Predicting Frictional Properties of Graphene Kirigami Using Molecular Dynamics and Neural Networks

Designs for a negative friction coefficient.

Mikkel Metzsch Jensen

Thesis submitted for the degree of
Master in Computational Science: Materials Science
60 credits

Department of Physics
Faculty of mathematics and natural sciences

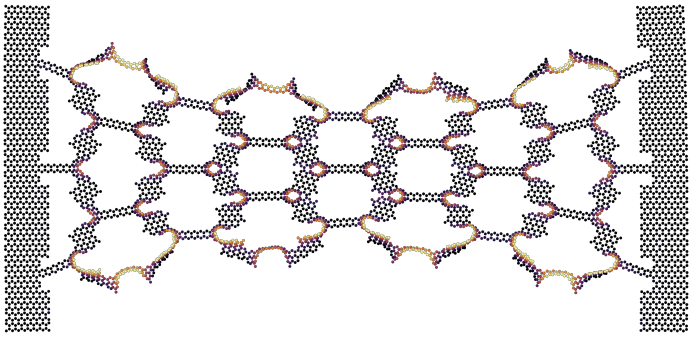
UNIVERSITY OF OSLO

Spring 2023

Predicting Frictional Properties of Graphene Kirigami Using Molecular Dynamics and Neural Networks

Designs for a negative friction coefficient.

Mikkel Metzsch Jensen



Predicting Frictional Properties of Graphene Kirigami Using Molecular Dynamics and Neural Networks

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Abstract.

Acknowledgments

Acknowledgments.

List of Symbols

F_N Normal force (normal load)

Acronyms

CM Center of Mass. 8

CNN Convolutional Neural Network. 30

FFM Friction Force Microscopes. 5

LJ Lennard-Jones. 51

MD Molecular Dynamics. 5, 23, 30, 37, 41, 44, 49, 50

ML Machine Learning. 38, 39, 40, 41, 42, 43

MSE Mean Squared Error. 31

NN Nearest neighbours. 10

SFA Surface force apparatus. 5

Contents

I Background Theory 1

II Simulations 3

1 Creating a graphene kirigami system 5
- 1.1 Region definitions . . . 5
- 1.2 Numerical procedure . . . 8
- 1.3 Setting up the substrate . . . 9
- 1.4 Setting up sheet . . . 9
  - 1.4.1 Indexing . . . 10
  - 1.4.2 Removing atoms . . . 11
- 1.5 Kirigami patterns . . . 12
  - 1.5.1 Tetrahedron . . . 12
  - 1.5.2 Honeycomb . . . 14
  - 1.5.3 Random walk . . . 15
    - 1.5.3.1 Fundamentals . . . 16
    - 1.5.3.2 Spacing of walking paths . . . 17
    - 1.5.3.3 Bias . . . 17
    - 1.5.3.4 Stay or break . . . 18
    - 1.5.3.5 Deployment schemes . . . 19
    - 1.5.3.6 Validity . . . 20
    - 1.5.3.7 Random walk examples . . . 21

2 Kirigami configuration exploration 23
- 2.1 Generating the dataset . . . 23
- 2.2 Data analysis . . . 24
- 2.3 Properties of interest . . . 26
- 2.4 Machine learning . . . 30
  - 2.4.1 Architecture . . . 30
  - 2.4.2 Data handling . . . 31
    - 2.4.2.1 Input . . . 31
    - 2.4.2.2 Output . . . 31
    - 2.4.2.3 Data augmentation . . . 31
  - 2.4.3 Loss . . . 31
  - 2.4.4 Hypertuning . . . 32
  - 2.4.5 Final model . . . 37
- 2.5 Accelerated Search . . . 41
  - 2.5.1 Patteren generation search . . . 41
  - 2.5.2 Genetic algorithm search . . . 45

3 Summary **49**
3.1 Summary and conclusions 49
3.1.1 Design MD simulations 49
3.1.2 Design Kirigami framework 49
3.1.3 Control friction using Kirigami 49
3.1.4 Capture trends with ML 50
3.1.5 Accelerated search 51
3.1.6 Negative friction coefficient 51
3.2 Outlook / Perspective 51

Appendices **53**

A Appendix A **55**

A Appendix B **57**

B Appendix C **59**

Part I

Background Theory

Part II

Simulations

Chapter 1

Creating a graphene kirigami system

The system definition plays an essential role in the "friction experiment" that we are going to carry out through MD simulations. The purpose of the simulations is to quantify the friction that arises when a stretched Kirigami graphene sheet slides over a substrate. Thus we aim to design the simplest possible system that allows such a measurement for different variations on Kirigami design, stretch and load.

For this purpose, two approaches were considered as sketched in Fig. 1.1. One approach is simply to mimic a FFM type experiment where the graphene sheet is resting on a substrate and a moving body scans across the graphene surface. This setup allows for a variety of tip designs connected to the moving body, and alternatively, the tip can be substituted with a flat surface making the setup resemble a SFA experiment instead. For this setup, we would have to attach a pre-stretched sheet to the substrate and require the edges to be fixated in order to ... the stretch sustain the stretch. We would essentially regard the sheet as a part of the substrate, and the possible applications would regard to certain effects being associated constant stretch. Another approach is to have the sheet ends fixated on the moving body instead. This allows for the connection to a nanomachine design which couples the load and the stretch of the sheet. Thus, the possible applications allow for a dynamic effect with changing stretch through the loading of the sheet. While both methods serve as novel approaches with prospects of providing valuable insight into a sparsely covered field, we choose the latter option due to the increased application possibilities.

We do not attempt to model the nanomachine explicitly, but we will use the conceptual idea of a coupling between load and stretch to motivate our study. Our system of choice consists of a 2D graphene sheet with locked ends, mimicking the attachment to a moving body, and a 3D silicon bulk substrate.

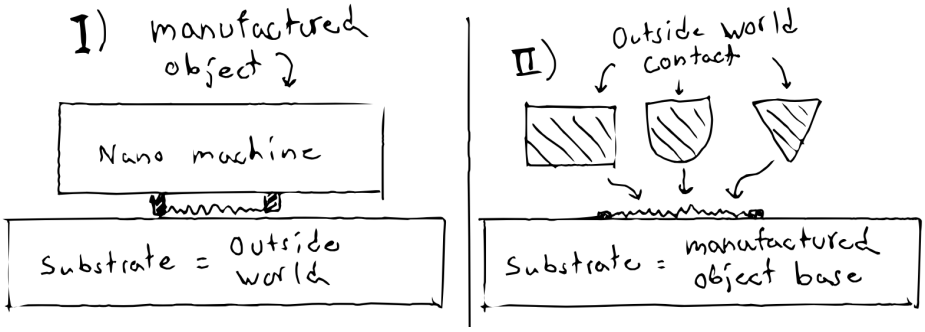


Figure 1.1: TMP System variations

1.1 Region definitions

We subdivide the two main parts of the simmulation, the sheet and the substrate, into specific regions according their functionality in the MD simmulations. For the sheet, we denote a subsection of the ends, with respect to the sliding direction, as so-called *pull blocks*, which is reserved for the application of normal load, stretching and dragging of the sheet, and for applying the thermostat. The remaining *inner sheet* is left for the kirigami cuts

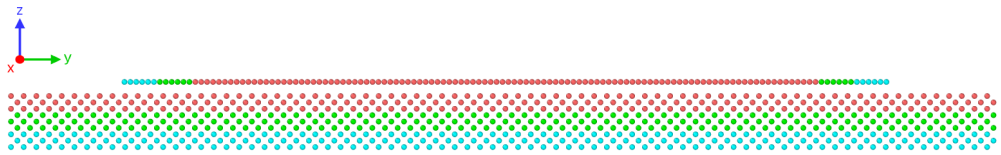
and are simulated as an NVE ensemble. The pull blocks are equally split between a thermostat part and a rigid part. The rigid part is however thermolized during the initial relaxation period but made rigid for the final duration of the simulation. Note that the rigid parts on both sides of the sheet is then considered as a single rigid object even though they are physically seperated. This means that all force interactions concerning the rigid parts will be applied as a common average making them move in total synchronization. The substrate is equally divided into three parts: The *upper layers* (NVE) responsible for the sheet-substrate interaction, the *middle layers* being a thermostat (NVT), and the *bottom layers* being frozen, made rigid and fixed, in the initial lattice structure to ensure that the substrate stays in place. In Fig. 1.2 the system is displayed with colors matching the three distinct roles:

1. Red: NVE parts which is governing the frictional behaviour of interest.

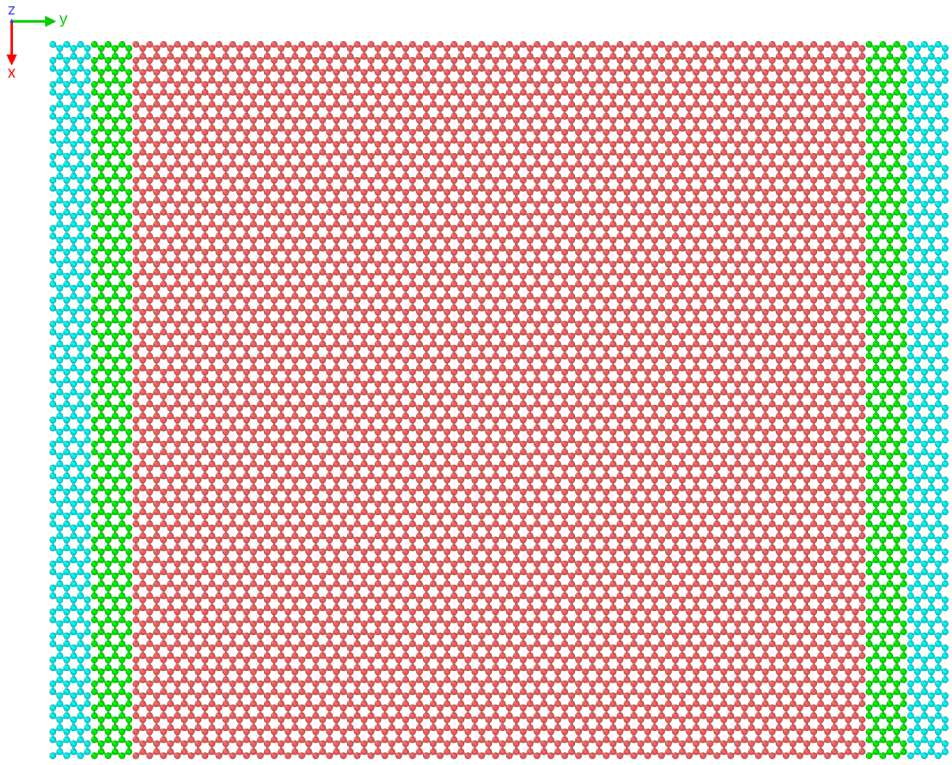
2. Green: Thermostats (NVT) sourrounding the NVE parts in order to modify the temperature without making disturbing changes to the interaction of the sheet and substrate.

3. Blue: Parts that are initially or eventually turned in to rigid objects. For the substrate this refers to an addtionally fixation as well.

The full sheet is given a size $\sim 130 \times 163$ Å while the substrate is scaled accordingly to the sheet which is further specified in Sec. 1.3. For an expected stretch of 200% the total system size is roughly 55k atoms. The specific distribution is shown in Table 1.2 along with the spatial x-y-measures in Table 1.1.



(a) Side view showing sheet on top of the substrate.



(b) Top view showing only the sheet.

Figure 1.2: System configuration colorized to indicate NVE parts (red), thermostat parts (green) and rigid parts (blue).

Table 1.1: Specification of the spatial size of the system for the x-y-dimensions with a substrate scaled for an expected stretch of 200%. The first column denotes the size relative to the full sheet size $x_S \times y_S$, while the second column denotes the corresponding length in Å.

Region	Dim	Dim [Å]	Area [nm^2]
Full sheet	$x_S \times y_S$	130.029 × 163.219 Å	212.23
Inner sheet	$x_S \times 0.81\ y_S$	130.029 × 132.853 Å	172.74
Pull blocks	$2 \times x_S \times 0.09\ y_S$	2 × 130.029 × 15.183 Å	2 × 19.74
Substrate	$1.16\ x_S \times 3.12\ y_S$	150.709 × 509.152 Å	767.34

Table 1.2: Specification of the system size in number of atoms for various system regions. These numbers corresponds with the case of no cuts applied to the sheet and a substrate scaled for the expected stretch of 200 %.

Region	**Total**	Sub region	Sub total	**NVE**	**NVT**	**Rigid**
Full sheet	7800	Inner sheet	6360	6360	0	0
		Pull blocks	1440	0	720	720
Substrate	47376	Upper	15792	15792	0	0
		Middle	15792	0	15792	0
		Bottom	15792	0	0	15792
All	55176			22152	16512	16512

1.2 Numerical procedure

After implementing the initial configuration to the system we first need to let the system relax and reach a stable equilibrium state. This involves an adjustment to the system temperature and the sheet-substrate distance. We then stretch the sheet to the desired length before applying normal load the pull blocks. The full numerical procedure can be arranged into the following steps. Some steps have been given a default duration denoted in parentheses in units of ps, 10^{-12} seconds.

1. **Relaxation** (15 ps): The sheet and substrate are relaxed for 15 ps after being added in their crystalline form with a separation distance of 3 Å. Given that the equilibrium separation distance will vary with temperature, this value is based on an average estimate suiting our temperature range of interest. In order to avoid any sheet drift we constrain it with the use of three hard spring forces with a spring constant $10^5\,\mathrm{eV/\AA^2} \sim 1.6 \times 10^6\,\mathrm{N/m}$: One spring attaches the sheet center of mass (CM) to its original position, preventing CM drift, while the remaining two springs are attached to the pull blocks CM, to prevent rotation. In principle, it would be sufficient to fixate only one of the pull blocks, but we fixate both for the sake of symmetry. During the relaxation phase, we consider the pull blocks to be rigid with respect to the z-direction only (perpendicular to the sheet). That is, all the forces in the z-direction are summed up and applied as a uniform external force, while the pull blocks are free to expand and contract in the x-y-plane. This feature is introduced to allow the sheet pull blocks to readjust the lattice spacing according to the temperature of the system. For the following phases, the pull blocks are made truly rigid with respect to all directions, and the spring forces are terminated.

2. **Stretch**: The sheet is stretched by seperating the two opposing rigid parts of the pullblock at constant velocity until the desired stretch amount is met. The duration of this phase is thus governed by the *stretch speed* and *stretch amount* parameters.

3. **Pause** (5 ps): The sheet is relaxed for 5 ps to ensure that the sheet is stable and equilibrized after the applied stretch deformation.

4. **Normal load** (5 ps): A normal load is applied to the rigid parts of the pull blocks. Initially a viscous damping force, $F = -\gamma \mathbf{v}$, is added to the sheet to resist the rapid acceleraction of the sheet and prevent a hard impact between the sheet and substrate. The damping coefficient is set to $\gamma = 8 \times 10^{-4}\,\mathrm{nN/(m/s)}$ and terminated after 0.5 ps which was found to be suitable for the extreme load cases of our intended range. The remaining 4.5 ps is simply devoted for further relaxation.

5. **Sliding**: A virtual atom is introduced into the simulation which exclusively interacts with the rigid parts of the pull blocks through a spring force with variable spring constant K in the x-y-plane. The z-direction is not affected by the spring force and is purely governed by the balancing forces of the normal load and the normal response from the sheet-substrate interaction. The virtual atom is immediately given a constant velocity, in accordance with the *sliding speed* parameter, which makes the sliding force increase linearly, $F_{slide} \propto K v_{\mathrm{slide}} t$, with sliding speed. An infinite spring constant can also be enforced for which the spring is omitted and the pull blocks are moved rigidly with a constant speed according to the sliding speed.

To limit the complexity of our prediction task, we want to consider systems without wear. To make sure that no wear is taking place for the sheet, we monitor the nearest neighbors for each atom throughout the simulation. At

the initial timestep the three nearest neighbors, sitting at a distance 1.42 Å, of all graphene atoms are recorded. If any of these nearest neighbors exceeds a threshold distance of 4 Å, indicating a bond breakage, this is marked as a rupture and we halt the simulation. The substrate was proven to be way more resistant to wear than the sheet, and by running a few test simulations of high load and sliding speed we confirmed visually that no wear is occurring.

1.3 Setting up the substrate

The substrate is created as a rectangular slab of Silicon (Si). We create the initial configuration according to its crystalline structure given as a diamond cubic crystal with a lattice parameter $a_{\mathrm{Si}} = 5.43\,\text{Å}$. The default substrate thickness is chosen such that 9 layers of atoms appear (2 unit cells) corresponding to a thickness of 10.86 Å. The x-y dimensions are chosen to match the dimensions of the sheet. That is, we define a margin between the sheet edge and the substrate edge for the x- and y-direction respectively. Since we use periodic boundary conditions a too small margin would result in the sheet edges interacting with themselves through the boundary. The absolute lower limit for the margin choice is thus half the cut-off distance for the Tersoff potential, governing the graphene sheet interaction, at $R + D = 2.1$ Å. However, due to fluctuations in the sheet, we cannot set the margin too close to that limit. Additionally, we must take into account the buckling of the sheet as it is stretched, which might induce an expansion in the x-direction for certain configurations. We choose a x-margin of 20 Å which provides $2 \cdot 20\,\text{Å} - 2.1\,\text{Å} = 37.9\,\text{Å}$ of additional spacing with respect to the absolute lower limit. By looking over the simulation result visually we confirm that this leaves more than enough room in the cases of extreme buckling. For the y-direction the rigid parts of the pull-blocks moves a certain distance based on the stretch value exclusively, and we define the y-margin based on the remaining distance to the edge after stretching. However, as the sheet travels through the periodic boundaries in the y-direction when sliding, we want to add some additional spacing through the y-margin in order to let the substrate surface relax before interacting with the sheet a second time. We choose a y-margin of 15 Å for which the preferred sliding speed of $20\,\text{m/s} = 2\,\text{Å/ps}$ gives 15 ps of relaxation time between encounters with the sheet, similar to the initial relaxation time.

1.4 Setting up sheet

The sheet consists of graphene, which is a single layer of carbon atoms arranged in a hexagonal lattice structure. The bulk version of graphene is graphite and is a stacked structure of multiple graphene layers. We can describe the 2D crystal structure in terms of its primitive lattice vectors $\mathbf{a_1}$ and $\mathbf{a_2}$ and a basis. The basis describes the atoms associated with each lattice site, and we populate the lattice by translating the basis by any linear combination of the lattice vectors

$$\mathbf{T}_{mn} = m\mathbf{a_1} + n\mathbf{a_2}, \qquad m, n \in \mathbb{N}.$$

For graphene, we have the primitive lattice vectors

$$\mathbf{a_1} = a\left(\frac{\sqrt{3}}{2}, -\frac{1}{2}\right), \qquad \mathbf{a_2} = a\left(\frac{\sqrt{3}}{2}, \frac{1}{2}\right), \qquad |\mathbf{a_1}| = |\mathbf{a_2}| = 2.46\ \text{Å}.$$

Notice that we deliberately excluded the third coordinate as we only consider a single graphene layer and thus we do not have to consider the stacking structure of 3D graphite. The basis consist of two carbon atoms given as

$$\left\{\Big(0,0\Big), \frac{a}{2}\Big(\frac{1}{\sqrt{3}}, 1\Big)\right\}$$

The crystal structure is visualized in Fig. 1.3. It turns out that the spacing between atoms is equal for all pairs of atoms with an interatomic distance

$$\left|\left|\frac{a}{2}\Big(\frac{1}{\sqrt{3}}, 1\Big)\right|\right| \approx 1.42\ \text{Å}.$$

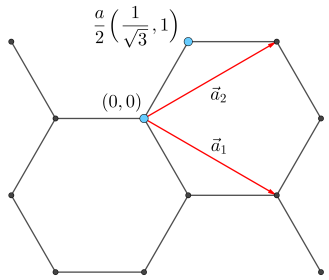


Figure 1.3: Graphene crystal structure with basis.

1.4.1 Indexing

In order to define the kirigami cut patterns applied to the graphene sheet we need to define an indexing system. We must ensure that this gives a unique description of the atoms as we eventually want to pass a binary matrix, containing 0 for removed atoms and 1 for present atoms, that uniquely describes the sheet. We do this by letting the x-coordinate allign with the so-called *armchair* direction of the sheet and making the y-coordinate increment along the so-called *zigzag* direction. Notice that the x-coordinate will point to *zigzag* chains of atoms for which the starting point, at $y = 0$ is not evenly spaced as illustrated in figure Fig. 1.4. Other solutions might naturally involve the lattice vectors, but since these are used to translate between similar basis atoms it introduces an unfortunate duality as one would then need to include the basis atom of choice into the indexing system as well. Additionally, we want an indexing system that conserves the relative physical position of neighbors. That is, atom (i, j) should be in the proximity of $\{(i+1, j), (i-1, j), (i, j+1), (i, j-1)\}$. However, due to the hexagonal structure of the lattice, only three said neighbor indexes willl be actual nearest neighbors in the lattice. While $(i, j \pm 1)$ is always a nearest neighbor, the index of the nearest neighbor in the x-direction oscillates for each incrementing of the x- or y-coordinate. That is, the nearest neighbors are decided as

$$\begin{aligned} (i+j) \text{ is even} &\rightarrow \text{NN} = \{(i-1, j), (i, j+1), (i, j-1)\}, \\ (i+j) \text{ is odd} &\rightarrow \text{NN} = \{(i+1, j), (i, j+1), (i, j-1)\}. \end{aligned} \tag{1.1}$$

By consulting Fig. 1.4 we can verify this visually, and it basically comes down to the fact whether the atom is oriented to the right or the left side in the zigzag chain.

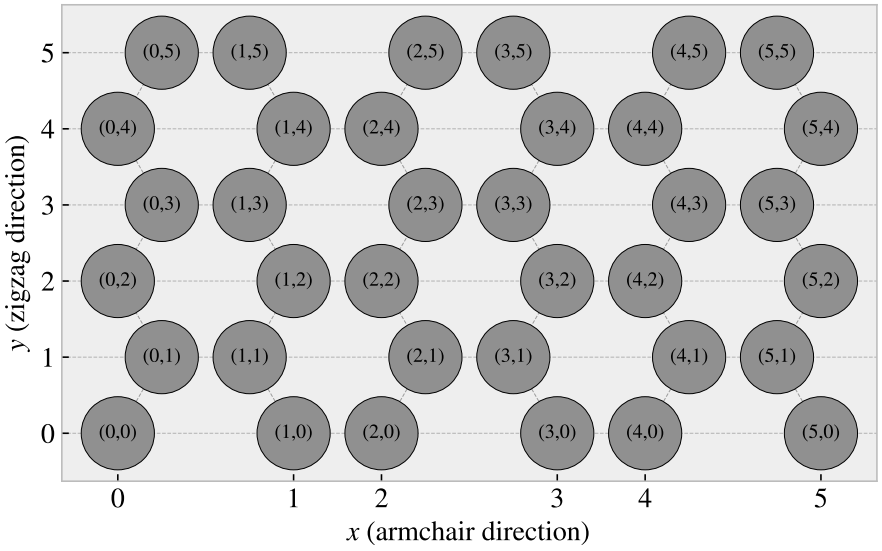


Figure 1.4: Graphene atom indexing

1.4.2 Removing atoms

As a means to ease the formulation of the cut patterns we introduce the *center element* placed in each gap of the hexagonal honeycomb structure as shown in figure Fig. 1.5. These are not populated by any atoms but will serve as a temporary reference for the algorithmic approaches of defining a cut pattern.

