DLCV HW3

R08922050 Chih-Chun YANG

December 2019

1 GAN

1.1 Describe the architecture & implementation details of your model. (5%)

- Data Cleansing:
  - Use the built-in model of opencv2 (CascadeClassifier with haarcascade frontalface alt2) to do face recognition and select the image can be recognized as front face.
- Image Preprocessing:
  - Instead of reading image as "RGB" format, I read the image as the "CIE-LAB" format and only select "L" channel which is a black-and-white image and represents the lightness of the image. The reason will be illustrated later.
  - Scale image to -1 1 by the following formula:

    $ScaledImage = (\frac{image}{255.0} - 0.5) * 2$

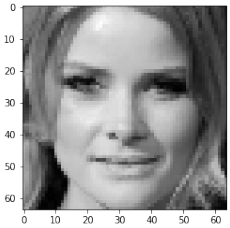

Figure 1: Image with only L channel

- Image Generating Procedure::
  - In order to get a better quality of image, I train 2 GAN model. The first one generate the black-and-white image while the other is responsible for coloring the gray image. Since the first model only need to learn the contour of the face, it can capture the details of the face thus generate image with better resolution and need less time to converge.
- Model Architecture:
  - DCGAN

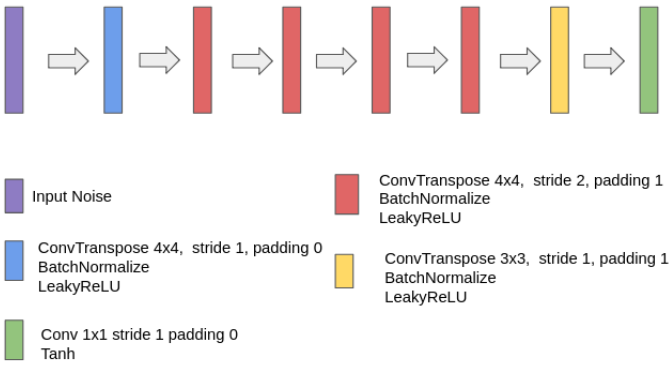

Figure 2: Generator

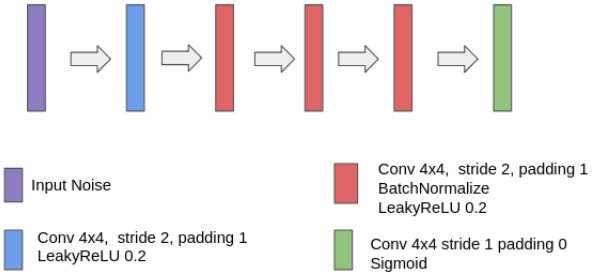

Figure 3: Discriminator

– Pix2Pix

  * Generater:

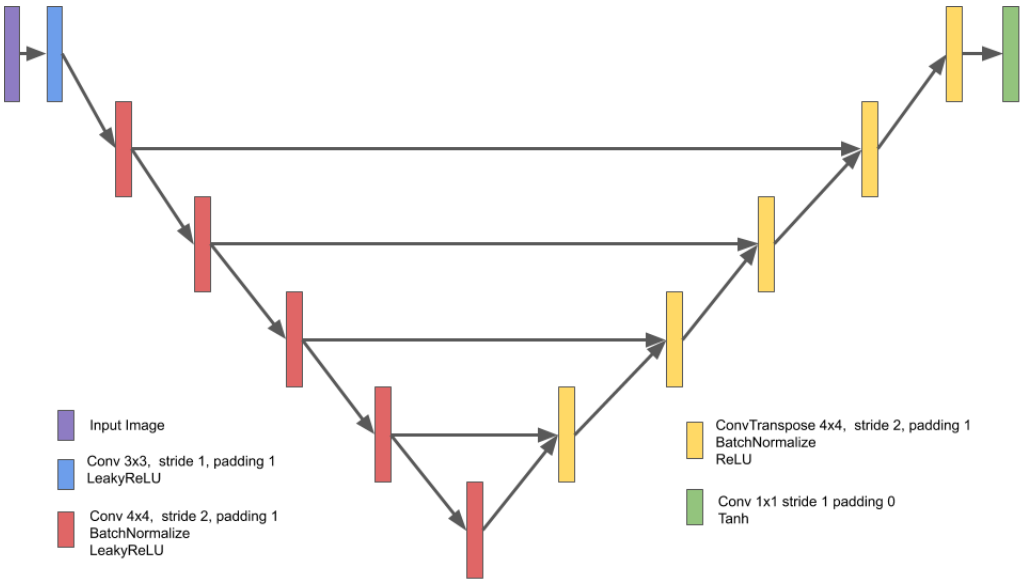

Figure 4: Generator of Coloring Model (U-Net)

