LINMA2472 - Algorithms in Data Science

Homework 3: GAN and CNN to conquer MNIST
Part 2: Theory and exploration of the latent space

Bastien Massion - bastien.massion@uclouvain.be

15 November 2022

Guidelines

Homework 3 is divided into 2 parts. The first part was given in a Notebook and used PyTorch in order to implement neural networks. More precisely, it asked you to implement two neural networks (a GAN and a CNN) in PyTorch in order to generate and classify handwritten numbers (and letters). The second part of the homework focuses on the theoretical aspects of the neural networks, their analysis and on the exploration of the latent space.

You will have to make a report answering the questions and completing the tasks presented in this assignment. The first two sections basically formalise the tasks present in the Notebook of part 1, only sections 3 and 4 bring new questions. Moreover, some questions are counted as bonuses and are optional, for the bravest and most interested among you. All answers should be concise and precise (max 10 lines). However, as a lot of plots are asked, your report could be somewhat long (up to 15 pages), but there is not too much to write. Finally, for some questions, you are advised to look up external resources (articles, blogs,...). Don't forget to cite them (and include a bibliography at the end of the report).

For this homework, be sure to take into account the general feedback from the previous homeworks published on Moodle.

As for the previous homeworks, you are making groups of 3, in the Moodle activity "Group Choice for Assignment 3". Register as soon as possible.

Per group, you have to submit one zip file **group_#_Name1_Name2_Name3_HW3.zip** containing:

- Your report (in PDF format **.pdf**),
- Your codes in Jupyter Notebooks (**.ipynb**),
- Savings of your trained models (generated with the saveModel function, see part 1),
- Savings of your chosen latent vectors (generated with the saveLatentVectors function, see Section 3).
- A READ_ME.txt file explaining what are the files in your zip useful for, how to easily run your codes and check your results.

The deadline for the homework is: **30 November 2022, 23:59.**

Your questions should be posted on the dedicated Moodle "Class forum" or sent directly to the mail address: **bastien.massion@uclouvain.be.**

Introduction

Since their introduction in 2014 by Goodfellow et al. [3], Generative Adversarial Networks (GAN) have taken a predominant place in the deep learning landscape. GANs are deep neural networks composed of two parts: a generator and a discriminator. Those two parts are trained as opponents (hence "adversarial"): from the dataset, the generator tries to create new fake data that can fool the discriminator into thinking they are real, while the discriminator tries to distinguish between true and fake data. Often, the most interesting part of this model is the generator, therefore we often refer to GANs as generative models (even if they are also discriminative models). This generative property is used in a lot of applications: mainly for image processing task (style transfer, segmentation, face generation, image inpainting, deblurring, super-resolution,...), but also for natural language processing tasks (text summarization, text generation,...), for music generation or even for medical tasks (tumor detection). [4]

Another powerful architecture is Convolutional Neural Networks (CNN). CNNs naturally exploit invariances that are present in data in order, for example, to classify them. This is done by applying successive convolutions on data in order to extract the main features. Together with the backpropagation algorithm, the invention of CNN marked the revival of neural networks and deep learning in the late 80's [7]. However, it is the groundbreaking performances of AlexNet in 2012 [6] for image recognition tasks that propelled CNNs and deep neural networks in general as workhorse models in artificial intelligence (AI). Since then, CNNs have found plenty of applications, mostly in image processing tasks. [1]

The goal of this homework in two parts is to implement your first GAN as well as your first CNN, to explore some properties and finally to combine them in order to generate what you desire.

Section 1. Train a GAN on MNIST for image generation

For this first section, you should not have to implement any additional code to the first part of the homework. You will be asked to report, analyze your results and answer some theoretical questions.

1.1 Your GAN model

If you kept the reference generator and discriminator architectures, describe them in great detail: number of layers, size of layers, types of layers, total number of parameters, dimension of the latent space for the generator, inputs, outputs.

If you modified the reference architectures, detail your new architectures and explain why you did the changes. If it contains other characteristics than the ones cited above, define them and give their values too.

If you trained several CNN, describe them all.

1.2 Training your GAN

On which datasets did you train your GAN(s): MNIST, EMNIST Letters or both? Be clear on which model was trained on which data.

Report the values of the optimization parameters (device used, number of epochs, batch size, Adam parameters).

Show new images generated by your generator and compare them visually with real data samples. Show predictions made by your discriminator. Comment briefly on your results.

1.3 Error metrics during training

Show the evolution of the two main error metrics in function of the number of epochs: loss functions (for generator and for discriminator) and classification accuracy (for training and testing sets, as defined in part 1). Report the duration of the training as well. Discuss your results.

