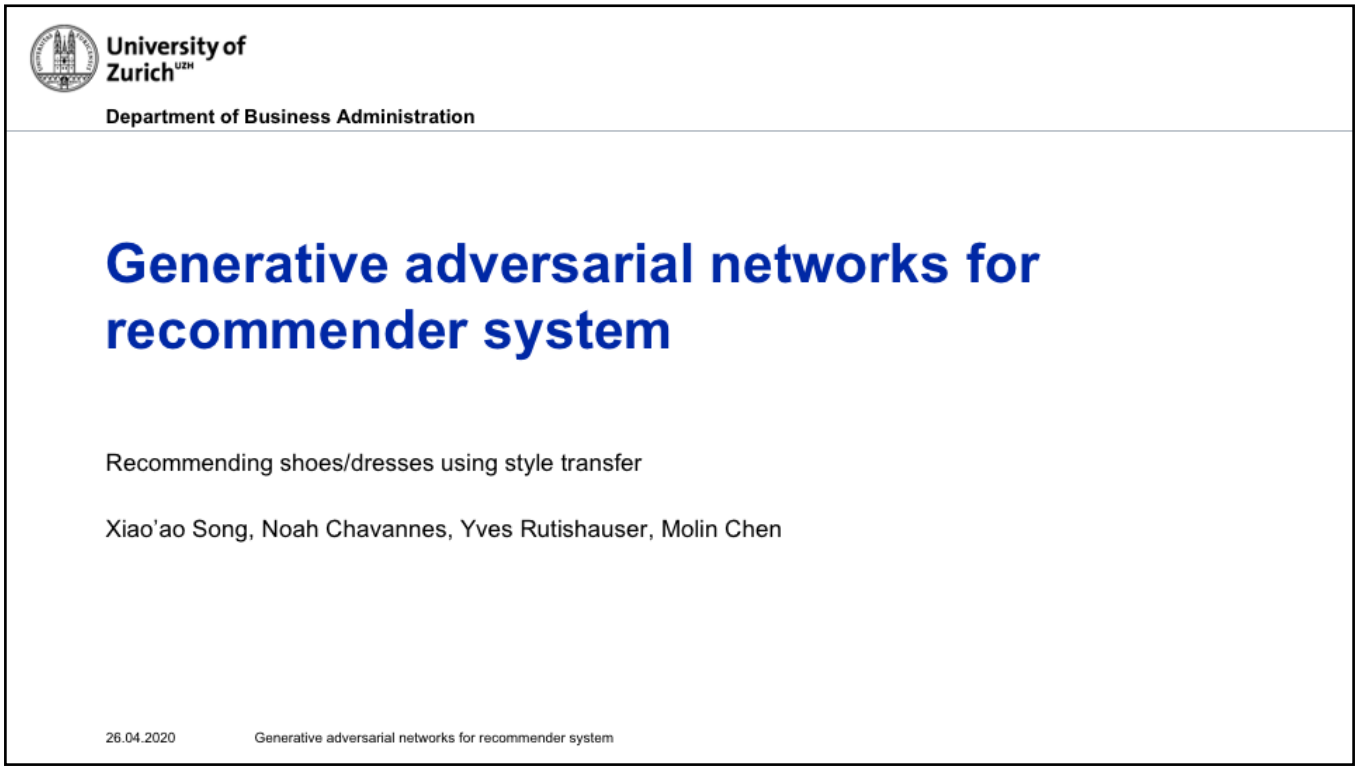


Hi, there! Welcome to our presentation. Our team member include Xiao'ao Song, Noah Chavannes, Yves Rutishauser and Molin Chen. Today, we are going to present how Generative adversarial networks can be used in recommender system. First of all, let's imagine you are Zalando and you have a customer browsing through the shoes. Then the shopper likes a pair of shoes and is going to the checkout. Before the shopper buys the shoes, you would like to show them what else you have in stock that they might like. Your product recommendation algorithm recommends a dress, so you show this to the shopper. They agree, the dress fits their style perfectly, so they throw it in the shopping cart as well! Good job!

