Self-Supervised Image-to-Image Translation

CSCI 3033-091 Project Proposal

Tanran Zheng
tz408@nyu.edu
New York University
Courant Institute of Mathematical Sciences

Yuxiang Zhu
yz7043@nyu.edu
New York University
Courant Institute of Mathematical Sciences

***Abstract*—In this project we aim to solve image-to-leverage self-supervised learning to pre-trained a backbone, and then perform transfer learning to GAN based model. We show that with this method we can achieve better performance in image colorization and Monet style transfer using only a small domain specific dataset.**

I. Introduction

Image-to-Image Translation (I2I) aims to transfer images from a source domain to a target domain while preserving the content representations. A good model will learn the mappings between different image domains, how to represent these mappings to generate the desirable results is explicitly related to the generative models. However, I2I needs domain specific data. For example, one will need the image data of zebras and horses in order to solve a zebra-to-horse translation (replace zebras in the image by horses). And this dataset is domain specific, i.e. it cannot be used to other target domain, such as Cartoon style transfer. These domain specific data are usually hard to find, or has small data size. New real data for some domains are not possible to get, such as Monet style transfer. So with this motivation, we want to solve the I2I tasks with fewer domain specific data by leveraging self-supervised learning.

Self-supervised learning has become one of the most promising areas of AI research today, and have demonstrate remarkable power to learn representations. The model trains itself to learn one part of the input from another part of the input. It aims to solve the similar problem that it's impossible to label everything in the world. Therefore, in this project, we would like to investigate combined these two methods. We use self-supervised learning to pre-train backbone with easily accessible data from other domain, and use transfer learning to replace the encoder of the generator of the GAN based model, such that the downstream GAN based model requires less amount of domain specific data for the I2I tasks. We implement and train SwAV and SimSiam as our self-supervised training part, and implement and train Conditional GAN and CycleGAN for the downstream task. We train our models on two downstream tasks: image colorization and Monet style transfer. In image colorization our model is given a grey-scale image, and it will output a faithful colorized version of that image; in Monet style transfer, our model is able to turn a painting of Monet into real pictures, or given a real picture it will turn it into a paint with Monet's style.

II. Dataset

A. Unlabeled data for self-supervised pre-training

We use half of the data from `ImageNet-1K`, which corresponding to 600,000 color images with resolution of 256×256. We did not use more data because the memory limit of Greene HPC. During the training, the labels of `ImageNet-1K` is not used.

B. Data for downstream training

We use COCO dataset to train SwAV model for the colorization task. After training, COCO dataset and photos taken by phones to evaluate how well the model can perform.

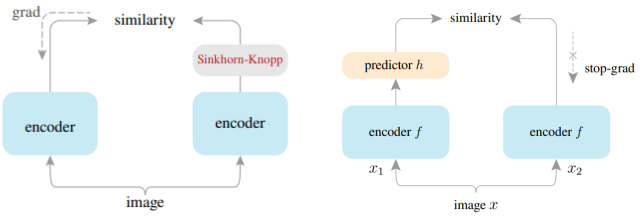

Fig. 2: Self-Supervised training algorithms. **Left**: SwAV; **Right**: SimSiam.

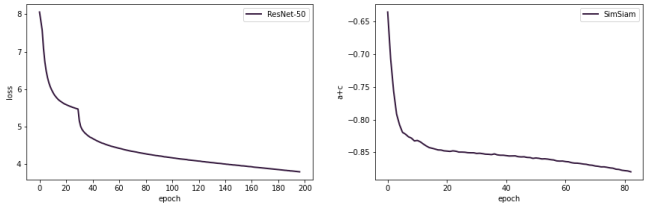

Fig. 3: Self-Supervised Backbone training loss. **Left**: SwAV; **Right**: SimSiam.

