Homework #3

Deep Learning for Computer Vision

NTU, Fall 2020

109/11/10

109/12/1 (Tue.) 02:00 AM due

Outline

- Problems & Grading
- Training Tips
- Dataset
- Rules

Problems – Overview

- **Problem 1**: VAE (30%) [Face dataset]
- **Problem 2**: GAN (20%) [Face dataset]
- **Problem 3**: DANN (35%) [Digits dataset]
- **Problem 4**: Improved UDA model (35%) [Digits dataset]

Please refer to “Dataset” section for more details about face/digits datasets.

Problem 1: VAE (30%)

A variational autoencoder (VAE) is a type of neural network that learns to reproduce its input, and also map data to latent space. it is worth noting that a VAE model can generate samples by sampling from the latent space.

In this problem, you will implement a VAE model and train it on a face dataset. Also, you will conduct some experiments to analyze your VAE model. Please follow problems below:

1. Build your VAE encoder and decoder model and print the architecture [fig1_1.png] (Please use "print(model)" in PyTorch directly). Then, train your model on the face dataset and describe implementation details of your model. (Include but not limited to training epochs, learning rate schedule, data augmentation and optimizer) (5%)

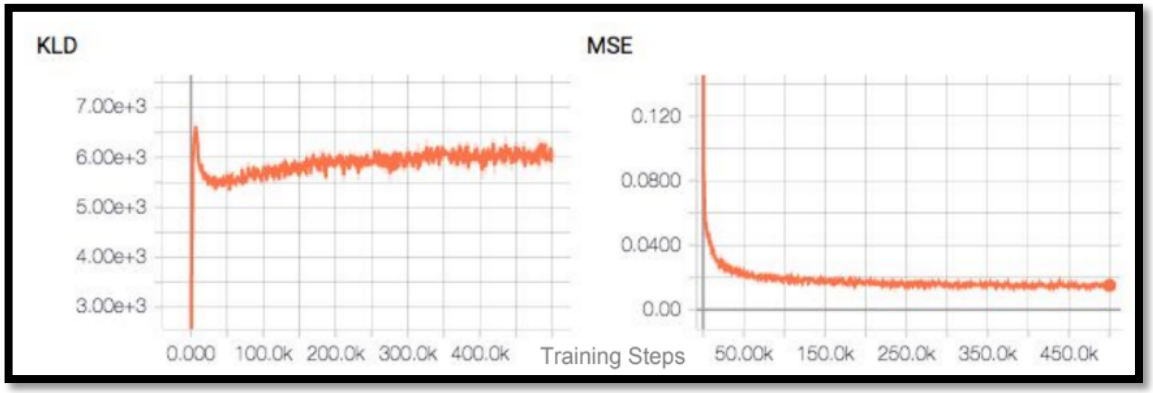
```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
```

Example for fig1_1.png

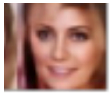
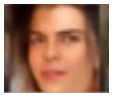
Problem 1: VAE (30%) (cont'd)

2. Please plot the learning curve (reconstruction loss and KL divergence) of your model. [fig1_2.png] (5%)

3. Please random choose 10 testing images and get the reconstructed images from your VAE model. Plot 10 testing images and their reconstructed results (reconstructed images and MSE) from your model. [Table 1_3] (5%)



Example for fig1_2.jpg

Testing Image		
Recon. Image		
MSE	15.22	23.1

...

Example for Table 1_3

Problem 1: VAE (30%) (cont'd)

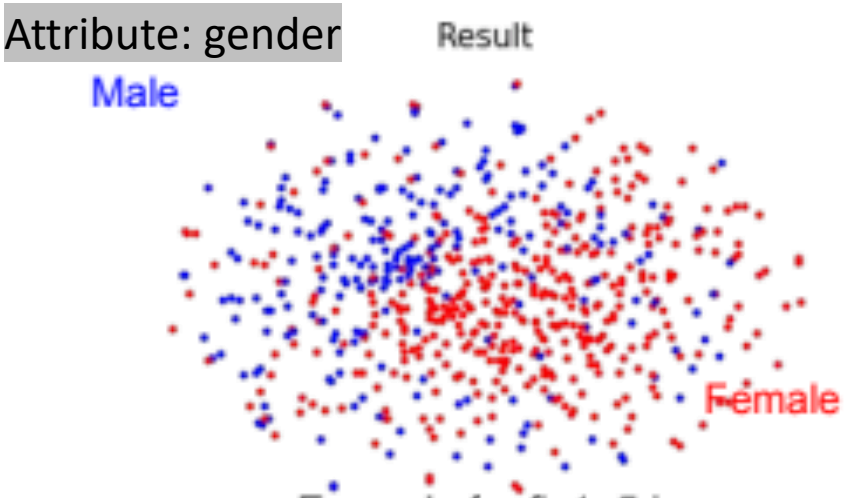
4. Now, we utilize decoder in VAE model to randomly generate images by sampling latent vectors from an Normal distribution. Please Plot 32 random generated images from your model. [fig1_4.jpg] (5%)

You should **fix the random seed** in your program such that the [fig1_4.jpg] produced by your script is exactly the same as the one in your report.

