen}(z)}[(D(G(z)))^2] \tag{4}$$

$$L_{ADV,G} = \frac{1}{2}E_{z \sim p_{gen}(z)}[(D(G(z))^2] \tag{5}$$

Here the objective of both generator and the discriminator are minimizing the corresponding loss function where generator maximizes and discriminator minimizes a common loss function in the original GAN paper. For simplicity the adversarial loss component will be notified as L_{ADV} in the general formulation. As it can be seen from the loss component, the main aim is to magnify the distances prediction error values so that vanishing gradients problem will less likely to occur.

*3) **Wasserstein GAN**:* As the third alternative for the adversarial loss component, Wasserstein GAN is the final alternative considered [5]. Here different than the previous two approaches, the aim is establishing stable learning where the changes in the loss values are steady. In order to establish the desired stability, the Wasserstein GAN method uses a critic function instead of a discriminator component. The main role of the critic is also to assess whether the generated sample is good enough to consider it as real. However, the main difference is in the form of the output of the criticizing function. Here the critic is also learned by a neural network but outputs a scalar value as a critic value. The main criterion that makes Wasserstein GAN unique is the fact that the critic function learned should be 1-Lipschitz continuous. Lipschitz continuity determines how fast the outputs of the critic function changes, which is the fundamental requirement for the stability of the gradients. The K-Lipschitz continuity rule for critic function f is determined by the rule below.

$$\frac{|f(x_1) - f(x_2)|}{|x_1 - x_2|} \leq K \tag{6}$$

In order to apply the constraint of 1-Lipschitz continuity, a gradient penalty is performed to enforce the rule $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$. This penalty is performed in the gradients, so that the magnitude of the gradients is enforced to be ≤ 1. The loss term added by the gradient penalty for this model is as follows. Here f denotes the critic function.

$$L_{PEN} = ||\nabla_w f(x) - 1||_1 \tag{7}$$

After defining the penalty term, the overall adversarial loss is defined as follows.

$$L_{ADV,D} = \frac{1}{m}\sum_{i=1}^{m}[f(x^{(i)}) - f(G(z^{(i)}))] + L_{PEN} \tag{8}$$

$$L_{ADV,G} = \frac{1}{m}\sum_{i=1}^{m}[f(G(z^{(i)}))] \tag{9}$$

In the formulation given in the equations above, the function f denotes the critic function whereas the implicit function denotes the generator. Just like the loss functions for the LSGAN model, the adversarial loss values for the critic and the discriminator are generalized as L_{ADV}.
By enforcing the 1-Lipschitz continuity on the critic, stability in GAN training is aimed. This step is taken against a possible vanishing gradient problem in training

C. Trained Model

In this study all of the three forms of GAN (Vanilla GAN, LSGAN and Wasserstein GAN) are attempted to find the best adversarial component for our dataset. As formulated in the previous section, the GAN variant type only effects the adversarial loss component used in the overall loss function. As a generalization the overall loss function can be generalized as follows.

$$L = L_{ADV} + L_{CYCLE} + L_{IDENTITY} \tag{10}$$

With the given GAN variants, this study aimed to study the impacts that these variants make in training. The comparison of these variants are given in the section titled as *Experiments & Results*.

D. Network Architectures

We have used a standard CycleGAN [2] structure in all of our experiments, with two discriminators, namely D_A and D_B, and two generators: G_A and G_B. This section further explains the structures of these networks.

1) Generator: The generator chosen is the `resnet-9blocks`, which contains an initial convolutional layer with a kernel size of 7 and a ReLU activation, two downsampling layers with `Conv-Instance Norm-ReLU`, 9 layers of ResNet blocks [6], and two upsampling layers, with `transpose convolution` instead of convolution in the downsampling layers. This enables the generator to embed the transform between the two domains in the weights of the deep residual network in the middle of the architecture. The ResNet blocks are inherently defined as two 3x3 convolutional blocks the activation layer, and a skip connection that adds the input to the output. We preferred this architecture to the U-Net [7], which is also widely used, due to the relatively faster training process.

