Programming Homework 3

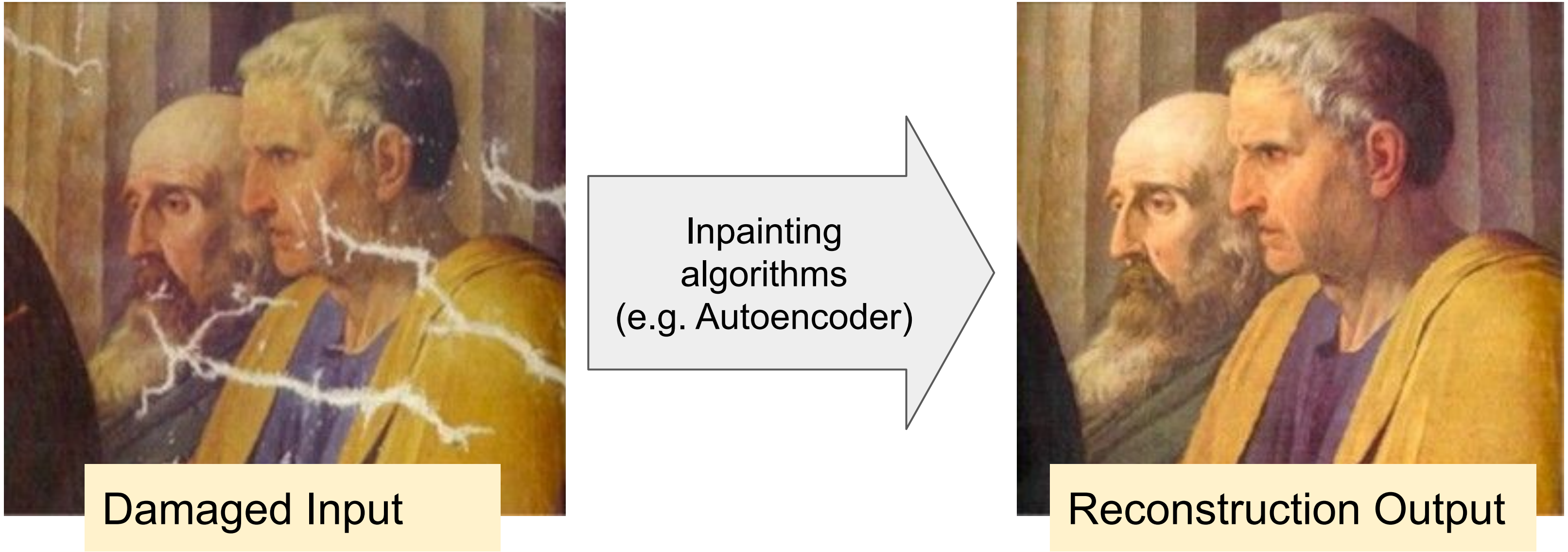
Image Inpainting using Linear Autoencoder

TA: Timmy S. T. Wan 萬世澤

Deadline: 2020/12/21 (Mon.) 23:59

Image inpainting

Modify an image in an undetectable form. *“SIGGRAPH 2000, Image Inpainting, Guillermo Sapiro”*