Example for fig1_4.jpg

Problem 1: VAE (30%) (cont'd)

5. To analyze the latent space of your VAE model, please visualize the latent space by mapping the latent vectors of the test images to 2D space (by tSNE) and color them with respect to an attribute (e.g., gender and hair color) of your choice. (5%) (An example is shown below.)
6. Discuss what you've observed and learned from implementing VAE. (5%)



Example for fig1_5.jpg

Problem 2: GAN (20%)

A generative adversarial network (GAN) is a deep learning method in which two neural networks (Generator and discriminator) compete with each other to become more accurate in their predictions.

In this problem, you will implement a GAN model and train it on a face dataset. Also, you will conduct some experiments to analyze your GAN model. Please follow problems below:

1. Build your generator and discriminator in GAN model and print the architecture [fig1_1.png] (Please use "print(model)" in PyTorch directly). Then, train your model on the face dataset and describe implementation details of your model. (Include but not limited to training epochs, learning rate schedule, data augmentation and optimizer) (5%)

Problem 2: GAN (20%) (cont'd)

2. Now, we can use the Generator to randomly generate images. Please samples 32 noise vectors from Normal distribution and input them into your Generator. Plot 32 random images generated from your model. [fig2_2.jpg] (5%)
3. Discuss what you've observed and learned from implementing GAN. (5%)
4. Compare the difference between image generated by VAE and GAN, discuss what you've observed. (5%)

You should **fix the random seed** in your program such that the [fig2_2.jpg] produced by your script is exactly the same as the one in your report.

Example for fig2_2.jpg

Problem 3: DANN (35%)

In this problem, you need to implement DANN on digits datasets (USPS, MNIST-M and SVHN) and consider the following 3 scenarios:
(Source domain → Target domain)

USPS → MNIST-M, MNIST-M → SVHN, SVHN → USPS

Note that during training DANN, we utilize the **images and labels** of **source** domain, and **only images (without labels)** of **target** domain.

1. Compute the **accuracy** on **target** domain, while the model is trained on **source** domain only. (lower bound) (3%)
   - Use source images and labels for training
2. Compute the **accuracy** on **target** domain, while the model is trained on **source** **and** **target** domain. (domain adaptation) (3+7%)
   - Use source images and labels + target images for training
3. Compute the **accuracy** on **target** domain, while the model is trained on **target** domain only. (upper bound) (3%)
   - Use target images and labels for training

Problem 3: DANN (35%) (cont'd)

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%)
   - Note that you need to plot the figures of all the three scenarios, so you would need to plot 6 figures in total in this sub-problem.
5. Describe the architecture & implementation detail of your model. (6%)
6. Discuss what you've observed and learned from implementing DANN. (7%)

Problem 4: Improved UDA model (35%)

In this problem, you need to implement an improved model on digits datasets (USPS, MNIST-M and SVHN) and consider the following 3 scenarios: (Source domain → Target domain)

USPS → MNIST-M, MNIST-M → SVHN, SVHN → USPS

Note that during training your improved UDA model, we utilize the **images and labels** of **source** domain, and **only images (without labels)** of **target** domain.

1. Compute the **accuracy** on **target** domain, while the model is trained on **source** **and** **target** domain. (domain adaptation) (6+10%)
   - Use source images and labels + target images for training

You should implement UDA methods other than DANN for your improved UDA model. Some methods are provided at the end of Problem 4 description (page 14).

Problem 4: Improved UDA model (35%) (cont'd)

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%)
   - Note that you need to plot the figures of all the three scenarios, so you would need to plot 6 figures in total in this sub-problem.
3. Describe the architecture & implementation detail of your model. (6%)
4. Discuss what you've observed and learned from implementing your improved UDA model. (7%)

Problem 4: Improved UDA model (35%) (cont'd)

Related UDA papers

- Adversarial Discriminative Domain Adaptation. CVPR 2017
- Domain Separation Network. NIPS 2016
- Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. CVPR 2017
- Generate To Adapt: Aligning Domains using Generative Adversarial Networks. CVPR 2018
- From source to target and back: Symmetric Bi-Directional Adaptive GAN. CVPR 2018

You may select any one of the papers above, or search for other related papers, or design a method by yourself to implement your improved model.

