Machine-Learning and Data Science Techniques in String and Gauge Theories

Edward Hirst[1*]

Thesis submitted in partial fulfillment of
the requirements for a doctorate of philosophy

March 2, 2023

[1] *Department of Mathematics, City, University of London, EC1V 0HB, UK*

[*] edward.hirst@city.ac.uk

Contents

Tables

Figures

Acknowledgements

I would like to take this opportunity to thank my many wonderful collaborators throughout the PhD, and in particular my academic siblings Jiakang Bao and Elli Heyes. I am grateful to STFC for the PhD studentship, and most of all I am indebted to my supervisor Professor Yang-Hui He for his unbounded and inspiring energy coupled perfectly with a plethora of enthralling ideas, and sagacious guidance.

Abstract

Techniques from supervised and unsupervised machine learning, along with those from data and network science, are applied to generated datasets of mathematical objects relevant to string and gauge theories. Investigations show success in identifying and learning new structure associated to these objects. Datasets considered in the research work completed for this thesis include: dessins d'enfants, quivers, Hilbert series, amoebae, polytopes, Calabi-Yau manifolds, brane webs, and cluster algebras.

1 Introduction

As physics seeks better description of the universe we live in, fundamental theory refines itself to describe smaller and smaller scales of the universe, of which larger scales can be statistically inferred. At these smallest scales quantum field theory dictates how particles interact via electromagnetism and the strong and weak nuclear forces, with one final obstacle of description: quantum gravity. Issues with the quantisation of the theory of general relativity [1] indicate the need for a paradigm shift for fundamental theory.

At present, techniques from gauge theory describe the quantisable forces through the language of group and representation theory [2], [3]. The standard model gauge group: $SU(3) \times SU(2) \times U(1)$, dictates the symmetry principles of these physical laws, which matter particles (here fermions) obey by existing as representations of this product group. The groups themselves are gauged, such that the degrees of freedom from the representation redundancy in description are transferred into gauge fields (here bosons). These gauge fields also exist as algebra representations, and hence particles in their own right.

In addition to the spin-1/2 spinor representations of fermions, and the spin-1 vector representations of bosons, to establish mass for these particles whilst allowing gauge symmetry spin-0 scalar Higgs fields must be introduced to spontaneously break the symmetry such that leading scalar coupling terms in the broken regime act alike traditionally mass terms via the Higgs mechanism [4], [5]. The standard model gauge theory may then be quantised to produce a quantum field theory with which practical perturbative computations may be performed [6]. However, as stated the canonical methods of quantisation are unsuccessful with gravity, producing a theory which is perturbatively non-renormalisable and hence not useful for computation.

String theory presents such a paradigm shift for fundamental theory, whereby the assumption that the universe is built from exclusively 0-dimensional particles is relaxed, and instead allows the presence of p-dimensional branes for varying p. Of these branes the 1-dimensional strings are the most relevant at lower energies, and hence dominate computation and consideration at the low coupling regimes pertinent to our world. The endpoints of these strings trace out D-branes, and these may be considered as additional dynamical objects in string theory also [7].

Starting as a methodology to describe hadrons [8]–[11], strings were then examined for bosonic vibrational patterns which had the potential to describe gravity in bosonic string theory [12], [13]. Extension to include fermionic vibrations, and relevant anomaly cancellations [14], [15] led to the consistent superstring theories: Type I, Type IIA, Type IIB, heterotic $SO(32)$, and heterotic $E_8 \times E_8$ [16], [17]. Later, dualities between the theories grouped them into a single larger M-theory [18], [19], where $p > 1$ dimensional branes become dominant dependent on coupling and dimension size in consideration.

For self-consistency, M-Theory must exist as a theory in 11 space-time dimensions (reducing to 10 dimensions in various superstring limits). To ensure a feasibly physical interpretation of these extra dimensions beyond our familiar (1,3)-space-time, these theories require compactification of the extra 6 dimensions on a Calabi-Yau (CY) manifold in the various superstring limits to describe a supersymmetric gauge theory consistent with our universe

[20]. However the specific manifold used, and the particular formulation of M-Theory leaves a huge amount of redundancy, with many methods of description via many mathematical objects; objects which are rarely considered from a data science perspective.