In this presentation, we will be using GANs to generate cross-domain product recommendations. The two domains we will be working with are shoes and dresses.

University of Zurich[UZH]

Department of Business Administration

Introduction to GAN and CycleGAN

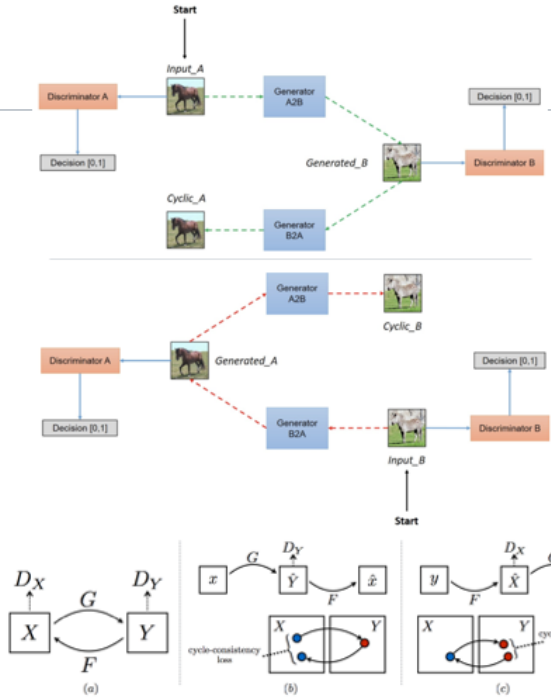
-Task:
Recommending shoes/dresses for shopper

-Input:
Unpaired training set from two domains:
- Dresses (20000 images)
- Shoes (20000 images)

-Output:
shoes/dresses image recommended by the model

-Loss:
Optimize LS loss + cycle-consistent loss + identity loss

-Network implementation difference with paper:
- 6 Residual Blocks
- Max replay buffer size is 100
- permute the training set during each epoch to increase randomization



The idea of generative adversarial network is invented by Ian Goodfellow and his colleagues in 2014. Basically GAN invovles two networks:a generator and a discriminator. The generator network is a network that generates data. In our case, the generator generates images that belong to a category; for example, shoes. The discriminator then has to distinguish between real images of shoes and the fake images of shoes that the generator created. The generator and discriminator are trained based on their successes. The generator tries to learn to generate features that fools the discriminator. The discriminator tries to learn how it can tell fakes apart from the real thing.

In our project, we want to recommend shoes/dresses to customers. To this end, we adopted CycleGAN and trained on our data set from the scratch.

A CycleGAN is a special type of GAN that is used for style-transfer. It has a few features that distinguish it from other GANs that we will outline quickly. These features are:

1. 2 sets of generators and discriminators.
   Since we are interested in making cross-domain recommendations, we need 2 sets of generator and discriminators. One for dresses and one for shoes.
2. The input to the generator are images.
3. The loss is cycle-consistent.
   The final difference is that the loss is cycle-consistent. This means, that we want the recommendations to be cyclical. If we recommend shoes X for a dress X, then it would make sense to get dress X recommended for shoe X.

In the bottom figure(a) , our model contains two mapping functions G and F, and

associated adversarial discriminators DY and DX. DY encourages G to translate X into outputs indistinguishable from domain Y , and vice versa for DX and F. (b) and (c) visually illustrate cycle-consistent loss.

The top figure shows a simplified overview of CycleGAN architecture except that we usw dresses and shoes instead of horses and zebras.

When implementing the cycleGAN network, we adopted the most of parameter setting specified in paper but only made a few changes.

-1 we use 6 residual blocks instead of 9 blocks. We believe that too much or too less residual blocks will cause overfitting and underfitting. In our experiment, 6 blocks is well-balanced.

