Milestone Report - Develop a Image-to-Image Translation Model to Capture Local Interactions in Mechanical Networks (GAN)

Introduction

This project is intensively related to my research work in physical sciences at the Georgia Institute of Technology on mechanical deformations in metamaterials.

Mechanical metamaterials are systems formed by connected building blocks, which grant them unconventional mechanical properties. How these blocks interact with one another determines the structure of the system and controls its mechanical deformations. Depending on its structure, a metamaterial can be entirely rigid or can possess soft regions; under certain circumstances, it can convey forces, stress and displacements in a particular direction. The model for these mechanical metamaterials, whether they are human-size objects or designed at the microscale, is the same: the blocks (also known as sites, nodes, or balls) are spread out in a roughly periodic structure and are connected by springs (also known as edges or bonds). These form the so-called **mechanical ball / spring networks** of interest in this work. For small systems, the components of these mechanical networks are tiny in size - of the order of the micron (μm). The nature of the local interactions, i.e. how the sites connect with one another, and which is commonly referred to as the *bond connectivity*, is then critical to understand the mechanical properties of the system.

Motivation / Goal

Consider the following situation: a researcher studies a micro-sized mechanical network and is searching for the local interactions in order to understand the mechanical properties of the given system. In principle, the researcher can identify where the sites are (a.k.a the building blocks) but not necessarily the bond connectivity. This project aims to design a neural network model that will automatically plot the bond connections between the sites of a given system. In particular, the goal is to build a **Image-to-Image Translation** model which inputs the image of the particles and outputs the same image with the bonds drawn:

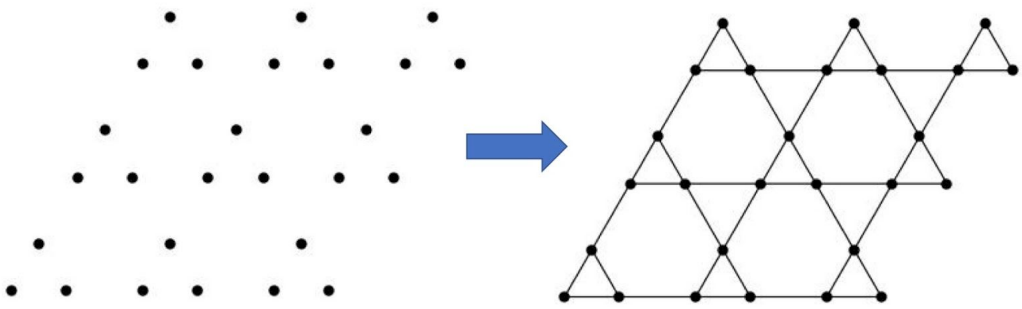

Figure 1: Ordered Kagome lattice without (left) and with (right) bond connectivity

A system often studied in the research on mechanical metamaterials is the Kagome lattice, drawn above without (left) and with (right, solid lines) the bond connectivity. On both images, the sites (black dots) are ordered periodically in space in groups of three. Each set of three neighboring sites constitute what we call a *cell*. However, in general, the systems encountered will be a bit more disordered, like in the following image.

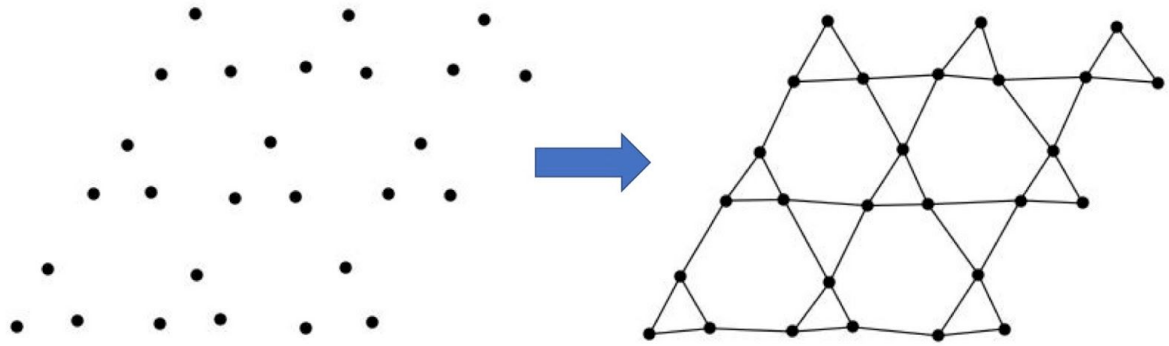

Figure 2: Disordered Kagome lattice without (left) and with (right) bond connectivity

As observed, the input image of figure 2 appears identical to the one of figure 1, at least to the human eye. But despite their apparent similarities, when the bonds are plotted, it becomes clear each image represents two very different systems. This “bond-plotting” model can facilitate the classification of these mechanical systems and could be very enriching for the scientific community that studies such materials.