2) Discriminator: For the discriminator architecture, we have used a fairly simple fully convolutional network. The architecture is defined as 5 layers of `Conv-Instance Norm-Leaky ReLU`, without the instance normalization only on the input and output layers. The α parameter of the leaky ReLU activation is always chosen to be 0.2. The maximum depth of the feature maps for both the generator and discriminator is chosen as 64 and each convolutional layer adapts to this depth depending on the layer.

IV. Experiments & Results

A. Evaluating Models

As one of the biggest problems of both unsupervised learning and generative models, there is no definitive metric that truly assesses the performance of the model like accuracy that directly evaluates the prediction made. To be able to assess the trained model statistically, we adapted Fréchet Inception Distance (FID) from [8].
This method mainly uses the idea from the original Inception Score [9], which uses InceptionV3 network [10] for feature extraction in the obtained samples and apply an entropy calculation to find the consistency of the model. Since this score does not include any comparison between real-world samples and the generated samples, we adopted FID instead of the inception score.
The formulation of FID metric is as follows.

$$\begin{aligned} d_{FID}(D_{real}, D_{gen}) = ||m_{real} - m_{gen}||_2^2 + \\ Tr(C_{real} + C_{gen} - 2(C_{real}C_{gen})^{\frac{1}{2}}) \end{aligned} \tag{11}$$

In the formulation given above, D_{real} and D_{gen} correspond to data distributions of real world images and the generated images for the target domain. The min principle of this score is to measure the similarity of the features of the images generated and the original images. In order to perform the feature extraction, the InceptionV3 network trained for ImageNet dataset has been used (for general purpose features of the images). To extract the features, the general practice is followed and the layer named `pool_3` is used. The terms m and C denote mean and covariance respectively, whereas the function Tr correspond to trace operation in linear algebra (sum of elements in the main diagonal).
At the evaluation step, the better model is expected to have

lower FID score. This is because if the generated cartoon images have similar features with cartoons found in the real-world, the FID metric will have a low value.

B. Training Details

To make a reliable experiment, the hyperparameters are kept constant between trains. For optimization of the models, we have used the Adam optimizer [11] with the parameters $\beta_1 = 0.5$ and a learning rate of 0.0002 for the 100 epochs. Then for 100 epochs, the learning rate is decayed down to 0 linearly, which makes a total of 200 epochs in training with a batch size of 2. The algorithm chooses the fake images from a pool generate with the last 50 images generated from the generator to train the discriminator. In all of the models, λ_A, λ_B, and $\lambda_{IDENTITY}$ is selected as $10, 10$ and 0.5 respectively, following the original paper [2]. All three models are run parallel on three GPUs namely two NVIDIA-RTX 2080 and one NVIDIA-GTX 1070 Max-Q, and all models took approximately 24 hours to train and 30 minutes to infer.

