Image-Image Translation with cGANs

John Curci, Tammy Qiu, Zahid Hasan
{jcurci92, tqiu, zhasan57}@bu.edu

Figure 1. An example of pix2pix on labels-to-facade and bw-to-color tasks

1. Task

For this project, we investigated and compared the current state of the art models which perform image-image translation, including pix2pix [1], BicycleGAN, and cVAE GAN. We also implemented the WGAN version of those models. Image-to-Image translation models can make arbitrary style modifications to images based on conditioned examples. Image-image translation models using conditional GANs are models which can function in multiple image categories without modifying and relying on handcrafted features. This task is difficult because it requires complex image processing as well as deep learning techniques to perform the translation. It is also difficult to measure success due to the somewhat subjective nature of the task, although papers have pursued various metrics. We will qualitatively compare and evaluate the results of original models and newly implemented WGAN versions of the models as well as test their generalities by testing on a wide variety of datasets, including those on which they have not previously been tested.

2. Related Work

The image-image translation task is an area that has seen significant progress in recent years. Prior to recent advances, this task required carefully handcrafted models and loss functions, and each new architecture would only work on a narrow set of tasks [2]. The popularization of Generative Adversarial Networks (GANs) has made it possible for new models to solve a generalized version of this task. Arjovsky et al. show that the original GAN generator's objective function suffers vanishing gradients and the proposed alternative cost function has a high variance to make the model unstable [3, 11]. Instead, they proposed to implement Wasserstein GAN (WGAN) [4] to get better stability and mode convergence.

GANs have been applied to a range of image translation tasks including style transfer, image superresolution, and object transfiguration [2, 6, 7, 8]. Early work with the addition to applying GAN loss, they focused on using simple loss functions such as L1 or L2 loss to condition the output on the target, which tended to produce blurry results [6]. Isola et al. introduced a unified framework known as pix2pix architecture improves upon these methods by explicitly using a conditional GAN architecture and loss [2]. Pix2pix performs well on the image translation task, but however, it suffers a low diversity of outputs [8].

Larsen et al. proposed a new method to learn an embedding to high-level abstract visual features and generate dataset samples called conditional variational autoencoder (cVAE) GAN [5]. Additional work has attempted to address the issue of mode-collapse [9]. Zhu et. al combine cVAE-GAN [5] and conditional latent regressor (cLR) [10] GAN to enforce connection between latent encoding and output jointly. This model known as BicyleGAN produces visually appealing output across a wide range of image-to-image translation problems, and more diverse output than other baseline models like pix2pix and cVAE-gan.

3. Approach

At the beginning of this project, we found and reused Tensorflow implementation of Pix2Pix, cVAE-GAN, and BicycleGAN, which are based closely on the original papers. These three codes are our baseline models. We trained and tested them on the existing datasets.

Secondly, we added two new datasets, which we call ADE20K (Semantic Segmentation) and RGBD Scenes, and we have trained and tested our three baseline models on these new preprocessed datasets.

Later, we modified the original versions of the baseline code for all three models – Pix2Pix, cVAE-GAN, and BicycleGAN. We modified the code to implement the Wasserstein GAN version of these models. This produces three more models for a total of six models which we compared in our testing.

Finally, we have in total six models for our evaluation of image-to-image translation: our three baseline models and the WGAN versions of each model. We trained and tested those models on the old and new dataset. We experimented with different combinations of WGAN hyperparameters for the modified models and draw some conclusion based on our observation.

Pix2Pix: It's our first baseline model. In this case, the generator colorizes the input image and the discriminator learns the difference between the true target colored image and generator output image.

The generator, G, in pix2pix has an encoder-decoder like structure. The idea is to compress the input data into a high-level representation using the encoders and the reverse by the decoder. The encoder and decoder has the structure of

Conv -> Batchnorm -> Leaky ReLU and DeConv -> Batchnorm -> ReLU

In pix2pix, the author uses a skip connection between encoder and decoder, called "U-Net", for performance improvement.

The discriminator, D, of pix2pix, differentiates between the true target image and generating a colored image. It has a structure like the Generator's encoder section and the architecture is called a "PatchGAN".