1.4 Improve your GAN model

Suppose that you want to improve your GAN for digit generation for some criterion (better outputs, faster training, more robust model or something else). Propose 3 techniques that could be used and for which purpose(s) you would use them.

1.5 BONUS: GAN loss function

How do you compute the generator and the discriminator loss functions in practice, i.e. what inputs do you give to the loss function (for your discriminator and for your generator)? How can you derive those empirical expressions from the GAN minimax problem formulation seen in the course?

1.6 BONUS: Discriminator output and loss function

The expected output from the discriminator is a number between 0 and 1. Usually, it is imposed by adding a sigmoid layer at the end of the discriminator before computing the Binary Cross Entropy loss function (BCELoss). However, this is not the case in the reference architecture: there is no sigmoid layer and the loss used is BCELoss-WithLogits. Why is it treated in this way? Explain.

Section 2. Train a CNN on MNIST for image classification

For this second section, you should not have to implement any additional code to the first part of the homework. You will be asked to report, analyze your results and answer some theoretical questions.

2.1 Your CNN model

If you kept the reference CNN architecture, describe it in great detail: number of layers, size of layers, types of layers, total number of parameters, inputs, outputs.

If you modified the reference architecture, detail your new architecture and explain why you did the changes. If it contains other characteristics than the ones cited above, define them and give their values too.

If you trained several CNN, describe them all.

2.2 Training your CNN

On which datasets did you train your CNN(s): MNIST, EMNIST Letters or both? Be clear on which model was trained on which data.

Report the values of the optimization parameters (device used, number of epochs, batch size, Adam parameters).

Show predictions made by your classifier. Comment briefly on your results.

2.3 Error metrics during training

Show the evolution of the two main error metrics in function of the number of epochs: loss function and classification accuracy (for training and testing sets, as defined in part 1). Report the duration of the training as well. Discuss your results.

2.4 Convolutional layers

Why are convolutional neural networks so widely used in image classification? How can you physically interpret convolutional layers? Why are they preferred to dense linear layers?

Convolutional layers can also be used in other types of neural networks, for example in GANs, as it was done in the reference GAN architecture. What are those GANs called?

2.5 Improve your CNN model

Suppose that you want to improve your CNN for digit classification for some criterion (better outputs, faster training, more robust model or something else). Propose 3 techniques that could be used and for which purpose(s) you would use them.

2.6 BONUS: CNN loss function

What does the cross-entropy loss function represent? Why it is so widely used for classification tasks?

Section 3. Explore the latent space

An important question about GANs is the ability to generate what we want. Indeed, we often give a random latent vector as input and see what comes out. How could we possibly find the optimal latent vector to generate what we desire? This problem is known as "GAN inversion" and is far from trivial. [9]

In this third part, you will make a first step in this direction: you will generate new images with the label you want and explore the latent space. At the end of the process, you should be able to generate exactly the messages you desire. The final goal of this homework is then to create encrypted (or not) messages that you want to tell your teaching team. This is for section 4, keep up until the end! For now, let's conquer MNIST!

For this third section, you will have to code some more. You will also be asked to report and analyze your results and answer some theoretical questions.

3.1 Combine your generator and your classifier

In part 1, you have created a functional generative model and an efficient classifier. You will now use both models to generate images from the labels you want. To solve this GAN inversion task, you will proceed in two steps.

First step: you generate a lot of triplets (latent vector, generated image, predicted label). To do so, you randomly generate latent vectors, which you pass in your generator to create the corresponding images, and then give the latter to your classifier to get its predicted labels.

Second step: select the images with the highest quality, i.e. the ones looking visually the most like true data. An idea to find them is to keep only the labels predicted with high probability, as they should be the most unambiguous ones. For example, you could set a minimum threshold on the prediction "certainty" (defined in part 1 as the "probability"/output of the most activated label in your classifier last layer). We will call these "labelled latent vectors". Notice that this achieves the goal: by choosing vectors with the desired label, you can generate any message you want!

Code these steps and show some of your results. Are you satisfied with your generated images?

3.2 Distribution of classes

Generate a lot of labelled latent vectors and count the proportion of each class. Are all labels represented? Are they represented equally? How does the certainty threshold influence that repartition? Explain and don't forget to report your experiment parameters (dataset, latent vector generating distribution, number of samples, certainty threshold).

3.3 Mean of your latent vectors

Take the mean of your labelled latent vectors from each class, we call these new vectors "mean latent vectors". Show the corresponding images. Are these results better than the ones from before? Comment briefly.

3.4 Norms and distances

Compute the norms of randomly generated latent vectors such as the labelled latent vectors. Compute the norms of the mean latent vectors. How to explain the observed gap?