Estimated to be at the order of 10^{500}, the number of flux vacua over CY manifolds for string theory compactification establishes an enormous dataset with little known structure [21]. This string landscape alone already manifests the importance of big data approaches to high-energy theory. Beyond CY data, many other objects used in this field can also be computationally represented and generated. With the exponential growth of computational power, it is becoming easier to build large datasets of these mathematical objects, allowing analysis beyond the archetypal cases considered.

As datasets grow in size, exhaustive calculation becomes extensively more infeasible. More traditional calculation using blackboard and chalk is quickly outcompeted by the speed and consistency of computational methods. Data science as a field represents a colossal number of these computational methods for data generation, management, and analysis, and as academic research calls for more detailed examination of more scenarios, statistical techniques become an essential tool to pragmatically extract data structure.

Machine learning (ML) is the modern umbrella term for these techniques of computational statistics, and application of these techniques to generated databases of physically relevant objects is the central theme to the work in this thesis. These techniques are applied with the hope that results may inspire analytic work in new directions, through conjecture from statistical analysis, that ultimately help to identify paths to stronger mathematical and physical understanding.

The computational work completed in these studies made use of high performance computing (HPC) resources made available at City, University of London [22], [23]. Scripts were largely completed in `python` with the use of ML libraries such as `tensorflow` [24], `sci-kit learn` [25], along with general libraries for topological data analysis (TDA) (`ripser` [26]), network analysis (`networkx` [27]), and analytic computation (`sagemath` [28]), as well as `mathematica` [29]. A GitHub of repositories relevant to works in this thesis is available at: `https://github.com/edhirst`.

This thesis is structured such that §2 gives a brief introduction to the key players of mathematical objects analysed in these studies, specifying how they relate to string and gauge theories. In §3 summaries of the research work performed for each of the selected papers is presented. Finally in §4 a general summary of this program of work is given, with some thoughts for further academic progression on these projects. Thoughout the PhD I have been fortunate to work on a large number of papers, in this thesis a selection of those more relevant to this theme have been chosen for inclusion [30]–[37], whilst the other works are acknowledged here [38]–[42].

2 Dramatis Personae

Fundamental theory addresses a wide range of potential extensions to the standard model. Whether it be the inclusion of further conformal or super symmetries, enlarging the con-

sidered gauge groups, changing the matter content, or changing the algebraic variety the theory exists on, all extensions require many parameter choices within themselves. To efficiently delineate similar theories different types of representation are chosen using different mathematical tools. Naturally forming datasets associated to each chosen extension.

In this section the mathematical objects used to represent and probe certain styles of theory are introduced, putting focus on those directly relevant to the research work of §3. The introductions are roughly split between those more relevant to general gauge theories §2.1, and those more relevant to a string theory interpretation §2.2 (although there is often significant overlap). Following the introduction of these key players, the data science and ML techniques used across the research projects are also introduced in §2.3.

2.1 Gauge Theories

Gauge theories are theories of fields that obey a set of symmetries, where the symmetries can be categorised to be global or local. Global symmetries (such as flavour) are fixed throughout the space-time variety of the theory, and in Lagrangian theories may be thought of as transformations of fields by a constant factor which leave the Lagrangian invariant. By contrast local symmetries act differently throughout the variety, and the action may be represented by a function on the variety. In Lagrangian theories local symmetries now transform fields by a factor dependent on their position on the variety, still leaving the Lagrangian unchanged. One may then note global symmetries are a subset of local symmetries for constant functions.

In a field theory, a global symmetry can be gauged to form a local symmetry by making the constant function a general function and introducing a gauge field that transforms exactly to cancel the variational effects of this change. The symmetry in its new form is a type of local symmetry known as a gauge symmetry, and depicts the redundancy in description of the physical theory by using the field representation. Any gauge transformation changes the fields but not the overall theory's behaviour, hence any choice of gauge leads to the same output and one is often chosen to simplify calculation. In the same way, to extract the pure physics from a theory one must consider gauge-invariant objects independent of this choice in the redundancy.

The most popular gauge theories in physics are Yang-Mills theories [3], which are based on gauged symmetries described by Lie groups [43]. Lie groups contain the famous matrix groups, including the orthogonal $O(N)$, unitary $U(N)$, and symplectic $Sp(N)$ groups (with their special counterparts) from the normal division algebras; commonly used as they allow simple representation theory interpretations of the fields. Further notable Lie groups are the exceptional groups: E_6, E_7, E_8, F_4, G_2, related to the octonionic division algebra and necessary for Dynkin classification [44] relevant to quiver mutation later discussed.