The conditional GAN objective can be expressed as

$$L_{cGAN}(G,\ D) = E_{x,y\sim p_{data}\ (x,y)}[log(D(x,y))\ ] + E_{x\sim p_{data}(x),\ z\sim p_{data}(z)}[log(1 - D(x, G(x,z)))] \quad (1)$$

Here, G tries to minimize this objective against its adversary D, and tries to maximize the following equation:

$$G^* = arg\ min_G max_D L_{cGAN}(G,D) \quad (2)$$

To compare this conditional GAN objective to the vanilla GAN objective, the G and D network of vanilla GAN do not directly observe x, the condition:

$$L_{GAN}(G,\ D) = E_{x,y\sim p_{data}\ (x,y)}[log(D(y))\ ] + E_{x\sim p_{data}(x),\ z\sim p_{data}(z)}[log(1 - D(G(x,z)))] \quad (3)$$

In addition to the conditional GAN loss, the generator is asked to generate image near the ground truth in an L2 sense, but L1 loss produce less blurring:

$$L_{L1}\ (G) = E_{x,y\sim p_{data}(x,y),\ z\sim p_{data}(z)}[||y - G(x,z)||_1] \quad (4)$$

So, the final objective for pix2pix becomes

$$G^* = arg\ min_G max_D L_{cGAN}(G,\ D) + \lambda L_{L1}(G) \quad (5)$$

The z here helps to produce diverse output. The training of the generator targets that objective function. The discriminator is trained to find the difference between the generated output and the target output.

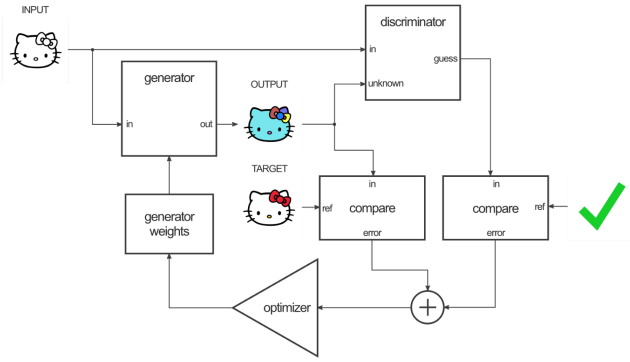

Figure 2. The Generator training framework for Pix2Pix

Conditional VAE-GAN:

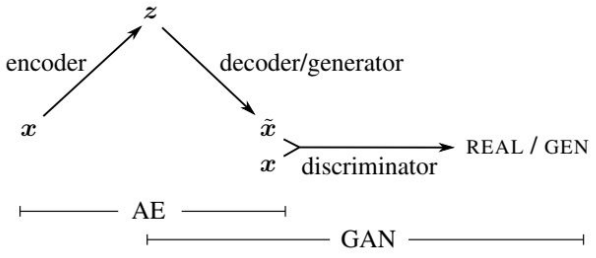

Figure 3. GAN discriminator is appended to autoencoder and compares decoder output to target [5]

The cVAE-GAN is a modification of the standard VAE, which treats the decoder as the generator of a GAN, and then adds a GAN discriminator. The GAN discriminator loss replaces the VAE reconstruction loss in the cVAE-GAN objective function:

$$L_{GAN}{}^{VAE}(G,\ D) = E_{A,B\sim p_{data}\ (A,B)}[log(D(A,B))\ ] + E_{x\sim p_{data}(A),\ z\sim p_{data}(z)}[log(1-D(A,G(A,z)))] \quad (6)$$

$$L_{KL}(E) = E_{B\sim p\ (B)}[D_{KL}(E(B)||N(0,I))\ ] \quad (7)$$

$$G^*,\ E^* = arg\ min_{G,\ E} max_D\ L_{GAN}{}^{VAE}(G,\ D,\ E) + \lambda L_1{}^{VAE}(G,E)\ \ + \lambda_{KL} L_{KL}\ (E) \quad (8)$$

This model provides an interesting alternative way to condition input images on targets, by providing the generator with a low-dimensional representation of target images during training. However, when random latent codes are used at test time, cVAE-GAN might not be able to produce realistic results.