The Data

The data I’ll use is one I generate for my research work, using the program *Mathematica*. The input and output images will be similar to those already shown on this document: a Kagome lattice of about 30 sites (the points) and as many springs (the edges). These images are artificial, but they correspond to the structure of metamaterials that present exciting mechanical properties.

Generating the data via Mathematica

For each figure, the points are generated as follow:

- The first cell (the original 3 points) is drawn in the lower-left corner.
- The remaining cells are generated by a set of discrete geometric translations to form a periodic structure. This is the ordered structure of figure 1.
- Each point in each cell is then shifted by a small and yet random displacement to create an image like figure 2: the structure appears to carry a long range order but contains small variations from one cell to another.

- For the images with bonds, the points are then connected by straight lines (the edges) as indicated on figure 1 and 2. The bond connectivity is the same for all figures.
- Each figure is then exported as a jpg image, roughly 5 KB in size and 350x225 pixels in dimension.

The practical advantage here is the potential to easily generate 1000s of pictures, without the need to further edit them. Therefore, the wrangling and cleaning steps of this project are small, if not non-existent.

More data

Being able to generate as much clean data as we want, in a relatively small amount of time, will also allow us to attempt multiple things:

1) If the model cannot train on the type of images provided in figures 1 and 2, we can assist it by providing easier images to build on such as:

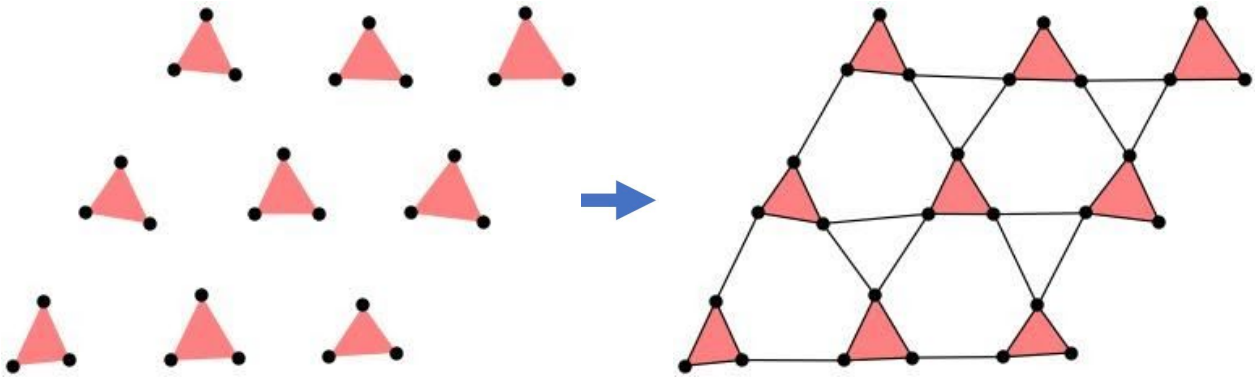

Figure 3: Kagome lattice with color-filled cells

2) If the program performs well, we can even attempt to model other mechanical systems beside the Kagome lattice, such as the square lattice:

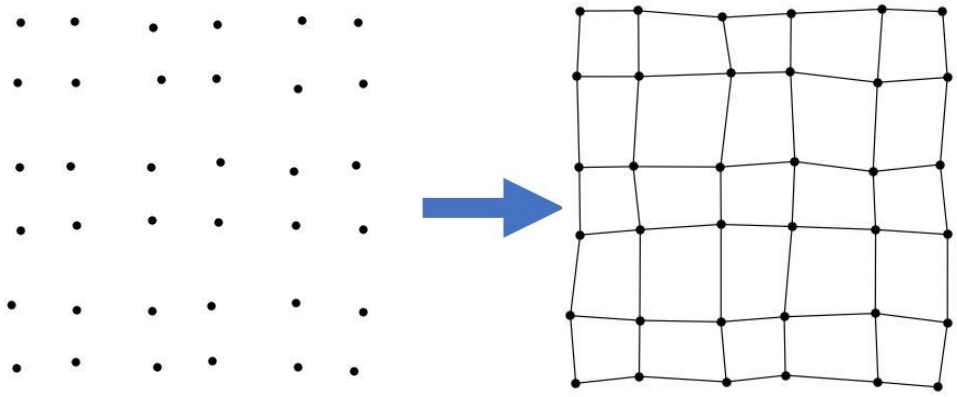

Figure 4: Square lattice

Loading data and verification

Regardless of the type of systems considered, the jpg images are uploaded on my Google Drive, which I will mount to the Google Collab notebook in order to build, train and test the deep learning model. The images are loaded onto the notebook using the Keras.preprocessing.image function and the pixels are stored in a 3D numpy array, the first two dimensions corresponding to the image size, the last one being the RGB code for each pixel (ranging from 0 to 255). A good way to make sure the images are properly deconstructed is to reconstruct the pixel arrays and to display the corresponding images using imshow from numpy.

Methodology

The deep learning model used to generate an image-to-image in this project is a **Generative Adversarial Network** (GAN). I will use as main reference the Pix2Pix modeled developed by Jason Brownless Ph.D. for Machine Learning Mastery and referenced here: https://machinelearningmastery.com/how-to-develop-a-pix2pix-gan-for-image-to-image-translation/

As mentioned earlier, I plan to use Google's Colab notebook, to take advantage of Google's computer resources and to access to their cloud-hosted GPUs in order to run the GAN. There are two options to do so:

	Price	GPU	Runtime	Memory
Colab	Free	K80	Up to 12 hours	12GB
Colab Pro	$9.99/m (before tax)	T4 & P100	Up to 24 hours	25GB with high memory VMs

Depending on the needs of this project, I will potentially enroll in the colab pro for a more high-end GPU, a longer runtime and a larger memory.

Summary of GAN

The GAN's architecture consists of two main models that train together in an adversarial manner - the discriminator and the generator - along with other features that bind and train them together.

The **discriminator**:

It consists of a deep Convolutional Neural Network (CNN) which takes as input a source image (the image with no bonds) and a target image (the image with bonds) and classifies the likelihood the two images are translation of one another

A **generator**:

It generates an image. The generator follows a U-net architecture and consists of three sections: the encoder (contraction), the bottleneck and the decoder (expansion).

1) The encoder section consists of multiple blocks that downsample the input image. Each block is a convolution layer with an ever-so-increasing number of filters which allows the architecture to learn the complex structure of the image effectively.
2) On the contrary, the decoder consists of several expansion blocks, with a number of filters ever-so-decreasing. Importantly enough, the input of each expanding layer features both the output of the previous expanding block AND the output of corresponding contracting block. This ensures the features learned during the encoding phase are re-used to reconstruct the image in the decoding one.
3) The bottleneck section connects the encoding and decoding sections.

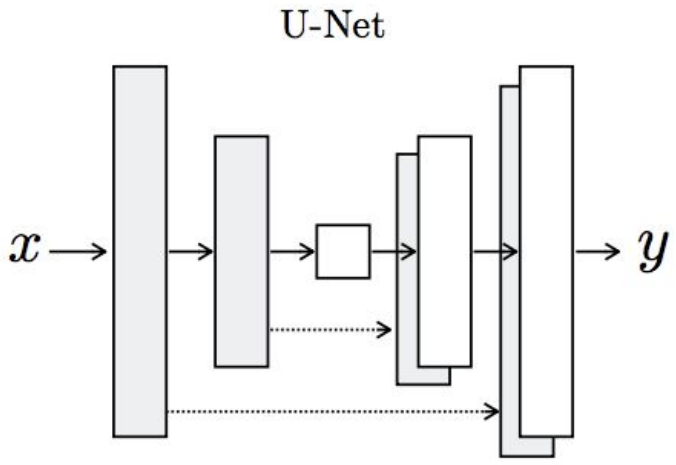

Figure 5: U-Net Architecture of the Generator Model

The **composite**:

It connects the discriminator and generator model together. A source image (image without bonds) is fed to the generator to create a generated image. Then, both corresponding target image (image with bonds) and the generated image are fed into the discriminator, which classifies if the two images match and updates the weight of the generator only (the weights of the discriminator are updated in a standalone manner).

The **train model**:

Each training step consists of using the composite but also fake generated images from the generator. The idea is that a combination of real target and fake generated images, respectively classified as 1 and 0 will help the generator build a model that converges faster toward a solution.