Grading – Problem 3 & 4

- If your accuracy of Problem 3-2 does not pass the baseline accuracy, you only get 3 points in this sub-problem.
  - Pass the baseline in one scenario: 3+3 points, two: 3+5 points, three: 3+7 points
- Baseline score for DANN:
  - USPS → MNIST-M: **40%**
  - MNIST-M → SVHN: **40%**
  - SVHN → USPS: **40%**
- Your accuracy of problem 4-1 should **surpass** the accuracy reported in problem 3-2. If not, you only get 6 points in this sub-problem.
  - Improve in one scenario: 6+4 points, two: 6+7 points, three: 6+10 points

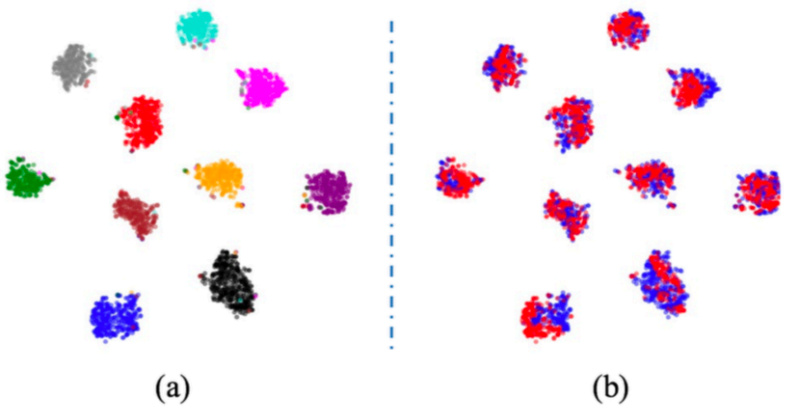
Grading – Problem 3 & 4

- Expected results on report:

  Accuracy: e.g.,

	USPS → MNIST-M	MNIST-M → SVHN	SVHN → USPS
Trained on source			
Adaptation (DANN/Improved)			
Trained on target			

t-SNE: e.g.,

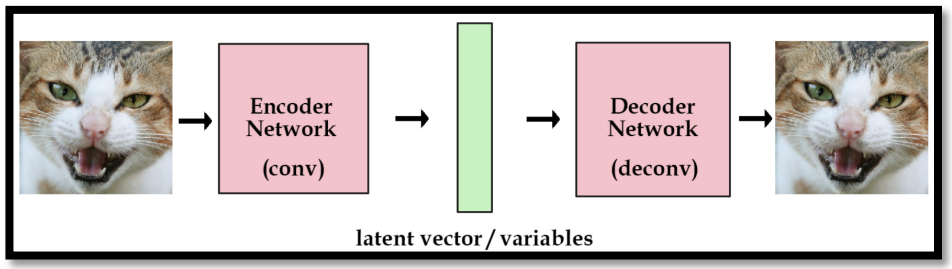


Outline

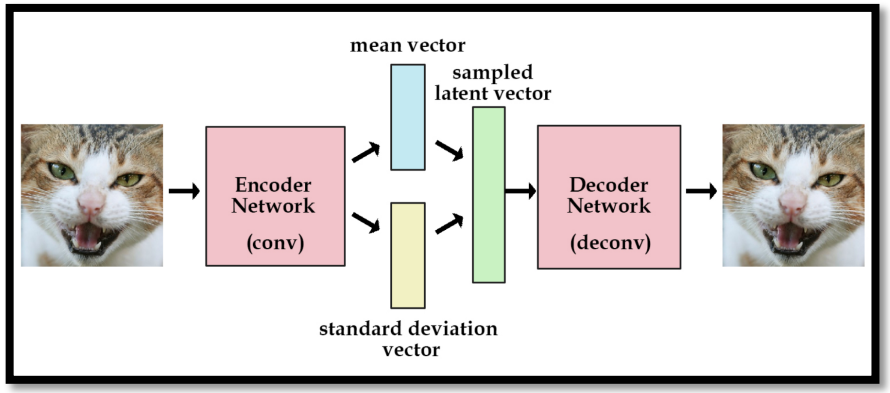
- Problems & Grading
- Training Tips
- Dataset
- Rules

Training Tips - VAE

- The VAE contains two modules:
  - **Encoder**: Learn to predict the mean and std of the input images in the latent space.
  - **Decoder**: Reconstruct an image from a latent vector sampled from the latent space.