-2 We increase the max replay buffer size from 50 to 100. The idea of this strategy is that we update the discriminators using a history of generated images rather than the ones produced by the latest generators. Increase the buffer size will therefore reduce the variance of the discriminators

-3 We also permuted the training image set during each training epoch. We did this to increase the randomization for the network and therefore can reduce overfitting issue.

The full loss function for this network include a LS loss, cycle-consistent loss and identity loss.

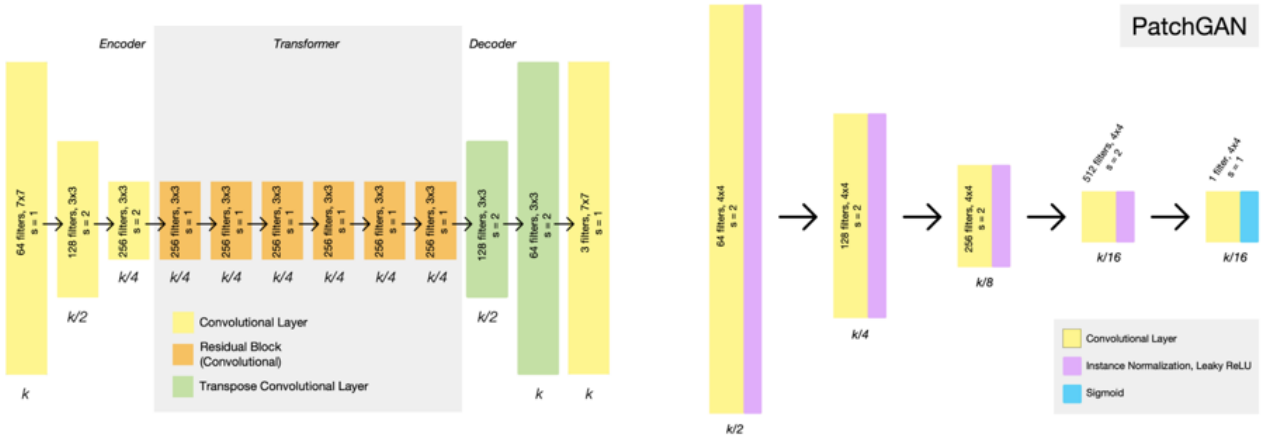
To stabilize the training, negative log likelihood objective is replaced by a least-squares loss.

In particular, the generator is also regularized by near an identity mapping when real samples of the target domain are provided as the input to the generator. In practice, the identity mapping loss helps preserve the color of the input paintings. The weight for the identity mapping loss was 0.5λ where λ was the weight for cycle consistency loss. λ is set to 10 in original setting.

University of Zurich[UZH]

Department of Business Administration

CycleGAN Architecture



Generator Architecture **Discriminator Architecture**

So far so good? Now, let's digest the network and explain the main components of it.

First, we introduce the generator architecture:

Each CycleGAN generator has three sections: an encoder, a transformer, and a decoder. The input image is fed directly into the encoder, which shrinks the representation size while increasing the number of channels. The encoder is composed of three convolution layers. The resulting activation is then passed to the transformer, a series of six residual blocks. It is then expanded again by the decoder, which uses two transpose convolutions to enlarge the representation size, and one output layer to produce the final image in RGB.

You can see the details in the left figure.

As you can see, the representation size shrinks in the encoder phase, stays constant in the transformer phase, and expands again in the decoder phase. The representation size that each layer outputs is listed below it, in terms of the input image size, k. On each layer is listed the number of filters, the size of those filters, and the stride. Each layer is followed by an instance normalization and ReLU activation, but these have been omitted for simplicity. Since the generators' architecture is fully convolutional, they can handle arbitrarily large input once trained.

Still clear now?