III. Methodology

The flow chart of our methodology is shown in Figure 1. We first use unlabeled to do self-supervised training using SwAV and SimSiam. With the trained ResNet-50 backbone, we will replace the UNet of the generator of the GAN based model

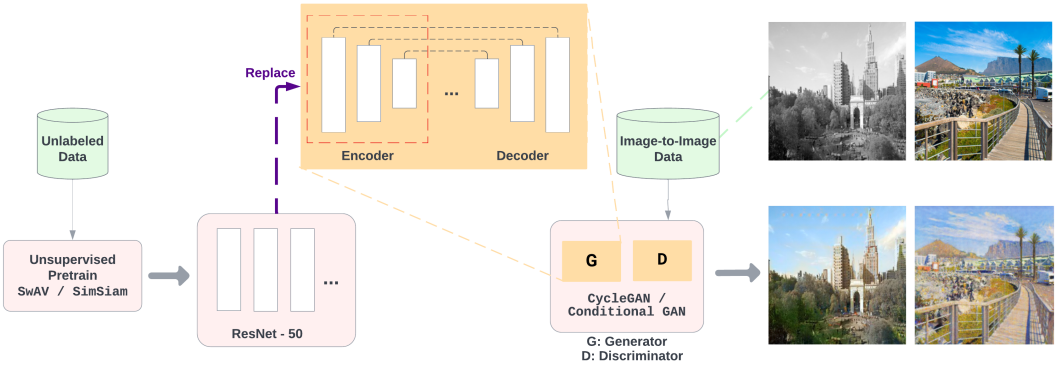

Fig. 1: Methodology

by this pretrained backbone, and then use only a portion of domain specific data to train the GAN based model for the downstream task.

A. Baseline - Conditional GAN with Untrained UNet

For the baseline, we use a untrained conditional GAN with a UNet as the generator. [1] The architecture of UNet is shown in Figure 4. The baseline will be trained using the same amount of data and same number of epochs as the other models we have.

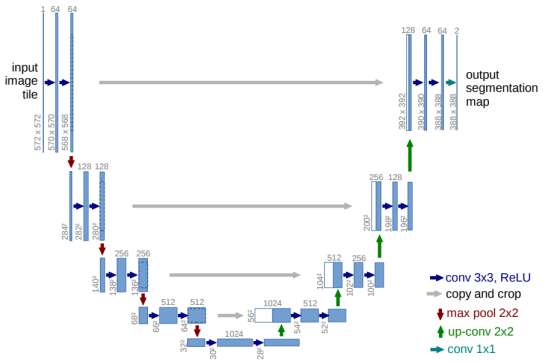

Fig. 4: UNet Architecture

B. Self-Supervised Pre-train: SimSiam

SimSiam (Simple Siamese) is a self-supervised techniques to train the models to learn meaningful visual representation. [2] Comparing to earlier self-supervised framework such as MoCO [3] and SimCLR [4], it doesn't need: (1) negative sample pairs, (2) large batches, (3) momentum encoders. It uses a simple approach, stop-gradient, to prevent the collapse of the model, which is one of the major difficulties in self-supervised learning domain. We choose it as one of our self-supervised learning approach because comparing to its counterparts, it has simpler architecture, requiring relatively less hardware resources, while not compromising its performance when transferring to other computer vision downstream tasks.

C. Self-Supervised Pre-train: SwAV

Swapping Assignments between Views (SwAV) is a self-supervised training methods for visual features representation [5]. Instead of learning by performing pairwise comparison for each image samples, it compares the corresponding cluster assignments of these images. As shown in Figure 2, it first creates a different view given a input image by performing Multi-crop. Then it uses two identical encoders to transfer these two views into feature space. With the feature space, it use Sinkhorn-Knopp algorithm to perform a soft cluster assignment. With the assignments, the model uses loss functions that measures the similarities and enforce consistency between cluster assignments produced by the same images.

We choose SwAV as our another self-supervised learning method because it shows very promising results in downstream task such as image classification.

D. Downstream: Conditional GAN

To transfer and fine-tune the SwAV model to colorize gray scale images, we follow the suggestion from [6]. We remove last two layers from the SwAV model which are one fully connected dense layer and one softmax output layer concatenating it with a UNet model. With model, we first make three transformations on the trianing images – 1. Resizing, 2. Randomly flipping, and 3. changing RGB channels to 3 gray scale channels. We trained the model for 20 epochs with the combined loss function from [6] shown below.

$$G^* = arg\min\max \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

For evaluation, we randomly choose images changing it to a gray scale image and feed it to the model.

