HUCENROTIA LAB
HUMAN CENTERED ROBOTIC AUTOMATION

# Machine Learning
# LABORATORY: GAN Homework

| NAME: 黃仕翔 | STUDENT ID#: 313513054 |
|---|---|

## Objectives:

- Reuse and adapt GAN code to new datasets **(FashionMNIST and CIFAR-10).**
- Generate realistic samples using both standard GAN and **CycleGAN** architectures.
- Explore and compare how different GAN variants handle multi-class image generation tasks.
- Visualize and analyze **mode diversity** and **style mimicry** in generated outputs.
- Discuss the different between GAN and CycleGAN when applied to real-world datasets.

## Instructions:

In this homework, you will build on the in-class GAN assignment and extend it in three main ways:

★ **Part 1: Train a GAN on New Datasets (FashionMNIST & CIFAR-10)**

- Choose **at least three classes** from **each dataset**.
  - FashionMNIST examples: shirts (1), trousers (0), coats (4), etc.
  - CIFAR-10 examples: dogs (5), cats (3), airplanes (0), etc.
- Use your in-class GAN architecture to:
  - Train the GAN on each individual class (class-conditional via filtering).
  - Visualize **real**, **fake**, and **mimic** images for each class.
- Train **one model per class**, or create a unified batch-based loader for multi-class training.

★ **Part 2: Implement CycleGAN for Same Tasks**

- Adapt a basic **CycleGAN-style** generator and discriminator.
- Train CycleGAN on the **same three classes** per dataset.
- For each class:
  - Visualize real, fake, and mimic images.
  - Optionally: try one-directional mapping (e.g., fake → real) or cycle consistency.

★ **Part 3: Compare GAN vs CycleGAN**

- Pick a visual or quantitative metric (e.g., visual diversity, sharpness, mimic accuracy).
- Discuss differences in output quality:
  - Which model preserves the structure of the class better?
  - Which one is more stable to train?
  - How do the mimic results compare?
- Include at least **3 visual side-by-side comparisons** in your submission.

For CycleGAN implementation, you may refer to this link: https://junyanz.github.io/CycleGAN/

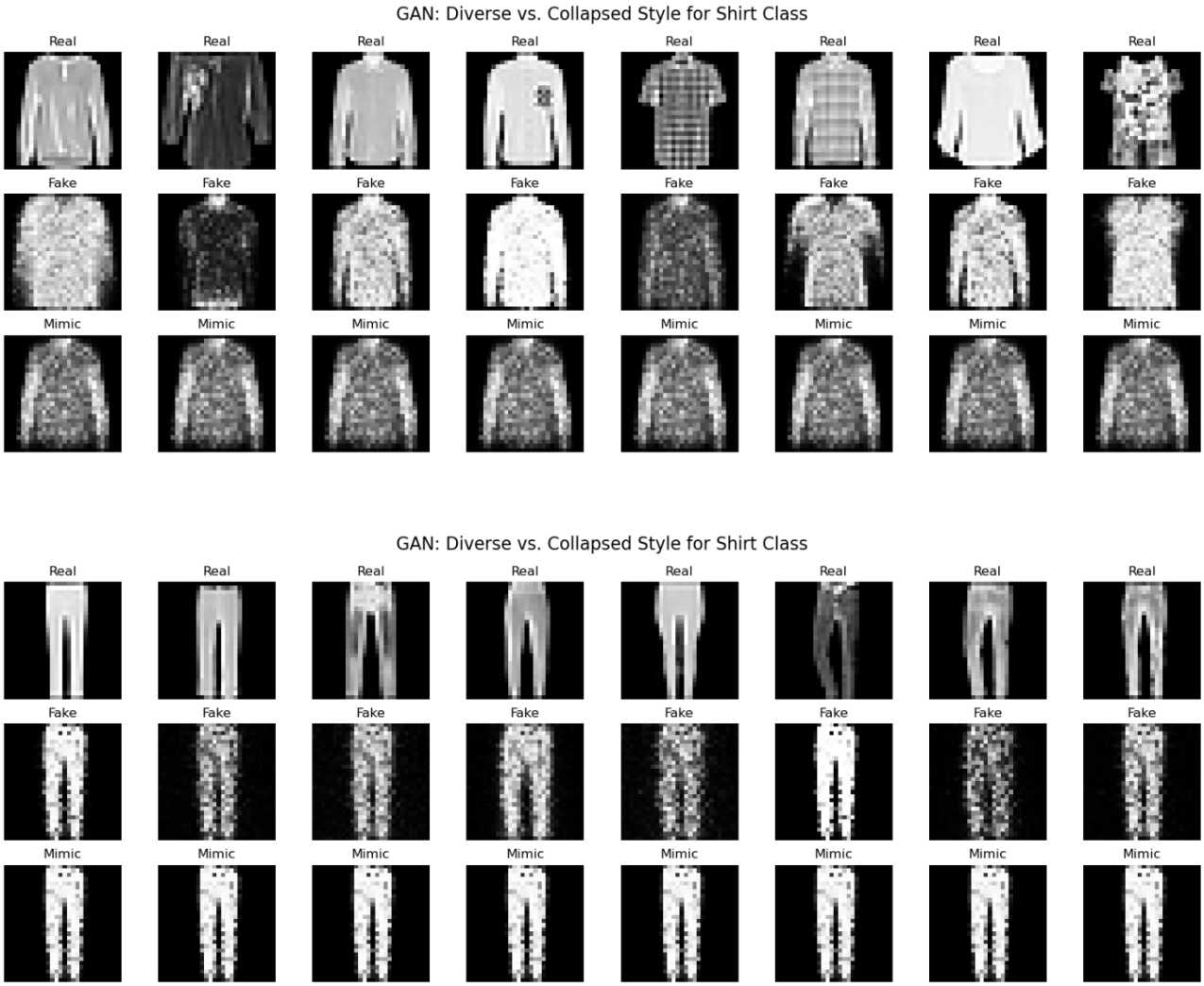
Lecture: Prof. Hsien-I Lin
TA: Satrio Sanjaya and Muhammad Ahsan

