Milestone Report 2

In this report, I describe in further details the architecture of the GAN (Generative Adversarial Model) I used to build a Image-to-Image translation model, how some functions are implemented and some of the preliminary results.

Recall of our Objective

Our goal is to build a Image-to-Image translation model able to plot the connections (aka bonds, springs) between the building blocks (aka sites, nodes) of a fictitious metamaterial. Practically speaking, we feed the image on the left below and expect the image on the right

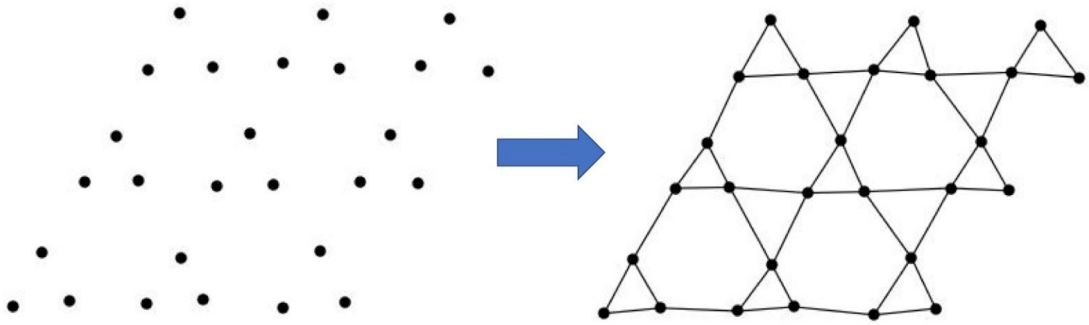

Figure 1: Disordered Kagome lattice without (left) and with (right) bond connectivity

The images considered so far are quite similar to those of figure 1, following a Kagome lattice periodic structure, with the position of the three sites carrying a small amount of variation from one cell to another (aka small disorder). While we anticipate the model to work well for this type of structures, we want to test how it will perform if fed a total different set of images, such as the triangular lattice (same as the Kagome lattice, except only triangles) or the square lattice (four sites per cell).

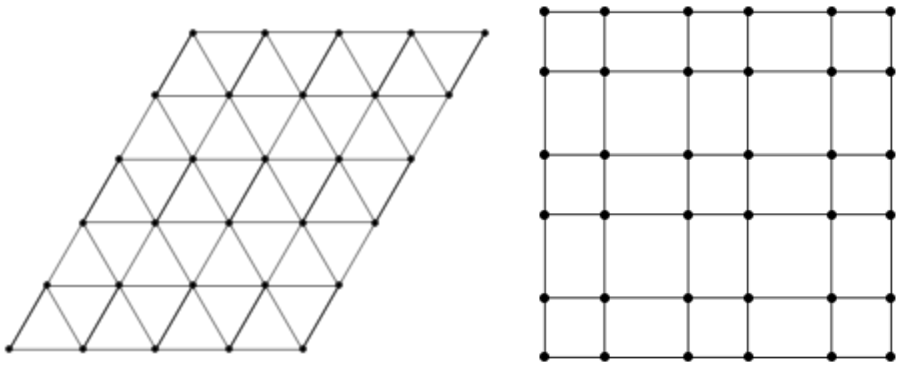

Figure 2: Ordered triangular lattice and square lattice

The GAN Model

Our generative adversarial network consists mostly of two models:

- The **discriminator** allows us to classify if two images, a source (without bonds) and a target (with bonds), are translation of one another.
- The **generator** generates the translation image.

These two models work collaboratively (I should say adversarially) so that the weights of the generator are updated given how well the discriminator performs. But tying it all together is a bit more complicated than that. Without getting in too much detail about the architecture of both discriminator and generator, let me explain how the training of the two models work.

To start with, the model is set by a training dataset of **sample size** = 2 (corresponding to 2 pairs of images), a **number of epochs** = 100 and a **number of batches** = 1. This means that the **batch size per epoch**, i.e. the sample size divided by the number of batches, is 2. Therefore, each epoch consists of going over the 2 pairs of images and hence there are a total of 2*100 = 200 **iteration steps**. During each step:

- The model picks two real images (a source and a target) and a 2D array of 1's. Real images are images drawn at random from the training sample. Then,
- The model generates a fake image from the source real image above and a 2D array of 0's. Fake Images are images generated by the generator. Then,
- the discriminator takes as input the real images and the 2D array of 1's and trains accordingly. Then,
- The discriminator takes as input the fake images and the 2D array of 0's and trains accordingly. Then,
- The generator takes thel source and target images as input and trains accordingly.

We summarize the performance of the model every 20 iteration steps, accounting for a total of 10 summaries. The very last performance output is the final model, which we save for re-use later on.

Early report

First, given the small size of our training sample, this training process turns out to be relatively fast. I plan to have a sample much bigger in size in the future, but up until now, the results are more or less satisfactory.

By using the final model, we can visualize what the generated output image is for a sample from the training dataset

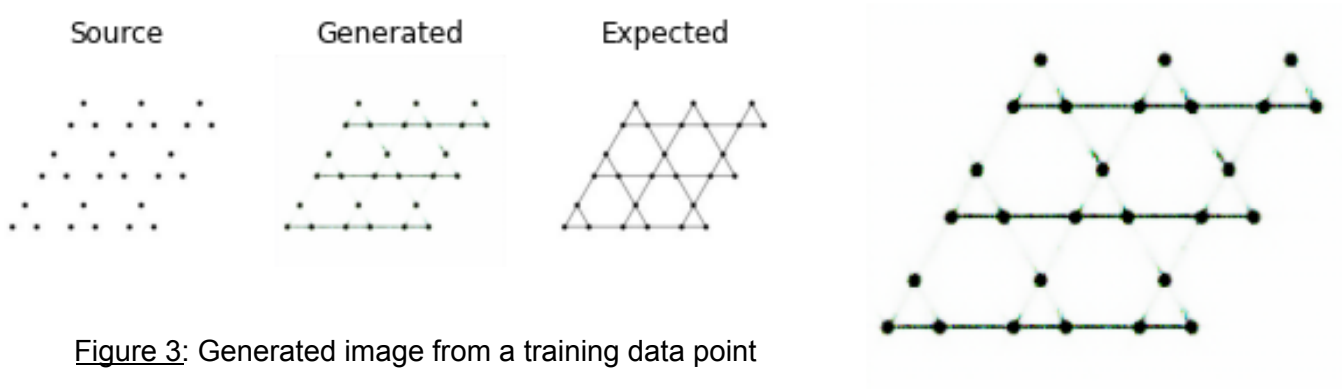

Figure 3: Generated image from a training data point

As we can see, the generated image plots the bond connection we expected. The horizontal lines are drawn well, the diagonal ones a little less. Importantly enough, the sites (the black dots) are drawn at the same spot as they are on the source image. This positional argument is critical to the mechanical properties of the system considered.

We can also visualize a generated output for an image that was not in the training set. To that end, we consider:

- Another deformed Kagome lattice, quite similar to those of the training data and which we expect the model to work well on.
- A square lattice with different number of sites per cell and different bond connectivity.

Figure 4: Testing the model on two validation samples

Two things to take from these results:

- Because the Kagome lattice are quite similar to one another, the model performs well on the left image, in that it generates the bonds we expect
- For the square lattice, the bond connectivity (which resembles the one of figure 1, image on the right) is totally different and hence the model cannot plot the bonds. I would expect better results if we include more of the square lattices in the training data. One exciting feature here though is the ability of the model to easily detect the sites.