  * Discriminator:

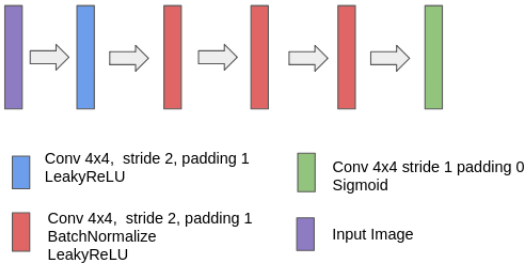

Figure 5: Discriminator of Coloring Model

– Training:

  * As the original paper said, the Discriminator will learn how to distinguish the generated color image from the original color image and the generator should learn how to fool the discriminator.

1.2 Plot 32 random images generated from your model. $\textbf{fig1}_2.jpg(10\%)$

Figure 6: Random 32 Image

1.3 Discuss what you' ve observed and learned from implementing GAN. (5%)

- I found that the model will learn the contour of the face such as cheek and eyes. Then the model will learn more details of the face.
- The learning should not be too high since GAN is sensitive to the learning rate. If capability of the discriminator and generator is imbalanced, the model will collapse.

2 ACGAN

2.1 Describe the architecture & implementation details of your model. (5%)

- Generating Details:
  - Following the same procedure of GAN, I generate the gray image first and then color the image.
- Model Architecture:
  - Following the same procedure of GAN, I generate the gray image first and then color the image.

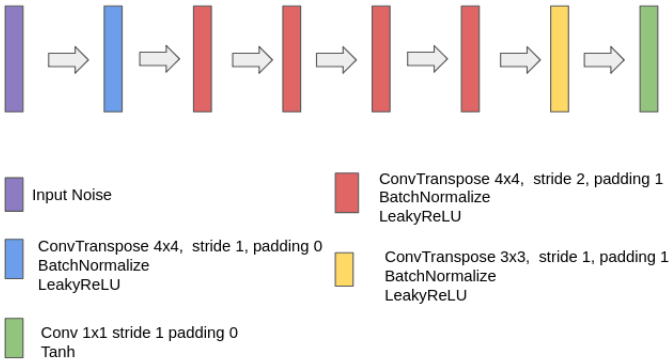

Figure 7: Generator

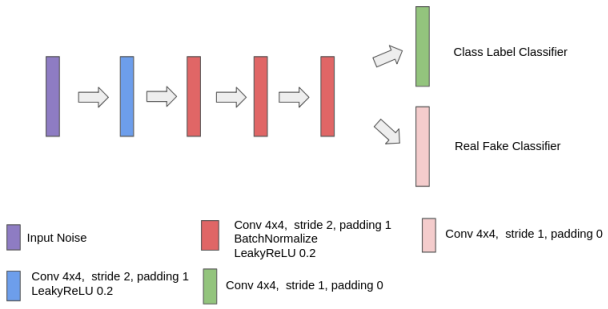

Figure 8: Discriminator

2.2 Plot 10 random pairs of generated images from your model, where each pair should be generated from the same random vector input but with opposite attribute. This is to demonstrate your model' s ability to disentangle features of interest. [fig2$_2$.*jpg*](10%)

Figure 9: 10 Pairs of Image

2.3 Discuss what you' ve observed and learned from implementing ACGAN. (5%)

- Since the model's input is the gray image, the model will learn the contour better than learning the contour and color simultaneously, the model learn almost as fast as DCGAN.