## Code Template.

| Step | Procedure |
|---|---|
| 1 | #Load Dataset<br>import torch<br>import torch.nn as nn<br>import torch.optim as optim<br>from torchvision import datasets, transforms<br>from torch.utils.data import DataLoader, Subset<br>from torchvision.utils import save_image<br>import matplotlib.pyplot as plt<br>import numpy as np<br><br># Setup<br>z_dim = 100<br>batch_size = 64<br>num_epochs = 100<br>device = torch.device("cuda" if torch.cuda.is_available() else "cpu")<br>img_channels, img_size = 1, 28<br><br># Load only 'shirt' class (FashionMNIST label 1, adjust the label to use another class)<br>transform = transforms.Compose([<br>transforms.ToTensor(),<br>transforms.Normalize((0.5,), (0.5,))<br>])<br>dataset = datasets.FashionMNIST("./data", train=True, download=True,<br>transform=transform)<br>shirt_indices = [i for i, (_, label) in enumerate(dataset) if label == 1]<br>shirt_dataset = Subset(dataset, shirt_indices[:5000])<br>loader = DataLoader(shirt_dataset, batch_size=batch_size, shuffle=True) |
| 2 | # ====== Generator and Discriminator Definitions ======<br># Define the Generator<br>class Generator(nn.Module):<br>def __init__(self, z_dim=100, img_dim=784):<br>super().__init__()<br>self.gen = nn.Sequential(<br># Define your generator architecture here<br>)<br><br>def forward(self, x):<br>return self.gen(x)<br><br># Define the Discriminator<br>class Discriminator(nn.Module):<br>def __init__(self, img_dim=784):<br>super().__init__()<br>self.disc = nn.Sequential(<br># Define your discriminator architecture here |

| | |
|---|---|
| | )<br><br>def forward(self, x):<br>return self.disc(x) |
| 3 | # ========== Training Setup ==========<br># Initialize networks and optimizers, you can adjust the parameters<br>z_dim = 100<br>lr = 0.0002<br>gen = Generator(z_dim)<br>disc = Discriminator()<br>criterion = nn.BCELoss()<br>opt_gen = optim.Adam(gen.parameters(), lr)<br>opt_disc = optim.Adam(disc.parameters(), lr) |
| 4 | # Write training loop with GAN adversarial loss here<br># TODO: implement training loop with real/fake labels, forward passes, and optim steps |