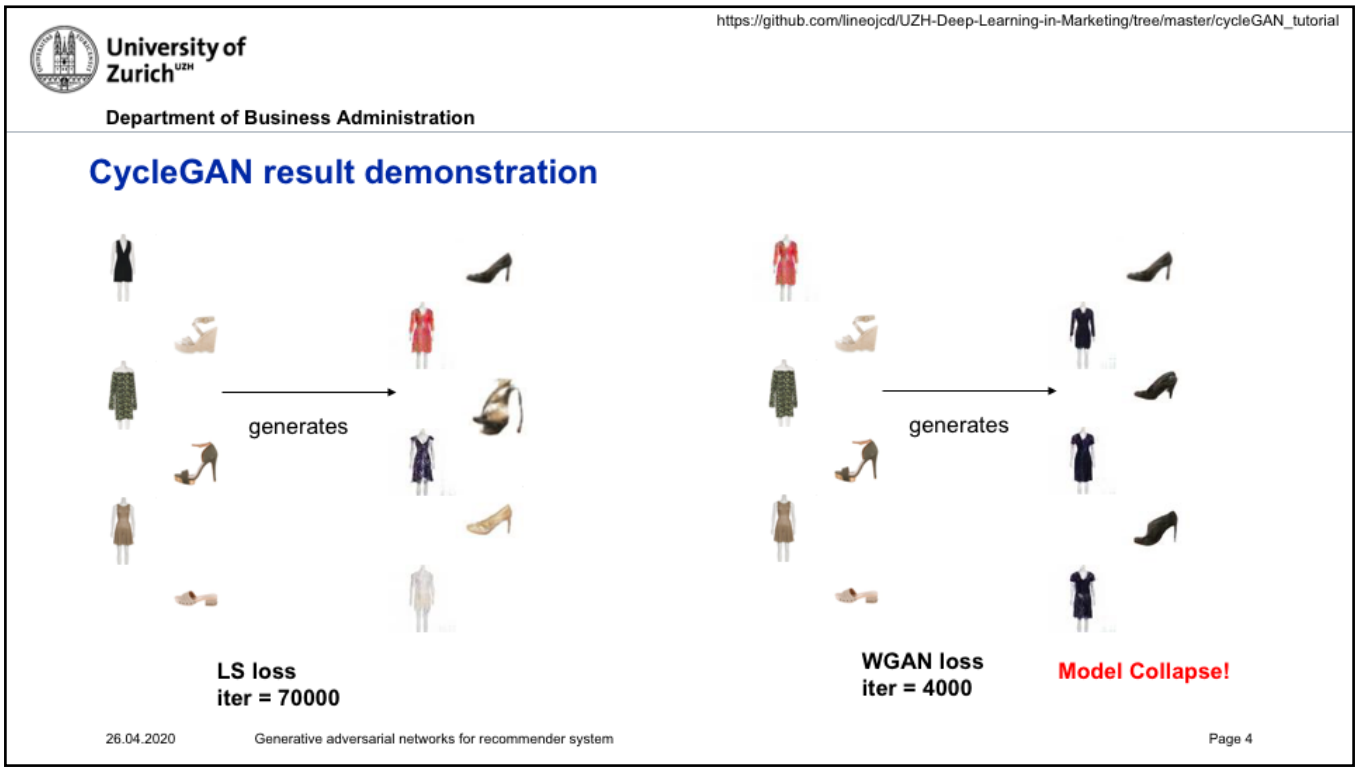
Cool! Next, let's see the discriminator architecture:

The discriminators are PatchGANs, fully convolutional neural networks that look at a "patch" of the input image, and output the probability of the patch being "real". This is both more computationally efficient than trying to look at the entire input image, and is

also more effective — it allows the discriminator to focus on more surface-level features, like texture, which is often the sort of thing being changed in an image translation task.