BicycleGAN:

BicycleGANs directly address the issue of mode collapse by combining a cVAE-GAN and cLR-GAN. They take advantage of the transformation (image->latent space->image) from cVAE-GAN, as well as the (latent space->image->latent space) transformation of cLR-GAN. In this way, the model avoids mode collapse, because mode collapse can be considered a loss of information, and for a mapping to be reversible it must retain all sufficient information to reconstruct the input.

The loss function for BicycleGAN combines the loss functions of the cVAE-GAN and cLR-GAN:

$$L_1{}^{latent}(G,\ E) = E_{A\sim p_{data}\ (A),\ z\sim p_{data}\ (z)}[||z - E(G(A,z))||_1\ ] \quad (9)$$

$$G^*,\ E^* = arg\ min_{G,\ E} max_D\ L_{GAN}\ (G,\ D) + \lambda_{latent}\ L_1{}^{latent}\ (G,E) \quad (10)$$

$$G^*,\ E^* = arg\ min_{G,\ E} max_D\ L_{GAN}{}^{VAE}(G,\ D,\ E) + L_{GAN}\ (G,\ D)\ + \lambda L_1{}^{VAE}(G,E)\ \ + \lambda_{KL} L_{KL}\ (E)$$

$$+ \lambda_{latent}\ L_1{}^{latent}\ (G,E) \quad (11)$$

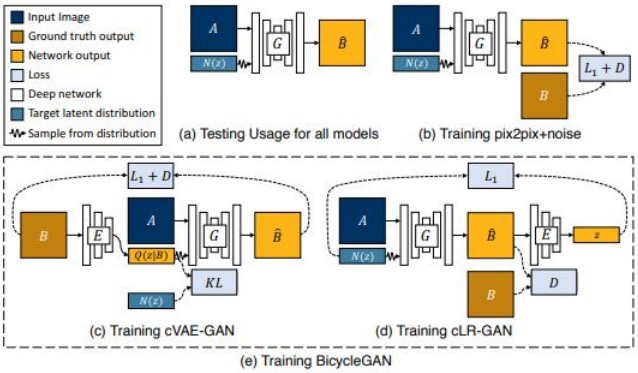

Figure 4. Overview of BicycleGAN structure[5]

Wasserstein GAN:

The original GAN paper addressed the gradient vanishing issue (figure below) by proposing a new loss function, but this one has a high variance that may make the model unstable.

Wasserstein GAN (WGAN) introduce a new cost function using Wasserstein distance. It has a smoother gradient for the distribution distance everywhere. It ensures WGAN learns no matter the generator is performing or not.

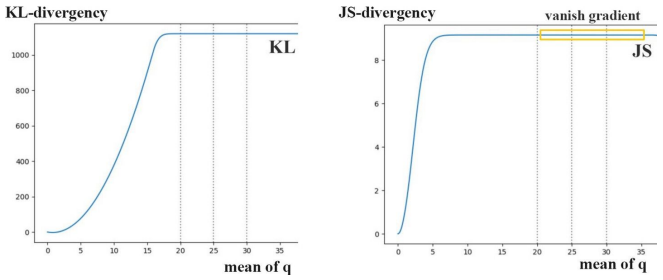

Figure 5: The gradient of two distributions p and q increase with the distance eventually gets saturated when p and q has a high mean difference. (Assuming p and q has the same variance)

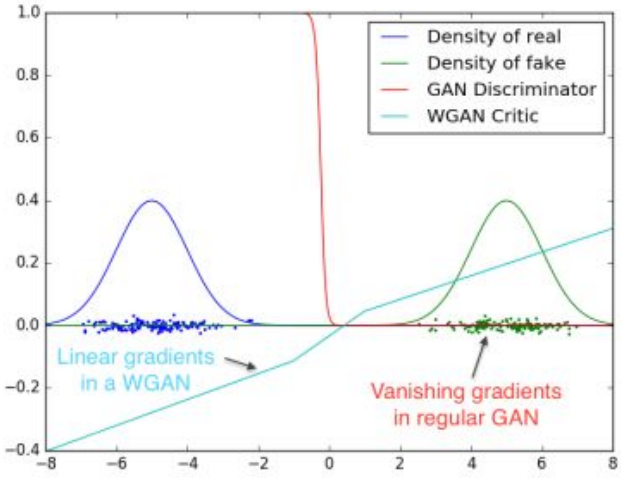

Figure: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space [4]