Quivers are multi-digraphs (discussed as graph theory objects in §2.3.2), often used to represent the structure of a category in category theory. However they also have use in representation theory, where vector spaces are affiliated to each node and linear maps to each edge. Due to this symbolic link with representations they can be used to diagrammatically represent the symmetry and matter content of certain supersymetric gauge theories. More generally, these theories can be considered as effective limits of special brane systems from string theory [45], [46].

Dependent on the number of supercharges under consideration, the content of the theory dictated by a quiver changes. Prototypical theories most closely related to our universe involve D3 branes, matching our (1,3)-dimensional space-time[1]. Systems of these branes for $\mathcal{N} = 2$, with 8 supercharges, are constructed as gauge theories with gauge groups and matter; where the matter content is split into: vector multiplets (forming the supersymmetric equivalent of gauge groups), and hypermultiplets (forming the core matter), in the simplest cases. The quiver representation of this information denotes the gauge groups by circular nodes, labelled with the respective gauge group (if unlabeled often assumed to be $U(1)$, or $U(N)$ if N is given); to each node an implicit vector multiplet in the adjoint representation of the respective gauge group is also assumed (analogous to the gauging process for non-supersymetric theories to give adjoint gauge vector bosons). Then undirected edges between nodes represent the hypermultiplets in a bifundamental representation of the two connecting nodes' gauge groups. Finally quivers may also show square nodes, which are related to global flavour symmetries, which the hypermultiplet edges may also be connected to. It is worth emphasising this restriction of quivers to theories with hypermultiplets only in bifundamental representations limits the types of theories they can represent.

4-dimensional theories with $\mathcal{N} = 1$, and hence 4 supercharges, have a slightly different notation due to their differing matter content. As there are less supercharges the multiplets are smaller, with gauge group vector multiplets now half the size (but still represented implicitly). The main change for 4 supercharges is the core matter is now portrayed by chiral multiplets instead of hypermultiplets. These chiral multiplets are now depicted with directed arrows between the two gauge groups they are in fundamental and anti-fundamental representations of respectively (direction dependent on convention). In fact $\mathcal{N} = 2$ quivers can be decomposed to $\mathcal{N} = 1$ quivers by adding to each node a directed loop (representing the chiral multiplet broken off of the $\mathcal{N} = 2$ vector multiplet), and replacing any undirected line with two directed lines in opposite directions (representing the two opposite chirality chiral multiplets the hypermultiplet breaks into) [40]. An example of this is given in Figure 2.1.1.

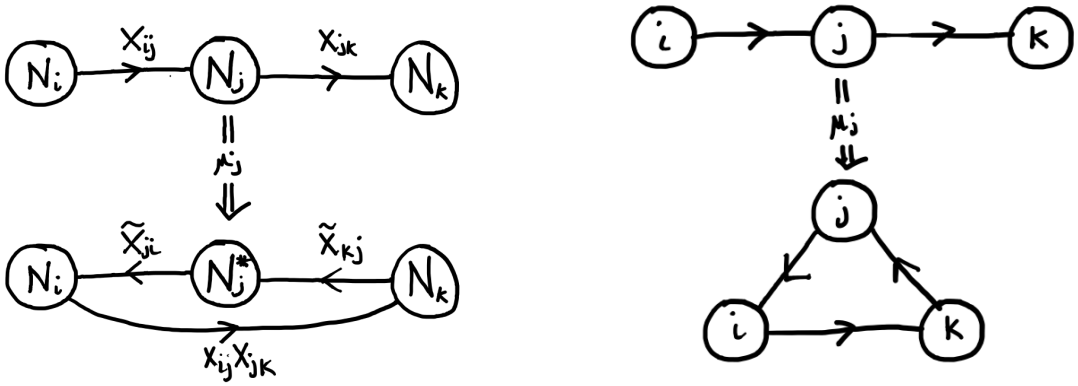
Quivers represent supersymmetric gauge theories directly, independent of their string interpretation, and through ideas of renormalisation [47], [48] in the low-energy infra-red (IR) limit some theories become equivalent. A combinatoric process on $\mathcal{N} = 1$ quivers connects theories with this same IR behaviour in a process known as Seiberg duality [49]. Under this process, also called mutation, a specific node and hence gauge group is first

(a) A $\mathcal{N} = 2$ quiver

(b) The equivalent $\mathcal{N} = 1$ quiver

Figure 2.1.1: A quiver gauge theory with gauge group $U(N_1) \times U(N_2)$ in the (a) $\mathcal{N} = 2$ and (b) $\mathcal{N} = 1$ notation.