As you can see in the example architecture from the right figure, the PatchGAN halves the representation size and doubles the number of channels until the desired output size is reached. The representation size that each layer outputs is listed below it, in terms of the input image size, k. On each layer is listed the number of filters, the size of those filters, and the stride. In this case, it was most effective to have the PatchGAN evaluate 70x70 sized patches of the input.

Ok, this is basically enough for now, for other details, you can check on the original paper.



After understanding the concepts and network architecture of CycleGAN, we will show you some results of our models including good attempts and bad attempts.

By following the network architecture and settings we introduced previously, we trained our model and predicted on the test image set. The left figure shows some sample result. Given the image input on the left hand side, the system will recommend you the product on the right hand side. As we can see, the model is able to transfer the color of the dresses/shoes and its style.

Great!

Except for the previous settings, we also trained another model by changing the LS loss to WGAN loss.

Researchers have pointed out that the original GAN is hard to train. When an optimal discriminator produces good information for the generator to improve. But if the generator is not doing a good job yet, the gradient for the generator diminishes and the generator learns nothing. To this end, WGAN proposes a new cost function using Wasserstein distance that has a smoother gradient everywhere. WGAN learns no matter the generator is performing or not. The concepts of WGAN is a bit

complicated, we will not talk too much here, however, if you are interested, feel free to talk with us!

The right figure shows the sample result by using WGAN loss.

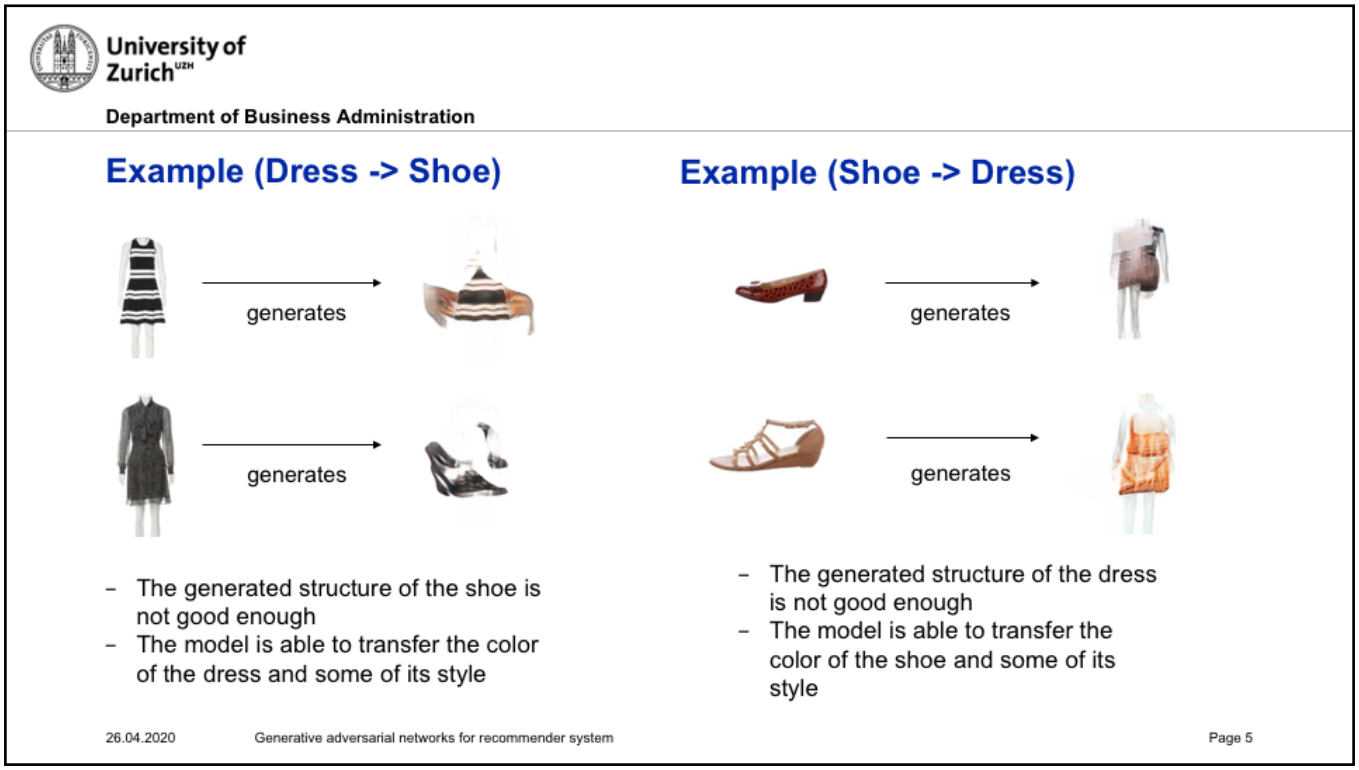
Wait! Did you notice something? It seems that no matter what input image is given, the model will always produce somewhat similar result. This phenomenon is called model collapse. Mode collapse refers to that the generator collapses which produces limited varieties of samples. This is often happened when training GAN and you need to avoid that.