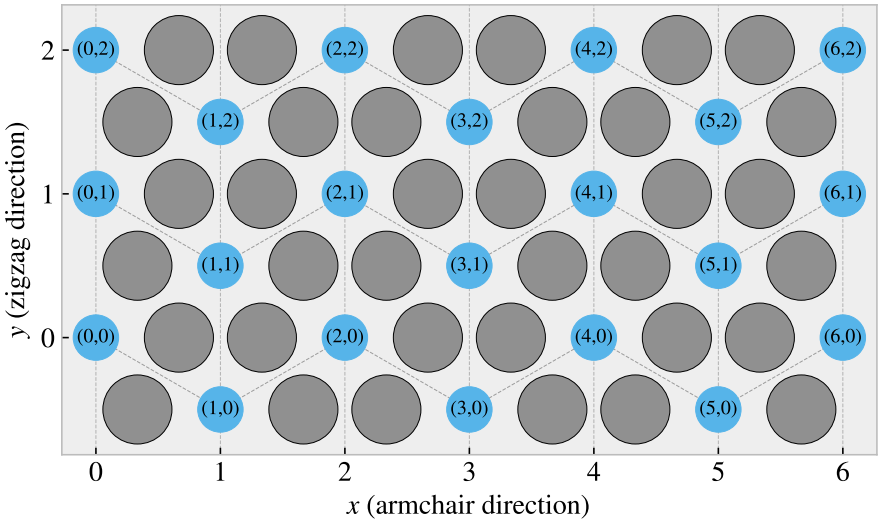


Figure 1.5: Graphene center indexing

Similar to the case of the atom indexing, the nearest neighbours center elements alternate with position, this time only along the x-coordinate index. Each center element has six nearest neighbours, in clockwise direction we can denote them: “up”, “upper right”, “lower right”, “down”, “lower left”, “upper left”. The “up” and “down” is always accesed as $(i, j \pm 1)$, but for even i the $(i+1, j)$ index corresponds to the “lower right” neighbour while for odd i this corresponds to the “upper right” neighbour. This shifting applies for all left or right oriented neighbours and the full neighbour list is illustrated in Fig. 1.6.

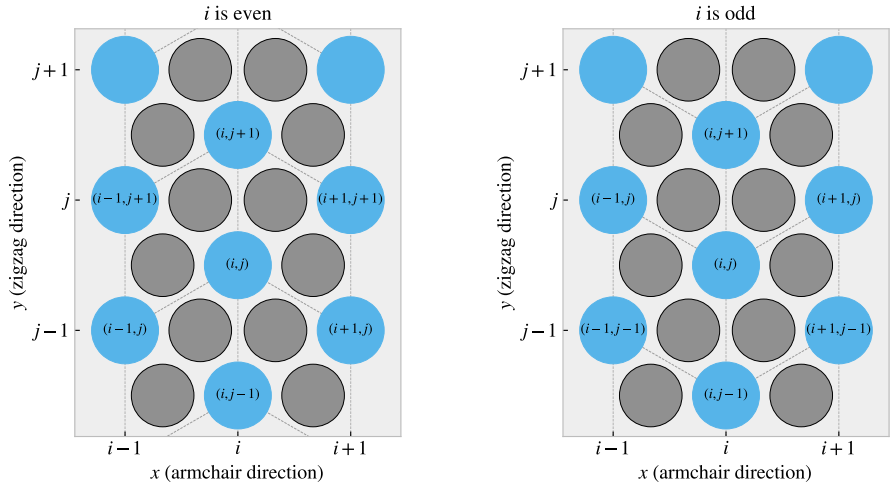


Figure 1.6: Graphene center elements directions

We define a cut pattern by connecting center elements into connected paths. As we walk from center element to center element we remove atoms according to one of two rules

1. Remove intersection atoms: We remove the pair of atoms placed directly in the path we are walking. That is, when jumping to the "up" center element we remove the two upper atoms located in the local hexagon of atoms. This method is sensitive to the order of the center elements in the path.

2. Remove all surrounding atoms: We simply remove all atoms in the local hexagon surrounding each center element. This method is indepdent of the ordering of center elements in the path.

We notice that removing atoms using either of these rules will not garuantee an injective, one to one, mapping. The first rule, being path dependent, will more often result in a unique result. However, for both methods it is possible to construct two different paths leading to the same cut pattern as shown in the following example:

$$\text{Path 1:}\quad (i,j) \to \underbrace{(i+1,j+1)}_{\text{upper right}} \to \underbrace{(i,j+1)}_{\text{up}} \to \underbrace{(i+1,j+2)}_{\text{upper right + up}} \to \underbrace{(i+1,j+1)}_{\text{upper right}}$$

$$\text{Path 2:}\quad (i,j) \to \underbrace{(i+1,j+1)}_{\text{upper right}} \to \underbrace{(i+1,j+2)}_{\text{upper right + up}} \to \underbrace{(i,j+1)}_{\text{up}}$$

Illustrate the example path, because I think it is otherwise impossible to follow.

For the second rule it is even more abovious that different paths can result in the same final pattern. For instance, if we incircle a center element completely there will be no surrounding atoms left to delete when jumping to that center element. This highlights the importance of defining the atom based indexing system will yield an injective mapping for the binary cut matrix. However, using the center elements for reference makes the following definitions of the cut patterns a lot easier to design as we always can always go in the same six directions as opposed to the atom indexing system which have alternating directions for its neighbours.

1.5 Kirigami patterns

We propose a series of kirigami inspired cut patterns for the altering of the graphene sheet. We seek inspiration from macroscale patterns that showcases a considerable amount of out of plane buckling when stretched. We choose to imitate two different designs: 1) An alternating repeating series of perpendicular cuts as shown in Fig. 1.7a popularly used in studies of morphable metematerials [1]. This patteren produce surface buckling with a tetrahedron (three sided pyramid) shape when stretched. 2) A more intricate pattern shown in Fig. 1.7b which is used commercially by Scotch$^{\text{TM}}$ Cushion Lock$^{\text{TM}}$ [2] as protective wrap for items during shipping. This pattern buckles into a hexagoal honeycomb structure when stretched. In addition to the modeling of the so-called *Tetrahedron* and *Honeycomb* patterns we also create a series of random walk cut patterns.

(a) **(b)**

Figure 1.7: Macroscale kirigami cut patterns used as inspiraton for the nanoscale implementation. (a) Tetrahedron: Alternating perpendicular cuts producing a tetrahedron shaped surface buckling when stretched [1]. (b) Honeycomb: Scotch$^{\text{TM}}$ Cushion Lock$^{\text{TM}}$ [2] producing a honeycomb shaped surface buckling when stretched.

1.5.1 Tetrahedron

The *Tetrahedron* pattern is defined in terms of center elements for which all atoms sourrounding a given center element are removed. The pattern is characterized by two straight cuts, denoted line 1 and line 2, arranged

perpendicular to each other. This is done in such a way such that one line aligns with the center of the other line and with a given spacing in between. This is illustarted in Fig. 1.8. In order to achieve perpendicular cuts we cannot rely purely on the six principal directions corresponding to the center element neighbours which is spaced by 60°. We let line 1 run along the center elemnts in the direction of the "upper right" (and "lower left") center elements while line 2 goes in the direction between the "down" and "lower right" ("up" and "upper left") center elements, corresponding to the direction $(1/\sqrt{3}, -1)$. We define variations of the pattern by the number of center elements L_1 and L_2 in line 1 and 2 respectively, together with the spacing between the lines d, as the tuple (L_1, L_2, d). The pattern is constructed by translating the two lines to the whole sheet according to the spacing. Due to the alignment criterias of having one line point to the center of the other line we can only have odd line length. Furthermore, in order to ensure that each center element is translated to an i-index of similar odd or evenness, we must in practice require that $|L_2 - L_1| = 2, 6, 10, \ldots$. Fig. 1.8 shows a visual representation of the pattern components for the $(7, 5, 2)$ patteren.

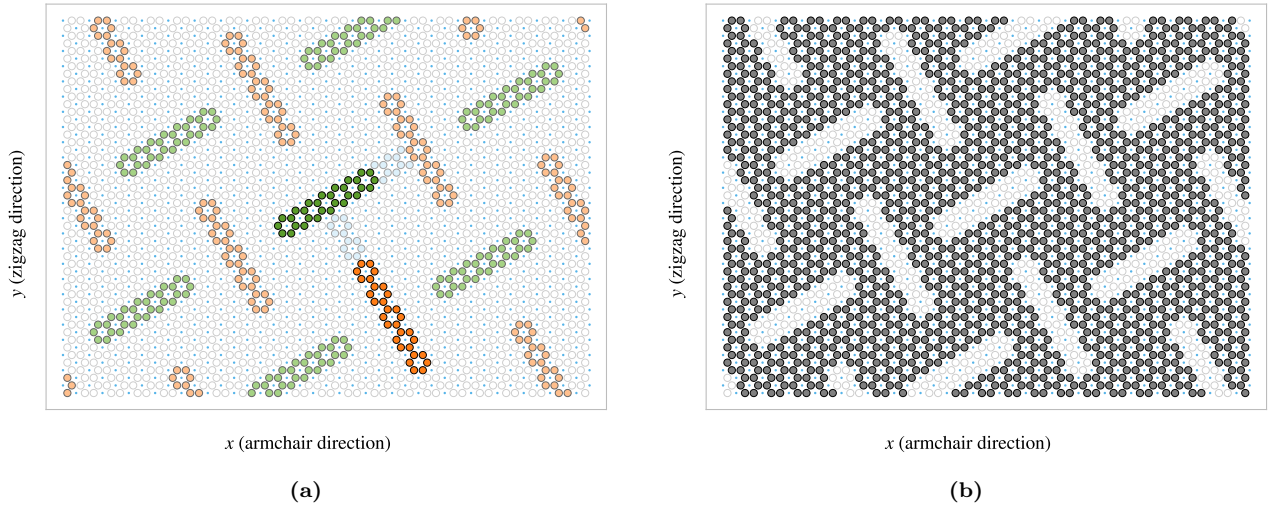


Figure 1.8: Visual representation of the tetrahedron pattern consisting of two perpendicular lines, line 1 and line 2, of length L_1 and L_2 respectively with spacing d. This example used $(L_1, L_2, d) = (7, 5, 2)$. (a) Highlight of the atoms removed. Line 1 is shown in green and line 2 in orange, with lighter colors for the translated variations, and the spacing is shown in light blue. (b) The sheet after applying the cut pattern where the grey circles denote atoms and the transparent white denotes removed atoms. The small blue circles show the center elements for reference. Sheet size used in example

In addition to the three parameters L_1, L_2, d, the pattern is also anchored to a reference point which describes the position of line 1 and 2 before translating to the whole sheet. Due to the repeating structure of the pattern there exist a small finite number of unique reference positions. For the pattern $(7, 5, 2)$ used as an example in Fig. 1.8, there are 140 [1] unique reference points. Some additional variation of the pattern is showcased in Fig. 1.9 all with a reference position in the center of the sheet. Note that a smaller sheet size is used in both Fig. 1.8 and Fig. 1.9 for illustrative purposes.

[1] The general formula for this number is rather complicated in comparision to the importance in this context. Thus, we exclude the formula for this calculation as the derivation is rather handwavy executed, but the number stated here is numerically backed for this specific parameter set.

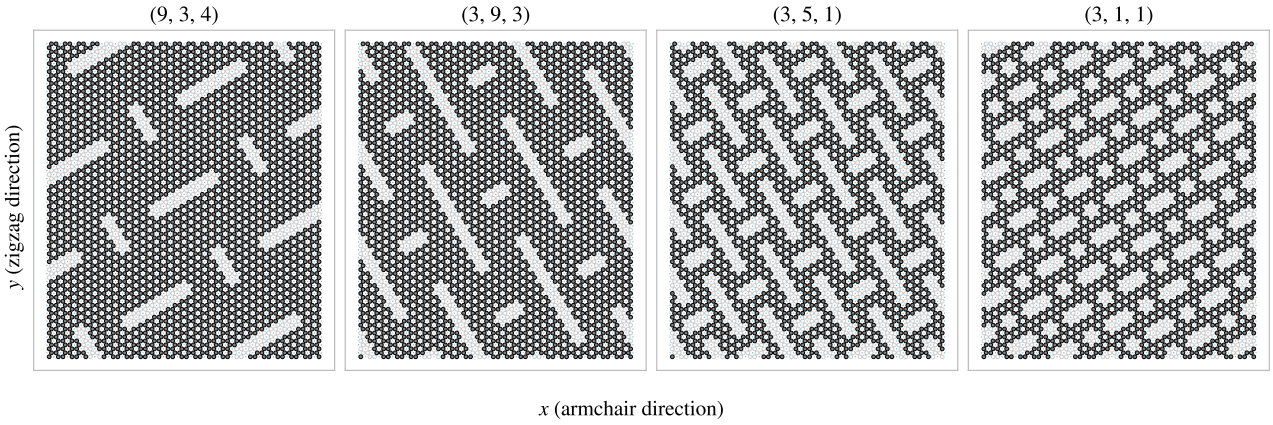
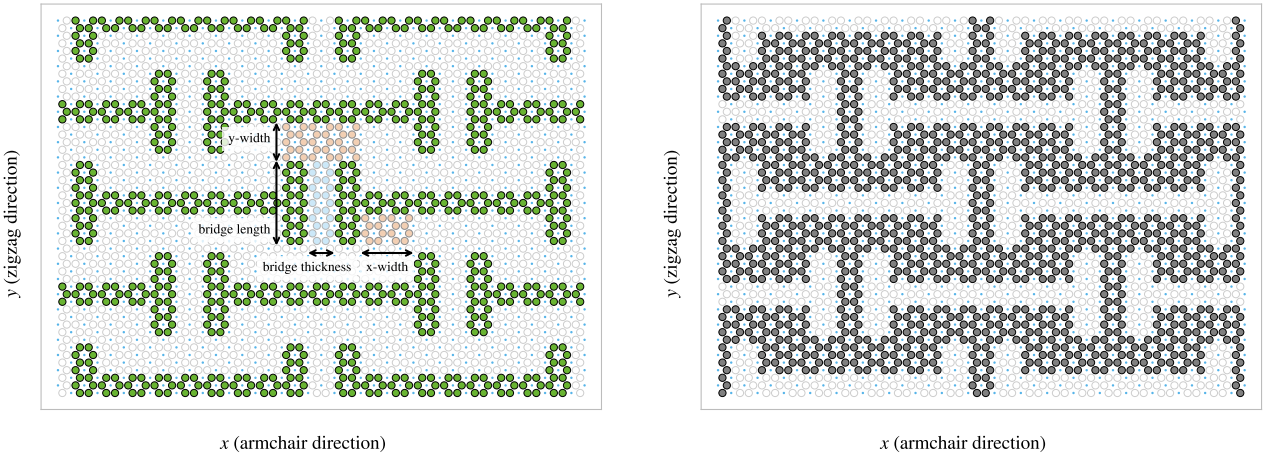


Figure 1.9: Sheet size used in example

1.5.2 Honeycomb

The *Honeycomb* pattern is defined, similarly to the Tetrahedron pattern, in terms of the center elements for which all sourrounding atoms are removed. The Honeycomb pattern is build from a repeating series of cuts remniscient of the roman numeral one rotated by 90° (⊢). For a given spacing these are put next to each other in the x-direction, ⊢ ⊢ ⊢, to achieve a row where only a thin *bridge* in between is left to connect the sheet vertically in the y-direction. By placing multiple rows along the y-direction with alternating x-offsett we get the class of honeycomb patterns as visualized in Fig. 1.10. The pattern is described in terms of the parameters: (x-width, y-width, bridge thickness, bridge length) which is annotated in Fig. 1.10a with the parameters (2, 2, 1, 5) used as an example. Some additional variations of the pattern class is showcased in Fig. 1.11.



(a) **(b)**

Figure 1.10

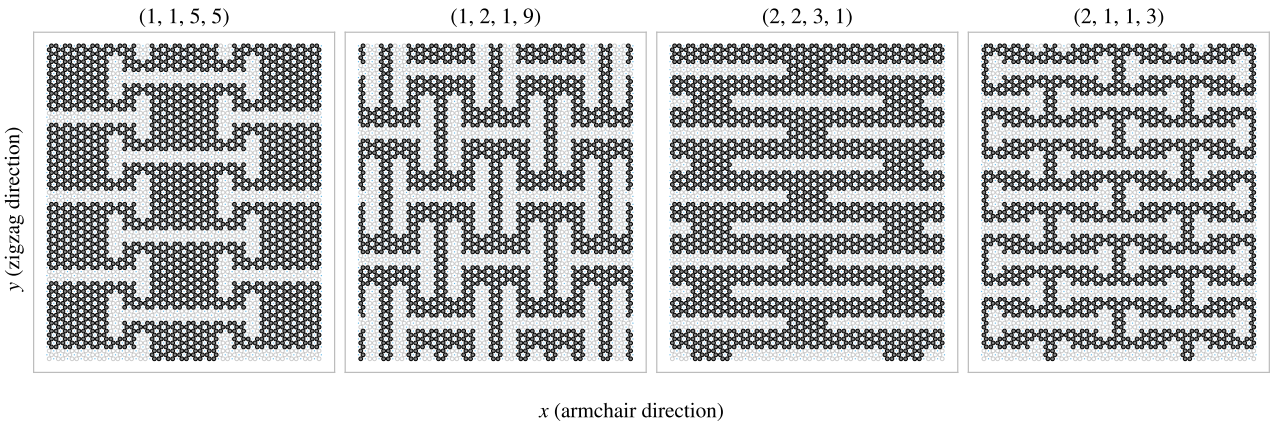


Figure 1.11

1.5.3 Random walk

The random walk serves as a method for introducing random patterns into the dataset with the scope of populating the configuration space more broadly than achieved purely with the more systematic patterns described above. By this argument, a straightforward way to create random configurations could be achieved simply by random noise, either uniform or gaussian. However, this would often leave the sheet detached with lots of non-connected atom clusters. Intuitively, we do not find this promising for the generation of large scale organized structures which we hypothesize to be of interest. The random walk pattern generation is characterized by the parameters summarized in Table 1.3 which will be introduced throughout the following paragraphs.

Table 1.3: Parameters for the random walk generator.

Parameters	Value	Description
Num. walkers (M)	Integer ≥ 1	Number of random walks to be initiated on the sheet (one at a time).
Max. steps (S)	Integer ≥ 1	The maximum steps allowed for any random walker.
Min. distance	Integer ≥ 0	The minimum distance required between any future paths and the previous paths in terms of the shortest walking distance in between.
Bias	(Direction, strength ≥ 0)	Bias direction and strength defining the discrete probability for the chocie of the next site.
Connection	Atoms / Center elements	Whether to walk between atom sites or center elements (removing all adjecent atoms).
Avoid invalid	True/False	Whether to remove already visited sites from the neighbour list before picking the next site. This prevents jumping to already visited sites and lowers the likelihood of early termination.
Stay or break	$p = [0, 1]$	Probability that the walker will maintain its direction for the next step.
Periodic	True/False	Whether to use periodic boundary conditions on all four sides.
Avoid clustering	Integer ≥ 0	Amount of times to restart the whole random walk generation in order to arrive at a non-detached configuration. If no valid configuration is reached after this amount of tries, the non-spanning clusters are removed.
RN6	True/False	Randomly change the bias direction between the deployment of each random walker to one of the six center element directions.
Grid start	True/False	The option to have the random walkers start in an evenly spaced grid.
Centering	True/False	Relocate the path of a random walk after termination such that the path center of mass gets closer to the starting point (without violating the rules regarding already visited sites).

1.5.3.1 Fundamentals

For an uncut sheet we deploy M random walkers, one at a time, and let them walk for a maximum number of S steps. We can either let the walker travel between atom sites, removing the atoms in the path as it goes, or between center elements, removing all sourrounding atoms — *Connection: Atom/Center elements*. The method of removing only the intersecting atoms between center elements was also incorporated, but we ended up not using it due to plenty of other interesting options. Nonetheless, we will always remove a site once visited such that the walker itself, or any other walkers, cannot use this site again. This corresponds with the property of a self avoiding random walk, but it futhermore constraint the walkers not to visit any path previously visited by others walkers which we might denote "other avoiding" then. By default, the walker has an equal chance of chosing any of its adjecent neighbours for the next step, i.e. we draw the next step from a discrete uniform distribution. Optionally we can use periodic bounary conditions, *Periodic: True/False*, allowing neighbouring sites to be connected through the edge in both the x and y-direction. When traveling on atom sites this ensures that we have three neighbour options for the next step while traveling on the center elements this gives six neighbour options. If the walker happens to arrive at an already visited site the walk is terminated early. Optionally, we can choose to remove any neighbouring sites already visited from the neighbour list, *Avoid invalid: True/False*, and choose uniformly between the remaining options instead. This prolongs the walking distance, but the walker

is still able to find itself in a situation where no neighboring sites are available. Note that it cannot backtrack its own path either, and in such a case the walk will be terminated despite the setting of *Avoid invalid*.

1.5.3.2 Spacing of walking paths

In order to control the spacing between the paths of the various walkers we implement a so-called, *minimum distance: 0, 1, ...*, parameter describing the spacing required between paths in terms of the least amount of steps. When a walker has ended its walk, either by early termination or hitting the maximum limits of steps, all sites within a walking distance of the minimum distance are marked as visited, although they are not removed from the sheet. This prevents any subsequent walkers to visit those sites in their walk according to the general behaviour introduced in the previous paragraph. In practice this is done through a recursive algorithm as described in algorithm Algorithm 1. For a given path the function *walk_distance()* is called with the input being a list of all sites in the given paths. For each site, the function then gathers all site neighbours (regardless of their state on the sheet) and call itself using this neighbour list as input while incrementing a distance counter passed along. This will result in an expansion along all possible outgoing paths from the initial path of interest which is terminated when the distance counter hits the distance limit. The function will then return the final neighbour lists which are accumulated into a final output corresponding to a list of all sites within the minimum distance to the path.