3 DANN

3.1 Compute the accuracy on target domain, while the model is trained on source domain only. (lower bound) (3%)

3.2 Compute the accuracy on target domain, while the model is trained on source and target domain. (domain adaptation) (3+7%)

3.3 Compute the accuracy on target domain, while the model is trained on target domain only. (upper bound) (3%)

	SVHN → MNIST-M	MNIST-M → SVHN
Trained on Source	0.426	0.4004
Adaptation (DANN)	0.8258	0.4101
Trained on target	0.9866	0.9216

3.4 Visualize the latent space by mapping the testing images to 2D space (with t-SNE) and use different colors to indicate data of (a) different digit classes 0-9 and (b) different domains (source/target). (6%)

Randomly sample 200 data for each class (0-9)

- SVHN → MNIST-M

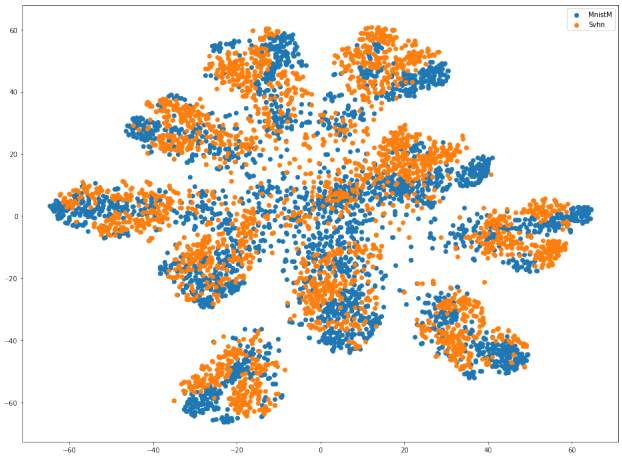

Figure 10: Different Domain

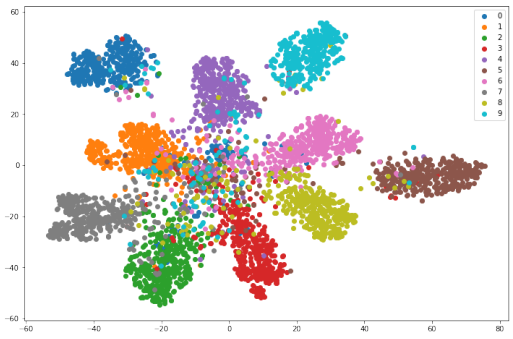

Figure 11: Different Class

- MNIST-M → SVHN

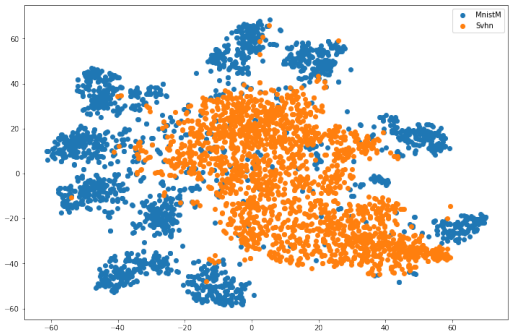

Figure 12: Different Domain

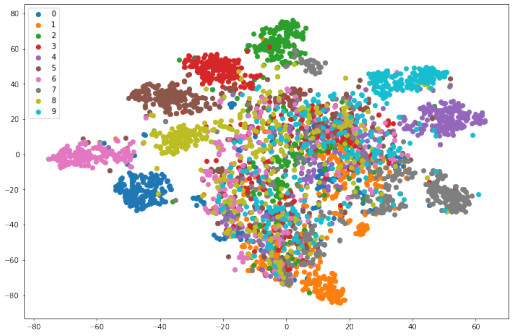

Figure 13: Different Class

3.5 Describe the architecture & implementation detail of your model. (6%)

The feature extractor that I implement is inspired from VGG Net. And the rest of the model is only slightly changed from the original model.