AutoEncoder



VAE

Training Tips - VAE (cont'd)

- Objective function of VAE:

$$L_{vae} = L_{reconstruction} + \lambda_{KL} L_{KL}$$

  - λ_{KL} controls the importance of KL.
- $L_{reconstruction}$ represents the mean squared error (MSE) of the input image and reconstructed image.
- L_{KL} denotes the KL divergence loss between generated latent vector and N-dimension gaussian noise.

To be more specific…

$$L_{KL} = -\frac{1}{2}(1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

In practice, we usually let encoder output mean μ and log variance $logvar$

$$L_{KL} = -\frac{1}{2}(1 + logvar - \mu^2 - e^{logvar})$$

what your code should look like

Training Tips - VAE (cont'd)

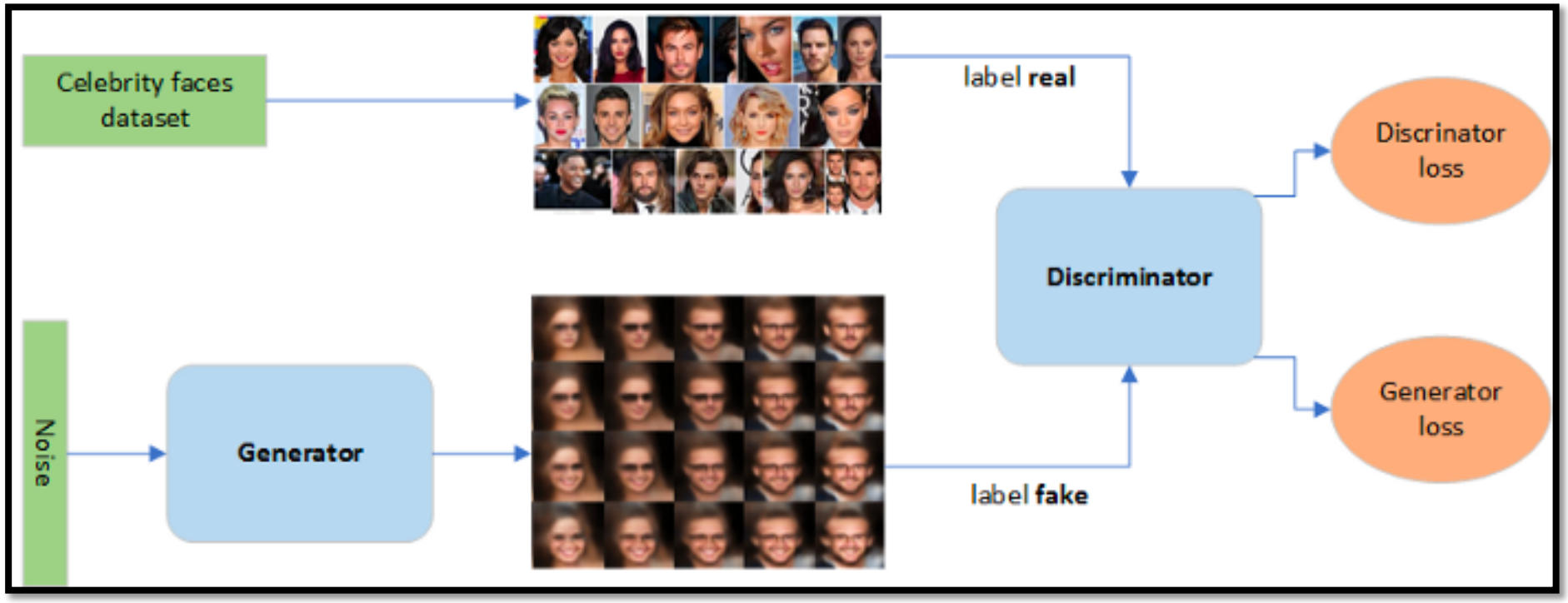
If you still have no idea how to begin, here are some instructions

- Try to implement a normal AutoEncoder and make sure it works first, then modify it to VAE
- Use (pixel-wise averaged) Mean Squared Error as reconstruction loss
- For the choice of latent space dimensional, you may first pick 512 or 1024 temporarily
- λ_{KL} is the weight between "Reconstruction quality" and "Generated image quality"
- You may select λ_{KL} = 1e-5 first and adjust it according to the image quality you received.
- If MSE is not less than 0.05 after first 20k steps , it's probably never going to be. Start over.

VAE is relatively friendly generative model in terms of training and tuning, you should finish it ASAP.