Damaged Input

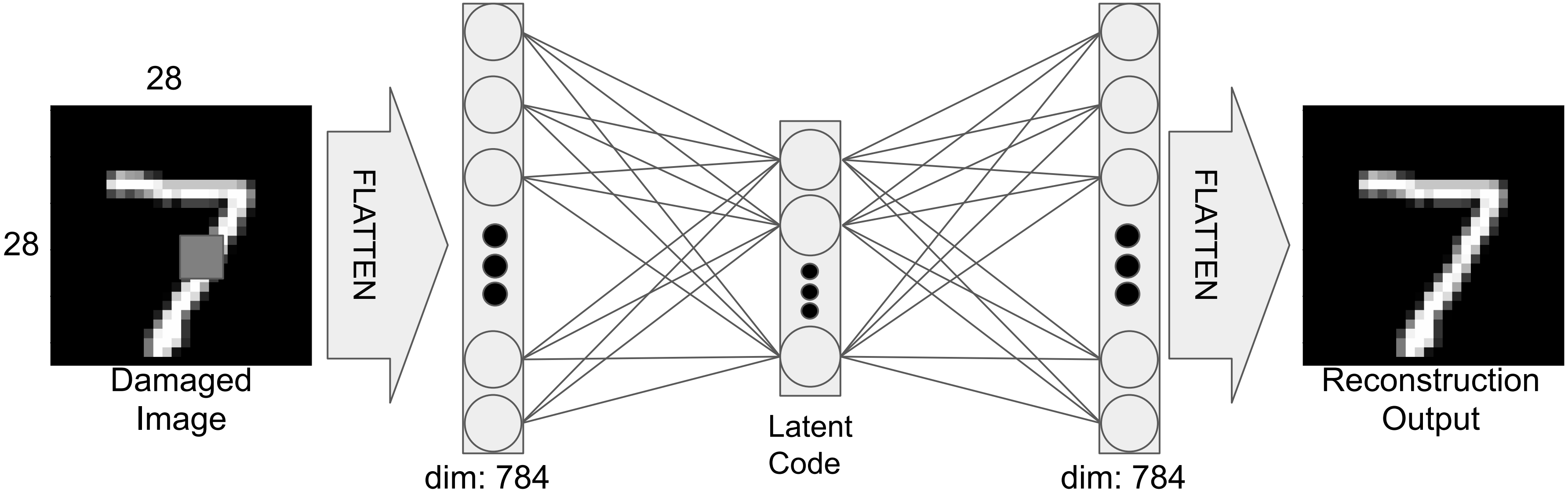
Inpainting algorithms (e.g. Autoencoder)

Reconstruction Output

The goals are widely from the restoration of old photos to the removal of selected objects.

Image courtesy of “Cornelia, mother of the Gracchi” by J. Suvee (Louvre)

Image inpainting using linear autoencoder

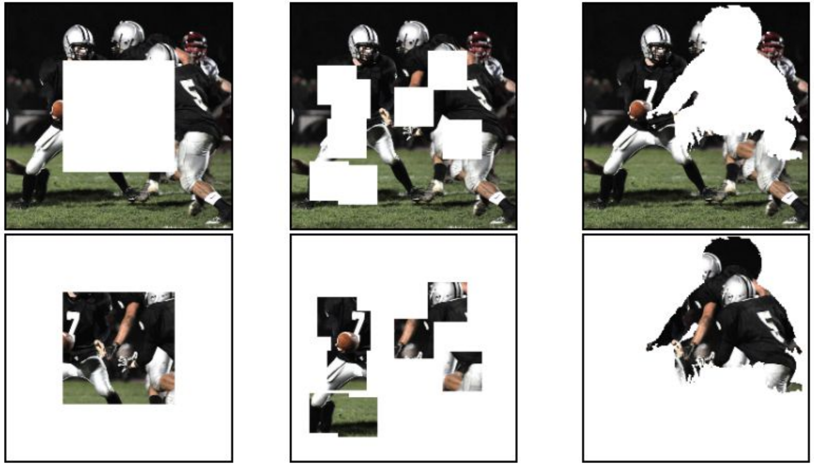


In this homework, the damaged image is a digit sampled from MNIST with a customized region mask.

Tips: you can draw a region mask in gray.

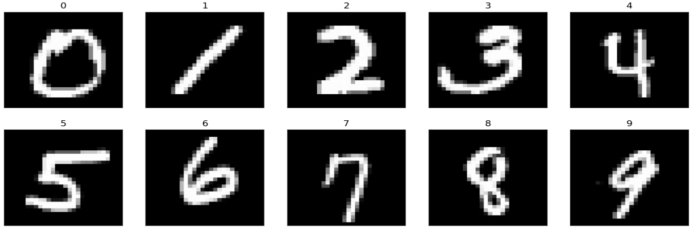
Problem description

- Given a damaged image (image with a customized region mask), reconstruct the undamaged one.
    - Design the region mask by yourself.
        - The region mask could be of any shape.
        - Put the region mask at any position.
        - The number of region mask is at least one.
- The image data we use is from MNIST
    - For the training data, choose data from at least 1 class in the training set.
    - For the testing data, sample images from the chosen class in the testing set.