[1]Note that both the general dynamical p-branes and the string endpoint branes (D and NS) are labelled by the number of space dimensions they are extended in, therefore with time their worldvolume is $(p + 1)$-dimensional.



(a) Seiberg duality of a quiver

(b) The equivalent cluster mutation process on the quiver.

Figure 2.1.2: Both processes show mutation on a node j of an A_3 quiver, mutation denoted μ_j. In (a) the physics interpretation shows the chiral fields X_{ij} and X_{jk} flipping to the conjugate representations, a meson field $X_{ij}X_{jk}$ being introduced, and the gauge group rank updating $N_j \longmapsto N_j^* = \sum_{\ell \mapsto j} a_{\ell j} N_\ell$ for quiver adjacency matrix a_{ij}. No hypermultiplets are introduced (as 2-cycles) to be integrated out. In (b) the equivalent cluster algebra mutation drops the field and gauge group interpretations but has the same combinatorics, whilst the initial cluster $\{x_i, x_j, x_k\} \longmapsto \{x_i, \frac{x_i+x_k}{x_j}, x_k\}$ under the mutation μ_j.

chosen, its rank is updated under the duality, then chiral multiplets connected to the group have their representation conjugated, additional meson fields are introduced to preserve any anomaly (or lack of), and finally any massive hypermultiplets are integrated out. This process is demonstrated for a different quiver in Figure 2.1.2a, since the quiver in Figure 2.1.1b is too symmetric for interesting mutation. Repeated application of mutation on all the nodes leads to a duality tree (note not a graph-theoretic tree) of connected theories [50], [51], whose combinatorics is probed with ML techniques in §3.2.

Cluster algebras are special algebras whose set of generators are built from a finite subset under an equivalent mutation process [52], [53]. A cluster algebra of rank r has r cluster variables in its initial cluster, which along with an initial quiver form the initial seed. As algebras they may be considered as vector spaces with the extra operation of multiplication between elements, and are generally defined through generators (alike vector space basis elements) and relations. Those of rank r are subalgebras of the field of rational functions in r variables.

Under mutation of the cluster at a chosen variable, the quiver structure dictates how that variable updates to a new variable producing a new cluster. After the cluster mutation the quiver is then also mutated on the equivalent node in a parallel manner to the Seiberg duality process, overall producing a new cluster and quiver and hence a new seed. For a rank r cluster algebra with initial seed $\{\{x_1, x_2, ..., x_r\}, \mathcal{Q}\}$ where the quiver $\mathcal{Q}$ has antisymmetric adjacency matrix[2] b_{ik}, the cluster mutation μ_j on the jth variable/node updates the cluster according to

$$x_i \longmapsto x_i' = \begin{cases} \left(\prod_{b_{\sigma i}>0} x_\sigma^{b_{\sigma i}} + \prod_{b_{\nu i}<0} x_\nu^{-b_{\nu i}}\right)/x_i & \Big| \; i = j \\ x_i & \Big| \; i \neq j \end{cases} \tag{2.1.1}$$

[2]For cluster algebras these antisymmetric quiver adjacency matrices are called exchange matrices.

whilst the quiver updates according to

$$b_{ik} \longmapsto b'_{ik} = \begin{cases} -b_{ik} & \mid j = i \text{ or } k \\ b_{ik} + b_{ij}b_{jk} & \mid b_{ij} > 0 \text{ and } b_{jk} > 0 \\ b_{ik} - b_{ij}b_{jk} & \mid b_{ij} < 0 \text{ and } b_{jk} < 0 \\ b_{ik} & \mid \text{otherwise} \end{cases} \tag{2.1.2}$$

An example of mutation of an initial seed for the rank 3 A_3 cluster algebra is given in Figure 2.1.2b. Repeated mutation at all variables produces a tree of seeds, alike the Seiberg duality trees, whose nodes represent seeds and edges indicate that mutation can transform either seed into the other[3]. These cluster algebra trees are called *exchange graphs*, and these algebras form a richer structure for mutation which can reduce to quiver mutation under removal of the clusters from the seeds. The depth of an exchange graph is the number of mutations computed from an initial seed, a term particularly useful when considering infinite exchange graphs truncated to a finite size.