Algorithm 1 Recursive algorithm implemented as class method to mark sites within a distance of the class attribute self.min_dis.

```
Require: self.min_dis > 0                          ▷ This pseudocode does not handle other cases
function WALK_DISTANCE(self, input, dis = 0, pre = [ ])
    new_neigh ← [ ]                                ▷ Initialize list for new neighbours
    for site in input do
        neigh ← get_neighbouring_sites(site)       ▷ Get sourrounding neighbours
        for n in neigh do
            if (n not in pre) and (n not in new_neigh) then   ▷ If not already added
                AddItem(new_neigh, n)
            end if
        end for
    end for
    dis += 1                                       ▷ Increment distance counter
    if dis ≥ self.min_dis then                     ▷ Max limit hit
        return input + new_neigh
    else                                           ▷ Start a new walk from each of the neighbouring sites
        pre ← input
        return pre + self.walk_distance(new_neigh, dis, pre)
    end if
end function
```

Do we need a figure supporting this?

1.5.3.3 Bias

We include the option to perform biased random walk through the Bias: (direction, strength) parameter option. We implement this by modelling each walking step analog to the canonical ensemble under the influence of an external force **F** representing the bias. For such a system each microstate i, corresponding to the sites in the neighbour list, has the associated probability p_i given by the Gibbs–Boltzmann distribution

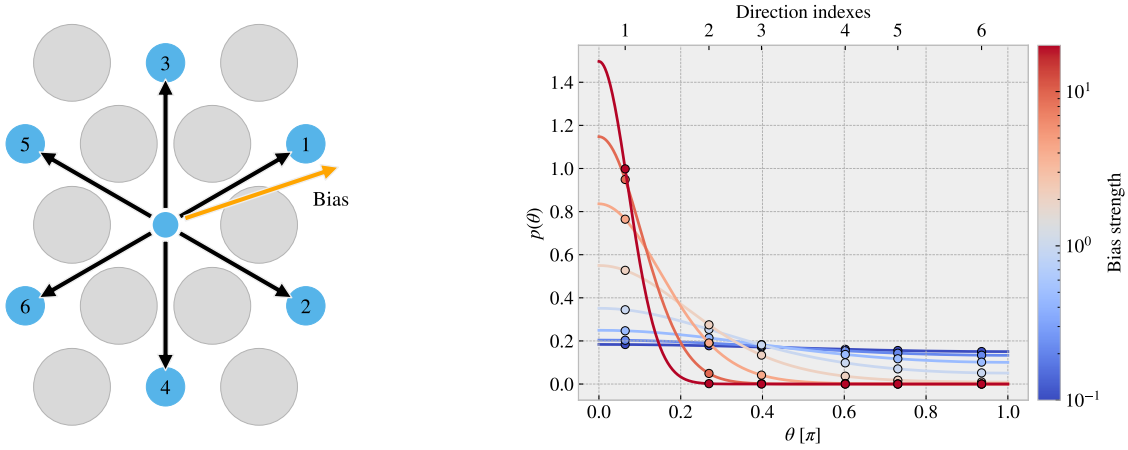
$$p_i = \frac{1}{Z} e^{-\beta E_i}, \qquad Z = \sum_i e^{-\beta E_i},$$

where Z is the canonical partition function, $\beta = 1/k_B T$ for the boltzmann constant k_B and temperature T, and E_i the energy of site i. We model the energy of each site as the work required to move there. For a step $\mathbf{s}$ the energy becomes $E_i = -\mathbf{s} \cdot \mathbf{F}$, where the sign is chosen such that the energy (difference) is negative when moving

for aling the bias, analogous to an energy gain by moving there. Due to the symmetry of both the atom sites and the center elements sites the step length to neighbouring sites will always be equal. By defining the bias magnitude $B = \beta|\mathbf{F}||\mathbf{s}|$ we the probability for jumping to site i as

$$p_i = \frac{1}{Z} e^{B\hat{\mathbf{s}}\cdot\hat{\mathbf{F}}} \propto e^{B\hat{\mathbf{s}}\cdot\hat{\mathbf{F}}},$$

where the hat denotes the unit direction of the vector. The bias magnitude B then captures the opposing effects of the magnitude of the external force and the temperature of the system as $B \propto |\mathbf{F}|/T$. We notice that $\hat{\mathbf{s}} \cdot \hat{\mathbf{F}} = \cos(\theta)$ for the angle θ between the step and bias direction. This shows that the bias will have the biggest positive contribution to the probability when the step direction is fully allinged with the bias direction ($\theta = 0$), have no contribution for orthogonal directions ($\theta = \pm\pi/2$) and the biggest negative contribution when the directions are antiparallel ($\theta = \pi$). The partition function serves simply as a normalization constant. Thus, numerically we can enforce this simply by setting $Z = 1$ at first, calculate p_i, and then normalize the result at the final stage as a divition by the sum of all p_i. In the numerical implementation we then pick the step destination weighted by the discrete probability distribtion p_i. In Fig. 1.12 we have illustrated how a bias of different strength impacts the probability disitrbution for a random walk between center elements. We can visually confirm that the bias will favorise the directions that lies closes to the bias direction. This favorization is more distinct at high bias strengths while at low strength $B \to 0$ we get a uniform distribution which alligns with the default unbiased random walk.



(a) **(b)**

Figure 1.12: Illustration of the probability distribution for the various step direction during a bias random walked between center elements. (*a*) Shows the possible step directions as black arrows pointing towards the neighbouring center elements shown as blue circles. The bias direction is denoted as an orange arrow and the numbering indicates the most likely direction to take (1) towards the least likely (6). The atoms sites are marked as grey circles for reference. (*b*) The probability distribution as a function of angle between the direction of choice and the bias direction. The disitrbution is normalized according to the discrete probabilities marked with dots for which the continous line simple highlights the curve of the distribution. The direction indexes corresponds to the numbering on figure (*a*). The color map indicates different strenghs of the bias.

1.5.3.4 Stay or break

The *Stay or break: True/False* parameter defines the probability p_{stay} that the walker will keep its direction or otherwise break into a different direction by probability $1 - p_{\text{stay}}$. That is, we manually substitute p_{stay} in the discrete probability for the next step corresponding to a continuation in the same direction. We then shift the remaining probabilities such that the distirbution remains normalized. In this way we can still perform bias random walk in combination. For the center element walk it is trivial to determine which of the neighbour directions correspond to a continuation of direction based on the last visited site. However, for an atom site walk, it is not possible to follow the same direction in a straight line due to the hexagonal layout of the lattice.

We recall that the nearest atom neighbour indexes alternates for each increment in x or y index (see Eq. (1.1)) which corresponds to the alternating neighbour directions D

$$(i+j) \text{ is even} \rightarrow D = \left\{ \frac{a}{2}\left(\frac{-2}{\sqrt{3}}, 0\right), \frac{a}{2}\left(\frac{1}{\sqrt{3}}, 1\right), \frac{a}{2}\left(\frac{1}{\sqrt{3}}, -1\right) \right\},$$
$$(i+j) \text{ is odd} \rightarrow D = \left\{ \frac{a}{2}\left(\frac{2}{\sqrt{3}}, 0\right), \frac{a}{2}\left(\frac{-1}{\sqrt{3}}, 1\right), \frac{a}{2}\left(\frac{-1}{\sqrt{3}}, -1\right) \right\}.$$

One way to mittigate this issue is to use the six directions from the center element walk as the common direction to "stay or break" from. As showcased in Fig. 1.13, for each center element direction (black arrows) there are two possible atom directions (red and orange arrows) that are equally close to the center element direction. The red and orange arrows represent $(i+j)$ being even or odd respectively, and we notice that these appear in pairs such that we can always determine which of the atom directions that are closest to the center element direction. Following this idea we can map each center direction to an atom direction depending on the even or oddness of the position. For $p_{\text{stay}} = 1$ this results in a garuanteed zigzag motion along the center element direction that it happens to start on.

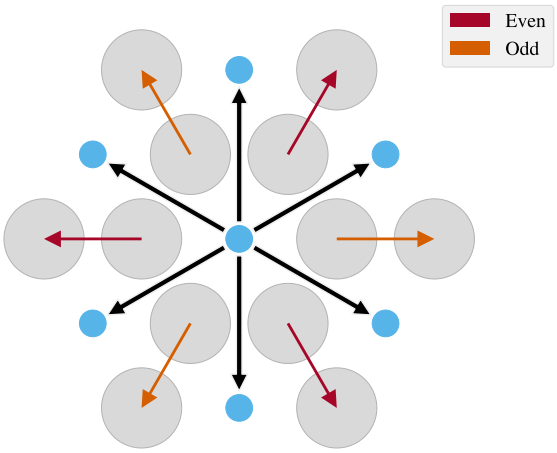


Figure 1.13: Caption

The *Stay or break* feature is still subject to previously defined rules. For instance, in the case where the preferred site is not available, the walker will either terminate when going there, or the preferred site is removed from the neighbor list when *avoid invalid: True*. In the latter case, the walker will be forced to break out of its direction and follow the new direction that it happens to go in.

1.5.3.5 Deployment schemes

By default, each random walker is given a uniform random starting point among the non-visited availble sites left on the sheet. This includes any modifications in relation to the minimum distance parameter. By toggling the *Grid start: True/False* parameter on, the starting points are instead predefined on an evenly spaced grid. That is, the sheet is subdivided into the least amount of squares that will accomodate a space for each starting point. 1 walker leads to a 1×1 partition, $\{2, 3, 4\}$ walkers lead to a 2×2 partition, $\{5, 6, 7, 8, 9\}$ walker lead to a 3×3 partition and so on. For each partition square the starting point is placed as central as possible. The lower left partition square is then choosen as a default starting place for the first walker and the remaining sites are filled according to the order that maximizes the minimum distance between a new starting point and the ones already used[2]. The population of the grid is visualized in Fig. 1.14 for 1-9 walkers in total with color coding for the order of deployment.

[2]In hindsight, it would have been less biased to choose a random partition square, but we do not consider this to be of great importance for the usage of this feature in final dataset

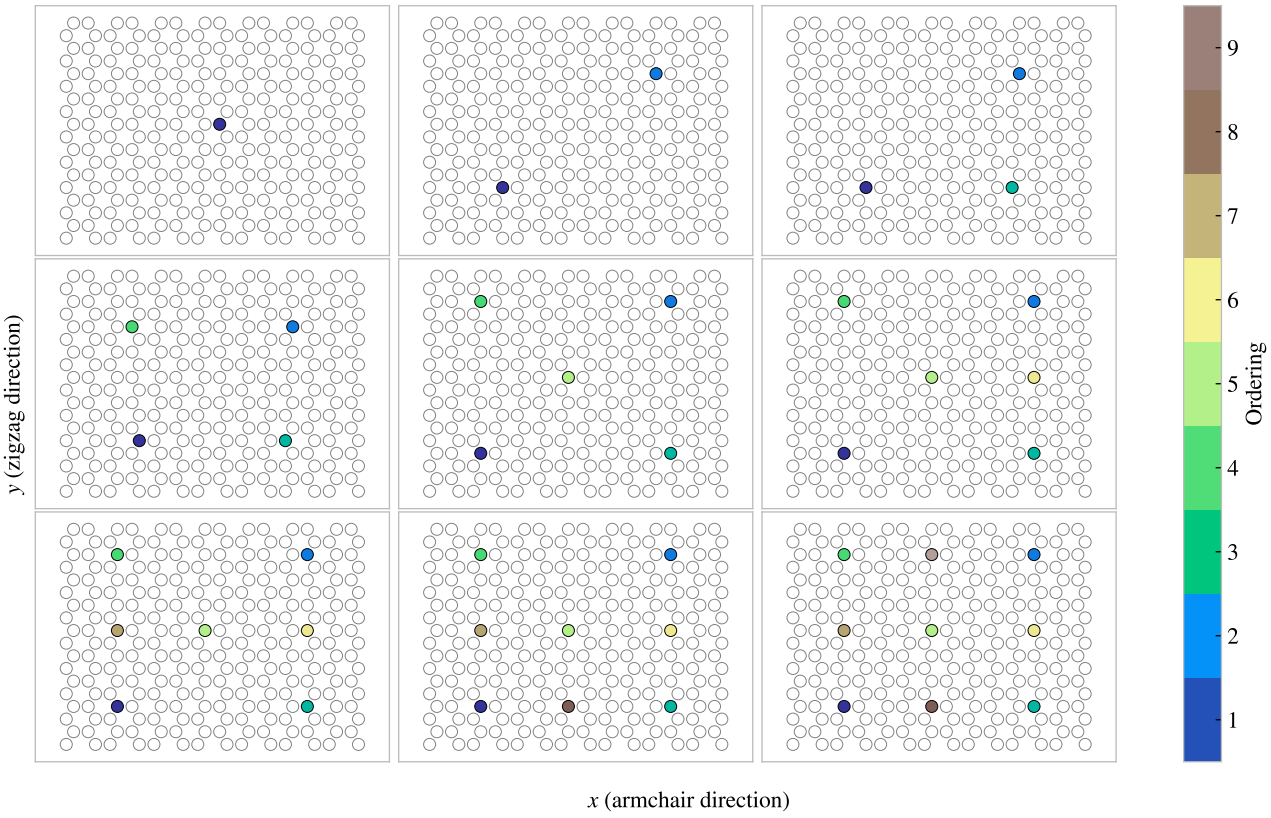


Figure 1.14: Populaiton of starting points with the centering parameter toggled on in a 14×18 sheet. This is shown for 1-9 number of random walkers with the color map conveying the order of the population.

The *Centering: True/False* parameter let us relocate the path of the random walker such that the path center of mass alligns better with the starting point. When toggled on, the path is moved in the direction defined by the center of mass and the starting point for which the closest valid relocation on the direct translation line is choosen. This can be used in combination with the grid start and the bias parameter to make rather ordered configurations. In addition, the *RN6:True/False* parameter can be used update the bias direction to on of the six directions of the center element walk for each new walker deplyoed. This lets us create confgiratiosn like the one shown in Fig. 1.15b.

1.5.3.6 Validity

The simulation procedure requires the sheet to be fully attatched, non ruptured, which can be summarized as the following requirements.

1. There exist only a single cluster on the sheet. We define a cluster as the set of atoms which can all be reached through nearest neighbour walking on the cluster.
2. The cluster of atoms is spanning the sheet in the y-direction. This means that there exist at least one path through nearest neighbour walks that connect the bottom and the top of the sheet. This is due to the reason that the sheet must be attatched to the pull blocks.

In order to accommodate these requirements we count the number of clusters and search for a spanning cluster after all walkers have terminated. If the requirements are not met we simply rerun the random walk from scratch. This is done, *Avoid clustering: 0, 1, ...*, amount of times. If the requirements are not met during any of those reruns the non-spanning clusters are simply removed. In the case of no spanning cluster the configuration is skipped. This crude scheme was later reinvented as a more refined repair scheme which alters the sheet by the intention of performing the least amount of changes (addition or subtraction of atoms) in order to meet the attatchment requirements. This was done as a part of the accelerated search procedure and hence it was not utilized in the creation of the random walk dataset.

1.5.3.7 Random walk examples

Some examples of the random walk patterns are illustrated in Fig. 1.15.

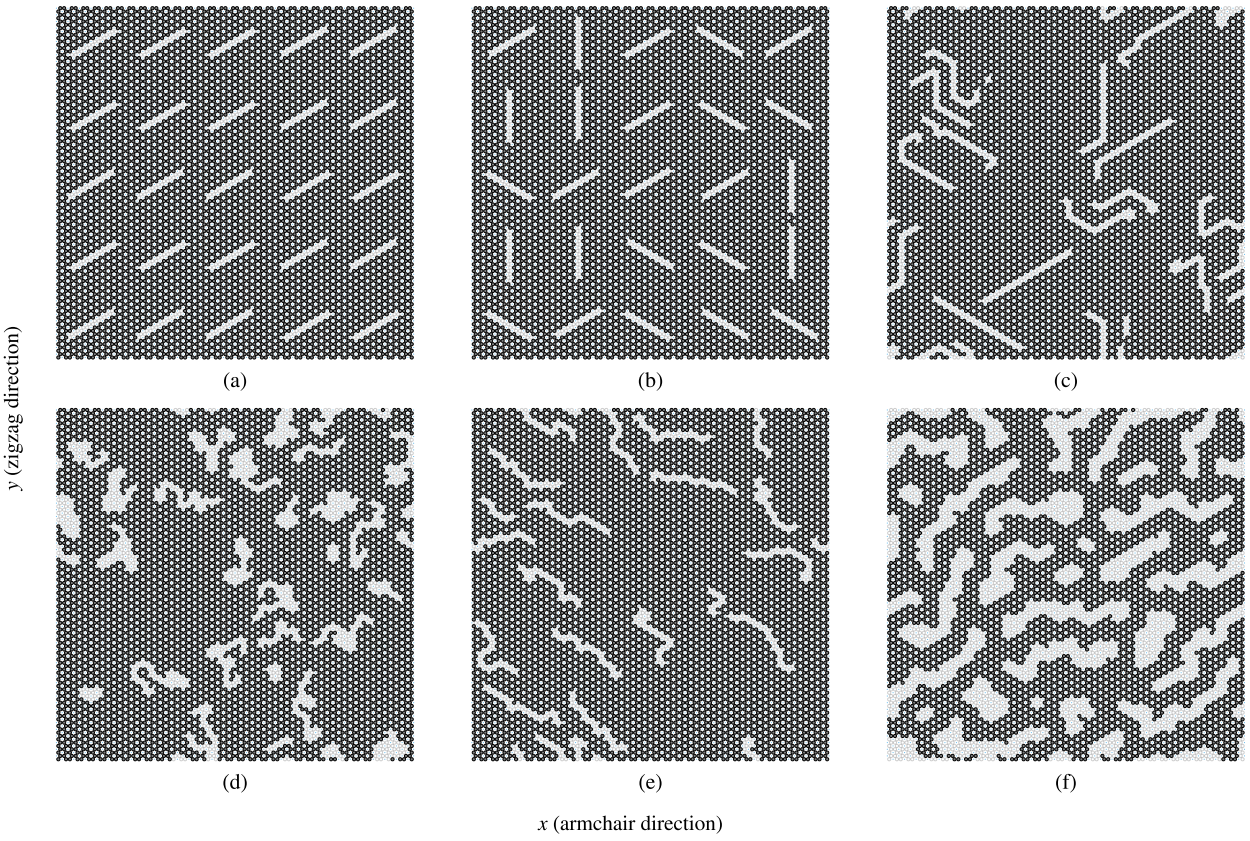


Figure 1.15: Some example uses of the random walking class. Give information of parameters? Henrik says Yes :D

Chapter 2

Kirigami configuration exploration

Building upon the discoveries of the Pilot Study **??**, we will further explore the impact of Kirigami designs on strain-dependent friction. Our focus is primarily to optimize the friction force and friction coefficient towards their maximum or minimum values. To achieve this goal, we will utilize MD simulations to generate an extended dataset that encompasses a wider range of Kirigami designs. This is motivated by the aim of gaining gain a broader understanding of the friction-strain relationship. We will then leverage this dataset to explore the potential of employing machine learning for predicting friction behavior based on Kirigami design, strain, and load. Finally, we plan to utilize the developed machine learning model for an accelerated search.

2.1 Generating the dataset

We aim to create a dataset that contains an extended series of Kirigami design configurations based on the pattern generation methods developed in Chapter 1 for which we will vary the strain and load for each configuration. For each configuration, we sample 15 pseudo uniform strain values (see **??**) between zero and the rupture strain according to the rupture test. Since the normal force did not prove to be dominant in the friction description we only sample 3 values per configuration, uniformly sampled in the range $[0.1, 10]$ nN. In total, this gives $3 \times 15 = 45$ data points for each configuration. For the remaining parameters, we use the values presented in the pilot study (see **??**). We are mainly concerned with the mean friction and whether the sheet ruptures during the simulation. However, we also include the maximum friction, the relative contact, the rupture strain (from the rupture test) and the porosity (void fraction) in the dataset. We generate 68 configurations of the Tetrahedron pattern type, 45 of the Honeycomb type and 100 of the Random walk type. For the Tetrahedron and Honeycomb patterns, we choose a random reference position that results in translation of the patterns. A summary of the dataset is given in Table 2.1 while all configurations are shown visually in **??**. The Tetrahedron and Honeycomb parameters are chosen to provide additional variations of the configurations evaluated in **??** which exhibited interesting properties. The Random walk configurations are chosen with the aim of introducing as much variety as possible within our Random walk framework. Notice that not all submitted data points "make it" to the final dataset. This is due to a small bug in the data generation procedure[3].

[3]The issue arises from the fact that the rupture point in the rupture test does not completely match the rupture point in the following simulations. After performing the rupture test the simulation is restarted with a new substrate size, corresponding to the measured rupture strain limit, but also with a new random velocity and thermostat initialization. The sheet is then strained and checkpoints of the simulation state (LAMMPS restart files) are stored for each of the targeted strain samples. However, if the rupture point arrives earlier than suggested by the rupture test, due to randomness from the initialization, some of the planned strain samples do not get a corresponding checkpoint file. Thus, these data points are not included in the dataset even though they ideally should have been noted as a rupture event. This could quite have been mitigated by a rewrite of the code, but it was first discovered after the dataset had been created. We notice, however, that the dataset still contains 11.57 % rupture events which provide a reasonable amount of rupture events to incorporate in the machine learning model

Table 2.1: Summary of the number of generated data points in the dataset. Due to slight deviations in the rupture strain and the specific numerical procedure not all submitted simulations "make it" to the final dataset. Notice that the Tetrahedon $(7, 5, 2)$ and Honeycomb $(2, 2, 1, 5)$ from the pilot study are rerun as a part of the Tetrahedon and the Honeycomb datasets separately. In the latter datasets, the reference point for the pattern is randomized and thus these configurations are not fully identical. This is the reason for the ambiguousness in the total sum.

Type	Configurations	Submitted data points	Final data points	Ruptures
Pilot study	3	270	261	25 (9.58 %)
Tetrahedon	68	3060	3015	391 (12.97 %)
Honeycomb	45	2025	1983	80 (4.03 %)
Random walk	100	4500	4401	622 (14.13 %)
Total	214 (216)	9855	9660	1118 (11.57 %)

2.2 Data analysis

In order to gain insight into the correlations in the data we calculate the correlation coefficients between all variable combinations. More specifically, we calculate the Pearson product-moment correlation coefficient which is defined, between data set X and Y, as

$$\mathrm{corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\langle (X-\mu_X)(Y-\mu_Y)\rangle}{\sigma_X \sigma_Y} \in [-1,1],$$

where $\mathrm{Cov}(X,Y)$ is the covariance, μ the mean value and σ the standard deviation. The correlation coefficients range from a perfect negative correlation (-1) through no correlation (0) to a perfect positive correlation (1). The correlation coefficients are shown in Fig. 2.1.

