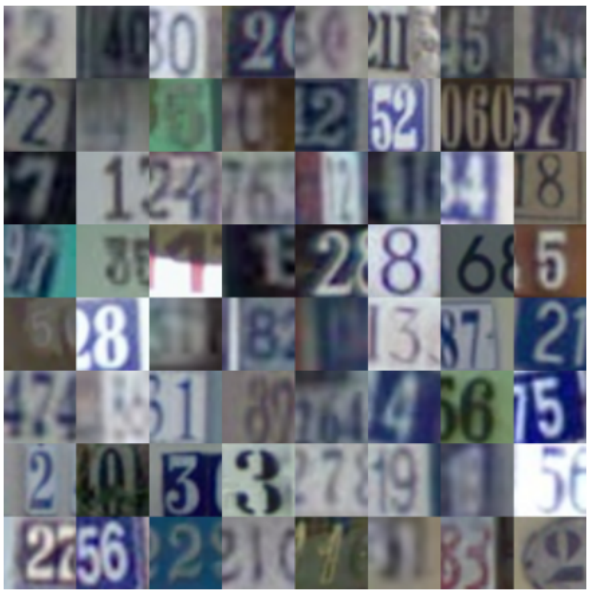
Machine Learning & Deep Learning
Advanced Image Analysis
MSCV VIBOT & ESIREM Robotique
Renato Martins - Universite de Bourgogne, October 2023

Assignment II - Style transfer with unpaired collections of images for classification

In this assignment, you will perform classification and style transfer using unpaired image samples in two different domain conditions. Although unpaired style transfer is a complex task, it has been intuitively modeled by the seminal work CycleGAN[1] (seen and discussed during the class). We will start working with a slightly more difficult dataset of numbers than MNIST, the street view house numbers dataset (SVHN)[2]:



You will also use MNIST later for generating colored samples with the style of SVHN. Both datasets are nicely available on the datasets module of Pytorch. You will also understand more closely how the training of a GAN works and how to evaluate your style transfer model with quantitative scores in the generation and how to apply it to improve a classification task.

Part I - Environment Setup

Please create a new notebook on Colab named ML_styleTransfer, based on your previous notebooks done in the class, or follow these steps to configure your conda environment if you are working in your local machine:

```
>> conda create -n ML_styleTransfer
>> conda activate ML_styleTransfer
>> conda install jupyter
# Go your workspace directory of the course and run jupyter
>> jupyter-notebook
```

[1] https://junyanz.github.io/CycleGAN/

[2] Please notice that the digit 0 has label 10 in SVHN. For more details about the dataset: http://ufldl.stanford.edu/housenumbers/

We will use mostly PyTorch tensors and numpy arrays during this exercise. So do not hesitate to check some definitions on the tutorials on PyTorch and for Numpy. There are also some helpful "Cheat Sheets" of basic commands for Numpy and Scipy.

IMPORTANT: You should use TensorBoard or wandb[3] for tracking and monitoring your models. Do not forget to include screenshots and plots of the losses/eval metrics from your generators and discriminators during the training on your report. Do not forget to comment their behaviors.

TIP I: Do not forget to select GPU when running your python notebooks. Otherwise, it will take forever!

TIP II: If your training is still taking more than 10 hours on GPU for training, please reduce the number of samples in your datasets for the development! I would recommend you start at first with a small number of samples for prototyping and debugging your model.

Part II - ResNet-18 Backbone for Classification

We will start defining classifiers for the recognition of numbers on MNIST and SVHN.

1. Implement a classifier of digits 0 to 9 that has as encoder backbone the ResNet-18 model. *Tip: Please remember you should first remove the last fully connected layer of the pre-trained ResNet model and then add your own output layer for the classification with the desired number of classes. While this is trivial in Tensorflow, in Pytorch this is slightly more difficult. One simple change you can make is to simply modify the fc layer of your model accordingly. For checking the layers of your model you can use print(model). Please also have a look on this documentation on how to finetune common existing models:* `https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html`

2. Make the required changes either in the image loaders or in the network model such as that your model can process either images from MNIST and SVHN without changes. Please justify your choices.

3. Please define your loss using categorical cross-entropy (using softmax in your output layer). Make a training script to train your model using a global random seed of 101, with 6000 samples of MNIST, for 20 epochs, with a batch size 64, and Adam optimizer with learning rate of 0.001. Your test set should be using 1000 samples of the test set, and please select accuracy as your main metric for evaluation.[4].