The key difference in WGANs is that it substitutes the original GAN objective, which is equivalent to minimizing KL divergence, to Wasserstein distance between the generated and true distribution. The Wasserstein distance can be simplified as:

$$W(P_r, P_g) = sup_{\|f\|_L <= 1} E_{x \sim P_r}[f(x)] - E_{x \sim P_g}[f(x)] \quad (12)$$

With f having the technical requirement of being a 1-lipschitz function. In this case, f is the original GAN discriminator with any sigmoid output layer removed, returning pure logits. In WGAN, the discriminator outputs an unbounded score of how real the sample is. This function is often referred to as a "critic" because it now technically performs regression instead of classification. Ignoring any other loss terms (e.g. GANs often include an L1 loss term) the WGAN loss can be expressed as:

$$L_{disc/critic} = E_{x \sim P_r}(log\ D(x)) - E_{x \sim P_g}[D(x)] \quad (13)$$

$$L_{gen} = - E_{x \sim P_g}(log\ D(x)) \quad (14)$$

Algorithm for implementing WGAN [4]:

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Implementation of Wasserstein GAN:

We have implemented WGAN in all of our three baseline models, discussed earlier. We followed a similar technique to modify the original source codes to WGAN version of that models. Apart from some practical issues depending on model, our implementation of WGAN on the all original codes generally follows the following steps:

1. Loss function should have no logarithmic loss. The output of D is no longer a probability and doesn't need any sigmoid. We first remove sigmoid from the discriminator output and use this output in the generator and discriminator loss for optimization. We have kept other loss like L1 loss, VAE-GAN loss, CLR loss unchanged.
2. The Discriminator weights need to be clipped to enforce the Lipschitz constraint on the critic's model to calculate the Wasserstein distance. However, this clipping is a crude solution to that requirement, and the clipping threshold is a difficult parameter to tune. The WGAN models were very sensitive to this hyperparameter.
3. The Wasserstein GAN paper recommends training the discriminator for more iterations than the generator, so this functionality must be implemented.
4. We had to use Root Mean Squared Propagation (RMSProp) as the optimization scheme rather than ADAM optimizer. The learning rate affects the generator. We selected a lower learning rate than ADAM optimizer for the original model by trial and error.

The WGAN version of the original code took more time to train because of the lower learning rate requirement and RMSProp optimizer.

4. Dataset and Metric

To explore the generality of the cGAN we would like to test this on different datasets. For semantic labels, the cityscapes datasets [12], containing 25000 images with the mixture of fine and course annotation, will be used. Next for architectural labels facade datasets of around 500 images will be used. The aerial to maps dataset contains 1096 training images scraped from Google Maps. Edges2Shoes contains 50k training images from UT Zappos50K. Edges2Bags contains 137k images from Amazon Handbags.
dataset

- Cityscapes: https://www.cityscapes-dataset.com/
- Facades: http://cmp.felk.cvut.cz/~tylecr1/facade/
- Edges2Shoes: http://vision.cs.utexas.edu/projects/finegrained/utzap50k/
- Edges2Bags: https://people.eecs.berkeley.edu/~junyanz/projects/gvm/datasets/handbag_64.zip
- Maps: https://people.eecs.berkeley.edu/~tinghuiz/projects/pix2pix/datasets/maps.tar.gz
- RGBD: https://rgbd-dataset.cs.washington.edu/
- ADE20K: http://groups.csail.mit.edu/vision/datasets/ADE20K/

We will also be using an unseen RGB-D scenes dataset to retrain the network on depth to image pairs. The RGB-D datasets contains color and depth images for 3975 training examples, 806 test examples, and 199 validation examples.

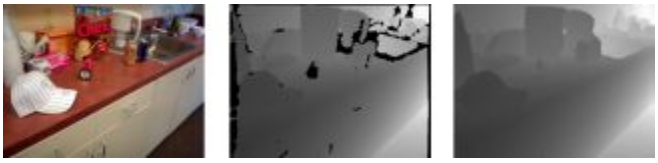

Figure 6: Sample set of images from the RGB-D dataset

The motivation for applying all the models to this dataset is to test the network's ability to either serve as depth estimation or to evaluate its ability to reconstruct scenes from depth information. All of the models can be applied in either direction.