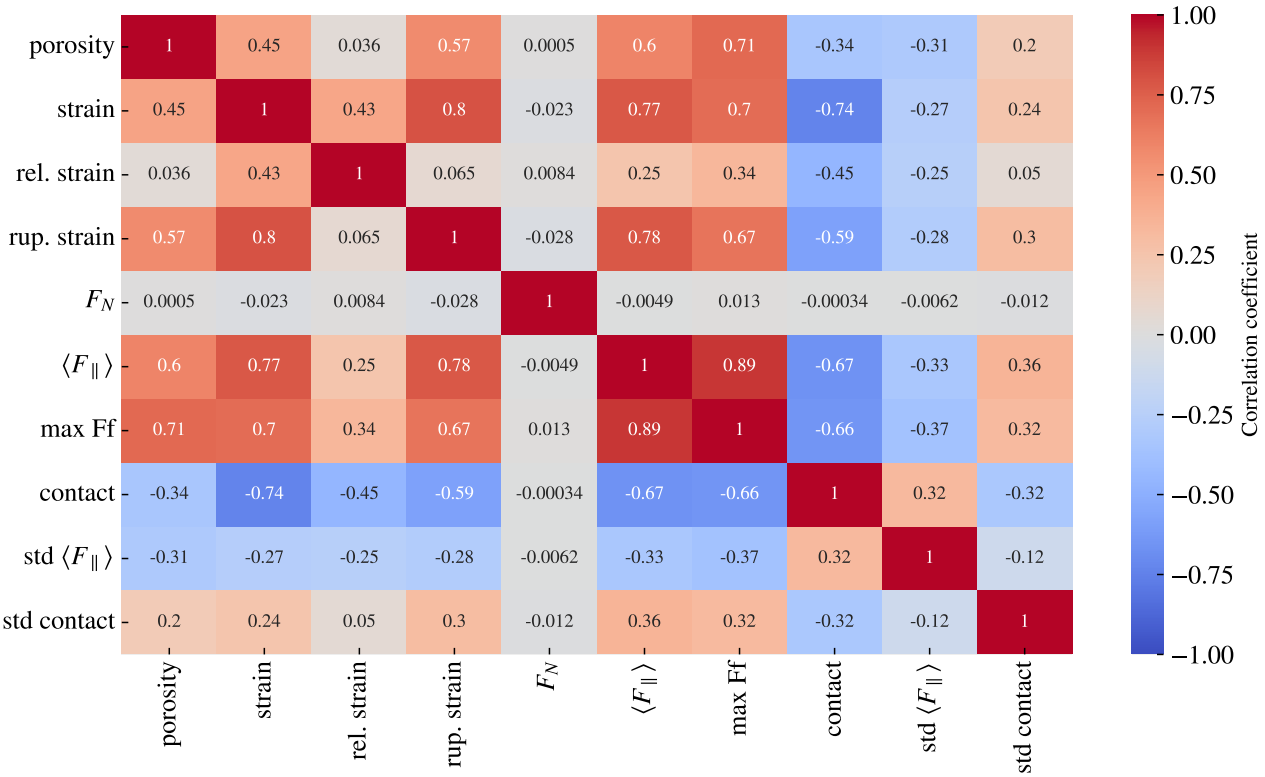


Figure 2.1: Pearson product-moment correlation coefficients for the full datset (see Table 2.1).

From Fig. 2.1 we especially notice that the mean friction force $\langle F_{\parallel} \rangle$ has a significant positive correlation with strain (0.77) and porosity (0.60). However, the relative strain, the strain scaled by the rupture strain, has a weaker correlation of only 0.25. This indicates that the correlation might be associated with the flexibility of the configurations since these can be taken to higher absolute values of strain. This is further supported by the fact that the mean friction and the rupture strain are also strongly positively correlated (0.78). From Fig. 2.1 we also observe that the contact is negatively correlated with the mean friction (-0.67) and the strain value (-0.74). This is generally consistent with the trend observed in the pilot study in **??** where the increasing strain was correlated with a decreasing contact and mainly increasing mean friction. However, we must note that the correlation coefficient is a measure of the quality of a forced linear fit on the data. Since we have observed a non-linear trend between friction and strain (**??**) we should not expect any near 100 % correlations. Additionally, we also notice that all correlations to normal load are rather low, which aligns well with the findings in the pilot study.

Fig. 2.2 shows a visualization of the data (excluding the pilot study configurations) for chosen variable pairs on the axes. This provides a visual clue on some of the correlations and provides a qualitative feeling for the diversity in various planes of the feature space that we eventually will base our machine learning model on. We notice that the honeycomb pattern is spanning a significantly larger range of strain, contact and mean friction.

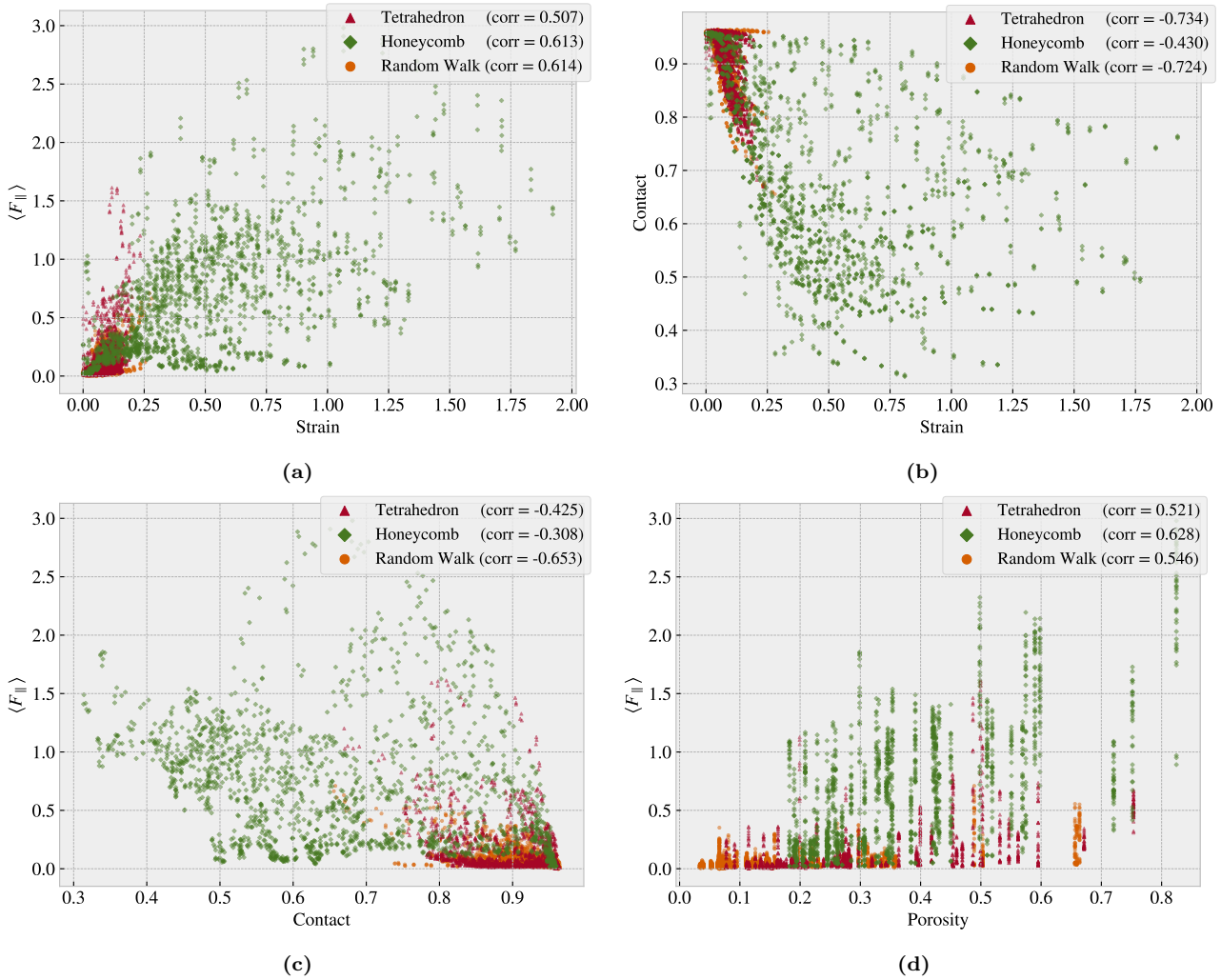


Figure 2.2: Scatter plot of the data sets Tetrahedron, Honeycomb and Random Walk (excluding the pilot study) for various variable combinations in order to visualize some chosen correlations of interest and distributions in the data

2.3 Properties of interest

In the pilot study (**??**) we found promising results for the idea of achieving a negative friction coefficient under the assumption of a system with coupled normal force and strain. Hence, we will consider this as a main property of interest for our further exploration. However, it is not obvious how one should rigorously quantify this. The friction coefficient is by our definition (**??**) given as the slope of the friction F_f vs. normal force F_N curve. Hence, for two data points $\{(F_{N,1}, F_{f,1}), (F_{N,2}, F_{f,2})\}$, $F_{N,1} < F_{N,2}$ we can evaluate the associated friction coefficient $\mu_{1,2}$ as

$$\mu_{1,2} = \frac{F_{f,2} - F_{f,1}}{F_{N,2} - F_{N,1}} = \frac{\Delta F_f}{\Delta F_N}.$$

In the pilot study, it became apparent that the effects of friction under the change of load is negligible in comparison to the effects related to strain ε. Thus, by working under the assumption $F(F_N, \varepsilon) \sim F(\varepsilon)$ and a coupling $F_N \propto R \cdot \varepsilon$ with linear coupling ratio R we get

$$\mu_{1,2}(\varepsilon_1, \varepsilon_2) = \frac{\Delta F_f(\varepsilon_1, \varepsilon_2)}{R(\varepsilon_2 - \varepsilon_1)} \propto \frac{\Delta F_f(\varepsilon_1, \varepsilon_2)}{\Delta \varepsilon}. \tag{2.1}$$

With this reasoning, we can in practice substitute load F_N for strain ε in the expression for the friction coefficient of our coupled system. This justifies the search for a negative slope on the friction-strain curve since this can be related to a negative friction coefficient in our proposed coupled system. The remaining question is then how to evaluate the strength of this property. By definition, the minimum (most negative) slope value would give the lowest friction coefficient. However, two data points with a small $\Delta\varepsilon$, corresponding to a small denominator in Eq. (2.1), would potentially lead to a huge negative slope value without any significant decrease in friction. Hence, we choose to consider the drop in friction with increasing strain as a better metric. Numerically we compute this by locating the local maxima on the friction-strain curve and then evaluating the difference to the succeeding local minima. The biggest difference corresponds to the *max drop* which serves as our indicator for a negative friction coefficient. In this evaluation, we do not guarantee a monotonic decrease of friction in the range of the biggest drop, but when searching among multiple configurations this is considered a decent strategy to highlight configurations of interest worthy of further investigation. In addition to the biggest drop in friction, we also consider the minimum, $\min F_{\text{fric}}$, and maximum, $\max F_{\text{fric}}$, friction along with the maximum difference, $\max \Delta f_{\text{fric}} = \max F_{\text{fric}} - \min F_{\text{fric}}$. The extrema of these four properties for each of the categories: Tetrahedron, Honeycomb, Random walk and Pilot study, are summarized in Table 2.2. The corresponding strain profiles and configurations are shown in Fig. 2.3 to 2.6 (excluding the pilot study). The strain profiles for the full dataset are shown in appendix **??**.

From the property comparison in Table 2.2 we find that both the Tetrahedron and Honeycomb subsets contain improved candidates for each of the property scores in comparison to the Tetrahedron $(7, 5, 1)$ and Honeycomb $(2, 2, 1, 5)$ examined in the pilot study. Overall, the Honeycomb pattern type is still resulting in the highest scores for the maximum properties while the minimum friction is still achieved by the non-cut sheet. This latter observation reveals that our dataset does not provide any indication that friction can readily be reduced for a Kirigami sheet under strain. However, the improvement in the remaining properties indicates that the dataset contains the necessary information to provide a direction for further optimization of the maximum properties. Considering the Random walk we find that the max property scores are generally lower than that of the Tetrahedron and Honeycomb structures. However, since these are found to be on a comparable order of magnitude we argue that contain relevant information for populating configuration space with respect to machine learning training. The Random walk patterns also contribute with some immediate insight into the structures that we can associate with each of the properties of interest due to its increased diversity in comparison to the other patterns.

For the $\min F_{\text{fric}}$ top candidates (Fig. 2.3) we find that the Random walk candidate has a rather cut density (low porosity) and with vertical cuts. Since these cuts run parallel to the stretching direction one can hypothesize that this minimizes the induced buckling effect which agrees with the constant level of contact. For the minimum candidate for the Tetrahedron pattern, we also observe a low decrease in contact, and in both these cases this corresponds with a relatively flat friction-strain curve. When considering the remaining friction-strain curve throughout Fig. 2.3 to 2.6 we find a rising friction-strain curve which is accompanied by a decreasing relative contact. When looking at the Random walk 96 pattern, which is the top candidate for both the $\max F_{\text{fric}}$ (Fig. 2.4) and $\max \Delta f_{\text{fric}}$ (Fig. 2.5) property, we find a rather porous configuration with mainly horizontal-orientated cuts.

This has some structural reminiscence with the Honeycomb pattern. Finally, the best random walk candidate for the max drop category, Random walk 01, did not produce a big drop. We find that the contact area is decreasing with strain but not as strongly as seen for the other configurations. The configuration contains some slanted cuts which might be reminiscent of parts of the Tetrahedron pattern.

Table 2.2: Evaluation of the properties of interest for our dataset.

Tetrahedron	Configuration	Strain	Value [nN]	Hon. $(2, 2, 1, 5)$ [nN]
$\min F_{\text{fric}}$	$(3, 9, 4)$	0.0296	0.0067	0.0262
$\max F_{\text{fric}}$	$(5, 3, 1)$	0.1391	1.5875	0.8891
$\max \Delta F_{\text{fric}}$	$(5, 3, 1)$	$[0.0239, 0.1391]$	1.5529	0.8603
max drop	$(5, 3, 1)$	$[0.1391, 0.1999]$	0.8841	0.5098

Honeycomb	Configuration	Strain	Value [nN]	Tetra. $(7, 5, 1)$ [nN]
$\min F_{\text{fric}}$	$(2, 5, 1, 1)$	0.0267	0.0177	0.0623
$\max F_{\text{fric}}$	$(2, 1, 1, 1)$	1.0654	2.8903	1.5948
$\max \Delta F_{\text{fric}}$	$(2, 1, 5, 3)$	$[0.0856, 1.4760]$	2.0234	1.5325
max drop	$(2, 3, 3, 3)$	$[0.5410, 1.0100]$	1.2785	0.9674

Random walk	Configuration	Strain	Value [nN]
$\min F_{\text{fric}}$	12	0.0562	0.0024
$\max F_{\text{fric}}$	96	0.2375	0.5758
$\max \Delta F_{\text{fric}}$	96	$[0.0364, 0.2375]$	0.5448
max drop	01	$[0.0592, 0.1127]$	0.1818

Pilot study	Configuration	Strain	Value [nN]
$\min F_{\text{fric}}$	No cut	0.2552	0.0012
$\max F_{\text{fric}}$	Honeycomb	0.7279	1.5948
$\max \Delta F_{\text{fric}}$	Honeycomb	0.7279	1.5325
max drop	Honeycomb	$[0.7279, 1.0463]$	0.9674

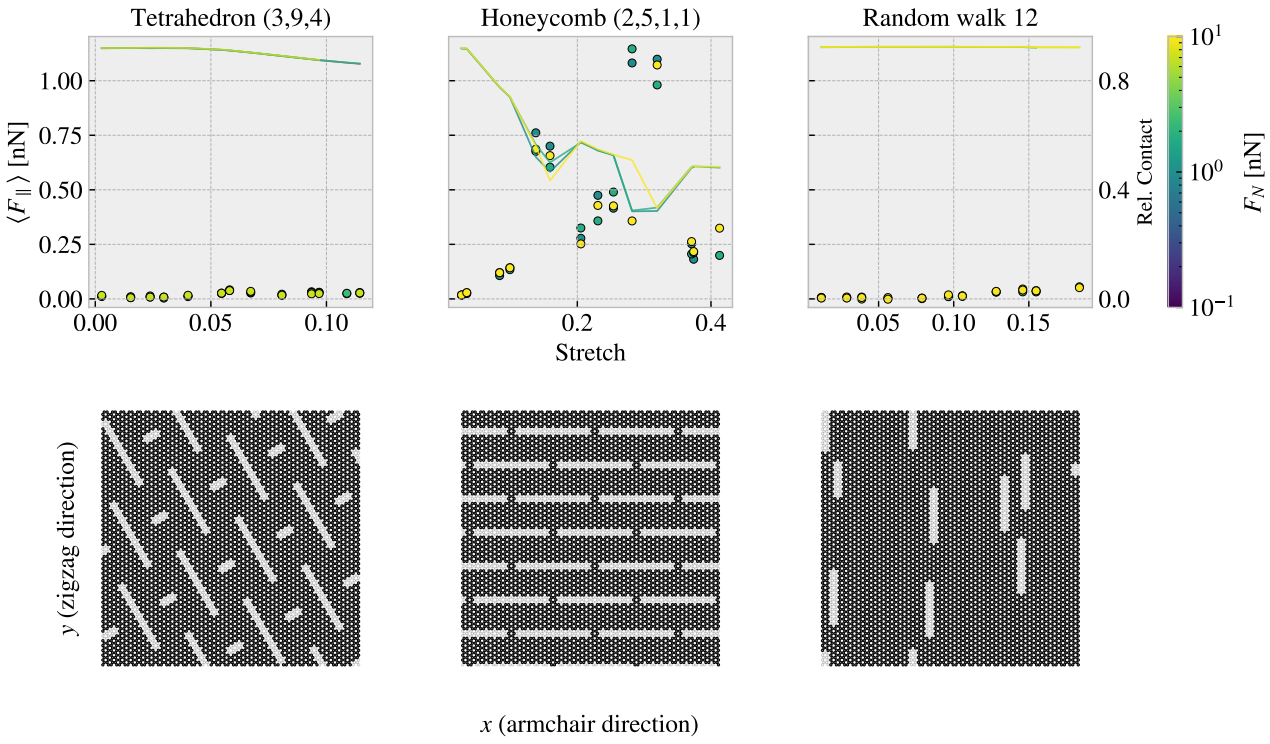


Figure 2.3: Minimum friction: Configurations corresponding to the minimum friction.

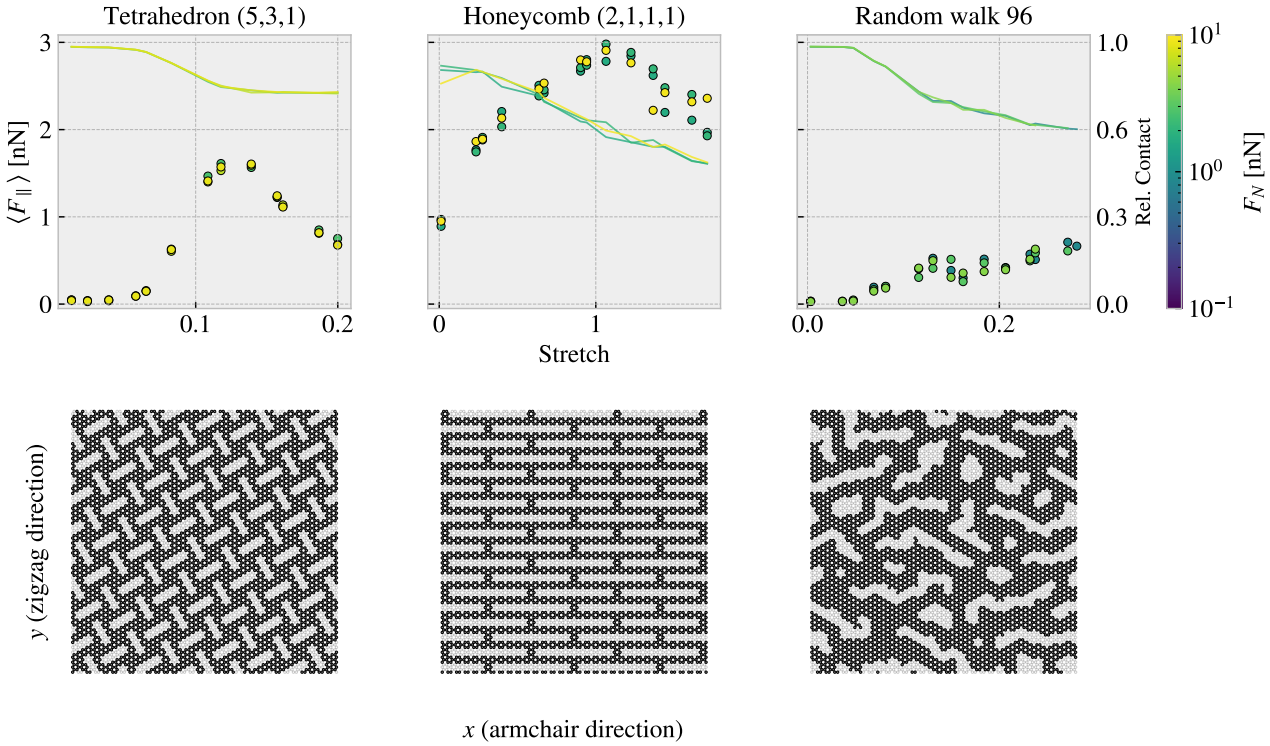


Figure 2.4: Maximum friction: Configurations corresponding to the maximum friction.