4. Please check and show the histogram of the classes of your cropped dataset. Are your datasets balanced?

5. Define two versions of your model, one that reuses the pre-trained weights of ResNet-18 and another trained from scratch[5]. What are the classification accuracies on MNIST of both models? What are the classification accuracies on SVHN of both models? Which dataset is easier to be classified and what is the configuration you would keep?

[3] `https://docs.wandb.ai/quickstart`

[4] Do not forget to set always the seed 101 for your data splits and all your model initializations

[5] Please check more generic tips of transfer learning here about reusing the weights: `https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html`

6. Do you observe any underfitting or overfitting in the obtained training loss and metrics curves?

Part III - Image-to-Image Translation

You will now design the CycleGAN model and evaluate the performance of your model for style transfer (aka as image-to-image translation), changing images from MNIST to look colored, or like as in SVHN. You will also test different ways to evaluate the performance of your model.

1. Write your own function compute_FID that receives two sets of data samples and returns the Frechet Inception Distance between the images in them. The Frechet inception distance measures the Frechet distance between two Gaussian distributions[6] from the averaged features extracted using the inceptionv3 network [7]. Please note you should implement your own FID metric using these definitions. You should however use the pre-trained inception_v3 model available on Pytorch for feature extraction. *Tips: Please notice you can remove the last fully connected layer by replacing it with an nn.Identity() layer. For the preprocessing of the input data expected by inceptionv3 you can use the following:*

```
## Please check https://pytorch.org/hub/pytorch_vision_inception_v3/
model_inception = torch.hub.load('pytorch/vision:v0.10.0', 'inception_v3',
    pretrained=True)

print(model_inception)
## TODO change your model for feature extraction here

# preprocessing operations for inceptionv3
preprocess = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize(299),
    transforms.CenterCrop(299),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485,0.456,0.406], std=[0.229,0.224,0.225])])

# applying the pre-processing per image
image_incep = preprocess(img) # where img is a numpy array
```

2. In order to evaluate your FID implementation, please compute the FID from the provided set of colored images generated with a previously learned CycleGAN model:

```
# Get samples CycleGAN generated data
!mkdir data
%cd data
!gdown https://drive.google.com/uc?id=16TdnYC1kCpKVR7yu8y8A-AvElvaQheOu
```

And for the real colored data samples please use the following code:

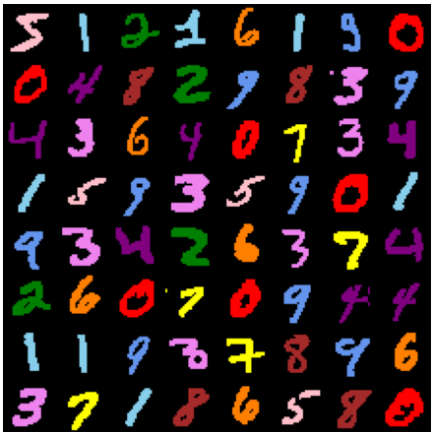
```
# convert MNIST to color images
def get_rgb(images_src, image_labels):
        colored_images = np.zeros((images_src.shape[0], 28, 28, 3))
        """Convert the digits to specific colored-version.
        0: red; 1: blue; 2: green; 3: violet; 4: purple; 5: pink;
        6: orange; 7: yellow; 8: brown; 9: cornflower blue."""
        colors = np.array(
                [[255,   0,   0],
                 [135, 206, 235],
                 [  0, 128,   0],
                 [238, 130, 238],
```