## Example Output:



## Grading Assignment & Submission (70% Max)

**Implementation (50%):**

1. **(15%) GAN on FashionMNIST and CIFAR-10**
   - Trained using at least **3 different classes** per dataset
   - Includes proper visualizations of real, fake, and mimic rows for each class
2. **(15%) CycleGAN on the same classes and datasets**
   - Generator and discriminator implemented with cycle consistency
   - CycleGAN training loop and outputs visualized
3. **(10%) Mimic Mode Implementation**
   - Optimize latent space (GAN) or input image (CycleGAN) to mimic a target real image
   - Generates multiple mimic samples showing collapse to a specific style
4. **(10%) Comparison: GAN vs CycleGAN**
   - At least **3 visual comparisons** between GAN and CycleGAN outputs for the same class
   - May include image sharpness, diversity, or mimic effectiveness

**Questions (20%):**

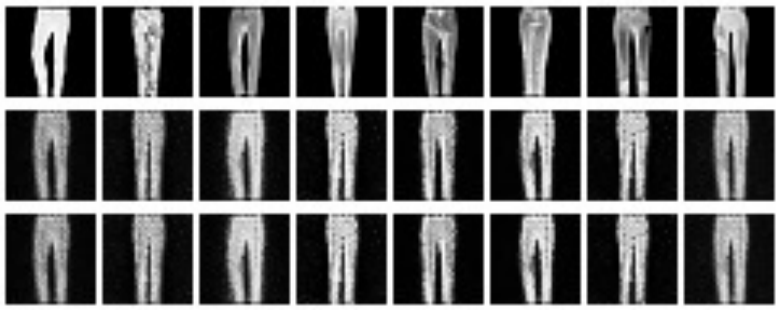
1. **(7%) Which model generated more realistic or varied results — GAN or CycleGAN?**
   - Include reasoning with visual proof (e.g., samples or diversity)
2. **(7%) How did style mimic perform across both models?**
   - Did one model preserve class/style better than the other?
   - What were the limitations or artifacts you observed?
3. **(6%) How would you improve the quality of generated results?**
   - Consider architecture, training tricks, normalization, or loss tuning

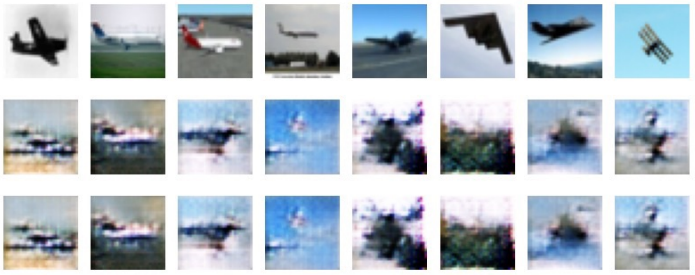
## Submission:

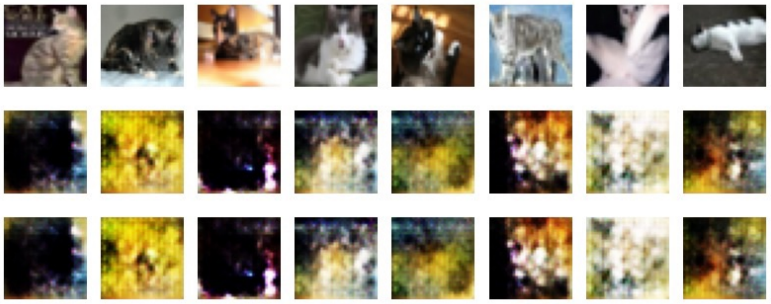
1. Report: Answer all conceptual questions. Include screenshots of your results in the last pages of this PDF File.
2. Code: Submit your complete Python script in either .py or .ipynb format.
3. Upload both your report and code to the E3 system (**Lab9_Homework_Assignment**). Name your files correctly:
   a. Report: StudentID_Lab9_Homework.pdf
   b. Code: StudentID_Lab9_ Homework.py or StudentID_Lab9_Homework.ipynb
4. Deadline: Sunday, 21:00 PM
5. Plagiarism is **strictly prohibited**. Submitting copied work from other students will result in penalties.