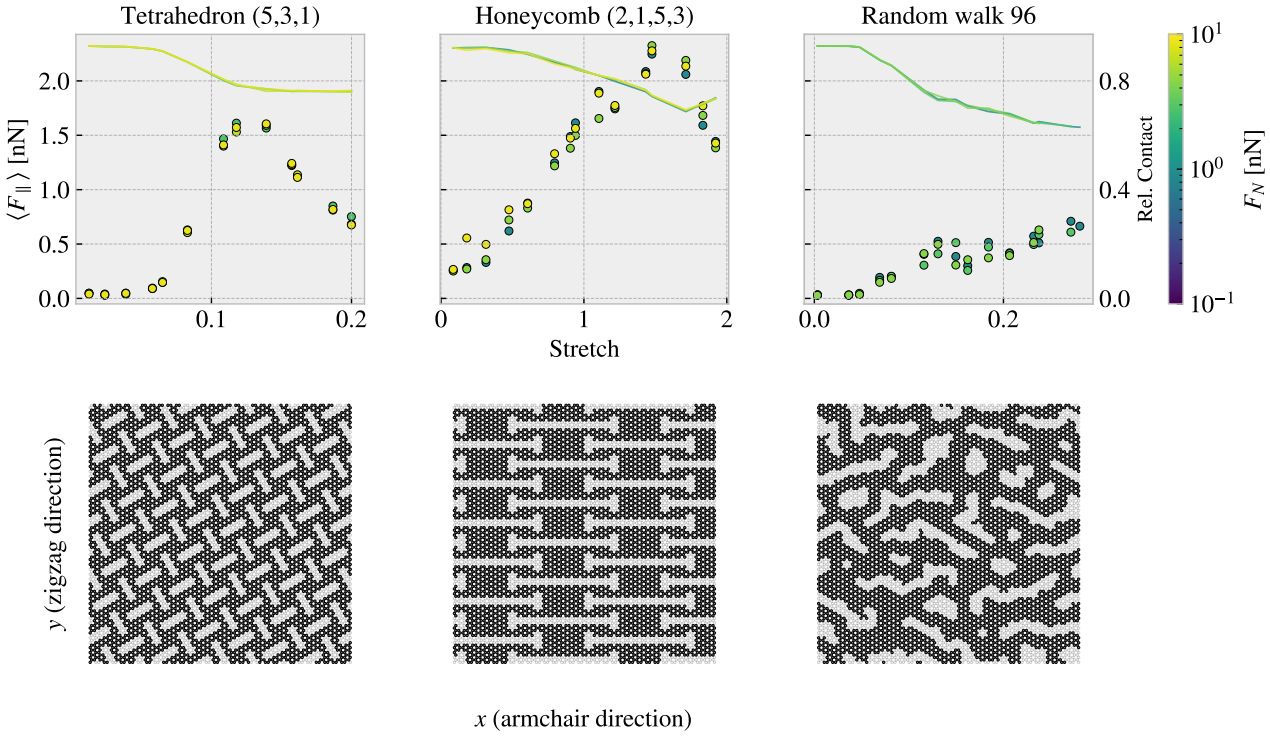


Figure 2.5: Maximum Difference: Configurations corresponding to the biggest difference in friction in the dataset for each pattern.

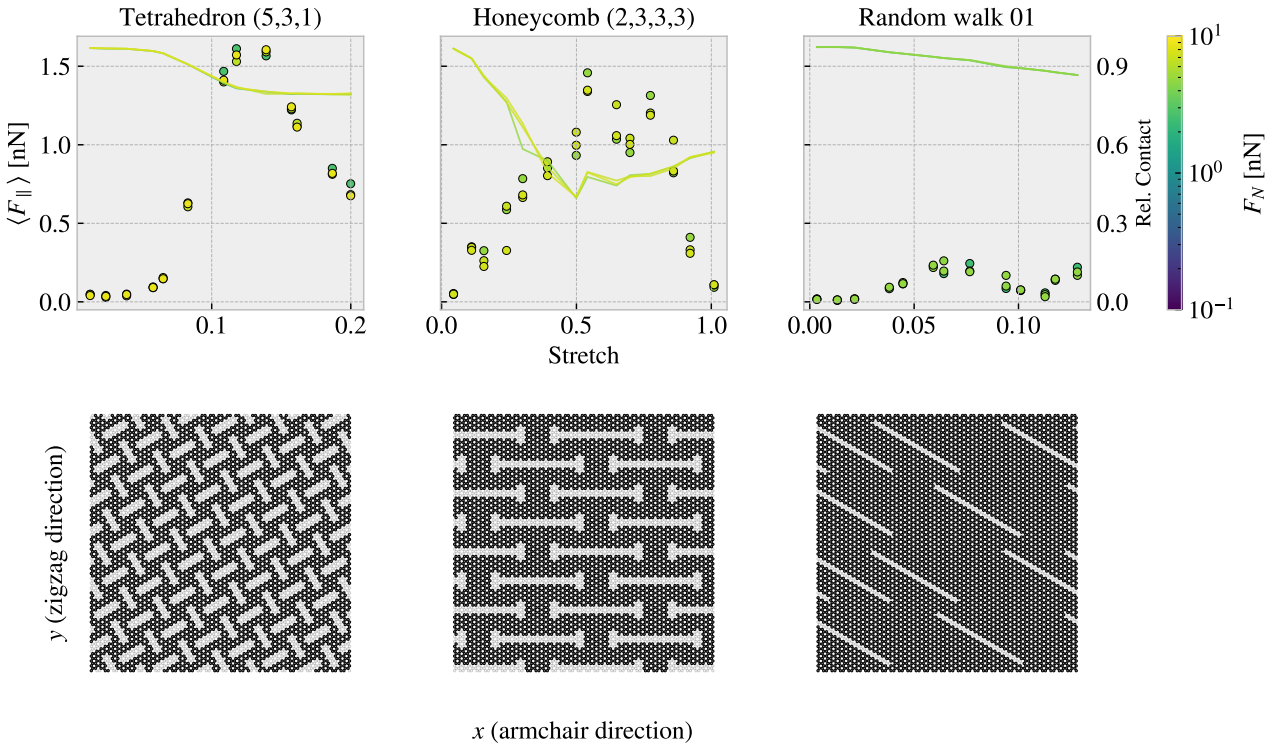


Figure 2.6: Maximum drop: Configuratiosn corresponding to the biggest friction drop in the dataset for each pattern.

2.4 Machine learning

Given the MD-based dataset we investigate the possibilities of training a machine learning model to predict the friction behavior from a given Kirigami configuration, strain and load.

2.4.1 Architecture

Due to the spatial dependencies in the Kirigami configurations, we use a convolutional neural network (CNN). Similar studies which predict mechanical properties for graphene sheets have used a VGGNet style network, Hanakata et al. [3, 4] and Wan et al. [5], which we adopt for this study as well. The VGGNet-16 architecture illustrated in Fig. 2.7 shows the key features that we will include:

- The image is processed through a series of 3×3 convolutional filters (the smallest size capable of capturing spatial dependencies) using a stride of 1 with an increasing number of channels throughout the network. We use padding to conserve the spatial size during a convolution. Each convolutional layer is followed by a ReLU activation function.
- The spatial dimensions are reduced by a max pooling, filter size 2×2 and a stride of 2, which halves the spatial resolution each time.
- The latter part of the network consists of a fully connected part using the ReLu activation as well. The transition from the convolutional to the fully connected part is achieved by applying a filter with the same dimensions as the last convolutional feature map. This essentially performs a linear mapping from the spatial output to the fully connected layer with the number of channels corresponding to the nodes in the first fully connected layer.

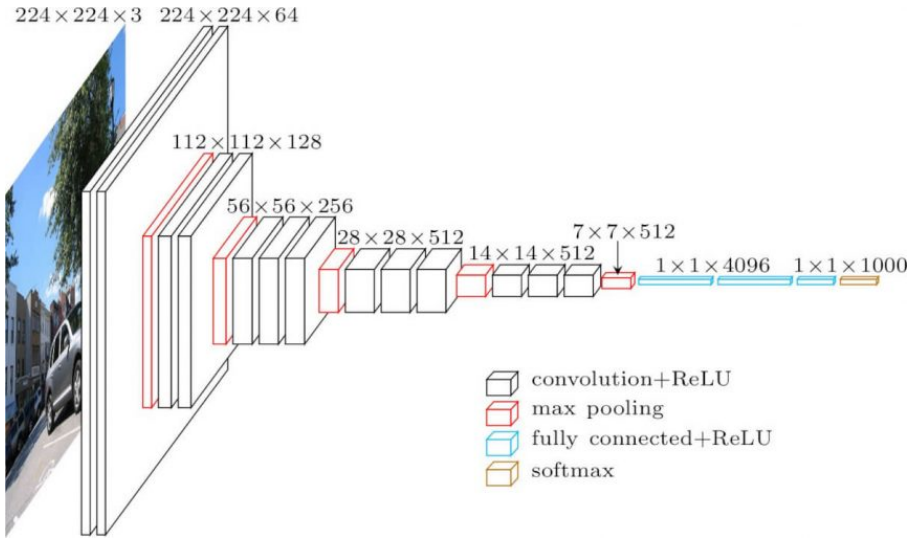


Figure 2.7: VGGNet-16. Source https://neurohive.io/en/popular-networks/vgg16/.

We deviate from the VGGNet-16 architecture by including batch normalization and restricting ourselves to setting up the convolutional part of the network in terms of the blocks: (Convolution → Batch normalization → ReLU → Max pooling). Similarly, we define a fully connected block by two elements (Fully connected → ReLU) which match the VGGNet model. Hanakata et al. and Wan et al. used a similar architecture with the parameters

$$\begin{aligned} \text{Hanakata et al. [3]} \quad & C16\ C32\ C64\ D64, \\ \text{Wan et al. [5]} \quad & C16\ C32\ D32\ D16, \end{aligned}$$

where C denotes a convolutional block with the number denoting the number of channels, and D a fully connected (dense) block with the number denoting the number of nodes. For the process of determining a suiting complexity for the architecture, we adopt the approach by Wan et al. [5] who used a "staircase" pattern for combining

convolutional and fully connected blocks. By defining a starting number of channels S and network depth D we fill the first half of the network layers with convolutional blocks, doubling in channel number for each layer, and the latter half with fully connected blocks having the number of nodes decrease in a reverse pattern. For instance, the architecture $S4D8$ will take the form

$$\text{Input} \rightarrow \underbrace{\overbrace{C4}^{S=4}\ C8\ C16\ C32\ D32\ D16\ D8\ D4}_{D=8} \rightarrow \text{Output}.$$

This provides a simple description where S and S can be varied systematically for a grid search over architecture complexity.

2.4.2 Data handling

2.4.2.1 Input

We use three variables as input: Kirigami configuration, strain of the sheet and applied normal load. The configuration is given as a two-dimensional input by the binary matrix while the strain and load are both scaler values. This gives rise to two different options for the data structure:

1. Expand the scalar values (strain and load) into 2D matrices of the same size as the Kirigami configuration by copying the scaler value to all matrix coordinates. This can then be merged into an image of three channels used as a single input.

2. Pass only the Kirigami configuration through the convolutional part of the network and introduce the remaining scalar values into the fully connected part of the network halfway in.

Both options utilize the same data, but the latter option is more directed towards independent processing of the data while the first makes for an intertwined ude of the configuration, strain and load. We implemented both options but found immediately that option 1 was producing the most promising results during the initial test runs, and thus we settled for this data structure.

2.4.2.2 Output

For the output, we are mainly concerned with mean friction and the rupture detection. In combination, this will make the model able to produce a friction-strain curve with an estimated stopping point as well. However, in order to retain the option to explore other relations in the data we include the maximum friction, relative contact, porosity and rupture strain in the output as well. Notice that we weigh the importance of these output variables differently in the loss as described in Sec. 2.4.3.

2.4.2.3 Data augmentation

In order to increase the utility of the available data one can introduce data augmentation. For most classification tasks this usually includes distortions such as color shifts, zoom, flip etc. However, such distortions are only valid since the classification network should still classify a cat as a cat even though it is suddenly a bit brighter or flipped upside down. For our problem, we can only use augmentation that matches a physical symmetry. Such a symmetry exists for reflection across the y-axis. We cannot use a reflection across the x-axis as the sheet is sliding in a positive y-direction. This would correspond to a change in the sliding direction which we cannot expect to be fully symmetric.

2.4.3 Loss

The output contains two different types of variables: scalar values and a binary value (rupture). For the scalar values we use the Mean Squared Error (MSE) **??** and for the binary output, we use binary cross entropy **??**. We calculate the total loss as a weighted sum of the loss associated with each variable

$$L_{tot} = \sum_v W_v \cdot L_v.$$

We choose the weights to be 1/2 for the mean friction and 1/10 for the remaining 5 variables, thus sharing the loss evenly for the remaining 50% of the weight. During the introductory phase of the training, we tried different settings for these weights, but we found that the results varied little. Hence, we concluded that this was of minor importance and stuck to the values defined above.

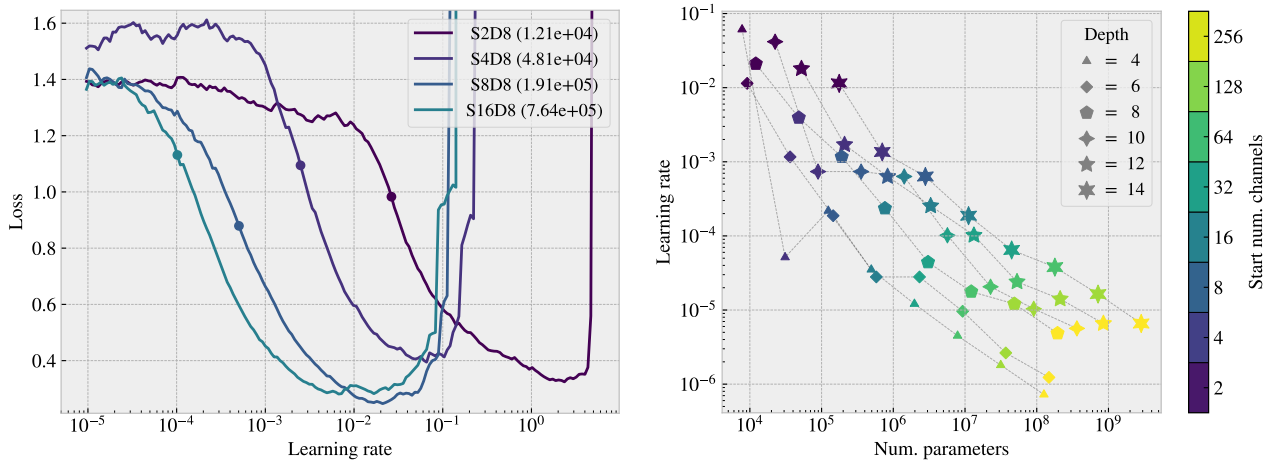
2.4.4 Hypertuning

For the hypertuning we focus on architecture complexity, learning rate, momentum and weight decay. We will use the ADAM optimizer with the initial default values of $beta_1 = 0.9, \beta_2 = 0.999$ and zero weight decay (we will change momentum β_1 and weight decay). We use a batch size of 32 and train the model for a maximum of 100 epochs while storing the best model based on the validation scores. Since the learning rate is considered to be one of the most important hyperparameters we will determine a suitable choice for the learning rate using the learning rate range test for each of the two major grid searches:

1. Architecture complexity grid search of S vs. D with individually chosen learning rates for each complexity combination.

2. Momentum vs. weight decay grid search with learning range chosen with regard to each momentum setting.

We consider first the architecture complexities in the range $S \times D = \{2, 4, 8, 16, 32, 64, 128, 256\} \times \{4, 6, 8, 10, 12, 14\}$. For each architecture complexity, we perform an initial learning rate range test and determine the suitable choice as the point for which the validation loss decreases most rapidly. The learning rate is increased exponentially within the range 10^{-7} to 10 with increments for each training batch iteration. This is done for just a single epoch where a batch size of 32 yields a total of 242 increments. This corresponds to an exponent increment of approximately 1/30 giving a relative increase $10^{1/30} \sim 108\%$ per batch iteration. The learning rate range test is presented in Fig. 2.8 for various model complexities. We notice that the suggested learning rate decreases with an increasing number of model parameters. This decrease is further independent of the specific relationship between S and D.



(a) **(b)**

Figure 2.8: Learning rate range test for various model complexities. We increase the learning rate exponentially from nume-7 to 10 during one epoch corresponding to an exponent increment of roughly 1/30 per batch iteration. (a) shows a few examples of the training loss history as a function of the learning rate. The exemplary architectures are S[2, 16]D8 with the corresponding number of model parameters shown in parentheses in the legend. The dot indicates the suggested learning rate at the steepest decline of the slope. (b) shows the full results of suggested learning rates depending on the number of model parameters with color coding differentiating the number of start channels and marker types differentiating different model depths.

With the use of the suggested learning rates from Fig. 2.8 we perform a grid search over the corresponding S and D parameters. We evaluate both the validation loss and the mean friction R_2 score for the validation data which is shown in Fig. 2.9 together with the best epoch and the number of model parameters. Additionally,

we evaluate the mean friction R_2 score for a selected set of configurations. This set consists of the top 10 configurations with respect to the max drop property for the Tetrahedron and Honeycomb patterns respectively. This is done as a way of evaluating the performance on the non-linear strain curves which we immediately found to be the more difficult trend to capture. The selected evaluation is shown in Fig. 2.10. Note that these configurations already are a part of the full dataset and thus the data points related to these configurations are most likely present in both the training and the validation data set. Hence, the performance must be considered in conjunction with the actual validation performance in Fig. 2.9.

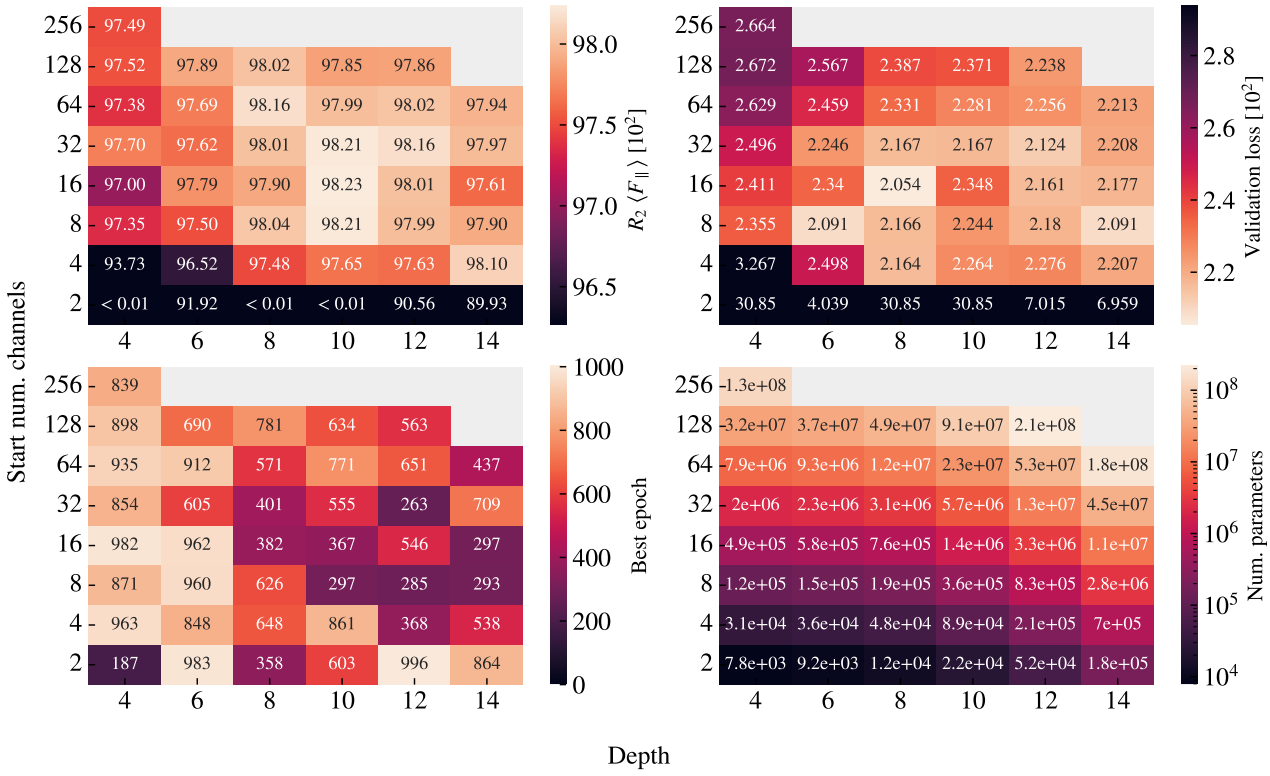


Figure 2.9: Architecture search.

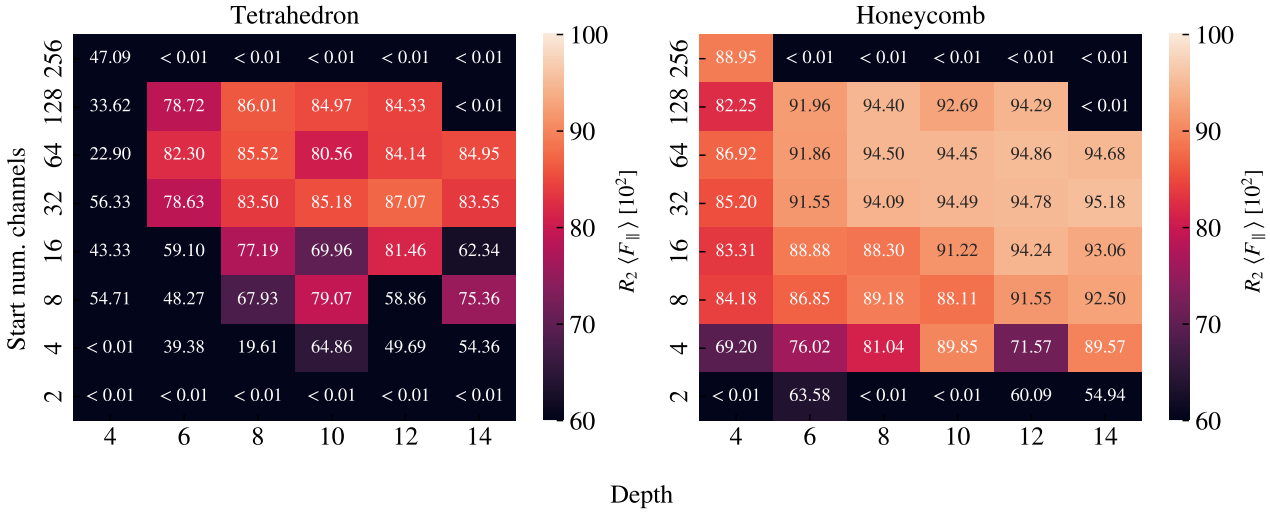
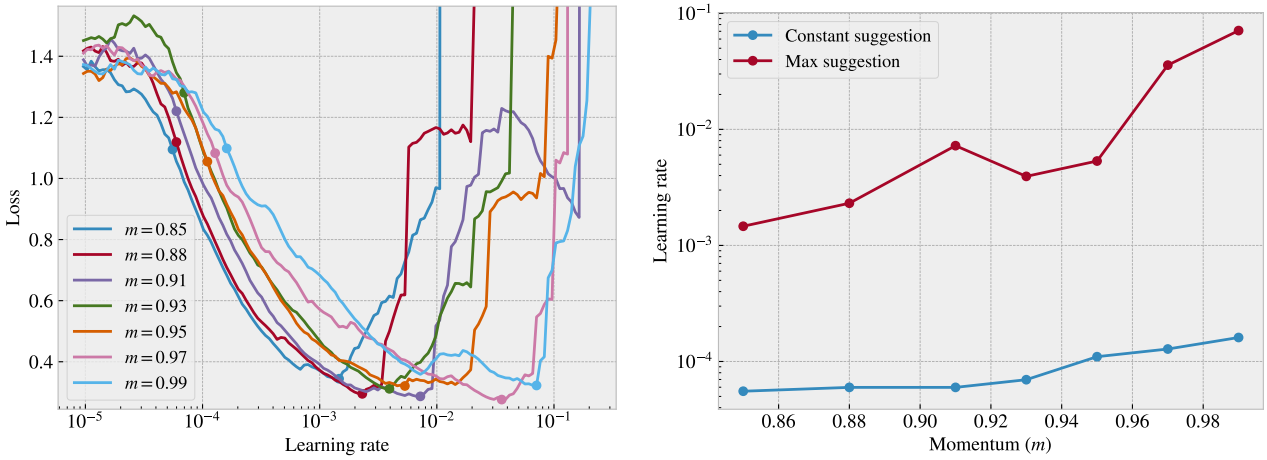


Figure 2.10: Selected pseudo validation set. Fix the missing grey fields in the top which are replaced by ¡ 0.01.