Cluster algebras can be classified into 3 distinct types, which can be characterised by the quiver exchange graphs (i.e. duality trees) and seed exchange graphs being either finite or infinite in size. These types are:

1) *Finite type*: These are defined by having a finite number of seeds and thus finitely many clusters, exchange matrices (quivers), and also cluster variables. All cluster algebras of this type are formed from quivers which take the form of oriented[4] ADE Dynkin diagrams [53], [54]. Or where generalisation to allow skew-symmetrisable exchange matrices in place of quivers (skew-symmetric adjacency) is performed, this extends the finite type classification to all Dynkin types (i.e. with B_n, C_n, F_4, G_2). Since there are finitely many seeds and exchange matrices both the seed exchange graph and the quiver exchange graph form finite compact polytopes. These polytopes are called generalised associahedra (for this finite case). This algebra type may also be identified by the condition that $|b_{ij}b_{ji}| \leq 3 \ \forall \ i, j$ in all exchange matrices in all cluster algebra seeds.

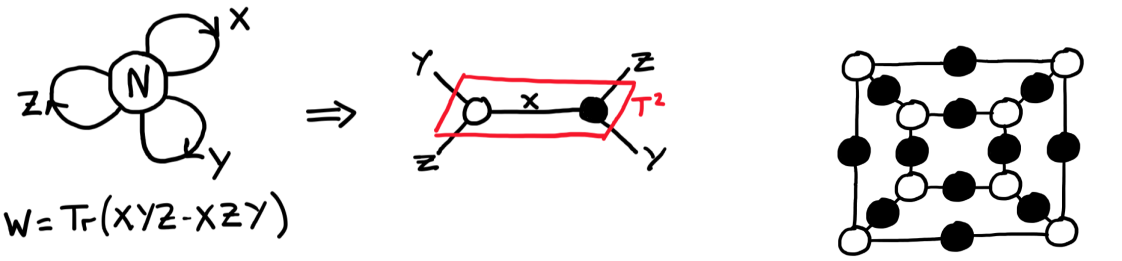
2) *Finite-mutation type*: Since the quiver exchange graph is strictly smaller than the seed exchange graph[5], it can be finite even if the seed exchange graph is infinite. Therefore finite-mutation type cluster algebras are defined by having finitely many exchange matrices (quivers) but may have infinitely many cluster variables, and hence also infinitely many clusters/seeds. Therefore whilst the seed exchange graph *may* be infinite, the quiver exchange graph is finite. These cluster algebras thus naturally include finite type but also others; in particular, the classification consists of cluster algebras formed from rank 2 quivers, from triangulations of marked surfaces, and a set of exceptional cases including: oriented versions of affine ADE types ($\tilde{A}_n, \tilde{D}_n, \tilde{E}_n$), oriented elliptic types ($E_n^{(1,1)}$), and two additional cases (X_6, X_7) [55], [56]. Extending the finite type condition, finite-mutation type may also be identified by $|b_{ij}b_{ji}| \leq 4 \ \forall \ i, j$ in all exchange matrices in all the seeds [57].

3) *Infinite type*: These encompass the remaining cases, with infinitely many clusters and exchange matrices leading to infinitely many cluster variables. Therefore both the seed

[3]Note that as the mutation process is an involution the edges are hence undirected.

[4]An orientation is introduced by adding arrows to the graph edges to make them directed.

[5]This is because multiple seeds may have the same quiver (perhaps infinitely many), therefore different nodes in the seed exchange graph become the same node in the quiver exchange graph.



(a) A $\mathcal{N}=1$ quiver and superpotential W, with their respective dimer.