Training Tips - GAN

- Architecture of GAN



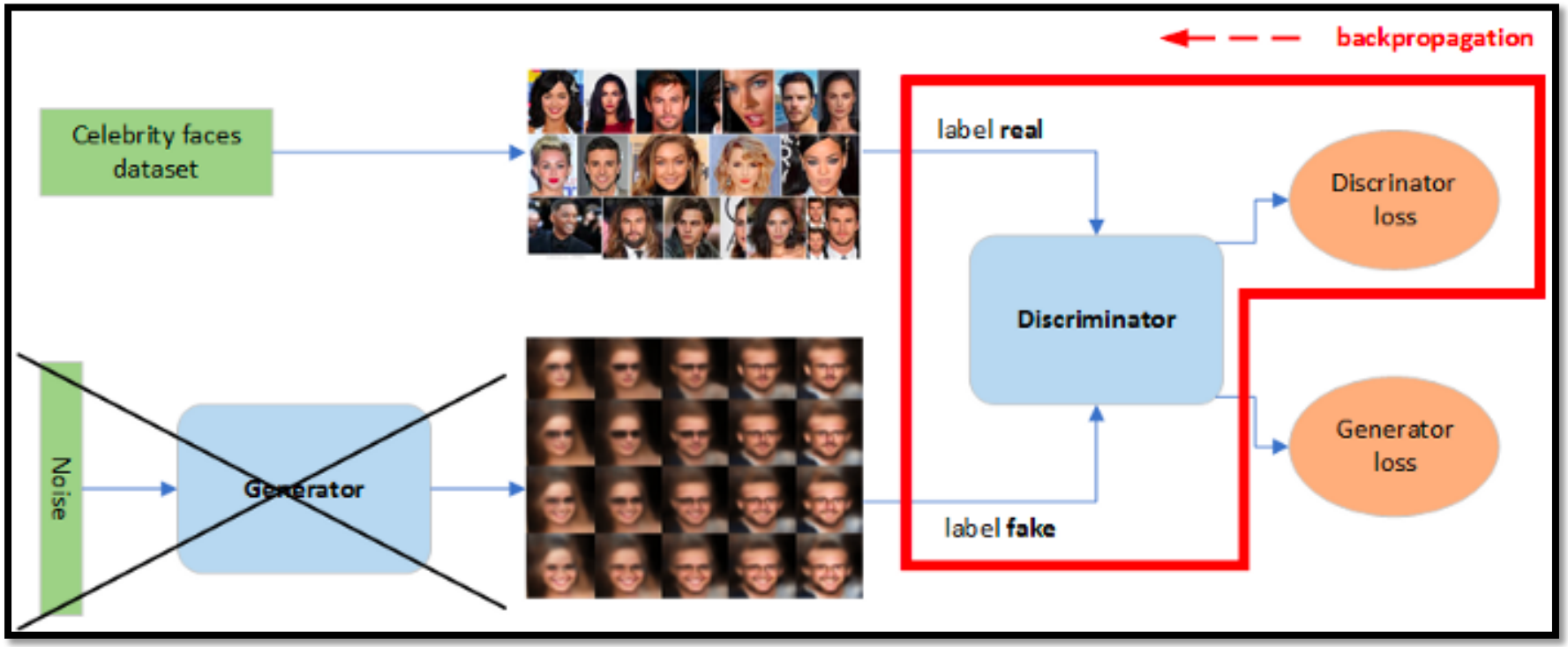
Reference: GAN for dummies

Training Tips - GAN (cont'd)

- Training

  (1) Discriminator

  - The generator's weights are fixed. Only the discriminator's weights are updated. Both real and fake image data are observed during training. The discriminator learns to detect fake image inputs.