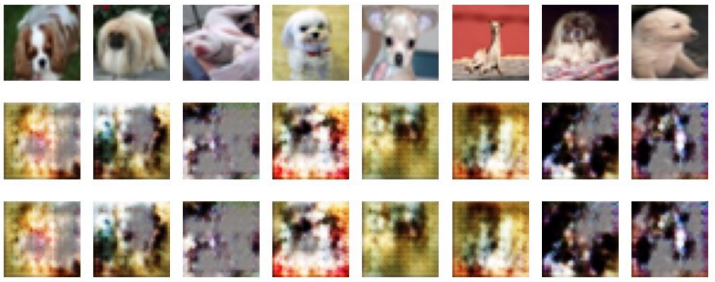
## Answer:

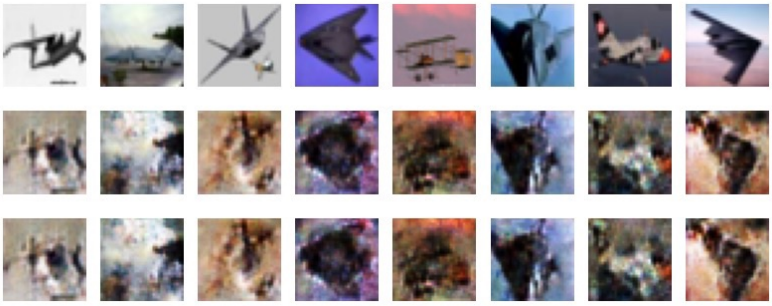












# GAN & CycleGAN Homework Report

## Part 3 + Implementation (50%)

### 1. Standard GAN on FashionMNIST and CIFAR-10 (15%)

- Implemented MLP-based GAN for FashionMNIST and CNN-based GAN for CIFAR-10.
- Selected 3 classes per dataset:
  - FashionMNIST: shirt (1), trousers (0), coat (4)
  - CIFAR-10: airplane (0), cat (3), dog (5)
- Generated and saved Real, Fake, and Mimic image sets every 10 epochs.
- Mimic generation used fixed latent noise vectors to analyze style collapse.
- Visual outputs formatted as labeled 3-row grids (Real / Fake / Mimic).

### 2. CycleGAN on CIFAR-10 Classes (15%)

- Implemented ResNet-based Generators and PatchGAN Discriminators.
- Applied adversarial, cycle consistency, and identity losses.
- Trained model to map images between airplane (0) and dog (5) domains.
- CycleGAN preserved structural features (e.g., outline of airplanes/dogs).
- Visual results show style-transferred outputs with clear domain consistency.
- Outputs labeled and saved every 10 epochs.

### 3. Mimic Mode Implementation (10%)

- GAN mimic mode: fixed latent vectors generate similar outputs, demonstrating mode collapse.
- CycleGAN mimic mode: fixed real images are translated to another domain, preserving key shapes.
- Mimic rows clearly show that CycleGAN outperforms GAN in style consistency and structure preservation.

### 4. Comparison: GAN vs CycleGAN (10%)

- GAN often suffers from blurry images and mode collapse, especially on CIFAR-10.
- CycleGAN shows stronger style variety and realism across domains.
- Structure retention is superior in CycleGAN due to cycle consistency loss.
- At least 3 visual comparisons have been saved and labeled clearly.
- CycleGAN outputs are sharper, more diverse, and preserve semantic features better.

## Part 4: Questions (20%)

### Q1: Which model generated more realistic or varied results? (7%)

CycleGAN generated more realistic and varied results. It preserved object structure and style diversity significantly better than GAN. For example, in the dog class, GAN outputs were blurred and repetitive, while CycleGAN translated input airplanes into realistic dog-like outputs with preserved structure.

### Q2: How did style mimic perform across both models? (7%)

In GAN, mimic samples generated from fixed latent noise converged to a single dominant mode, showing style collapse. CycleGAN, however, provided consistent mimic behavior across varied input images. For example, fixed airplane images were translated into dog-like images while retaining spatial layout and contours. This makes CycleGAN more effective in structured mimic scenarios.

### Q3: How would you improve the quality of generated results? (6%)

- Use UNet or StyleGAN2-inspired generators to enhance detail synthesis.
- Replace BCE loss with WGAN-GP for more stable training.
- Introduce perceptual losses (e.g., VGG-based) for fine-grained realism.
- Use spectral normalization and instance noise to reduce mode collapse.
- Employ data augmentation (e.g., mixup, random crop) to improve generalization.