(b) An example dessin d'enfant

Figure 2.1.3: Dessin examples from the (a) dimer construction, and (b) congruence subgroup construction. In (a) a new toric $\mathcal{N}=1$ quiver is given along with a superpotential W; and the equivalent dimer model is drawn on the torus T^2 (note genus 1). The fields become dimer edges, and the gauge group the enclosed hexagonal face after tiling of the dimer. The superpotential then comes from a positive term going anticlockwise around the black node added to a negative term going clockwise around the white node. In (b) a genus 0 dessin of the type machine-learnt is given, this has 8 white nodes from $\beta^{-1}(0)$ and 12 black nodes from $\beta^{-1}(1)$, all with valencies given by the map degeneracy's of $\beta^{-1}([0,1])$. The 6 faces come from $\beta^{-1}(\infty)$ and include the external face from drawing the dessin on a genus 0 closed surface.

exchange graph and quiver exchange graph are infinite and non-compact.

The algebra type and the respective combinatorics of the exchange graphs are examined through the lens of ML and general network analysis in §3.8.

Dessins d'enfants are bipartite graphs originally devised to represent the degeneracy of Belyĭ maps on Riemann surfaces [58], [59]. These covering maps $\beta : \mathcal{V} \to \mathbb{P}^1_{\mathbb{C}}$ from a Riemann surface $\mathcal{V}$, to the Riemann sphere $\mathbb{P}^1_{\mathbb{C}}$, are many-one. Specifically they have exactly three ramification points, which can be mapped to $\{0, 1, \infty\}$. This degeneracy for $\{\beta^{-1}(0), \beta^{-1}(1)\}$, $\beta^{-1}([0,1])$, and $\beta^{-1}(\infty)$ are mapped to the bipartite graph nodes, edges, and faces respectively, allowing a combinatoric representation of the Riemann surface structure.

Further to this dessins are important objects in Galois theory as they are acted on faithfully by the elusive absolute Galois group over the rationals $Gal(\overline{\mathbb{Q}}/\mathbb{Q})$. This group looks at extensions of the rational base field $\mathbb{Q}$ by adding in non-rational numbers. The absolute Galois group then includes all non-rational numbers that can be formed as roots of rational polynomials in the extension. The dessin Belyĭ maps may include non-rational numbers as their coefficients, and where they do these numbers will be roots of some 'minimal' polynomial[6]. Swapping this non-rational numbers for other roots of its minimal polynomial changes the Belyĭ map and hence the respective dessin – taking the dessin through its Galois orbit. How the size of these Galois orbits can be machine-learnt is the topic of §3.1.

Dessins d'enfants have also seen physical use in relation to supersymmetric gauge theories as novel methods of factorising Seiberg-Witten curves [60], [61], and also as dimer models/brane-tilings [62]–[64]. These dimer models are realised as toric quiver gauge theories describing $\mathcal{N}=1$ superconformal theories from D3-brane stacks at the tip of a toric CY cone [65], [66]. They are built from the quiver and superpotential by taking the dual

[6]The lowest degree polynomial with coefficients in the base field (here $\mathbb{Q}$) that has this number (here some non-rational) as a root.

graph to the quiver and drawing it on a torus, such that combining the fields associated to each edge incident to a chosen node (in a clock/anti-clockwise manner dependent on node colour) gives the terms in the superpotential. This construction is exemplified in Figure 2.1.3a, along with an example dessin of the type machine-learnt in Figure 2.1.3b.

Hilbert series are a tool from algebraic geometry with appealing application in physics. In general, they provide information about the grading of the coordinate ring used to define a complex variety embedded in a complex projective space. The series act as a generating function for the dimensions of these graded pieces, effectively enumerating the number of independent polynomials of each degree that can be defined on the variety.

On the physics side, these series give a way to count the number of single and multi trace BPS operators in gauge theories formed from brane stacks probing CY singularities, alike dimer models [67], [68]. In the closed forms of the Hilbert series, the parameters of the rational polynomials give information about the variety and its embedding, and hence information about the theory's moduli space. Via the Hilbert-Serre theorem [69], the series can be written in 2 further closed forms equivalent to the defining form:

$$H(t;\mathcal{V}) = \sum_{i=0}^{\infty}(\dim_{\mathbb{C}} R_i)t^i \ , \tag{2.1.3}$$

$$H(t;\mathcal{V}) = \frac{P(t)}{\prod_{i=0}^{s}(1-t^{p_i})^{q_i}} \ , \tag{2.1.4}$$