C. Results Obtained

We trained three different networks in parallel to find the cyclic network with the most suitable GAN variant. Here, one of the networks use vanilla GAN, one uses LSGAN and the last network uses WGAN (Wasserstein GAN) as the GAN variant. In training, no convergence in Generator and Discriminator is observed, which means that they continue to learn from each other. If the other case would have observed, it would mean that training stops which is not a desirable case for training GANs. Our main expectation for the success of the learning was to see the decreasing trend in the cycle-consistency loss (L_{CYCLE}) and identity loss ($L_{IDENTITY}$). This trend is observed for networks using Vanilla GAN and LSGAN but not for WGAN.
As mentioned, the success of the training process is evaluated by the decrease in loss components L_{CYCLE} and $L_{IDENTITY}$. For the cycle-consistency loss, as training proceeds, it is expected that the cyclic translation would produce more accurate results considering that the aim is to learn a two-way translation. In terms of identity loss, another desired outcome of the translation is to preserve the pixel-wise characteristics of the original image as training proceeds. Considering this fact, the decreasing value of identity-loss shows another aspect of successful learning. The loss plots for L_{CYCLE} and $L_{IDENTITY}$ are given in Fig. 2 and Fig. 3 respectively. Since we considered the attempt using WGAN as a failure, the loss plot of this variant is not provided.
To evaluate the models, the Fréchet Inception Distance (FID) is used. The explanation of this metric is given in section *Evaluating Models*. Since the main aim was to generate cartoon-like images from real-world images, the FID score is calculated by using cartoon images generated and the cartoon image dataset. To evaluate the models, three different distance values are calculated. For the first score, we tested the model on the cartoon images that are used on training to see how well the given model learned the distribution of the cartoon images in the training set. As the second experiment, the FID metrics of the models are measured for a separate test set that contains 1000 cartoon images are measured. Finally, the model performance is measured on all of the cartoon images present in our dataset. All of these measurements lead to consistent results, which shows that the LSGAN variant was the most effective model for learning image translation $A \rightarrow B$ where A is the domain of natural images and B is the domain of cartoon-like images. The failure of the model using the WGAN variant is also observed statistically with the experiments performed. The FID values obtained for the mentioned experiments are given in Table I.To illustrate the performance of the trained models visually, the inference results for translation $natural \rightarrow cartoon$ are given in Fig. 4.

FID to	LSGAN	VanillaGAN	WGan
Train Cartoon	**58.113**	58.803	138.299
Test Cartoon	**61.110**	62.877	140.432
All Cartoon	**54.485**	56.629	136.696

TABLE I
FID (FRÉCHET INCEPTION DISTANCE) BETWEEN THE GENERATED CARTOON DISTRIBUTION AND THE ORIGINAL SHINKAI STYLE CARTOON DISTRIBUTION

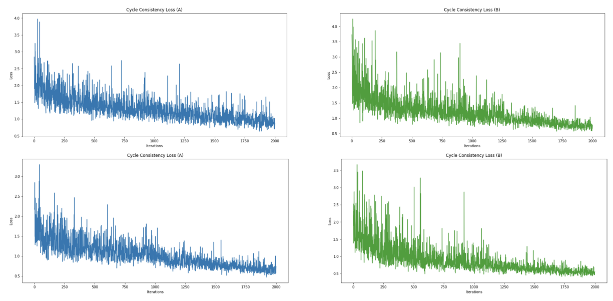

Fig. 2. Plots of L_{CYCLE} for network with Vanilla GAN variant (first row) and LSGAN variant (second row)

V. DISCUSSION

In this study, CycleGAN framework has been applied to learn the translation from natural images to cartoon images as a domain adaptation problem. While applying CycleGAN three different GAN variants which are namely Vanilla GAN, LSGAN, and WGAN (Wasserstein GAN). After training these

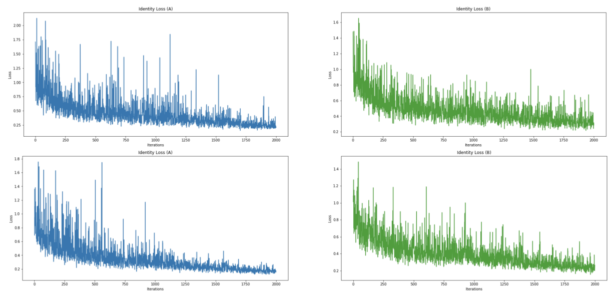

Fig. 3. Plots of $L_{IDENTITY}$ for network with Vanilla GAN variant (first row) and LSGAN variant (second row)