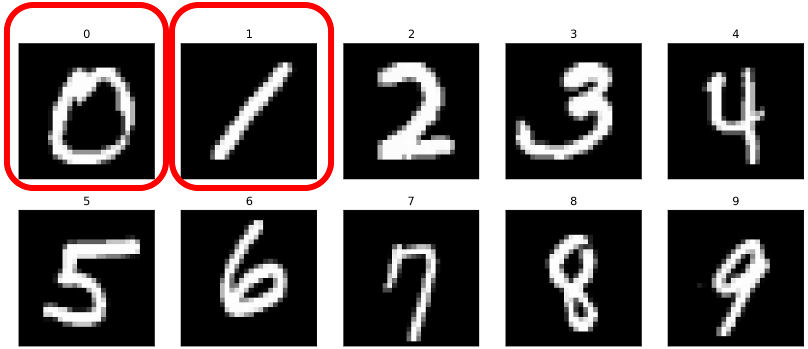


(a) Central region (b) Random block (c) Random region

Ref. Context Encoders: Feature Learning by Inpainting, CVPR 2016



Construct the dataset



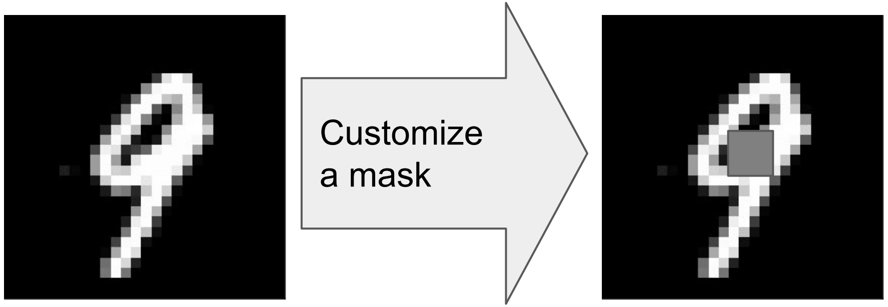
Example: we pick all samples from class 0 and 1 in both training set and testing set.

With TA’s code, the image pixel value is between [0.0, 1.0].
=> 1.0 for white, 0.0 for black, and 0.5 for gray

```
from torchvision.datasets import MNIST
from torchvision import transforms
from torch.utils.data import DataLoader
import torch
trainset = MNIST('./data', download=True, train=True,
transform=transforms.ToTensor()) # load training set
testset = MNIST('./data', download=False, train=False,
transform=transforms.ToTensor()) # load testing set
def get_target(dataset,target_label):
    samples, targets = [], []
    for sample,target in zip(dataset.data,dataset.targets):
        if target in target_label:
            samples.append(sample)
            targets.append(target)
    dataset.data, dataset.targets = torch.stack(samples), torch.stack(targets)
    return dataset
tar = [0,1] # suppose the target class is digit 0 and digit 1
trainset, testset  = get_target(trainset,tar), get_target(testset,tar)
trainloader = DataLoader(trainset, batch_size=128, shuffle=True)
testloader = DataLoader(testset, batch_size=128, shuffle=False)
for data in trainloader:
    sample, target = data # "target" can be ignored (useless)
    # do something below
```

Homework Requirements

1. Implement a **linear** autoencoder for image inpainting.
   a. Train and test a linear autoencoder (**Don't use non-linear** activation function like ReLU)
   b. Choose at least 1 classes from MNIST for experiments.
   c. Make an inference on images with your customized mask using your well-trained model.
2. Prepare a report to describe your experimental settings, model configurations or even interesting findings.

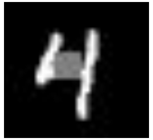


Customize a mask

the region mask is drawn in gray
(R,G,B) = (127,127,127) or (HEX) = 7F7F7F

Homework report

1. Show visualization results (at least 1 example). E.g.

Ground Truth	Damaged Input	Reconstruction Output
		

2. Settings for the customized autoencoder (at least 1 setting)
   a. Include implementation details like architecture(number of neurons, number of layers), optimizer(Adam/SGD/...), model initialization, learning rate, etc.
3. Quantitative study
   a. For the chosen class in the testing set, please report the L2-loss between original images and the reconstruction outputs. (take the average loss w.r.t batch size)
4. What you have learned
   a. Difficulties you encounter, interesting things you find, or special techniques you apply. E.g. describe your customized masks, training techniques, etc.