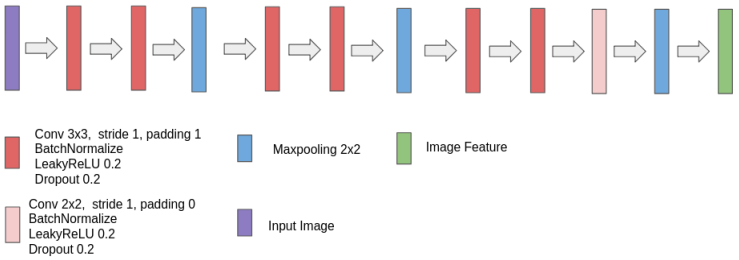

Figure 14: Feature Extracter

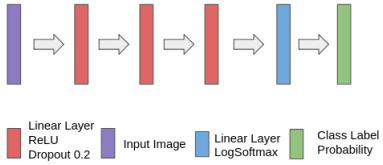

Figure 15: Class Label Classifier

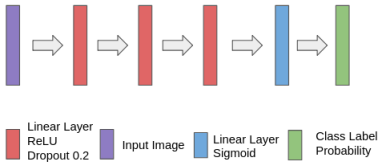

Figure 16: Domain Label Classifier

3.6 Discuss what you' ve observed and learned from implementing DANN. (7%)

- The adversarial gradient could not be too large during the first few epochs since the model are not ready to learn well in the source domain. If the adversarial gradient is too large, it will cause model cannot learn the correct information.

4 Improved UDA model

4.1 Compute the accuracy on target domain, while the model is trained on source and target domain. (domain adaptation) (3+7%)

	SVHN → MNIST-M	MNIST-M → SVHN
Trained on Source	0.4925	0.389
Adaptation (DANN)	0.8438	0.4284
Trained on target	0.9696	0.8995

4.2 Visualize the the latent space by mapping the testing images to 2D space (with t-SNE) and use different colors to indicate data of (a) different digits classes 0-9 and (b) different domains (source/target). (6%)