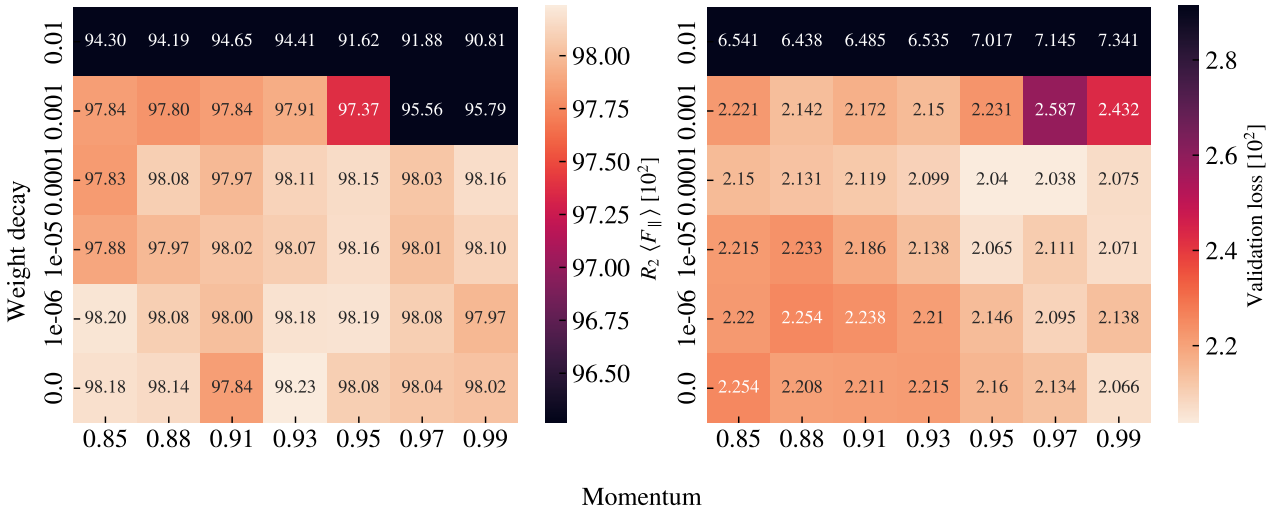
From the validation scores in Fig. 2.9, looking at both the loss and the R^2 scores, we find that models S(8-32)D(8-12) generally give the best performance. When looking at the best epoch we find that models of low depth result in a later best epoch which is compatible with underfitting. As the depth is increased we find more models with a lower best epoch, in the range $\sim [300, 600]$, which on the other hand suggest cases of overfitting. Since our training stores the best model during training, we do not have to worry too much about overfitting, but we can take this transition from underfitting to overfitting as a sign that our search is conducted in an appropriate complexity range. When consulting the evaluation on the selected set in Fig. 2.10 we find significantly lower R_2 scores, especially for the Tetrahedron pattern. Considering, that some of these data points are also present in the training data, this shows clearly that these configurations are more challenging to predict. While the peak R_2 value for the validation score in Fig. 2.9 was found for the model S16D10 model (98.23 %) the selected set test shows a slight preference for more complexity in the model. In the Tetrahedron selected set grid search, we find the best model to be S32D12, with an R_2 score of $\sim 87\%$. This model choice is more or less compatible with the overall performance as it is among the top candidates for the R_2 score and loss in Fig. 2.9 and the R_2 score for the Honeycomb pattern in Fig. 2.10 as well. Hence, we settle on this architecture.

Next, we consider momentum m and weight decay λ in the range $m \in [0.85, 0.99]$ and $\lambda \in [0, 1 \times 10^{-2}]$. For each choice of momentum, we perform a learning rate range test. We propose two learning rate schemes: A constant learning rate as used until this point and a one-cycle policy. In the one-cycle policy, we set a maximum bound for the learning rate and start from a factor $1/20$ of this bound and increase towards the maximum bound during the first 30% of the training. For the final 70% of training, we decrease towards a final minimum given as a factor $1e-4$ of the maximum bound. The increase and decrease are done by a cosine function. The suggested learning rate for the constant learning rate scheme is once again determined by the steepest slope on the loss curve while the maximum bound used for the one-cycle policy is determined as the point just before divergence. We find that the minimum point on the loss curve is a suitable choice that approaches the diverging point without getting too close and causing instabilities. The learning rate range test for momentum is shown in Fig. 2.11. We observe generally that a higher momentum corresponds to a higher suggested learning rate for both schemes. Using these results we perform a grid search of momentum and weight decay. We examine again the validation loss and validation mean friction R_2 score in addition to the friction mean R_2 score for the selected set of Tetrahedron and Honeycomb patterns. This is shown for the constant learning rate scheme in Fig. 2.12 and for the cyclic scheme in Fig. 2.13.

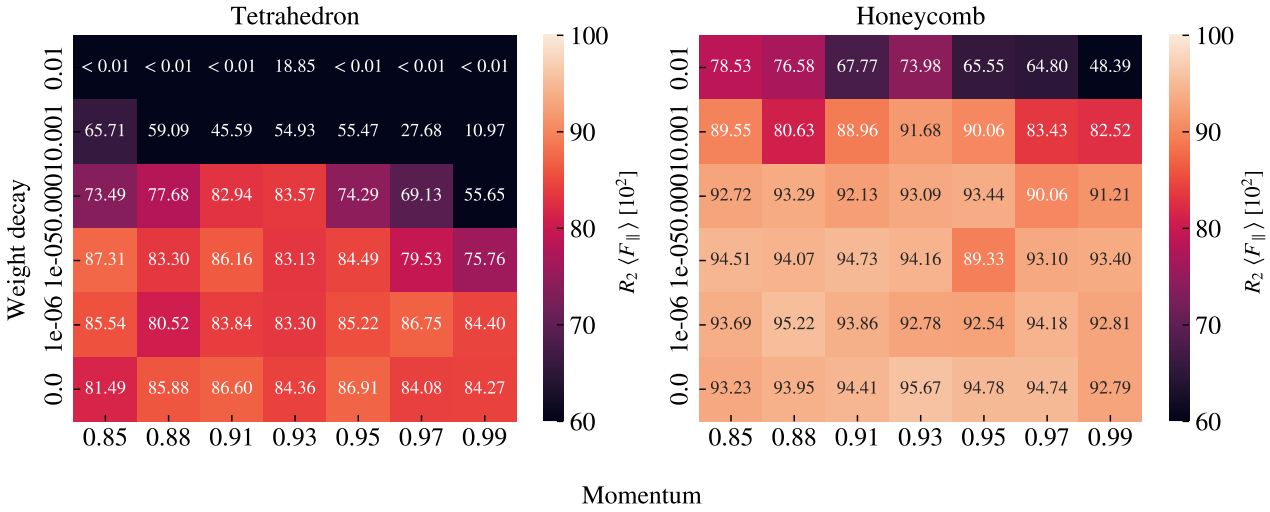


(a) (b)

Figure 2.11: Momentum learning rate range tets

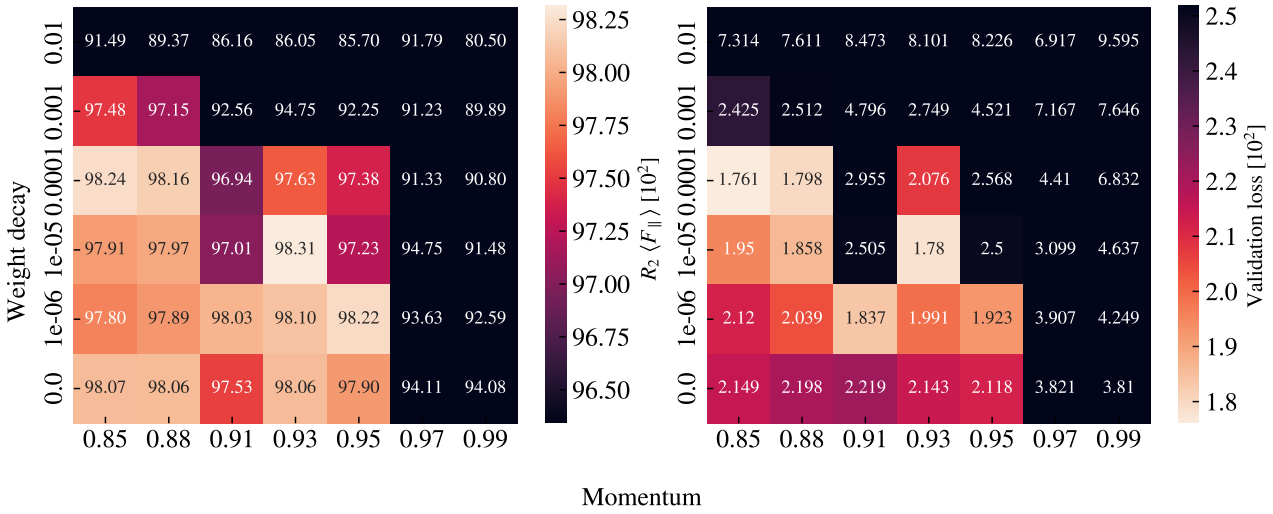


(a) Validation performance.

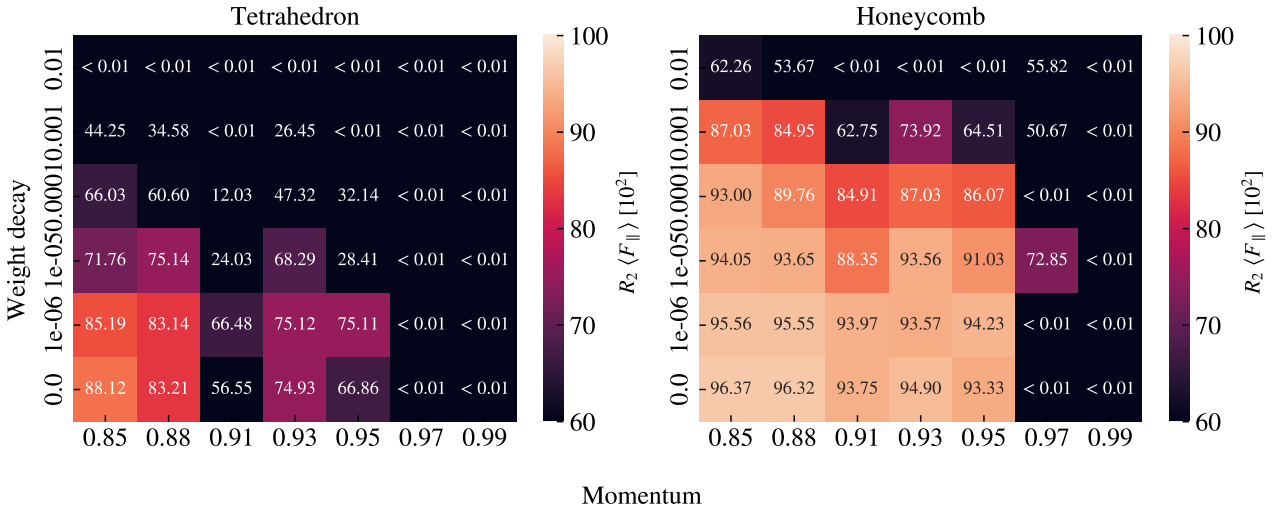


(b) Selected set comparison.

Figure 2.12: Constant learning rate and momentum scheme



(a) Validation performance.



(b) Selected set comparison.

Figure 2.13: Cyclic learning rate and momentum scheme

The original validation scores, before varying momentum and weight decay, were a validation loss of 0.02124 and a mean friction R_2 score of 0.9816. By varying momentum and weight decay, we can improve these scores slightly for the constant learning rate scheme (loss: 0.02038, R_2: 0.9823) and even more for the cyclic scheme (loss: 0.0176, R_2: 0.9831). However, these values are not taken from the same hyperparameter settings and we did not find a better option across all evaluation metrics. The comparison among best scores is summarized in Table 2.3. In general, the constant scheme shows rather stable results for all momentum settings $m \in [0.85, 0.99]$ in combination with a low weight decay $\lambda \leq 10^{-4}$. For the cyclic scheme the performance peaks towards a low momentum $m \leq 0.93$ and low weight decay $\lambda \leq 10^{-4}$. Looking at the summary in Table 2.3 we see that the cyclic scheme can produce a high score among all four performance metrics, but since these scores do not share common hyperparameters we need to choose which of them to prioritize. Due to our interest in capturing the non-linear trends, we prioritize the score from the selected set of Tetrahedron patterns as this provided the greatest challenge for our model to capture. We recognize that this choice introduces a greater risk of overfitting since the data points within this evaluation set are partly included in the training set as well. This is especially alarming since the abscent of weight decay allow for more overfitting in general. However, for the purpose of performing an accelerated search, we find it more important to increase the chances of discovering novel designs than to reduce the chance of getting false positive results. Since we retain the option to verify the properties of a

given design through MD simulations afterward, we do not have to rely on the machine learning prediction as a final score. Thus we choose the cyclic trained model with low momentum $m = 0.85$ and zero weight decay as our final model. On a final note we also point out that our choice of hyperparameters corresponded to the edge of our grid search, and it would have been natural to perform an extended search in that range. This was omitted due to time prioritization and the belief that the potential gain of doing so was not huge.

Table 2.3: Momentum and weight decay grid search using S32D12 model.

		Score $[10^2]$	Momentum	Weight decay
Validation loss	Original	2.124	0.9	0
	Constant	2.038	0.97	10^{-4}
	Cyclic	1.761	0.85	10^{-4}
Validation R_2	Original	98.16	0.9	0
	Constant	98.23	0.93	0
	Cyclic	98.31	0.93	10^{-5}
Tetrahedron R_2	Original	87.07	0.9	0
	Constant	87.31	0.85	10^{-5}
	Cyclic	88.12	0.85	0
Honeycomb R_2	Original	94.78	0.9	0
	Constant	95.67	0.93	0
	Cyclic	96.37	0.85	0

2.4.5 Final model

From the hypertuning process, we choose the S32D12 model trained by a cyclic training scheme with a momentum of 0.85, an accordingly chosen learning rate from the learning range test and zero weight decay. The model contains 1.3×10^7 model parameters. The main performance metrics are shown in Table 2.4 where "Tetrahedron" and "Honeycomb" refer to the selected set scores. Although we have mainly considered the mean friction R^2 score during the hypertuning we find that the performance on the remaining parameters is reasonable as well. The validation set reveals a final R^2 score for the mean friction of $\sim 98\%$ and a rupture accuracy of $\sim 96\%$. Since the data only contains roughly 12% ruptures this should be compared to the accuracy score of 88% corresponding to never predicting a rupture. Looking at the relative error for the rupture strain we find a large error of $\sim 13\%$. This is lower for the Tetrahedron (5.9%) and Honeycomb (1.5%) set. Thus we might get the idea that the relative error is especially high on the validation set due to a few cases of very low rupture strains in the Random walk patterns which would shift up the average.

I find it a bit disturbing to use the "$\sim$" too much, but I would like to indicate that a round the numbers... can I omit it?

Fig. 2.14 shows the mean friction, max friction and relative contact in comparison to the Tetrahedron $(7, 5, 1)$ and Honeycomb $(2, 2, 1, 5)$ used in the pilot study. We note that these configurations are also partly contained within the training data, but this serves as a visualization of how the fit looks for R^2 scores above 98%. We will evaluate a true test set later on based on the proposals from the accelerated search.

Table 2.4: Mean values are used over different configurations.

	Loss $[10^2]$	R_2 $[10^2]$			Abs. $[10^2]$	Rel. $[10^2]$	Acc. $[10^2]$
	Total	Mean F_f	Max F_f	Contact	Porosity	Rup. Strain	Rupture
Validation	2.1488	98.067	93.558	94.598	2.325	12.958	96.102
Tetrahedron	4.0328	88.662	85.836	64.683	1.207	5.880	99.762
Honeycomb	8.6867	96.627	89.696	97.171	1.040	1.483	99.111

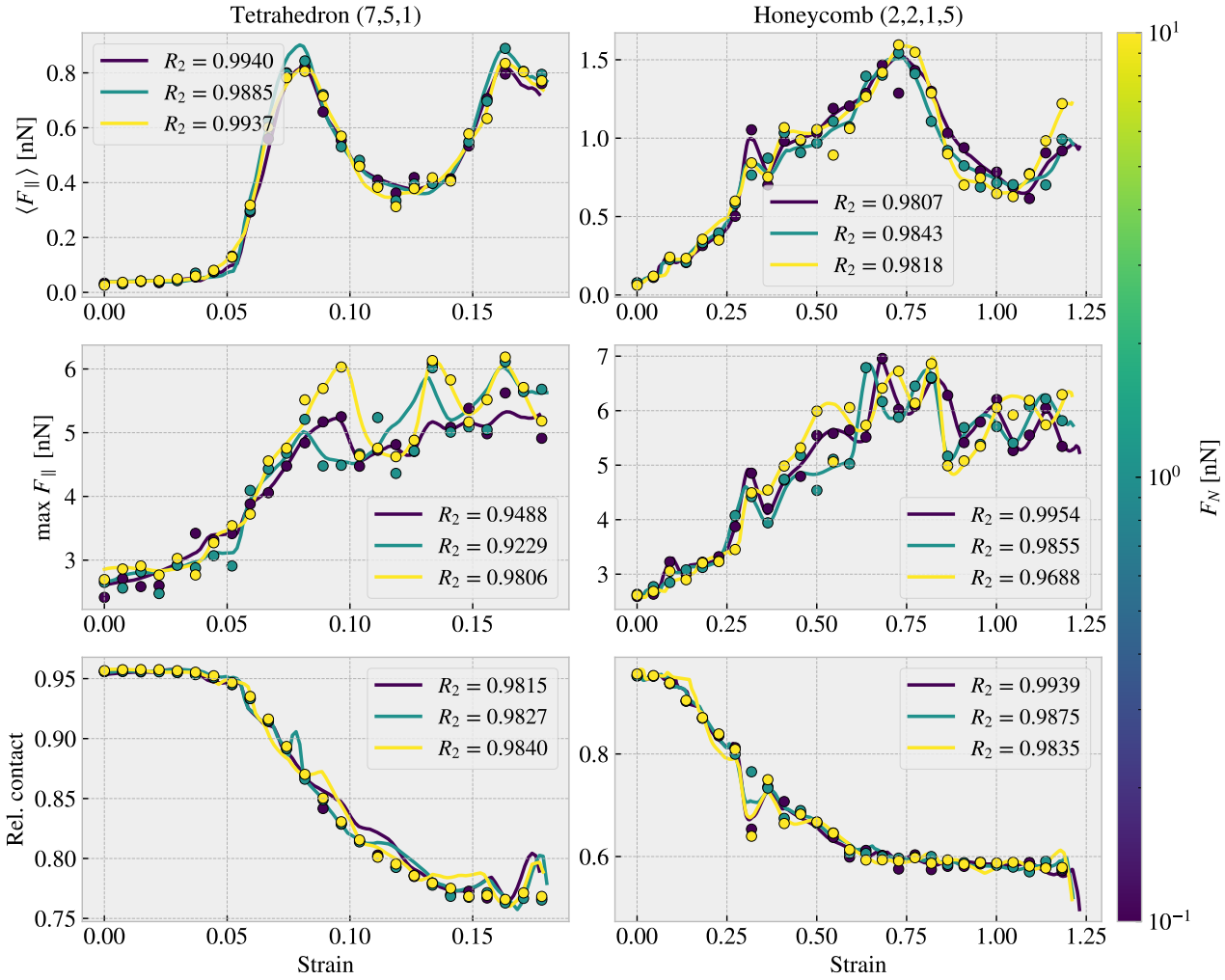


Figure 2.14: With 10^3 points in the strain range [0, 1.5] and stopping after first rupture prediction.

Using our final model, we evaluate the performance for the task of ranking the configurations by the properties of interest. That is, we go through all the configurations in the dataset, for the Tetrahedron, Honeycomb and Random walk respectively, calculate the properties of interest and sort the configurations accordingly. This is shown in **??**–2.7 in comparison to the actual ranking in the dataset. Generally, we find that the ML performs rather well in the ranking of the maximum properties by getting the right configurations into the top 3 three, while it is performing a lot worse for the minimum friction property. This latter observation can be attributed to the fact that the precision needed for an accurate ranking among the minimum friction cases is a lot greater than for the remaining properties. This is supported by the fact that the model predicts the Random walk 12 (Table 2.7) to have negative mean friction, showing a lack of precision. For the maximum categories, we find that the model gives a slightly better ranking for the Tetrahedon and Honeycomb in comparison to the Random walk patterns. When considering the actual predicted property scores for the maximum properties we find that the model predictions are generally within a ~ 0.2 nN in the top 5. This gives an incentive that our model can be used to perform an accelerated search of new configurations yielding a meaningful ranking of property scores.