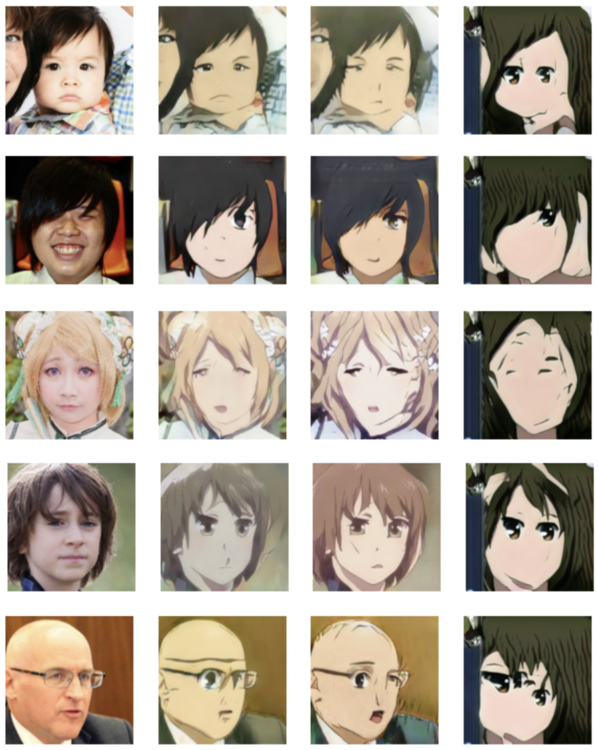

Fig. 4. Sample images from the dataset (first column) Sample results obtained by LSGAN (second column), Vanilla GAN (third column) and Wasserstein GAN (fourth column)

three models in parallel, their performance is evaluated quantitatively by using Fréchet Inception Distance (FID) metric. Results showed us that model using LSGAN achieved the best performance whereas the worst performance is observed in the model using WGAN. Comparing the models using LSGAN and Vanilla GAN, there is a difference in terms of network stability as it can be seen from the loss plots given in Fig. 2 and 3. Since LSGAN uses Least-squares error as the loss metric rather than Binary cross-entropy, the gradients obtained were larger even for samples in the correct side of the decision boundary, the decision boundary learned by the discriminator network was able to learn further compared to Vanilla GAN [4]. For the case of WGAN, the main promise was to achieve stable learning but with the use of momentum in learning and a short amount of training for the architecture, the attempt was not successful. Overall, after evaluating the model both quantitatively and qualitatively, the CycleGAN variant that uses LSGAN was declared as the best performing task four our task of adapting natural image domain to cartoon image domain.

VI. Future Work

While observing the results, we observed that the cartoon dataset was biased towards age, gender, and racial properties. The first improvement that should be made is to select a broader cartoon dataset in terms of context and size. This would help further generalize the network inferences.

One other observation was that the network was aimed to confuse the surroundings of the faces with hair. To solve that, a mixed training procedure can be done such that in every n iterations, a scenery image can be passed into the network to learn the details of the surroundings, while keeping the focus on the faces. Also, we believe that we can use U-Net [7] as the generator architecture since it is a more reliable network in terms of learning segmentation, which could help learn face content better than the ResNet architecture since we observed that most of the time, mouths are diminished in the resulting images.

Furthermore, we could improve the Wasserstein GAN architecture by changing its parameters and its optimizer. We have come to understand that momentum-based optimizers don't work well with WGAN since momentum loss for the critic is nonstationary. Hence, an improvement can be using SGD (Stochastic Gradient Descent) when training. However, it is observable that all of these changes require computational power and time, so a better configuration of parallel working GPUs can be considered essential while training CycleGANs with any objective function.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Networks*, 2014. arXiv: 1406.2661 [stat.ML].

[2] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[3] A. Brock, J. Donahue, and K. Simonyan, *Large Scale GAN Training for High Fidelity Natural Image Synthesis*, 2019. arXiv: 1809.11096 [cs.LG].

[4] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least Squares Generative Adversarial Networks," *arXiv preprint arXiv:1611.04076*, 2016.

[5] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein GAN*, 2017. arXiv: 1701.07875 [stat.ML].

[6] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[7] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015. arXiv: 1505.04597 [cs.CV].

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*, 2018. arXiv: 1706.08500 [cs.LG].

[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, *Improved Techniques for Training GANs*, 2016. arXiv: 1606.03498 [cs.LG].

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the Inception Architecture for Computer Vision*, 2015. arXiv: 1512.00567 [cs.CV].

[11] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, 2017. arXiv: 1412.6980 [cs.LG].