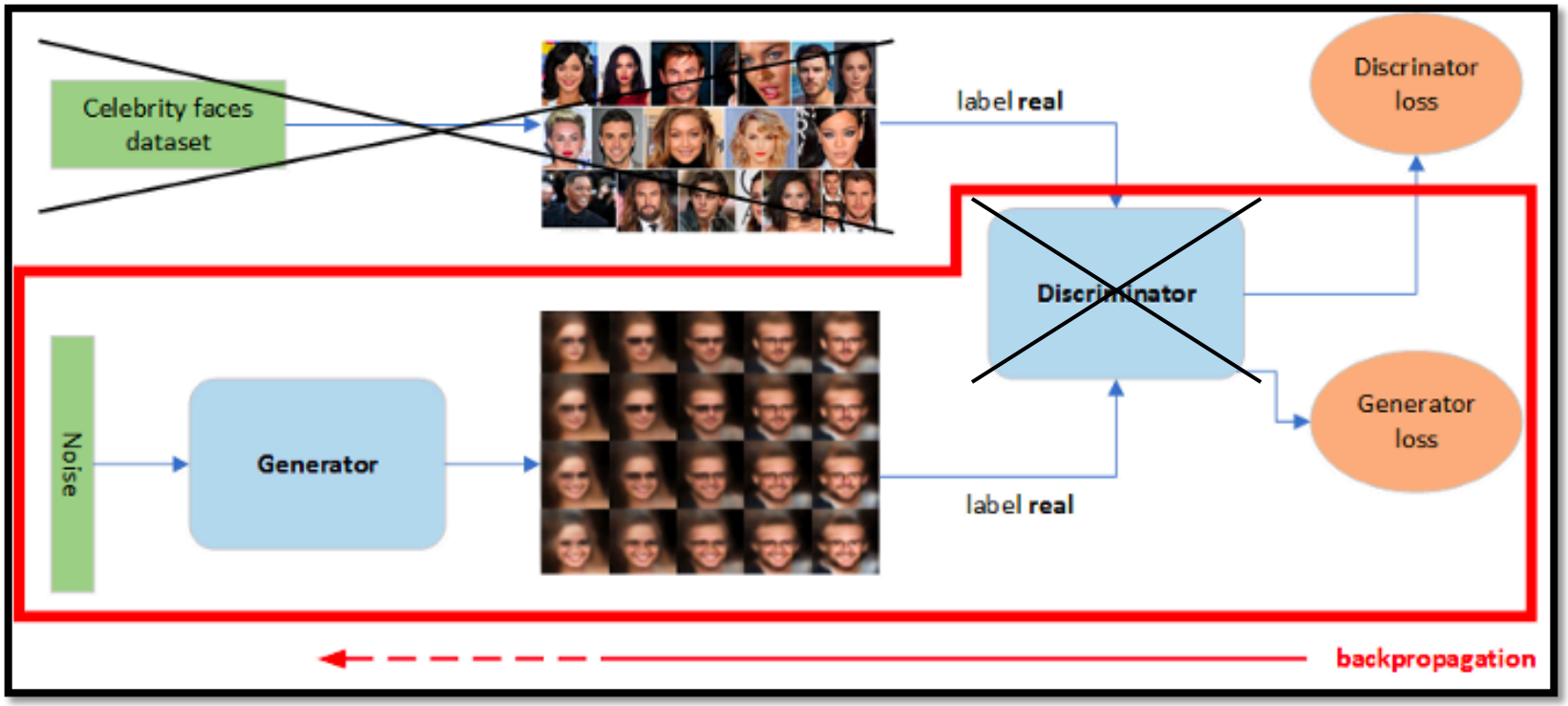
Click the link below for more details:
Reference: GAN for dummies

Training Tips - GAN (cont'd)

- Training

  (2) Generator

  - The discriminator's weights are fixed. Only the generator's weights are updated. The generator learns to fool the discriminator.



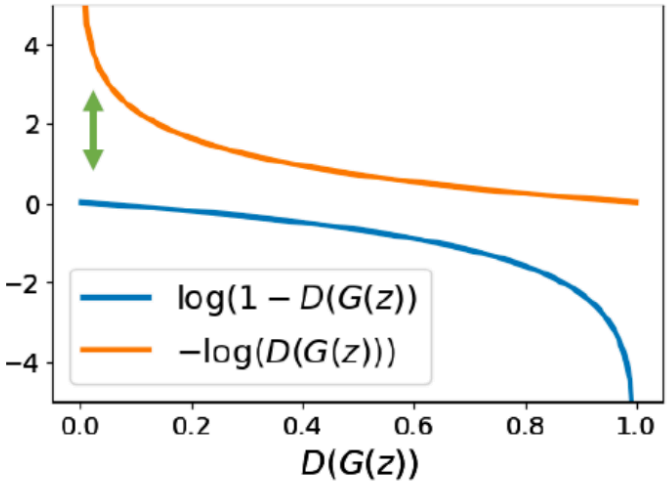
Click the link below for more details:
Reference: GAN for dummies

Training Tips - GAN Loss function (cont'd)

Potential Problem

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

- At start of training, G is not OK and D easily tells apart real/fake data (i.e., D(G(z)) close to 0)
- Solution:
  - Instead of training G to minimize log(1-D(z)) in the beginning, we train G to minimize -log(D(G(z)).
  - With strong gradients from G, we start the training of the above min-max game.



Training Tips - GAN (cont'd)

- Architecture Guideline proposed by Radford et al. in DCGAN

> Architecture guidelines for stable Deep Convolutional GANs
>
> - Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
> - Use batchnorm in both the generator and the discriminator.
> - Remove fully connected hidden layers for deeper architectures.
> - Use ReLU activation in generator for all layers except for the output, which uses Tanh.
> - Use LeakyReLU activation in the discriminator for all layers.