In addition, we have also tested the networks on the ADE20K dataset, which contains RGB scenes and their semantic segmentation. The dataset contains 13,887 training examples, 2950 test examples, and 751 validation examples.

Figure 7: Sample set of images from the ADE20K dataset

For both new datasets, we split and reorganized the images into the test, train, and validation sets using a preprocessing script that we wrote. Then we paired the images using a script provided in the original pix2pix github (https://github.com/phillipi/pix2pix) for training. We trained them on the pix2pix, cVAE-GAN, and BicycleGAN(https://github.com/kvmanohar22/img2imgGAN). We used three baselines Tensorflow codes from these sources and made them runnable in SCC. We modified them to build WGAN version of the original Tensorflow code and ran the WGAN-models and original models to experiment with the datasets.

5. Results

We first focused on implementing the original models (pix2pix, cVAE-GAN, BicycleGAN) on the datasets discussed in the paper: facades, cityscapes, maps, edges2shoes, edges2handbags. For this task, we collected code from the above links, which we modified to run on the SCC. After successfully re-using the modified code, we implemented the WGAN version of each of these models. Here are some of the result from the original models for comparison.

Figure 8: Sample test result from the test data of the cityscapes dataset after 2 epochs (BicycleGAN)

Figure 9: Sample test result from the test data of the maps dataset after 2 epoch (Pix2pix)

Figure 10: Sample test result from the test data of the facades dataset after 6 epoch (pix2pix)

Figure 11: Our results after 200 epochs of training (pix2pix)

Experiment details

Facades
400 images, 50 epochs
Pix2pix baseline performs a realistic reconstruction by suffers from mode collapse:

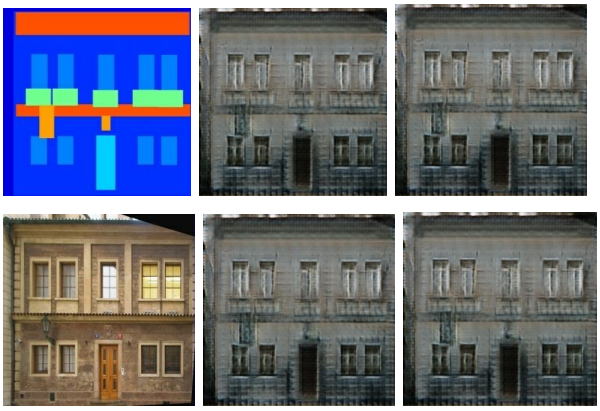

Figure 12: Basic pix2pix and WGAN-pix2pix suffer from mode collapse. Outputs (right) given input (top left) and target (bottom left) are almost identical.

Cityscapes:
2975 images, 10 epochs
Pix2pix with WGAN still experiences mode collapse. These are from the same input/target:

Figure 13: Three results for Pix2pix with WGAN still experiences mode collapse. Outputs are close to identical (but not exactly).

Edges2Handbags:
138576 images, 1 epoch

Figure 14: Six Bicycle-WGAN results for input (top left) and target (bottom left). Outputs are both realistic and diverse.

Figure 15: Sample output for WGAN-cVAE-GAN (second to right) and cVAE-GAN (rightmost) and original output (leftmost)

We have tested and trained all of our six models (Baseline-pix2pix, cVAE-GAN, BicycleGAN, and their modified WGAN versions). The results are all stored in the following SCC location on read-only mode.

/projectnb/dl-course/jcurci92/Project/results/

In terms of hyperparameters, the original pix2pix offers tuning the weight of the two losses (L1 and GAN loss). The cVAE-GAN has three types of losses and associated weights, and Bicycle has five types of losses and associated weights. They can be changed to see the variety of learning styles of the generator weights. Moreover, all of the models have the flexibility to choose the depth of the neural network.

The WGAN offers three additional parameters to the existing parameters of the original models. They are the learning rate of RMSProp, the number of times of Discriminator trains over the Generator and finally the clipping value for the weights.