Submission

1. Compress the following items into DSP2020_prog3hw_[STUDENT_ID].zip.
   E.g. DSP2020_prog3hw_r08944004.zip
   a. Source code and your model pretrained weights
      i. All source codes (training, testing, etc.) and model pretrained weights
      ii. Specify how to execute your program clearly. *E.g. README.md*
   b. Electronic files of your report
      i. It must be a pdf file. Please name it DSP2020_prog3hw_[STUDENT_ID]. *E.g. DSP2020_prog3hw_r08944004.pdf*
      ii. No more than 4 pages.
2. Send to iis.sinica.1518@gmail.com with email title "DSP2020_prog3hw_[STUDENT_ID]".
   E.g. DSP2020_prog3hw_r08944004
3. Due on 2020.12.21 (Mon.) 23:59
4. If you have any problems in this homework, please send email to iis.sinica.1518@gmail.com.

Grading Policy

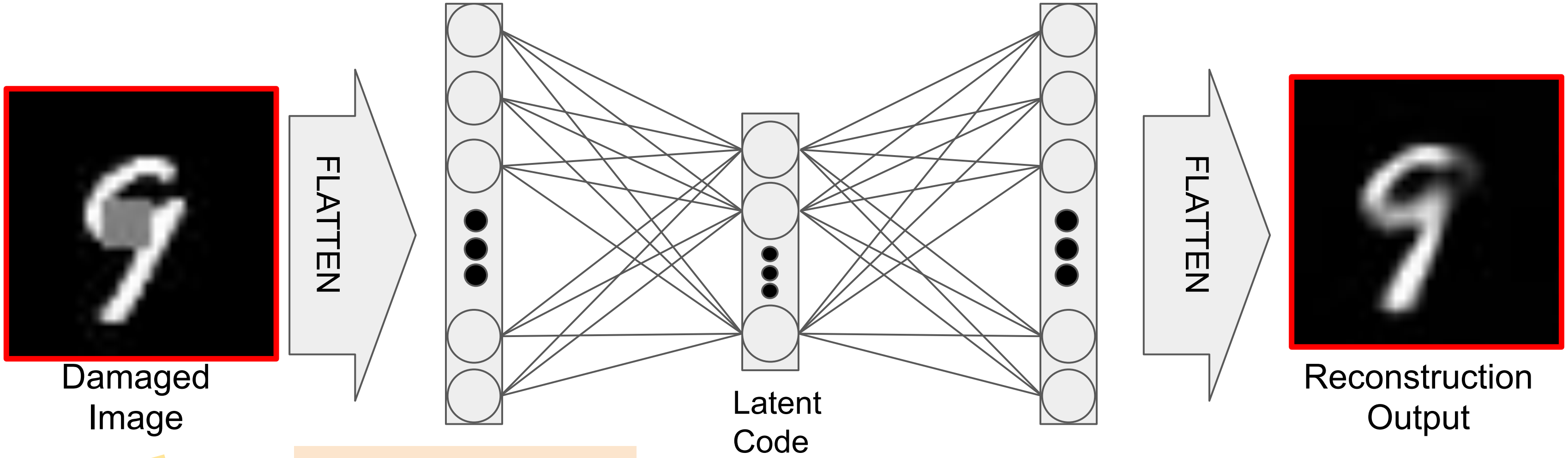
1. Source code (30%)
2. Pretrained model weights (10%)
3. Homework Report (60%)
   a. Visualization (20%)
   b. Model setting (10%)
   c. Quantitative study (10%)
   d. What you have learned (20%)