E. Downstream: CycleGAN

CycleGAN is an unsupervised GAN model that has shown promising preformance in I2I tasks. As shown in Figure 5 that it consists of two generators and two discriminators. It uses a cyclical path that contains two mapping functions $G : X \rightarrow Y$, and the reverse $F : Y \rightarrow X$. D_Y encourages G to translate X into outputs that indistinguishable from domain Y, and vice versa for D_X and F. It uses cycle consistency

loss in equation 2 to enforce the cycle-sonsistency within the same domain. More specifically, the forward cycle-consistency will enforce $x \to G(x) \to F(G(x)) \approx x$, and the backward cycle-consistency will enforce $y \to F(y) \to G(F(y)) \approx y$.

$$\mathcal{L}_{cycle}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||] \quad (1)$$
$$+ \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||] \quad (2)$$

Then, we finetune this pretrained model using only a portion of the labeled data, and evaluate the result.

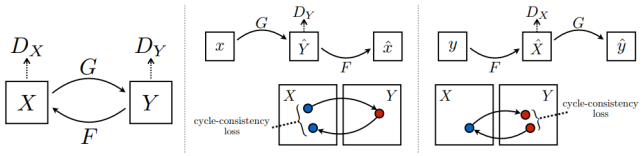

Fig. 5: CycleGAN Architecture

IV. Distributed Training

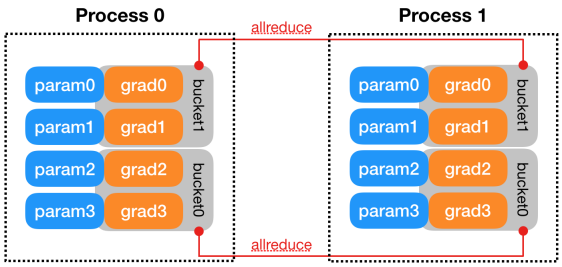

Fig. 6: With insufficient batch size,

As both methods of self-supervised training needs to be trained on large high-resolution data, we need to deploy distributed training. Especially for SwAV, where the batch size of the training must be large, otherwise the model is prone to collapse, as shown in Figure 7.

As a result, we use PyTorch `distributed data parallel` module and `Apex` (Pytorch extension with NVIDIA-maintained utilities to streamline mixed precision and distributed training) to train our backbone using data parallelism. The flowchart of Pytorch `distributed data parallel` is shown in Figure The SimSiam model was trained on 8 Nvidia V100 GPU of single node for 80 wall-clock hours on NYU Greene HPC. SwAV model was trained on 8 Nvidia V100 GPU of single node for 130 wall-clock hours on NYU Greene HPC. Moreover, distributed version of Sinkhorn is used with allreduce method provided by PyTorch.

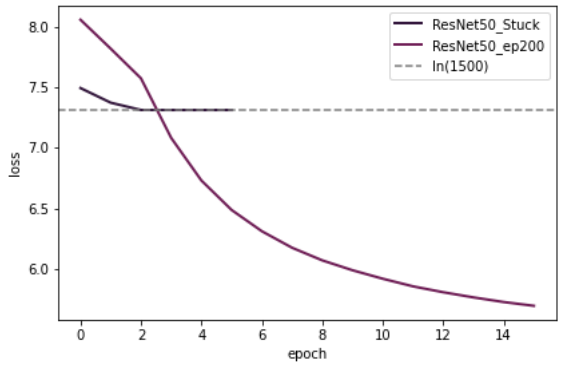

Fig. 7: With insufficient batch size and wrong syncBatch methods, SwAV's training loss will get stuck at ln(number of prototypes)

V. Resutls

A. Downstream: Colorization

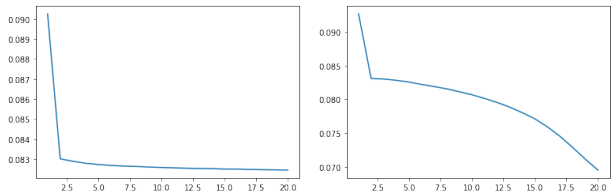

Fig. 10: Colorization training loss

Above are two training loss charts. In the first image, we concatenate our own SwAV model with UNet while we concatenate the Facebook pre-trained SwAV model with UNet in the second image. As the two charts show, the model with Facebook backbone perform better since they trained their model with much more data than us. However, we can still achieve satisfactory training loss and result with smaller dataset, which is noticeable.
From Figure 8 we can see the results of four colorization models. Condtional GAN SwAV, CycleGAN SwAV, and CycleGAN SimSiam all performs better than the baseline and they only requires around 20 epochs of training from their backbones. In addition, the Cond.GAN SwAV models tends to colorize plants better while the other two models colorize the sky and buildings better. Difference in dataset used for training and difference between UNet and CycleGAN may account for this performance nuance.