- Randomly sample 200 data for each class (0-9)
- SVHN → MNIST-M

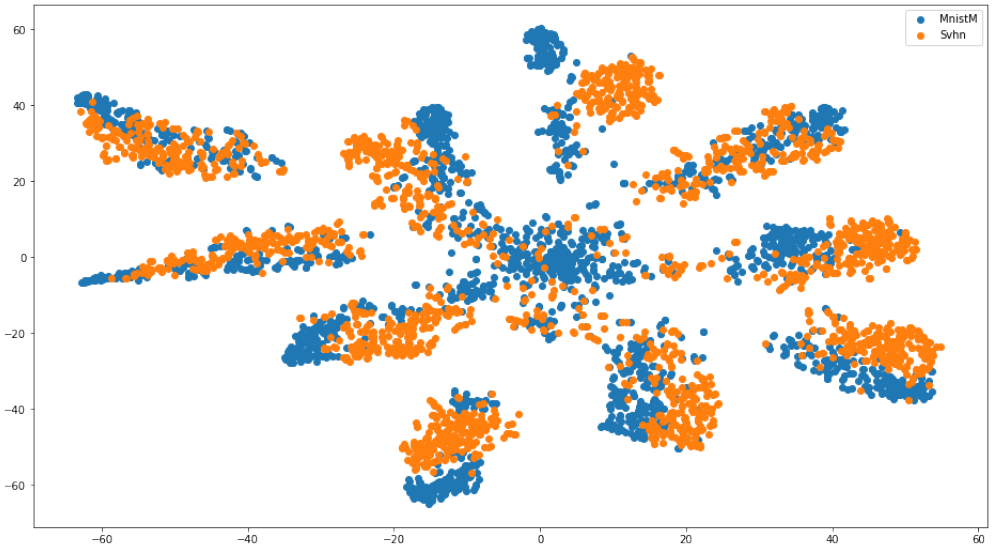

Figure 17: Different Domain

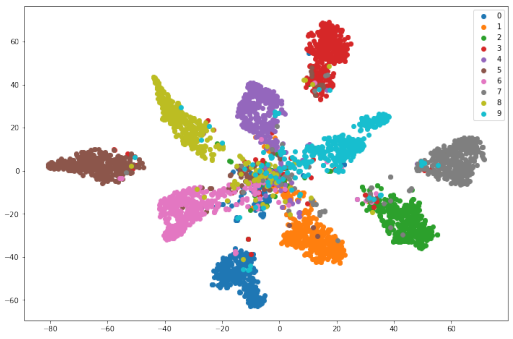

Figure 18: Different Class

- MNIST-M → SVHN

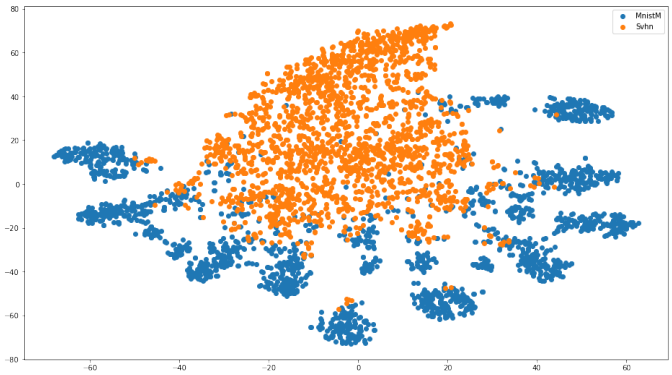

Figure 19: Different Domain

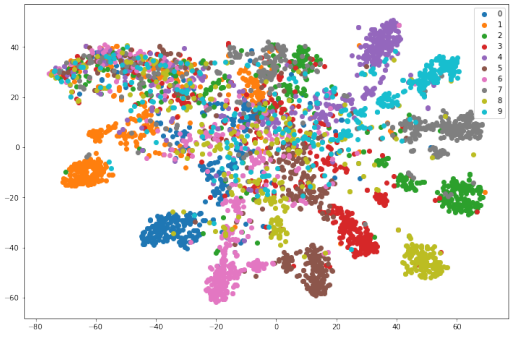

Figure 20: Different Class

4.3 Describe the architecture & implementation detail of your model. (6%)

- I implement the GTA (Generate To Adapt: Aligning Domains using Generative Adversarial Networks) model

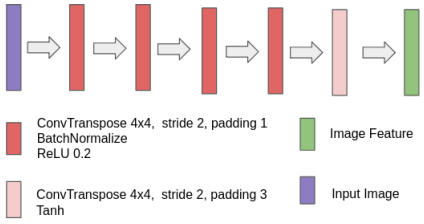

Figure 21: Generator

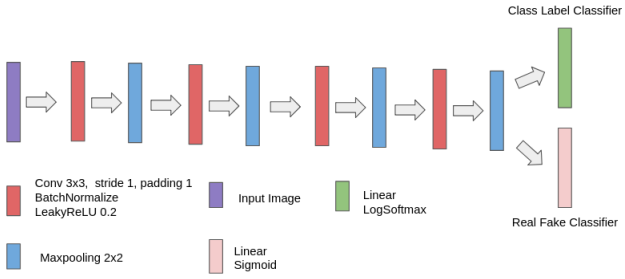

Figure 22: Discriminator

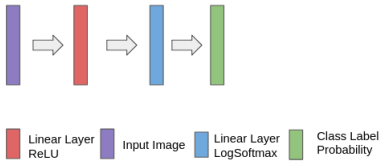

Figure 23: Image Classifier

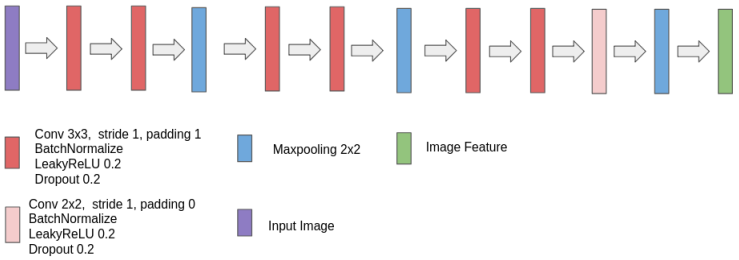

Figure 24: Feature Extracter

4.4 Discuss what you' ve observed and learned from implementing your improved UDA model. (7%)

- The GTA model is not very stable since Svhn to MnistM is very good but MnistM to Svhn is not as good as expected. The GAN module in GTA model can somehow boost the performance but might not be a stable choice.

4.5 Collaborators

- 林子淵 R07921100
- 黃孟霖 R07922170
- 張緣彩 R07922141

References

- https://arxiv.org/pdf/1704.00028.pdf
- https://arxiv.org/pdf/1611.07004v1.pdf
- https://github.com/ImagingLab/Colorizing-with-GANs
- https://blog.floydhub.com/colorizing-b-w-photos-with-neural-networks/
- https://arxiv.org/pdf/1704.00028.pdf
- https://arxiv.org/pdf/1409.1556v6.pdf
- https://reurl.cc/oDXGRg
- https://reurl.cc/L1DGW7