$$H(t;\mathcal{V}) = \frac{\tilde{P}(t)}{(1-t^{\mathcal{J}})^{dim+1}} \ , \tag{2.1.5}$$

where (2.1.3) shows the defining form in terms of dimensions of the graded pieces of the coordinate ring $R = \bigoplus_{i\geq 0} R_i$; (2.1.4) is in terms of a numerator polynomial $P(t)$ over a denominator with parameters which define the ambient weighted projective space $\mathbb{P}_{\mathbb{C}}(p_0^{q_0},\ldots,p_s^{q_s})$; and finally (2.1.5) shows an alternative form where $\mathcal{J}$ is the Gorenstein index, dim the variety's dimension, and if the numerator $\tilde{P}(t)$ is palindromic the variety is also Gorenstein in nature. All Hilbert series for the variety $\mathcal{V}$ are written in terms of a dummy 'fugacity' variable t and they may be Taylor expanded to give the full series used for the equivalent enumerations. If the expansion of the plethystic logarithm [68] of the series terminates at a finite order the variety is a special type of variety known as a complete intersection[7].

An additional point worth noting is that the moduli space variety in the physics is actually the cone over the variety $\mathcal{V}$ of the series. Learning the properties of the full variety from just the first terms in the series is the focus of the work in §3.3.

To exemplify the use of Hilbert series in physics the first few counts for the number of single trace BPS operators at each order for the quiver gauge theory in Figure 2.1.3a are shown below:

[7]Complete intersections are defined to be generated by as many independent polynomials as their codimension in the ambient projective space.

Order	#	Operators
0	1	1
1	3	$\mathrm{Tr}(X),\ \mathrm{Tr}(Y),\ \mathrm{Tr}(Z)$
2	6	$\mathrm{Tr}(X^2),\ \mathrm{Tr}(Y^2),\ \mathrm{Tr}(Z^2),\ \mathrm{Tr}(XY),\ \mathrm{Tr}(XZ),\ \mathrm{Tr}(YZ)$

where higher order generators are found using the F-terms of the superpotential $0 = \partial_X W = \partial_Y W = \partial_Z W$, which here just states all the 3 generators are freely commuting so can be combined in all ways. In the simplest case where $N = 1$ for the gauge group, the field representations are just complex numbers, giving a Hilbert series

$$H(t; X) = \frac{1}{(1-t)^3} \ , \tag{2.1.6}$$

$$H(t_1, t_2, t_3; X) = \frac{1}{(1-t_1)(1-t_2)(1-t_3)} \ , \tag{2.1.7}$$

where the second form shows the 'refined' case for toric varieties, used when one wishes to use multiple fugacities to distinguish the generators as opposed to combining them in the unrefined case discussed before. Since the Hilbert series shows 3 generators with no relations (seen via the trivial numerator) the moduli space here is simply $\mathbb{C}^3$, which may be considered as the cone over the $\mathbb{P}^2_{\mathbb{C}}$ variety. Additionally, the numerator is trivially palindromic so the variety is Gorenstein, and taking the plethystic logarithm produces a terminating expansion (again somewhat trivially as there are no relations), indicating this variety is a complete intersection.

2.2 String Theories

String theories are more general theories of branes of varying dimension, of which strings become the most important in more commonly studied weak string coupling scenarios. These 5 consistent anomaly-free string theories mentioned in §1, along with theories of supergravity, are connected via dualities into the larger M-theory. One of the main dualities relates amplitudes calculated in one string theory at weak string coupling g_S, with amplitudes in another string theory now calculated at strong string coupling $1/g_S$, called S-duality. The other main duality is T-duality and relates sets of winding and momenta string states at a small compactification radius of R in one string theory with the conjugate momenta and winding states in another string theory at large compactification radius $1/R$.

Developing from these more fundamental dualities of M-theory: the effects of S- and T-dualities can be combined into U-duality [18], connections have been made from T-duality to mirror symmetry between CY manifolds [70], and the AdS/CFT correspondence has opened the door to holographic dualities that relate gravitational theories to non-gravitational theories in lower dimensions [71].