Compute the distances between labelled latent vectors and the mean latent vectors of each class. What do you observe? This brings you to consider classifying to the nearest mean vector, which is known as Minimum Distance to Mean (MDM) classification [8]. Compare MDM (criterion = minimum distance mean vectors) and CNN classification (criterion = maximum activation of the last layer). Do both classifiers agree? Try with different values of the certainty threshold.

3.5 Interpolation for exploration

The goal is to examine how digits (or letters) transform one into the other according to your model. First, within the same class, then from one class to another.

First, choose two triplets within the same class that you find visually satisfying. We call those latent vectors "extreme latent vectors". Then, create intermediate triplets by interpolating linearly between the extreme latent vectors. Show how one image evolves into the other one by linear interpolation in the latent space. Compare the interpolants to the extremes, for 3 distinct examples.

Now, choose two extreme latent vectors from different classes (for example, corresponding to two mean latent vectors). Then, repeat the instructions from above for these new extreme latent vectors.

3.6 Noise for exploration

Generate "noise latent vectors" with $0 < \sigma < 1$: either from a normal zero-mean distribution (sampled in $\mathcal{N}(0, \sigma I_{\text{dim\_lat}})$), either from a uniform distribution (sampled from $[-\sigma, \sigma]^{\text{dim\_lat}}$). You then add those noise vectors to the mean latent vectors to create "variation latent vectors". These vectors are close variations of the mean latent vectors, i.e. they are neighbours in the latent space. Compare your variation latent vectors with the mean latent vector, in terms of labels and in terms of visual quality. Then, analyze the impact of the value of σ (how far from the mean latent vectors) on the variation latent vectors. More precisely, try $\sigma \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

For your information, exploring the latent space is key to the incredible results of models such as StyleGAN2 [5]. The latent space is a condensed space where the inherent features of images can be understood. Moreover, it gives insights into how features influence the generated image, how features are linked together, or how to search for better variations of an image in the immense space of possible images. This exploration is absolutely delightful, and I can warmly suggest you have a look at the amazing YouTube video "AI Art Evolution" by Emergent Garden [2] to be convinced. Even if the models in the video are not GANs, the ideas are similar and have been a source of inspiration for this project.

3.7 BONUS: Mode collapse

Sometimes, a GAN learns to generate only a few classes. This phenomenon is called "mode collapse". Why could this happen? How to avoid it?

Section 4. Create what you want

In the third section, you gained some understanding of the latent space. In this fourth and last section, you are asked to create and transmit messages to the teaching staff. Your messages can be encrypted or not and will show that you have mastered MNIST thanks to neural networks!

4.1 Mandatory: Open message

You are asked to generate one list of digits/letters that will represent one message. Therefore, you have to find corresponding latent vectors that we call the "message latent vectors" and put them, in order, in a PyTorch tensor. Then, you have to save it in a file thanks to the saveLatentVectors function provided below. You can also load previously saved latent vectors with loadLatentVectors.

```
def saveLatentVectors(filename, latent_vectors):
    torch.save(latent_vectors, '../Latent_Vectors/' + filename)
    return

def loadLatentVectors(filename):
    latent_vectors = torch.load('../Latent_Vectors/' + filename)
    return latent_vectors
```

For the functions described above, a folder "Latent_Vectors" was created, which can be a good idea. However, you are free to change those functions as you desire.

Your task is to show your created message, store the message latent vectors in files and add a little commentary about the message. For example, if you wrote GPS coordinates, you can explain why the located place is important/interesting to you. Be creative!

Please make it easy for the reader to run your codes to recreate your results from this section, he will be in a better mood. First of all, explain clearly in the READ_ME file how to use your code. Also, prepare in your codes the load functions that should be run for the models and the message latent vectors to be loaded. Finally, do not forget to include any file in your zip, and take care of writing the correct paths in your codes to your different files.

4.2 Optional: Encrypted message

Using the same procedure as before, you can, if you want, add some other messages. However, this time, you don't have to report the generated images and explain their significance (but you can if you want). You are just asked to provide a list of latent vectors just as before, and the reader will have to pass them through your generator in order to understand them. The reader will probably be the only one to get your message as they are encrypted, and he will not necessarily tell the rest of the teaching team. In your report, simply mention that you have done extra messages. Don't forget to add the files in the zip and the details in the READ_ME.

Stay polite, be creative and have fun!

References

[1] 7 Applications of Convolutional Neural Networks - FWS.

[2] Emergent Garden. AI Art Evolution, October 2022.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[4] Jonathan Hui. GAN — Some cool applications of GAN, March 2020.

[5] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN, March 2020. arXiv:1912.04958 [cs, eess, stat].

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[7] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[8] Pal Mahesh. [PDF] Factors influencing the accuracy of remote sensing classifications: a comparative study.

[9] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. GAN Inversion: A Survey, March 2022. arXiv:2101.05278 [cs].