Table 2.5: Tetrahedon

ML Rank	Data Config	Data Value [nN]	ML Config	ML Value [nN]	Data Rank
min F_{fric}					
20	(3, 9, 4)	0.0067	(3, 1, 2)	0.0041	5
5	(3, 1, 3)	0.0075	(1, 3, 4)	0.0049	11
6	(5, 3, 4)	0.0084	(1, 3, 3)	0.0066	6
21	(1, 7, 3)	0.0084	(3, 1, 4)	0.0066	8
1	(3, 1, 2)	0.0097	(3, 1, 3)	0.0078	2
max F_{fric}					
1	(5, 3, 1)	1.5875	(5, 3, 1)	1.5920	1
2	(1, 3, 1)	1.4310	(1, 3, 1)	1.2739	2
4	(3, 1, 2)	1.0988	(9, 3, 1)	1.1162	4
3	(9, 3, 1)	1.0936	(3, 1, 2)	0.7819	3
5	(7, 5, 1)	0.7916	(7, 5, 1)	0.7740	5
max ΔF_{fric}					
1	(5, 3, 1)	1.5529	(5, 3, 1)	1.5578	1
2	(1, 3, 1)	1.3916	(1, 3, 1)	1.2331	2
4	(3, 1, 2)	1.0891	(9, 3, 1)	1.0807	4
3	(9, 3, 1)	1.0606	(3, 1, 2)	0.7778	3
5	(7, 5, 1)	0.7536	(7, 5, 1)	0.7399	5
max drop					
1	(5, 3, 1)	0.8841	(5, 3, 1)	0.8603	1
2	(3, 5, 1)	0.4091	(3, 5, 1)	0.3722	2
4	(7, 5, 1)	0.3775	(1, 1, 1)	0.2879	5
5	(9, 7, 1)	0.2238	(7, 5, 1)	0.2478	3
3	(1, 1, 1)	0.1347	(9, 7, 1)	0.1302	4

Table 2.6: Honeycomb

ML Rank	Data Config	Data Value [nN]	ML Config	ML Value [nN]	Data Rank
min F_{fric}					
1	(2, 5, 1, 1)	0.0177	(2, 5, 1, 1)	0.0113	1
9	(2, 4, 5, 1)	0.0187	(2, 5, 5, 3)	0.0149	7
7	(2, 4, 1, 1)	0.0212	(2, 5, 5, 1)	0.0182	4
3	(2, 5, 5, 1)	0.0212	(2, 5, 3, 1)	0.0186	5
4	(2, 5, 3, 1)	0.0226	(2, 4, 1, 3)	0.0198	15
max F_{fric}					
1	(2, 1, 1, 1)	2.8903	(2, 1, 1, 1)	2.9171	1
2	(2, 1, 5, 3)	2.2824	(2, 1, 5, 3)	2.4004	2
6	(2, 1, 3, 1)	2.0818	(2, 1, 5, 1)	2.1060	5
4	(2, 1, 3, 3)	2.0313	(2, 1, 3, 3)	1.9458	4
3	(2, 1, 5, 1)	2.0164	(2, 4, 1, 1)	1.9381	6
max ΔF_{fric}					
1	(2, 1, 5, 3)	2.0234	(2, 1, 5, 3)	2.1675	1
2	(2, 1, 1, 1)	1.9528	(2, 1, 1, 1)	2.0809	2
3	(2, 4, 1, 1)	1.8184	(2, 4, 1, 1)	1.9157	3
4	(2, 1, 3, 3)	1.7645	(2, 1, 3, 3)	1.6968	4
5	(2, 4, 1, 3)	1.4614	(2, 4, 1, 3)	1.5612	5
max drop					
1	(2, 3, 3, 3)	1.2785	(2, 3, 3, 3)	1.3642	1
2	(2, 1, 3, 1)	1.1046	(2, 1, 3, 1)	0.9837	2
3	(2, 3, 3, 5)	0.8947	(2, 3, 3, 5)	0.9803	3
4	(2, 1, 5, 3)	0.8638	(2, 1, 5, 3)	0.9556	4
13	(2, 5, 1, 1)	0.8468	(2, 4, 5, 3)	0.8999	8

Table 2.7: RW

ML Rank	Data Config	Data Value [nN]	ML Config	ML Value [nN]	Data Rank
min F_{fric}					
1	12	0.0024	12	-0.0011	1
24	76	0.0040	06	0.0036	27
6	13	0.0055	14	0.0074	23
31	08	0.0065	05	0.0082	19
26	07	0.0069	63	0.0085	57
max F_{fric}					
3	96	0.5758	99	0.5155	2
1	99	0.5316	98	0.4708	3
2	98	0.4478	96	0.4356	1
4	97	0.3624	97	0.3503	4
11	58	0.3410	55	0.2817	7
max ΔF_{fric}					
3	96	0.5448	99	0.4669	2
1	99	0.4769	98	0.4314	3
2	98	0.4085	96	0.4128	1
4	97	0.3268	97	0.3080	4
78	57	0.2978	55	0.2542	7
max drop					
3	01	0.1818	00	0.1883	3
2	96	0.1733	96	0.1654	2
1	00	0.1590	01	0.1532	1
11	37	0.1022	04	0.0591	8
28	34	0.0879	56	0.0552	20

2.5 Accelerated Search

From Sec. 2.4 we have found promising results that we can use a machine learning model to predict the friction behavior. This enables us to omit the MD simulations in the evaluation process and perform an accelerated search through new configurations. We will use the friction properties of interest as our main optimization metrics. We approach the accelerated search by two different methods:

1. Using the generative algorithms developed for the creation of the Tetrahedron, Honeycomb and Random walk patterns, we create an extended dataset and evaluate the performance using the ML model.

2. Using the genetic algorithm method we perturb (mutate) the configurations and optimize for the maximum drop property using the ML model to evaluate the fitness function.

2.5.1 Patteren generation search

We utilize the pattern generators developed in Chapter 1 to create an extended dataset for our search. For the Tetrahedron and Honeycomb patterns, the increment of the parameters will lead to an increased spacing within the pattern. This will eventually lead to the main pattern structures exceeding the size of the sheet. Thus, we can essentially perform a full search "maxing out" the parameters of these patterns. We estimate that this is done with the max parameters, $(60, 60, 30)$ for the Tetrahedron, and $([30, 30, 30, 60])$ for the Honeycomb pattern. We use a random reference position and regenerate each unique parameter 10 times to explore translational effects. This gives in total 135k configurations for the Tetrahedron pattern and 2025k for the Honeycomb pattern. For the Random walk generator, we do a Monte Carlo sampling. In each sample, we draw the scalar values,

either from a uniform (U) or logarithmic uniform (LU) distribution as follows.

Num. walks $\sim$ U$[1, 30]$	Max. steps $\sim$ U$[1, 30]$	Min. dis. $\sim$ U$[0, 4]$
Bias direction $\sim$ U$[0, 2\pi]$	Bias. strength $\sim$ LU$[0, 10]$	$p_{\text{stay}} \sim$ U$[0, 1]$

Notice that we use discrete distribution for the parameters requiring integers. For the binary parameters *Connection*, *Avoid invalid*, *RN6* and *Grid start* we simply set the values by a 50–50 chance. The remaining parameters are kept constant at *Periodic: True* and *Centering: False* throughout the search. For the handling of clustering, we implement the repair algorithm such that the sheet is repaired by the least modifications approach rather than retrying the generation several times Make sure that this is introduced somewhere. Due to the extra computation time associated with the random walk and the repair algorithm, we only generate 10k configurations within this class. For the ML evaluation of the configurations we use a normal load of 5 nN and generate a strain curve in the domain $[0, 2]$ using 100 evenly spaced points. We compute the properties of interest and rank the configurations accordingly. The top candidates for each property are shown in Table 2.8 including a comparison to the original dataset top candidates from Table 2.2. The random walk top five candidates are visualized in Fig. 2.15.

Table 2.8: Pattern search. The values are in units nN.

	Search			Data		
Scores	Tetrahedron	Honeycomb	Random walk	Tetrahedron	Honeycomb	Random walk
min F_{fric}	-0.062	-0.109	-0.061	0.0067	0.0177	0.0024
max F_{fric}	1.089	2.917	0.660	1.5875	2.8903	0.5758
max ΔF_{fric}	1.062	2.081	0.629	1.5529	2.0234	0.5448
max drop	0.277	1.250	0.269	0.8841	1.2785	0.1818
Configs.	Tetrahedron	Honeycomb	Random walk	Tetrahedron	Honeycomb	Random walk
min F_{fric}	$(13, 11, 14)$	$(14, 25, 7, 19)$	*no name*	$(3, 9, 4)$	$(2, 5, 1, 1)$	12
max F_{fric}	$(1, 3, 1)$	$(2, 1, 1, 1)$	*no name*	$(5, 3, 1)$	$(2, 1, 1, 1)$	96
max ΔF_{fric}	$(1, 3, 1)$	$(2, 1, 1, 1)$	*no name*	$(5, 3, 1)$	$(2, 1, 5, 3)$	96
max drop	$(1, 7, 1)$	$(3, 3, 5, 3)$	*no name*	$(5, 3, 1)$	$(2, 3, 3, 3)$	01

First of all, we notice that the top candidates for the minimum friction are all predicted to have a negative friction value. This unphysical prediction aligns with the previous observations that our model does not have the required precision to yield accurate predictions for this property. Moreover, we can argue that pursuing the optimization for a low friction value will eventually highlight the weaknesses of the model as we reward an unphysical negative value. In order to resolve this problem one might need to extend the training dataset and possibly include a physical constraint for positive friction values. However, by consulting the proposed minimum candidates we find that they all share the same feature of being sparsely cut. For the Random walk, we see this visually in Fig. 2.15, while for the Tetrahedon and Honeycomb patterns, this is evident from the configuration parameters shown in Table 2.8 where the parameters reveal a high spacing between the cuts. The porosity of the minimum friction top candidates are all rather low being 1.5%, 5.6%, and 1.6% for the Tetrahedron, Honeycomb and Random walk respectively. This further supports the idea that the Kirigami sheet can not readily be used to reduce friction within our system domain since the results point toward a non-cut sheet as the best minimum friction candidate.

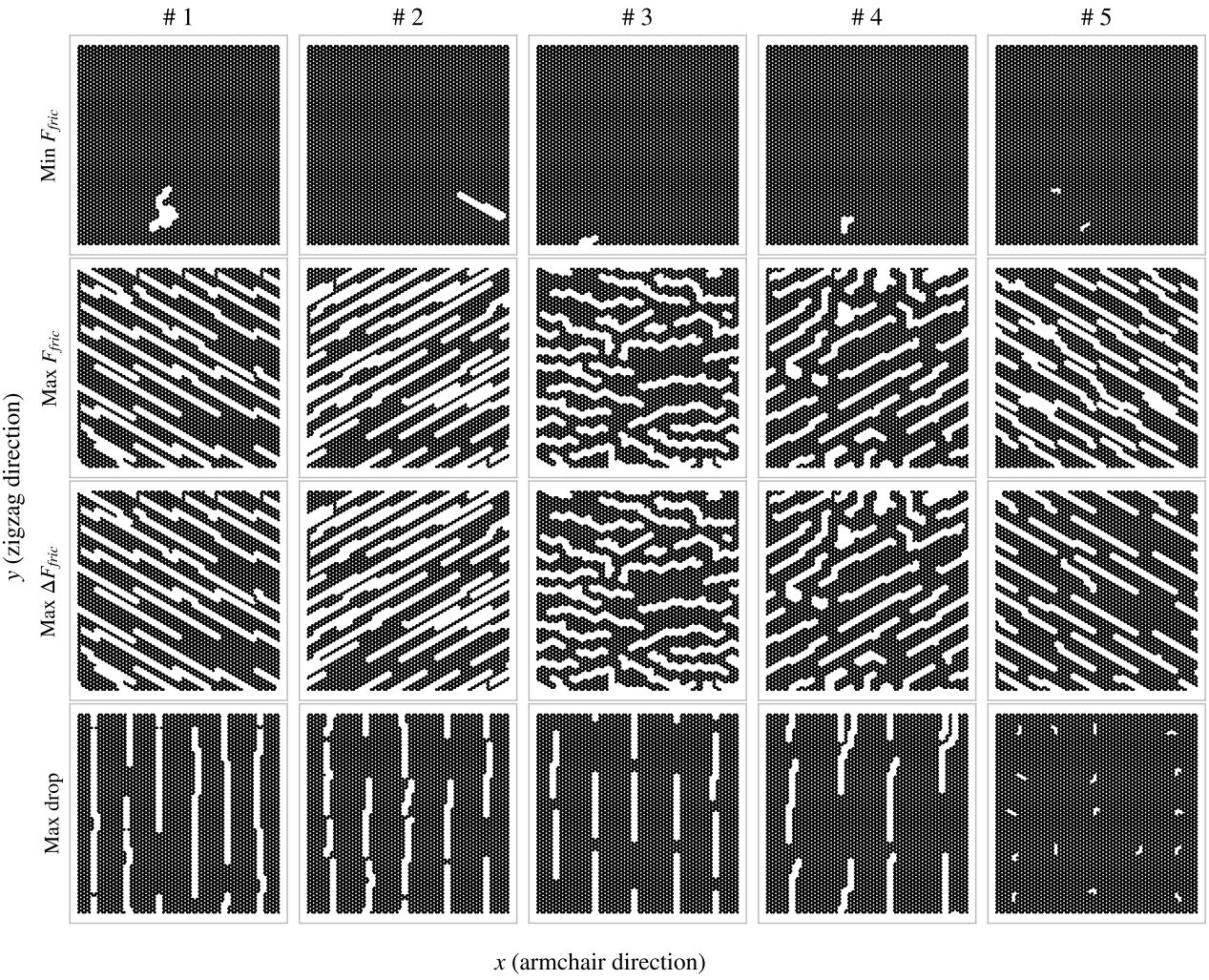
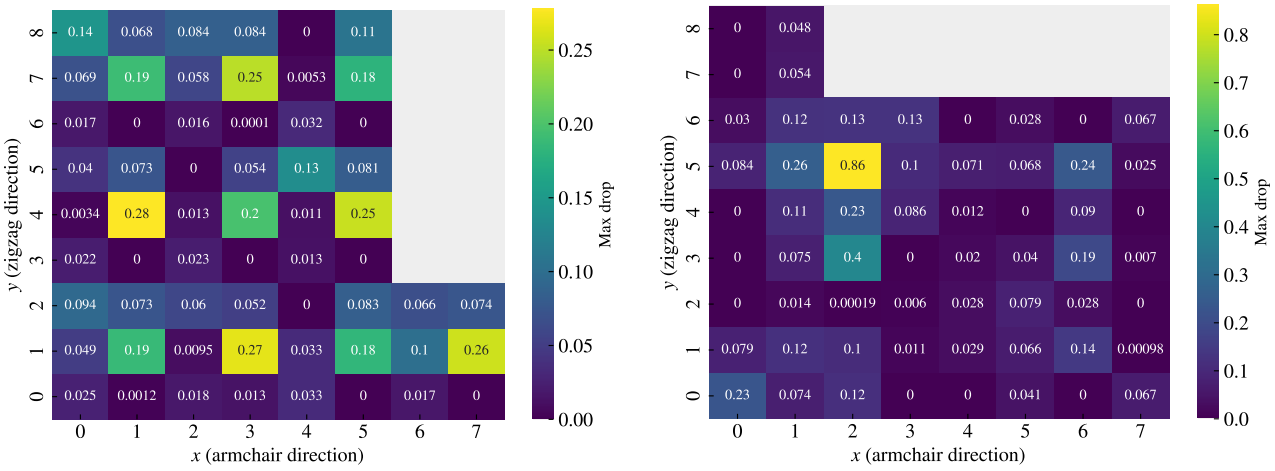


Figure 2.15: RW search top results.

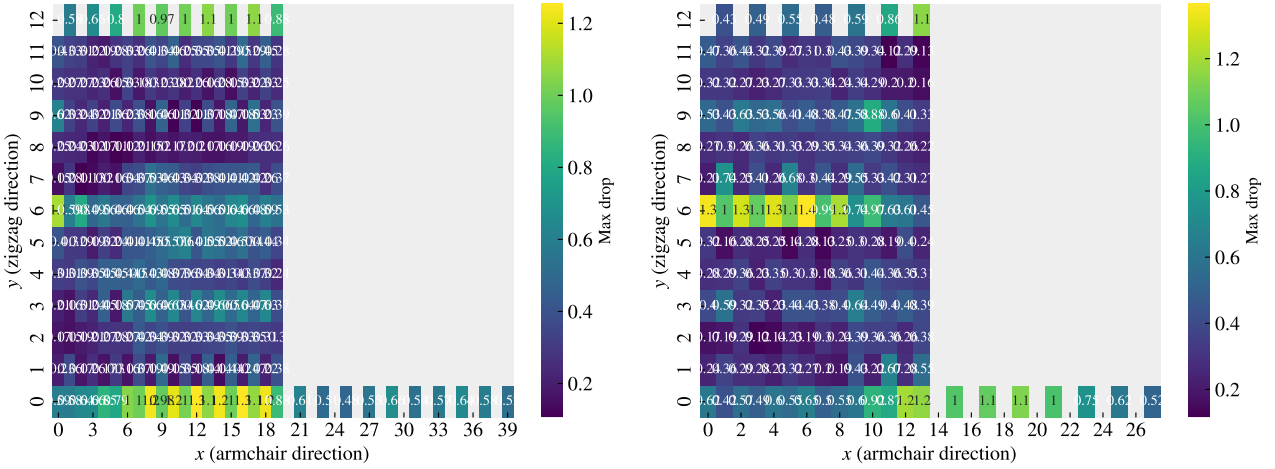
Among the remaining maximum properties, we find competing values for the Honeycomb and Random walk classes. However, the top-scoring values for the Honeycomb correspond to configurations within the original dataset which is also the case for the Tetrahedron top candidates. The only difference is the randomized reference position. When taking a closer look at the ranking for each property it becomes apparent that the predictions are highly sensitive to the reference position parameter used for the Tetrahedron and Honeycomb pattern. Since we repeated each pattern parameter 10 times with a random reference position, we initially expected to get a ranking in sets of 10. However, the ranking only shows contiguous appearing sets in the range of 1–5 which points toward a dependency on pattern translation. Hence we investigate this further by evaluating the scores for a systematic change of the reference position. We generally find the max drop parameter to give the highest variation and thus we show the max drop scores for the max drop top candidates Tetrahedron $(1, 7, 1)$, $(5, 3, 1)$ and Honeycomb $(3, 3, 5, 3)$, $(2, 3, 3, 3)$ in Fig. 2.16. The results show that the max drop property prediction varies drastically with the translation of these patterns. The emerging question is then whether this is grounded in a physical phenomenon or simply a deficiency in the ML. Even though the patterns are periodic in the x-y-plane, with a period according to the unique number of translations shown in Fig. 2.16, the translation will determine the specific configuration of the edge. Previous studies of static friction and stick-slip behavior point to the importance of edge effects refer to theory or directly to the sources?, and thus for a sheet where the atoms on the $\pm x$ free edge constitute about 2.5% of the inner sheet atom count, it is not unreasonable that the translation might result in a significantly different outcome. In that case, the search through reference positions highlights that the translation can be key to optimizing for certain properties. However, the results might also indicate that the model is either overfitted or that we simply did not provide enough data to reach a generalization of

the complex physical behavior of the system. The sensible way forward to unravel this would be to generate additional translational variants of the same configurations to investigate for any physical edge dependencies or otherwise strengthen the model by this data. We earmark this suggestion for another study. When considering some of the friction-strain curves we also find that the prediction of the rupture point plays an important role for the max drop property. As the rupture is often predicted on a descending part of the curve any variation to the rupture strain will affect the max drop property quite significantly.



(a) Tetrahedron $(1, 7, 1)$ (60). Std = 0.08, Rel. Std = 1.13

(b) Tetrahedron $(5, 3, 1)$ (60). Std = 0.13, Rel. Std = 1.61



(c) Honeycomb $(3, 3, 5, 3)$ (260). Std = 0.25, Rel. Std = 0.58

(d) Honeycomb $(2, 3, 3, 3)$. (182) Std = 0.27, Rel. Std = 0.60

Figure 2.16: CAPTION Fix overlapping annotation text

In order to get insight into the generalization of the model, we evaluate the performance on a true test set. We use the 20 configurations given by the top 5 candidates for each of the properties of interest in the Random walk search shown in n Fig. 2.15. We calculate the ground truth using MD simulation by sampling 30 pseudo uniform strain values using a normal load of 5 nN. Unfortunately, the test set reveals a significantly worse performance than the validation scores reported so far. It shows a loss of 2.13, which is two orders of magnitude higher than the validation scores, an average absolute error for the mean friction of 0.14 nN and a rupture accuracy of 70 %. The mean friction average R^2 score is negative which indicates that our model performs worse than simply guessing on a constant value given by the true data mean. This reveals that our model is not generalized enough to provide accurate predictions on the newly generated Random walk configurations. This can mainly be attributed two reasons: 1) The test set data distribution is not similar to that of the training and validation data drawn from the original data set. 2) The considerations of the selected Tetrahedron and Honeycomb dataset, which overlapped with the training data, has led to an overfitting of the model. However, by going back to the hypertuning process and choosing the best model (C16D8) from the architecture complexity grid test given by

the lowest validation score, trained with a constant learning rate, we get similar results. This indicates that the poor test result is not simply caused by the hypertuning choices. Instead, it points to the fact that our original training data does not contain a generalized distribution that accurately captures the full complexity of our system. This aligns with the high fluctuation in prediction value when translating the patterns. Thus we conclude that a machine learning approach might be feasible, given the promising validation scores, but that we need a bigger and more generalized dataset for a reliable prediction of new configurations.

2.5.2 Genetic algorithm search

Despite the fact that our machine learning analysis indicates that the model are not generalized for the prediction on new configurations we investigate the appraoch of using a gentic algorithm for an accelerate search.