B. Downstream: Monet Style Transfer

We further adapt our model to another I2I task, Monet style transfer, the results of which are shown in Figure 9. We can see that even the model is only trained with small dataset, it shows promising results in Monet style transfer task. The transformation in each direction is faithful.

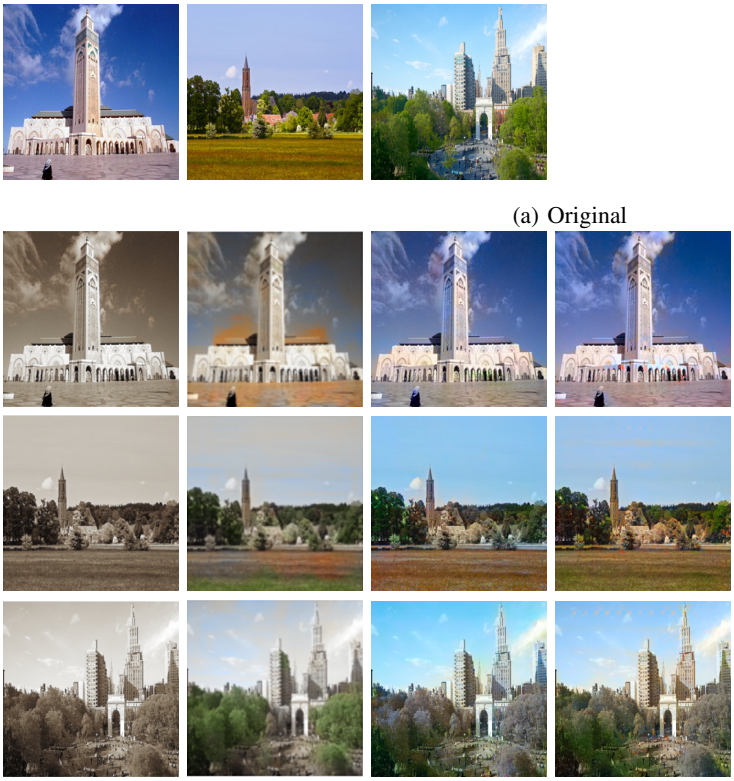

Fig. 8: **Left:** Baseline **Middle left:** SwAV + Cond GAN **Middle right:** SwAV + CycleGAN **Right:** SimSiam + CycleGAN

(a) Monet to Real

(b) Real to Monet

(c) NYU to Monet

Fig. 9: Monet style transfer using SwAV + CycleGAN

VI. Conclusion

In this project, we leverage self-supervised learning techniques and transfer learning techniques, and train Conditional GAN and CycleGAN for Image-to-Image translation task, including image colorization and Monet style transfer. Our approaches and models show promising performance in both tasks with only a few amount of domain specific data in terms of quantitative metrics and visualization.

In the futher, we would like to explore more domains in Image-to-Image translation task. We also like to explore and use more common evaluation metrics for GAN, such as Frechet Inception Distance (FID).

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[2] X. Chen and K. He, "Exploring simple siamese representation learning," *CoRR*, vol. abs/2011.10566, 2020. [Online]. Available: https://arxiv.org/abs/2011.10566

[3] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," *CoRR*, vol. abs/1911.05722, 2019. [Online]. Available: http://arxiv.org/abs/1911.05722

[4] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *CoRR*, vol. abs/2002.05709, 2020. [Online]. Available: https://arxiv.org/abs/2002.05709

[5] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *CoRR*, vol. abs/2006.09882, 2020. [Online]. Available: https://arxiv.org/abs/2006.09882

[6] M. Shariatnia, "Colorizing black amp; white images with u-net and conditional gan — a tutorial," Nov 2020. [Online]. Available: https://towardsdatascience.com/colorizing-black-white-images-with-u-net-and-conditional-gan-a-tutorial-81b2df111cd8