After comparing the two figures, we conclude that WGAN loss does not help in our project.

However, this will not reduce the importance of WGAN loss. By using WGAN loss, the model has been converged within 4000 iterations whereas the model with LS loss still does not converge after 70000 iterations.

In the next page, we will demonstrate some sample result produced by another model: the GANILLA network.

This part is done by my teammate Noah Chavannes and Yves Rutishauser.



We used a sample of 1000 images to test the Ganilla network. We decided on this number due to the environmental limitations and the fact that the authors of the ganilla paper also used a relatively low amount of samples and produced astonishing results. Also, this allowed us to train this computationally intensive model on our local systems and execute experiments faster.

We started by training the model and manually analysing the results. We quickly found that the generator network that is responsible for generating shoes does not do a good job in transferring the structure. The style of the dresses is matched acceptably.

In general, we found that both generator networks perform better on colors that occur often (e.g. black, dark blue, beige, rose) than on more exotic colors (e.g. light blue, green).

The generator for dresses works a little bit better when it comes to transferring the structure of a shoe to a dress.

The generator is able to reproduce the mannequin and place a dress on

top. When it comes to the colors, it faces the same problems as the generator for the shoes.

University of Zurich[UZH]

Department of Business Administration

Observations

- Ganilla expects the structure of the image to stay the same
- The domains of dresses and shoes have different structures. Therefore, when mapping a dress to a shoe, the structure cannot be maintained and has to be inferred
  - Ganilla incorporates low-level information gained during downsampling through skip-connections into the upsampling process (blue line) (P.5)

GANILLA

  - This helps to preserve the structure. To improve the model in our use-case, one could try to remove the skip-connections -> Ablation Model-II
    - Due to restrictions in the infrastructure, we could not train the Ablation Model-II