[6] Frechet distance definition: https://core.ac.uk/download/pdf/82269844.pdf

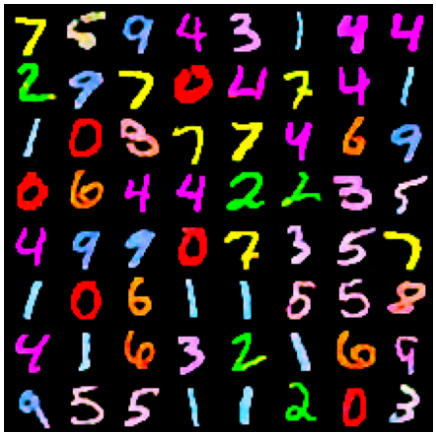
[7] The introduction of the metrics as we use is from NeurIPS 2017 paper "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium" (page 6, Section 3): https://papers.nips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

```
                  [128,   0, 128],
                  [255, 192, 203],
                  [255, 128,   0],
                  [255, 255,   0],
                  [165,  42,  42],
                  [100, 149, 237]], np.float32)

        for k in range(images_src.shape[0]):
                for i in range(28):
                        for j in range(28):
                                if images_src[k][i][j] > 0:
                                        colored_images[k][i][j][:] = colors[int(
                                            image_labels[k])]
        return colored_images

###############################################
### Create colored MNIST data              ###
###############################################
## Dataset preparation
# number of images to process
img_number = 1000
index_sub = np.arange(img_number)

# slicing dataset
dataset_mnist = datasets.MNIST(root='./data', train=True, download=True)
dataset_mnist.data = dataset_mnist.data[index_sub]
dataset_mnist.targets = dataset_mnist.targets[index_sub]
images_np = dataset_mnist.data.numpy()
labels_np = dataset_mnist.targets.numpy()

# get colored version
images_np = images_np.reshape(-1, 28, 28, 1).astype(np.float32)
colored_real = get_rgb(images_np, labels_np)
colored_real = torch.tensor(colored_real)

plt.figure(figsize=(8,8))
plotviews = colored_real[0:64,:,:,:]
plotviews = torch.swapaxes(plotviews,1,3)
print(plotviews.shape)
plt.axis("off")
plt.imshow(np.transpose(vutils.make_grid(plotviews, padding=0, normalize=True).cpu
    (),(2,1,0)))

###############################################
### Generated data with CycleGAN           ###
###############################################
## data MNIST X_test using CycleGAN
with open('data_sem.npy', 'rb') as f:
  colored_cycle = np.load(f)

colored_cycle = colored_cycle[0:img_number,]

plt.figure(figsize=(8,8))
plotviews = colored_cycle[0:64,:,:,:]
plotviews = torch.tensor(plotviews)
plotviews = torch.swapaxes(plotviews,1,3)
print(plotviews.shape)
plt.axis("off")
plt.imshow(np.transpose(vutils.make_grid(plotviews, padding=0).cpu(),(2,1,0)))
```

Your first 64 image samples should look similar to this:



Domain A (real colored samples)



Domain B (generated samples)

Please compute the FID distance between two sets of images in the following scenarios:

- Domain A: real colored MNIST and domain B: generated colored MNIST
- Domain A: Original MNIST (first 1000 images of the test set) and domain B: generated colored MNIST
- Domain A: Original MNIST (first 500 images of the test set) and domain B: first 500 images of training set of SVHN

Which one presents the smallest FID and highest FID? Comment and analyse the results from what you would expect.

3. Please see the GAN notebook and model seen during the lecture (**DL_GAN_DemoGeneration.ipynb** and slides) to help implementing a training loop of a GAN model. Then implement your CycleGAN model for changing the style of MNIST images. Please notice your two generators now should follow an encoder-decoder structure. Please select a U-Net encoder-decoder for your generator (as seen during the class) taking care of the number of parameters of your overall model. Please comment and justify your design choices.

4. Train your CycleGAN model to perform style transfer between the following domains:

   - Domain A: Original MNIST (first 1000 images of the test set) and domain B: first 1000 images of real colored MNIST
   - Domain A: Original MNIST (first 500 images of the test set) and domain B: first 500 images of training set of SVHN

5. Please provide the generators and discriminators loss curves. Please show the 64 inference results of your style transfer for the two transferring scenarios. *Tip: for the plot you can use the following code (or the one previously provided in the Assignment I)*:

```
plt.figure(figsize=(8,8))
plt.axis("off")
plt.imshow(np.transpose(vutils.make_grid(views2plot,padding=0).cpu(),(2,1,0)))
```

6. Please compute the FID distance and comment the results between the generated and real samples:

   - Domain A: real colored MNIST and domain B: generated colored from your CycleGAN

- Domain A: first 500 images of training set of original MNIST and domain B: first 500 images of training set of SVHN
- Domain A: first 500 images of training set of SVHN and domain B: generated SVHN numbers with your CycleGAN (500 first samples of MNIST training set)

Part IV [Bonus Points] - Data Augmentation for Classification

Describe and comment the steps of how you would employ this style transfer to improve the accuracy of your classifiers seen in **Part II**. Please comment in the context of limited number of annotated samples in one domain. Justify and discuss your steps for training your classifiers.

Submission

Please return a concise PDF report explaining your reasoning and visualization of your results. You should also submit your python notebook (commented) with the corresponding implementations, together inside a [name]_list_2_DL.zip file (replacing [name] with your name ;-)). The submission should be done via the teams channel of the course using teams assignment.
Deadline: 20/11/2023 at 23:59pm.

Note: This assignment is individual. Copying and sharing the work from another student will be awarded a 0 mark. In case there are multiple submissions with shared parts of code, each one will receive a 0.