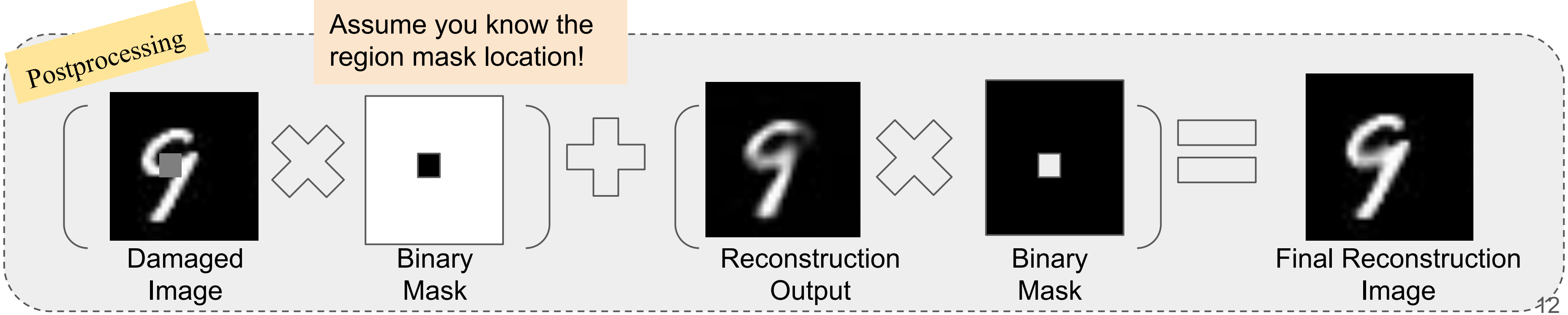
FAQ

1. Encounter “ModuleNotFoundError: No module named torchvision”
   Solution: pip install torchvision==0.4.1
2. Encounter “AttributeError: ‘MNIST’ object has no attribute ‘data’ “
   Solution: pip install torchvision==0.4.1
3. Encounter “ModuleNotFoundError: No module named torch”
   Solution for CPU user: pip install torch
   For GPU user, please see pytorch.org

Useful tips

Reference Implementation





Assume you know the region mask location!

Example

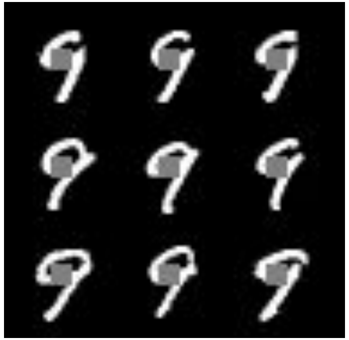
Network architecture:

- Latent code dimension: 128
- Run 10 epcohs
- use Adam optimizer with learning rate 0.001

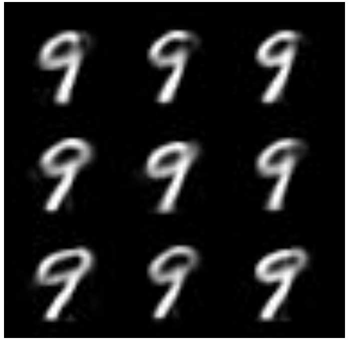
Hardware information:

- CPU: i5-5200U
- memory: 4GB
- No GPU
- Elapsed Time: 22s

Damaged input

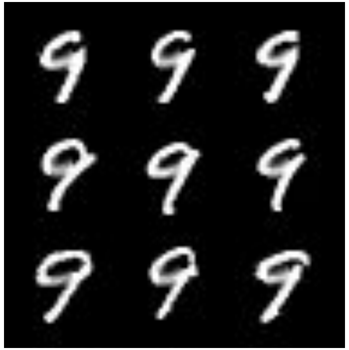


Reconstruction Output
w/o post-processing



Ground Truth

Reconstruction Output
w/ post-processing



Other techniques

Data augmentation

- For each batch training, the region mask can be of different shape and can be placed in different location.

Useful link

- TA’s provided materials (e.g. Model I/O, Save Image, etc.)
  - https://drive.google.com/drive/u/1/folders/1r2jpZPJ_bVU9wP4tgnQuOERLM-B_M0zw
- Cifar-10 Classification
  - https://github.com/kuangliu/pytorch-cifar
- MNIST Classification
  - https://github.com/pytorch/examples/tree/master/mnist
- AutoEncoder
  - https://github.com/L1aoXingyu/pytorch-beginner/tree/master/08-AutoEncoder

Notice: **Ignore the non-linear** activation function (E.g. Relu) in these examples!