The learning rate of RMSProp suggested by the original WGAN paper [4] is 0.00005. The same value works in our WGAN versions. We experimented with some higher values to check if they would work better and lead to better results using fewer epochs. All of the Generators of our WGAN version models were able to learn faster with a RMSProp learning rate 0.0005.

For the second WGAN hyperparameter, We also experimented with training the Discriminator more times than the Generator. We found that the more we trained Discriminator over the Generator, the more time the WGAN models would take to train while offering no visible improvement of the generator performance. Sometimes it would even cause generator performance to decrease. The generator generates the best visual samples when the G and D are trained equally.

The third parameter for WGAN models is the value to which we bound the Discriminator weights. This parameter is quite crude and a little change may cause the gradient to vanish or explode. The original [14] paper suggested the value to be 0.01, but can be varied at different values. In our experiment with the WGAN models, a clipping value of 0.1 works better than the suggested 0.01 clipping value. In some of our WGAN models, the 0.01 value failed to make the generator trained well.

Compared to the original models (pix2pix, cVAE-GAN, BicycleGAN) and its corresponding WGAN version, both models seem to generate similar images in terms of quality. WGAN versions of the models slightly sharpen the image compared to its vanilla counterpart.

As we have shown in the result earlier, both pix2pix and WGAN-pix2pix gives quite quality output image but they lack diversity for a particular input. We have overcome the issue using BicycleGAN and WGAN-Bicycle gan. We have added noise to the input image for pix2pix and WGAN-pix2pix but still, it does little to generate diverse output for a particular image. Meanwhile, BicycleGAN and WGAN bicycle are capable of producing multiple outputs for the same input image. In general, WGAN-version of the original model behaves like the original model apart from slight deviation. WGAN implementation of pix2pix didn't contribute to its mode collapse issue, it's BicycleGAN that addresses the diversity issue and so, WGAN-BicycleGAN also learned to produce diverse coloring scheme for a particular input.

From the experiment of this project, we cannot confidently conclude if the WGAN version of pix2pix, WGAN version of cVAE-GAN or WGAN version of BicycleGAN is better than its original vanilla version. Visually, the WGAN versions were slightly outperformed by the original model. Additionally, WGAN-versions of the models took significantly more times to run, due to the additional complexity. The table below gives a comparison of the running times. The hyperparameter of the number of times we want to train discriminator more than the generator directly affects the time for training.

Table1: Average running time for each epoch in cityscapes dataset for WGAN-pix2pix. (Original pix2pix time - 300s)

Disc training Times than generator	2	3	4
Rmsprop LR			
0.0002	356s	392s	412s
0.0001	332s	392s	413s
0.00005 [14]	341s	388s	439s

6. Detailed Timeline and Roles

Table2: Detailed timeline and roles

Task	**Deadline**	**Lead**
Run code on SCC and compare results	13/12/18	John

Implement code to load and process new datasets	13/12/18	Tammy
Implement of WGAN models in Tensorflow	13/12/18	Zahid
Prepare reports and presentation	13/12/18	all

In terms of future works, better metrics can be applied to for the models' output image. Secondly, the gradient penalty version of WGAN (WGAN-GP) can be incorporated with the baseline models to check their performance.

7. Code repository

We have uploaded all the codes and instruction to run any models in the SCC BU. Please feel free to email us if any of the code (original or WGAN) fails or runs into issues.

https://github.com/tqiu8/cgan_model

References

1) P. Isola, J. Zhu, T. Zhou and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR): 5967-5976, 2017.
2) I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.
3) J.-Y. Zhu, P. Krahenb uhl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In ECCV, 2016.
4) M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein GAN", Arxiv.org, 2018.
5) J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. Efros, O. Wang, and E. Shechtman. Toward Multimodal Image-to-Image Translation, arXiv.org, 2018.
6) A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300, 2015
7) D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. CVPR, 2016.
8) J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, arXiv.org, 2017.
9) O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, pages 234–241. Springer, 2015.
10) Ledig, C., Theis, L., Husz´ar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image superresolution using a generative adversarial network. Proceedings of CVPR (2017)
11) M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", Arxiv.org, 2018.
12) M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In CVPR, 2016