Well known tips and tricks published on GitHub.
It is suggested that you take a look before you start training.

TA's experience and hints

- Surveying related papers and using similar architectures may help.
- Trace the accuracy of discriminator network to see if G_{net} and D_{net} performance matches.
- Improved GAN algorithm is harder to implement but easier to train (e.g. WGAN, WGAN-GP)
- Use your VAE's decoder/encoder as the generator/discriminator prototype.

GAN is often difficult to train and tune, starting this part early may help a lot.

Outline

- Problems & Grading
- Training Tips
- Dataset
- Rules

Dataset – Generation

A subset of human face dataset CelebA

- Images are cropped and downscaled to 64x64
- 40000 training samples (about 21% of complete CelebA)
- 13 out for 40 attributes are provided for feature disentanglement experiments

Dataset – Generation

Format

data/

- face/
  - train/ # 40000 images for training (00000.png ~ 39999.png)
  - train.csv # training file attributes
- digits/

Header in first row, <img_filename>, <attribut_1>, <attribut_2>, …, <attribut_13> afterwards

```
image_name,Bangs,Big_Lips,Black_Hair,Blond_Hair,Brown_Hair,Heavy_Ma
keup,High_Cheekbones,Male,Mouth_Slightly_Open,Smiling,Straight_Hair
,Wavy_Hair,Wearing_Lipstick
00000.png,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,1.0,1.0,0.0,0.0,0.0
00001.png,0.0,0.0,0.0,0.0,0.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,1.0
00002.png,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,1.0,0.0,0.0,0.0
00003.png,0.0,0.0,1.0,0.0,0.0,0.0,1.0,1.0,1.0,1.0,1.0,0.0,0.0
...
```

Dataset – Unsupervised Domain Adaptation

Format

```
data/
└── digits/
    ├── usps/
    │   ├── train/        # images for training (*.png)
    │   ├── train.csv     # labels for training (0, 1, 2, …, 9)
    │   ├── test/         # images for testing (*.png)
    │   └── test.csv      # labels for testing (0, 1, 2, …, 9)
    ├── svhn/
    │       ...
    └── mnistm/
            ...
└── face/
```

Dataset – Unsupervised Domain Adaptation

- USPS Dataset
  - # of data: 7,291 / 2,007 (testing)
  - # of classes: **10** (0~9)
  - Image size: **28 * 28 * 1**

- MNIST-M Dataset
  - # of data: 60,000 / 10,000 (testing)
  - # of classes: **10** (0~9)
  - Generated from MNIST
  - A subset of MNIST. The digit images are normalized (and centered) in size **28 * 28 * 3** pixels.

Dataset – Unsupervised Domain Adaptation

- SVHN Dataset
  - # of data: 73,257 / 26,032 (testing)
  - # of classes: **10** (0~9)
  - Real-world image dataset for developing machine learning.
  - MNIST-like (size: **28 * 28 * 3**) images centered around a single character

You need to resize the images of the above three datasets to **28 * 28 * 3 pixels!**

Outline

- Task Description
- Dataset
- Grading
- Rules

Rules – Deadline & Policy

- Report and source code deadline: **109/12/1 (Tue.) 02:00 AM (GMT+8)**
- Late policy: Up to **3** free late days in a semester. After that, late homework will be deducted 30% each day.
- Taking any unfair advantages over other class members (or letting anyone do so) is strictly prohibited. Violating university policy would result in F for this course.
- Students are encouraged to discuss the homework assignments, but you must complete the assignment by yourself. TA will compare the similarity of everyone's homework. Any form of cheating or plagiarism will not be tolerated, which will also result in F for students with such misconduct.

Rules – Deadline & Policy

- Searching for online materials or discussing with fellow classmates are highly encouraged. However, you must provide the code or solution by yourself.
- Please specify, if any, the references for any parts of your HW solution in your report (e.g., the name and student ID of your collaborators and/or the Internet URL you consult with). If you complete the assignment all by yourself, you must also specify “no collaborators”.

Rules – Deadline & Policy

- We will adopt relative grading for Problems 1 & 2. That is, your scores will depend on the quality of your report compared to others'.
- For Problems 3 & 4, do not use the testing data to select your models. You can split the training data into training / validation sets and use the validation set to select models.
- Using external dataset is forbidden for this homework.