For the second approach to an accelerated search, we consider the genetic algorithm. So far we have concluded that a minimization of the friction is not promising, and hence we discard this property for further studing. We have also seen that the maximum style properties often share similar top candidates, and thus we choose to only investigate the max drop property, associated with the aim of creating a negative friction coefficient. We are going to use the extended search top candidate as a basis for the genetic algorithm population algorithm search. We generate a population of 100 configurations using the settings associated with the max drop top candidate for the Tetrahedron, Honeycomb and Random walk patterns respectively. We run the search for 50 generations as we did not see much difference for any longer runs. The Tetrahedron and Honeycomb search did immediately not give any improvements as the highest scoring individual from the population was not improved through the search even though the average score was rising initially. For the Random walk, we choose to perform a genetic algorithm search for each of the top 5 candidates. Most of them gave a similar result as seen from the Tetrahedon and Honeycomb pattern with only a single new candidate generated. The score of this candidate was 0.240 nN which is a small improvement from the best score of 0.182 nN. However, from some of the other runs the population initialization provided a top score of 0.345 nN which shows that we have better hopes of optimizing this property by simply generating more configurations from the top candidate parameters. The fact that starting from an existing design did not give any useful results we attempted to start from a population of random noise as well. We made one population from mixed porosities, having even 20 individuals each for porosity $\{0.01, 0.05, 0.1, 0.2, 0.3\}$, and two populations based on a porosity of 0.25 and 0.5 respectively. This time the algorithm improved the top candidate throughout, but the actual scores are still not impressive. The mixed porosity start gave the highest score, being 0.299 nN. When considering the corresponding patterns from the top five candidates in this search, they were all visually quite similar to the starting configurations; they still looked like random noise. Thus, we do not find signs of a generation of any noticeable patterns worth further investigating. By the use of the Grad-CAM method, we examined for any noticeable patterns for the model predictions on this mixed porosity top candidate as shown Fig. 2.17. For comparison, we included a similar examination for the top candidates in the pattern generation search with respect to the max drop category for the Tetrahedron, Honeycomb and Random walk shown Fig. 2.18 to 2.19. For the mixed porosity top candidate, the Grad-CAM method highlights some areas in the noise configuration as contributing more positively than others. However, we do not see any obvious patterns. For the more structured patterns of the Tetrahedron, Honeycomb and Random walk configurations we see in many cases throughout the straining that the cut parts are highlighted. This gives some confidence to the idea that the model does consider some of the relevant features in the pattern, but it still varies too much to make any strong conclusion. However, we notice that for certain strain values, the Grad-CAM reveals an "attention" towards the edge of the configuration. This especially relates to the top and bottom edge, which we for instance see for the Honeycomb pattern at strain 0.396 regarding the bottom edge in Fig. 2.19. Considering that the top and bottom of the configuration are not a true edge, since these are connected to the pull blocks in the simulation, this is a bit surprising. One interpretation is that the dissipation associated with the thermostat in the pull blocks might be of importance. This is not strongly supported and we simply note that as an interesting topic for further studies.

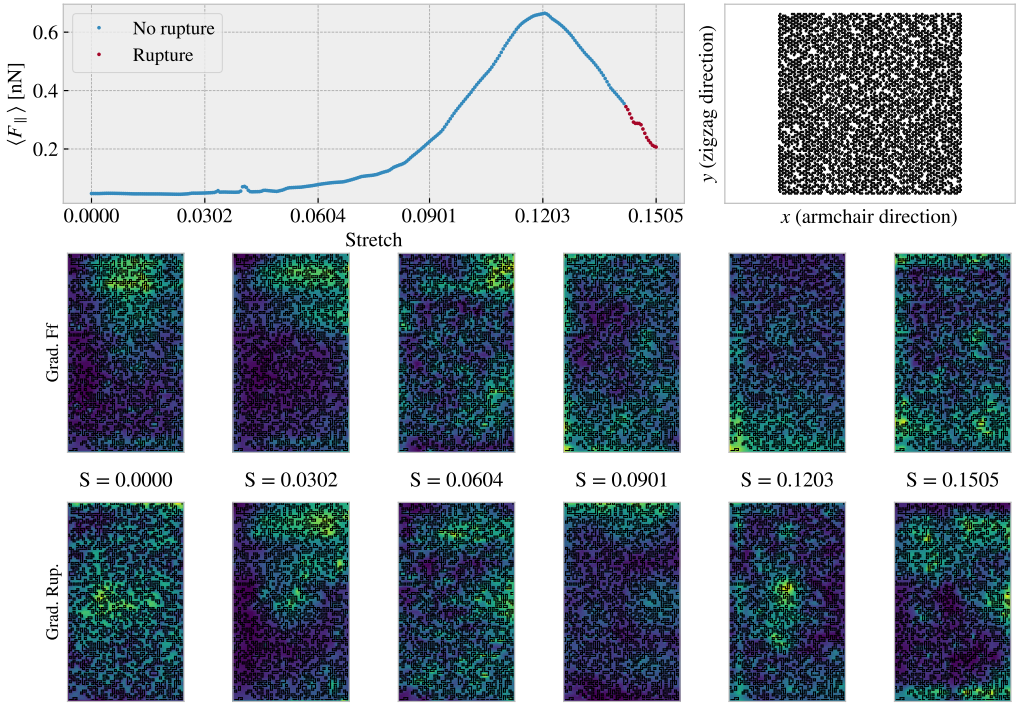


Figure 2.17: $p \in \{0.01, 0.05, 0.1, 0.2, 0.3\}$.

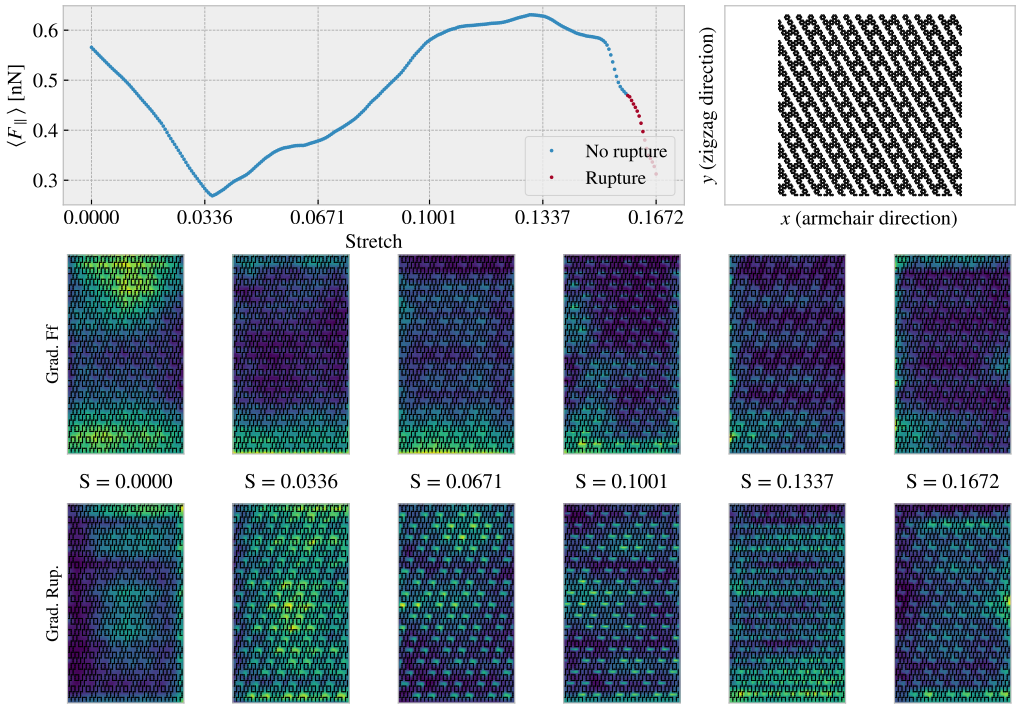


Figure 2.18: Tetrahedron $(1, 7, 1)$, ref $= (1, 4)$

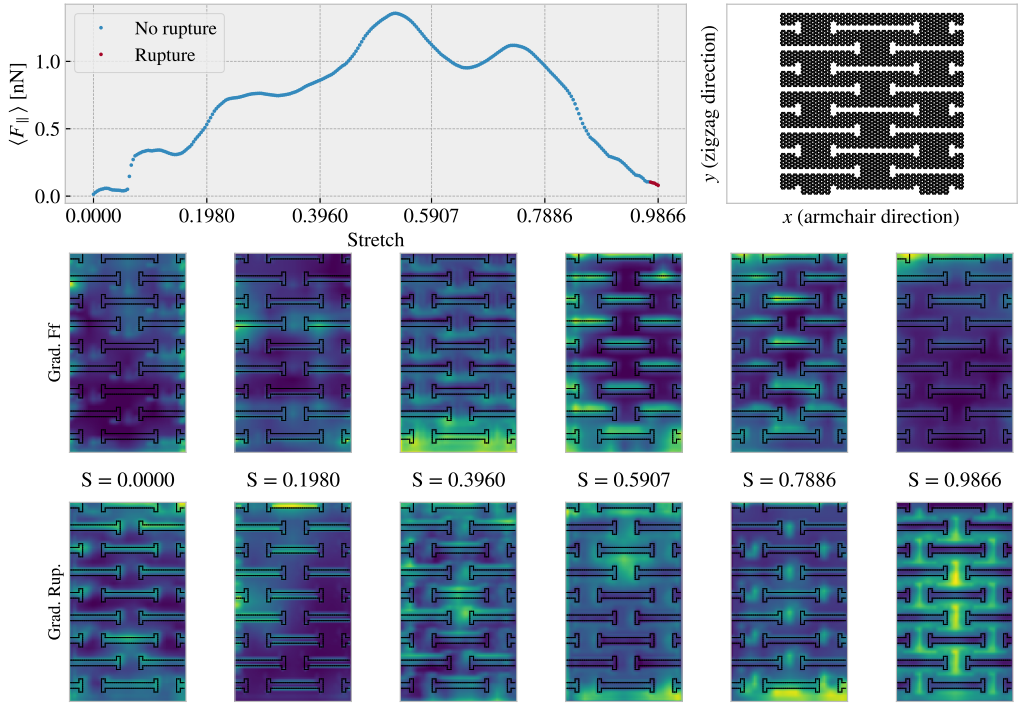


Figure 2.19: Honeycomb $(3, 3, 5, 3)$, ref $= (12, 0)$

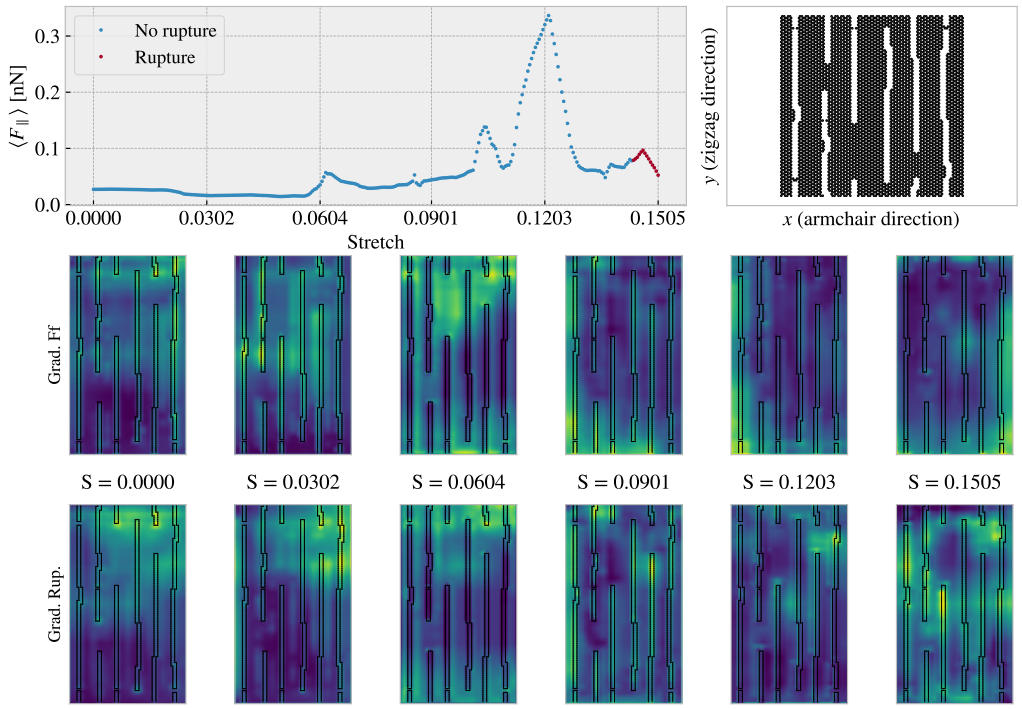


Figure 2.20: RW.

Chapter 3

Summary

The work presented in this thesis covers several topics (find a better opening line?). We have created an MD simulation which enabled us to study the frictional behavior of a graphene sheet sliding on a Si substrate. In addition, we have created a numerical framework for creating Kirigami design patterns and introducing these into the friction simulations. This was used to study the effects of the out-of-plane buckling induced by a selected pair of Kirigami designs in relation to a non-cut sheet under the influence of strain. Further, we have created a dataset of various Kirigami designs for the scope of investigating the possibilities with Kirigami design. We have investigated the possibility to use machine learning on this dataset and attempted an accelerated search. Finally we look into the prospects of achieving a negative friction coefficient for a system with coupled load and stretch. In this chapter we will summarize the findings and draw some final conclusions. We will also provide some topics for further research.

3.1 Summary and conclusions

3.1.1 Design MD simulations

We have designed an MD simulation for the examination of friction for a graphene sliding on a substrate. Some of the key features for the numerical procedure were that we managed the sheet through pull blocks in the ends. We could then apply load and stretch the sheet without acting directly on the inner parts. Say something about parameter dependencies from the Pilot study.

3.1.2 Design Kirigami framework

We have desgined a numerical framework for creating Kirigami designs. By defining an indexing system for the hexagonal lattice structure we were able to define the Kirigami designs as 2D matrix for numerical implementation. We digitalized two different macroscale designs, which we named the *Tetrahedron* and *Honeycomb* pattern respectively, that successfully produced out-of-plane buckling when stretched. Through a numerical framework we could create an ensemble of perturbed variations which gave approximately 135k configurations for the Tetrahedron pattern and 2025k patterns for the size of the sheet used in our study. When considering the possibility to translate the patterns this gave roughly a factor 100 more of unique pertubations. We also created a framework for creating Kirigami designs through a random walk. This was further controlled by introducing features such as bias, avoidance of existing cuts, preference to keeping a direction and procedures to repair the sheet for simulation purposes. The capabilities of the numerical framework for generating Kirigami designs was far larger than the capabilities for producing MD designs within the time constraint of this thesis. Thus we believe that this contains the possibility to benefit more extended studies and for the creation of a larger dataset.

3.1.3 Control friction using Kirigami

We have investigated the friction behavior of the non-cut sheet and a selected Tetrahedron and Honeycomb pattern under various stretch and load. The non-cut sheet did not exhibit significant out-of-plane buckling as opposed to the Tetrahedron and Honeycomb pattern. This is even when considering that the non-cut sheet had

a yield strain of 0.35 while the Tetrahedron had a lower yield strain of 0.21 and the Honeycomb a considerable larger one at 1.27 based on a stretch in vacuum. The out-of-plane buckling resulted in a signifciant reduction of the contact area as the sheet were stretched, towards a minimum of X for the Tetrahedron and y for the Honeycomb pattern. However, this disagreed with the asperity theory hypothesis of a decreasing friciton with decreasing contact area. We found that the strain-induced buckling was initially (at low relative strain) assocated with an increase in friction. Moreover, the friction-strain curve produced a no-linear behaviour which was not compatible with the approximately monotonic decreasing contact area as strain were increased. This is shown in **??**. This led us to the conclusion that the contact area cannot be attributed a dominant mechanism for friction throughout the straining of the studied Kirigami sheets. In general we found a non-existing relationship between friction and load considering the uncertainties in the simulation. This is best attributed to the superlubric state of the graphene sheet on the substrate. The slope of the friction-load curves were not significantly affected by the straining of the Kirigami sheet and thus we conclude that the load effect on friction is neglible cimpared to the strain effects.

3.1.4 Capture trends with ML

With the use of MD simulations, we have generated an extended dataset of 9660 data points based on 216 Kirigami configurations (Tetrahedron: 68, Honeycomb: 45, Random walk: 100, Pilot study: 3) under various strains (15 values) and normal load (3 values). The dataset shows some general correlation, such as a positive correlation between the mean friction force and strain (0.77) and porosity (0.60), and a negative correlation to contact area (-0.67). These results align with the findings from the pilot study suggesting that these features are involved, but not necessarily the cause, in the observed phenomena. By defining the friction property metrics: $\min F_{\text{fric}}$, $\max F_{\text{fric}}$, $\max \Delta f_{\text{fric}}$ and max drop, we investigated the top candidates within our dataset. This suggested that we cannot readily reduce friction using the investigated Kirigami configurations under strain as the non-cut sheet provided the lowest overall friction. Regarding the maximum properties, we found an improvement from the original pilot study values and with the Honeycomb pattern producing the highest scores. This provides an incentive that the data contains information for optimization with respect to these properties. Among the top candidates, we generally found that a rather flat friction-strain profile is mainly associated with little decrease in the contact area and vice versa.

We implemented a VGGNet-16-inspired convolutional neural network with a deep "stairlike" architecture: C32-C64-C128-C256-C512-C1024-D1024-D512-D256-D128-D64-D32, for convolutional layers C with the number denoting channels and fully connected (dense) layers D with the number denoting nodes. The model contains 1.3×10^7 and was trained using the ADAM optimizer for a cyclic learning rate and momentum scheme for 1000 epochs while saving the best model during training based on the validation score. The final model gives mean friction R^2 of $\sim 98\%$ and a rupture accuracy of $\sim 96\%$. However, we got lower scores for a selected subset of the Tetrahedon ($R^2 \sim 88.7\%$) and Honeycomb ($R^2 \sim 96.6$) pattern based on the top 10 max drop scores respectively. These scores were lower despite the fact that the selected set was partly included in the training data and that the hypertuning favored the performance of the selected Tetrahedron set. Thus we conclude that these configurations, associated with a highly non-linear friction-strain curve, are more challenging to predict. One interpretation is that these depend on more complex dynamics and perhaps that this is not readily distinguished from the behavior of the other configurations within the data. By evaluating the ability of the model to rank the dataset according to the property scores we found in general a good representation of the top 3 scores for the maximum categories, while the minimum friction property ranking was deviating a lot more. We attribute this to the higher required precision to rank the lowest friction values properly which the model does not possess. We created a test set based on MD simulations for the top 5 candidates for each property of interest generated by Random walk configurations in the accelerated search. This showed a significantly worse performance with a two-order of magnitude higher loss and a negative friction mean R^2 score which corresponds to the prediction being worse than simply guessing on a constant value based on the true data mean. However, by going back to the beginning of the architecture complexity hypertuning and choosing the model purely based on the lowest validation score we get similar results. This indicates that it is not simply a product of overfitting in hypertuning based on the selected set (which overlapped with the training data). Instead, it points to the fact that our original dataset did not cover a wide enough configuration distribution to accurately capture the full physical complexity of the Kirigami friction problem.

3.1.5 Accelerated search

Using the machine learning model we performed two types of accelerated search. One by going generating an extended dataset of Tetrahedron, Honeycomb and Random walk patterns and evaluating the results using the model. Another was based on the genetic algorithm were we pertubated the candidates in order to optimize for the max drop friction peroperty. The generative accelerated search method did produce some new candidates, but it is unsure how reliable this is due to the insufficienty in the machine learning model. We concluded that the minimization of friction was not applicable. The model revealed a sensitivity to edge effect which is likely due to a model insufficienty, but this serves as an interesting topic for future research. For the gentic algorithm search it did mainly not find any new candidates and we believe that the simplicity of the genetic algorithm was not immediately suited for creating new designs with the necessary spatial correlations presented

3.1.6 Negative friction coefficient

By enforcing a coupling between load and stretch, mimicking a nanomachine attached to the sheet, we investigated the load curves arising from load the Tetrahedron and Honeycomb pattern from the pilot study. The non-linear trend observed for increasing strain carried our to this simulation setup and produce a highly non-linear friction-load curve. This demonstrated a negative friction coefficient say something about the values.

3.2 Outlook / Perspective

Having successfully demonstrated a non-linear effect on friction with increasing strain of the sheet our results invite a series of further studies to investigate this relation. First of all, it would be valuable to investigate how the friction-strain curve depend on temperature, sliding speed, spring constant, and on load for an increased range $F_N > 100nN$. This is especially interesting in the context up conditions leading to a stick-slip behvaior as our results were carried in the smooth sliding. Moreover, it would be important to verify that the choices for relaxation time and pauses are not critical for the qualitative observations as well as trying a different interatomic potential for the graphene and perhaps an entirely different substrate material. Especially the Adaptive Intermolecular Reactive Empirical Bond Order (AIREBO) potential for the modeling of the graphene sheet might be of interest. The effects from excluding adhesion (the LJ interaction) can also be useful for the investigation of the observed phenomena.

In order to get a better understanding of the underlying mechanism for the friction-strain relationship we might investigate commensurability effects further by varying the scan angle. we might also consider investigating the friction-strain relationship under a uniform load to get insight into whether the loading distribution is of importance. Another topic worth studying is the relation to scale. Thus it would be interesting to study size effects but also further look into edge effects by translating the pattern. With this regard, we would also suggest a more detailed study of the effect from the thermostat in the pull blocks which is suggested to have a possibly importance by judging from the machine learning model Grad-CAM analysis.

For machine learning, we can either try to extend the data set to resolve the issue of the model not being generalized enough. We can also create a dataset for a single kirigami design and include some of the mentioned physical variables above and attempt to use machine learning for unraveling these relations. In that context we would advice for as more detailed investigation of machine learning techniques. If succesfull this would invite a study of inverse design methods such as GAN or diffusion models.

- How is this behavior effected by scaling?
- How does the distirbution of normal load effect the Kirigami friction behavior?
- Things to vary: load range, scan directions, adhesive forces, longer relaxation time, different potential (AIREBO)
- Investigate if the contact area is effecting the friciton non-linear by turning of friction force for atoms corresopnding to those that lift of from the sheet during the out-of-plane buckling.
- Investigations of commensurability effects.
- Study dependency of translation of the patterns as suggested by the ML results.

- Investigate effects from pull blocks...
- Investigate effects from the thermostat since the top and bottom edges was shown interest by the model prediction.

Appendices

Appendix A

Appendix A

Appendix A

Appendix B

Appendix B

Appendix C

Bibliography

[1]L. Burrows, *New pop-up strategy inspired by cuts, not folds*, (Feb. 24, 2017) https://seas.harvard.edu/news/2017/02/new-pop-strategy-inspired-cuts-not-folds.

[2]*Scotch cushion lock protective wrap*, https://www.scotchbrand.com/3M/en_US/scotch-brand/products/catalog/~/Scotch-Cushion-Lock-Protective-Wrap/?N=4335+3288092498+3294529207&rt=rud.

[3]P. Z. Hanakata, E. D. Cubuk, D. K. Campbell, and H. S. Park, "Accelerated search and design of stretchable graphene kirigami using machine learning", Phys. Rev. Lett. **121**, 255304 (2018).

[4]P. Z. Hanakata, E. D. Cubuk, D. K. Campbell, and H. S. Park, "Forward and inverse design of kirigami via supervised autoencoder", Phys. Rev. Res. **2**, 042006 (2020).

[5]L.-K. Wan, Y.-X. Xue, J.-W. Jiang, and H. S. Park, "Machine learning accelerated search of the strongest graphene/h-bn interface with designed fracture properties", Journal of Applied Physics **133**, 024302 (2023).