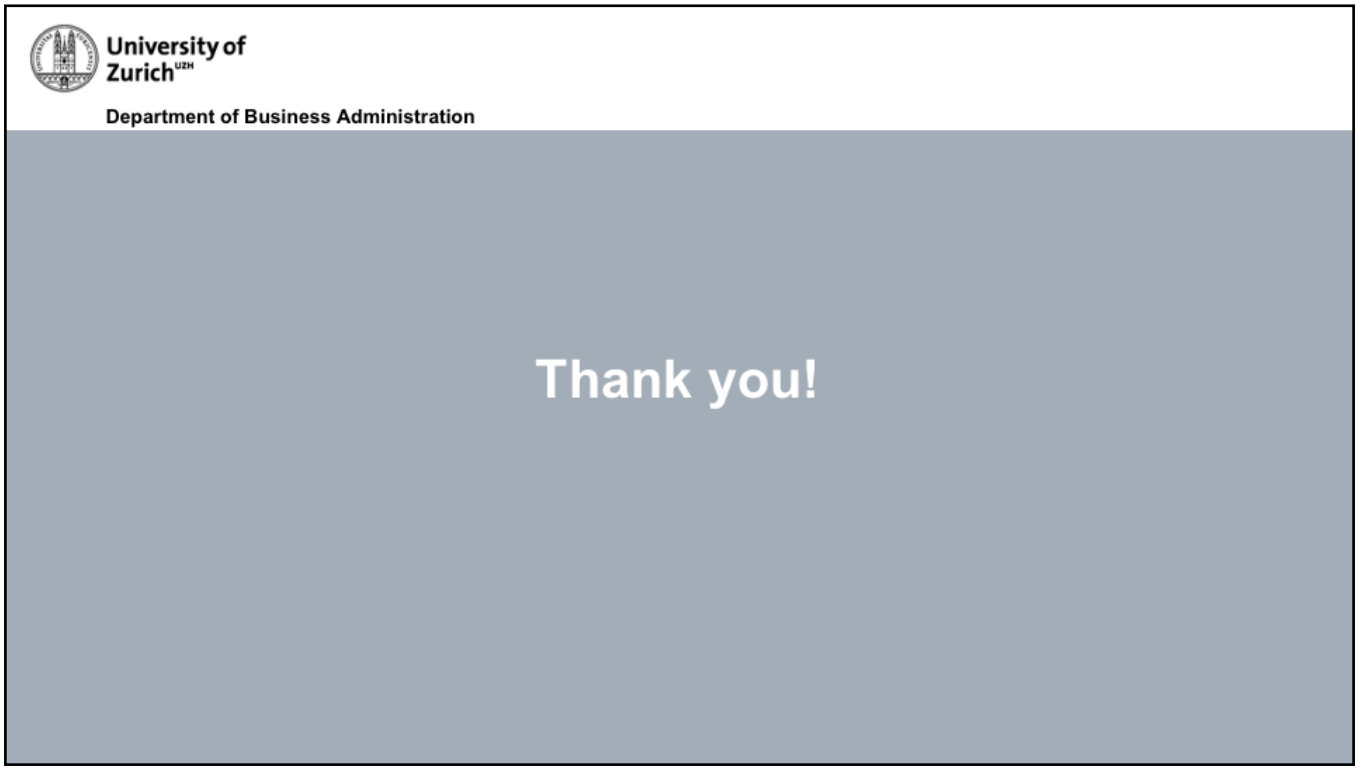
Ablation Model-II

We think that the rather poor performance of the ganilla model for the recommender system is due to its nature.

One of the main goals of ganilla is to preserve the structure of the image. When it comes to converting a dress into a shoe or the other way around, this is a rather poor ability.

This ability comes from the skip connections, that feed the outputs from previous downsampling layers to the feature pyramid network in the upsampling phase. We tried to solve this issue by reducing the weight of the skip connections or fully remove them, but the results were blurry images without any details.

We then thought the ablation model 2 would be the better fit for the recommender system, as it uses a different method to upsample images that do not include skip connections. We could not train this network due to its size. Even when reducing the batch size to 1 and reducing the number of filters used in the network.



Although the GANILLA network cannot help in this case, don’t be frustrated!! There are still some ways to improve the recommender system for example, you can work on WGAN-GP loss function or even try some other models such as Disco GAN and progressive growing GAN and etc.

Thanks for listening to our presentation! I hope it is interesting to you all and the topic could inspire you in your future research.

Key references

- Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks, Available at: https://arxiv.org/pdf/1703.10593.pdf
- Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of GANs for improved quality, stability, and variation, Available at: https://arxiv.org/pdf/1710.10196.pdf
- S. Hicsonmez, N. Samet, E. Akbas, P. Duygulu (2020). GANILLA: Generative Adversarial Networks for Image to Illustration Translation, Available at: https://arxiv.org/pdf/2002.05638.pdf

For further information, please check the key reference for more details.