Rules – Submission

- Click the following link and sign in to your GitHub account to get your submission repository:

  https://classroom.github.com/a/CuUOijqP

- After getting your GitHub repository (which should be named "hw3-<username>"), be sure to **read the README carefully** before starting your work.
- **By default, we will only grade your last submission before the deadline (NOT your last submission). Please e-mail the TAs if you'd like to submit another version of your repository and let us know which commit to grade.**
- **We will clone the main branch of your repository.**

Rules – Submission

- Your GitHub repository should include the following files:
  - hw3_<studentID>.pdf
  - hw3_p1.sh (for Problem 1)
  - hw3_p2.sh (for Problem 2)
  - hw3_p3.sh (for Problem 3)
  - hw3_p4.sh (for Problem 4)
  - your python files (e.g., training code & testing code)
  - your model files (can be loaded by your python file)
- **For more information, please check the README.md in your repository.**
- **Don't upload your dataset.**
- If any of the file format is wrong, you will get zero point.

Rules – Submission

- If your model is larger than GitHub's maximum capacity (100MB), you can upload your model to another cloud service (e.g., Dropbox). However, your script file should be able to download the model **automatically**.
  - Dropbox tutorial: https://drive.google.com/file/d/1XOz69Mgxo67IZNQWnRSjT2eZAZtpUAgZ/view
- Do not delete your trained model before the TAs disclose your homework score and before you make sure that your score is correct.
- Use the **wget** command in your script to download you model files. Do not use the curl command.

Rules – Bash Script

- TA will run your code as shown below:
  - bash ./hw3_p1.sh $1
  - bash ./hw3_p2.sh $1
    - **$1**: the path to your output generated images (Problem 1-4 and 2-2) (e.g. ***hw3/VAE/fig1_4.png*** or ***hw3/GAN/fig2_2.png***).
  - bash ./hw3_p3.sh $1 $2 $3
  - bash ./hw3_p4.sh $1 $2 $3
    - **$1**: directory of testing images in the target domain (e.g. ***hw3_data/digits/mnistm/test***).
    - **$2**: a string that indicates the name of the target domain, which will be either ***mnistm***, ***usps*** or ***svhn***.
    - **$3**: the path to your output prediction file (e.g. ***hw3_data/digits/mnistm/test_pred.csv***).
- Note that you should **NOT** hard code any path in your file or script.
- Your testing code have to be finished in **10 mins.**

Rules – Bash Script

- You must **not** use commands such as rm, **sudo, CUDA_VISIBLE_DEVICES**, cp, mv, mkdir, cd, pip or other commands to change the Linux environment.
- In your submitted script, please use the command **python3** to execute your testing python files.
  - For example: python3 test.py < --img_dir $1> < --save_dir $2>
- We will execute your code on **Linux** system, so try to make sure your code can be executed on Linux system before submitting your homework.

Rules - Packages

- python: 3.6
- numpy: 1.18.1
- pytorch: 1.4.0
- torchvision: 0.5.0
- cudatoolkit: 10.1
- scikit-learn: 0.21.3
- pandas: 1.1.3
- matplotlib: 3.3.2
- pillow: 8.0.0
- imageio: 2.9.0
- scipy: 1.2.1
- and other standard python packages
- **E-mail or ask TA first if you want to import other packages.**

Rules - Packages

- Do not use imshow() or show() in your code or your code will crash.
- Use **os.path.join** to deal with path as often as possible.

Rules - Penalty

For **problem 1-4, 2-2, 3-2 and 4-1**:

- If we can not reproduce your generated image/accuracy on the testing set, you will receive a 50% penalty in the sub-problem.
- If we can not execute your code, we will give you a chance to make minor modifications to your code. After you modify your code,
  - If we can execute your code and reproduce your results on the testing set, you will still receive a 30% penalty in the sub-problem.
  - If we can execute your code but cannot reproduce your results on the testing set, you will receive a 50% penalty in the sub-problem.
  - If we still cannot execute your code, you will get 0 in the sub-problem.

Reminder

- Please start working on this homework as early as possible.
- The training may take a few hours on a GPU or days on CPUs.
- Please read and follow the HW rules carefully.
- If not sure, please ask your TAs!

How to find help

- Google!
- Use TA hours (please check course website for time/location)
- Post your question under HW3 Q&A section in FB group
- Contact TAs by e-mail: ntudlcv@gmail.com

DOs and DONTs for the TAs (& Instructor)

- Do NOT send private messages to TAs via Facebook.
  - TAs are happy to help, but they are not your tutors 24/7.
- TAs will NOT debug for you, including addressing coding, environmental, library dependency problems.
- TAs do NOT answer questions not related to the course.
- If you cannot make the TA hours, please email the TAs to schedule an appointment instead of stopping by the lab directly.