Modern day string theory has provided a wealth of new ideas for exploration in algebraic geometry. Not least from the physical importance put on CY manifolds from compactification of the theory [20], as well as the later discovered mirror symmetry of these manifolds [72]–

[74], pairing CY manifolds which lead to the same physical theories when compactified on. Many of the objects surveyed in the ensuing projects have strong links to CYs, of which we introduced more thoroughly now.

Calabi-Yau manifolds are traditionally defined to be compact Kähler manifolds with holonomy group $\subset SU(N)$, vanishing first Chern class, having a trivial canonical bundle, and are by nature Ricci-flat [38], [75]–[77]. However, there are definition generalisations where one looks at: non-compact constructions, perhaps as more general varieties (i.e. manifolds with singular points), or where the canonical bundle is not trivial. Sometimes reducible manifolds are also included, such as complex tori and other locally product spaces. Some of the conditions are sufficient in certain subsets, but importantly the Kähler property ensures the manifold has compatible Riemannian, complex, and symplectic structures by requiring a positive semi-definite metric g_{ab} (to allow definition of a Hodge star, Laplacian, and harmonic forms) which is Hermitian and hence preserves the complex structure J^b_a via $g(J^a_c u^c, J^b_d v^d) = g(u^a, v^b)$ and allows definition of a Kähler form $\omega = ig_{a\bar{b}} dz^a \wedge d\overline{z}^{\bar{b}}$ with $d\omega = 0$ and Kähler potential $K(z, \overline{z})$ from $g_{a\bar{b}} = \partial\overline{\partial} K$ for coordinates z^a. For further details of the algebraic geometry surrounding their construction reference is made to appendix A of [78].

Elliptic curves make up the irreducible complex dimension 1 CYs, and K3 surfaces those of dimension 2, in both cases there is only one topological type (excluding products of tori). Those of complex dimension 3 are the most interesting for string theory compactification of the additional 6 real extra dimensions. For complex dimension 1 the topological type is parameterised by the genus (always 1) or Euler number χ (always 0), for complex dimension 2 the topological type is related to the Poincaré conjecture on real 4-manifolds so is notoriously difficult to determine. Importantly though for complex dimension 3, these CY 3-folds[8] are topologically classified by their non-trivial Hodge numbers, second Chern class, and triple intersection numbers: $\{(h^{1,1}, h^{2,1}), [c_2]_\alpha, d_{\alpha,\beta,\gamma}\}$. There are three core methods of constructing CY manifolds: as complete intersections, as toric hypersurfaces, and as elliptic fibrations, each briefly detailed in the following.

Before jumping into each of these constructions, generally when defining CYs as complex manifolds through the language of algebraic geometry they are considered as the zero locus of polynomials in complex coordinates. However to ensure compactness (non-compact equivalents are examined later) these complex coordinates instead parameterise a complex projective space $\mathbb{P}^n_{\mathbb{C}}$, as homogeneous coordinates from the $n+1$ coordinates z_i of $\mathbb{C}^{n+1} \setminus \{0\}$ under the identification

$$(z_0, z_1, ..., z_n) \sim (\lambda z_0, \lambda z_1, ..., \lambda z_n) \quad \forall \lambda \in \mathbb{C} \setminus \{0\} \ . \tag{2.2.1}$$

The simplest CYs are hence hypersurfaces in some ambient $\mathbb{P}^n_{\mathbb{C}}$, and to ensure vanishing of the first Chern class the hypersurface polynomial must have degree $n+1$ in the projective space homogeneous coordinates [74].

This leads to the *first* set of CYs being the complete intersection CYs, which generalises the above construction to instead consider hypersurfaces in some product of projective spaces $\mathbb{P}^{n_1}_{\mathbb{C}} \times \mathbb{P}^{n_2}_{\mathbb{C}} \times ... \times \mathbb{P}^{n_m}_{\mathbb{C}}$ defined through the zero locus of a set of homogeneous polynomials in all the spaces' coordinates. To be complete intersection these polynomials must intersect

[8] In general an n-dimensional CY is called a CY n-fold.

minimally such that the variety they define as the zero locus has codimension equal to the number of polynomials in the set (i.e. each new polynomial slices out one additional dimension of the ambient space). In the simplest cases $m = 1$ and the ambient space is only one $\mathbb{P}^n_{\mathbb{C}}$ giving the cyclic CYs. More generally though the complete intersection CYs can then be parameterised as

$$\begin{bmatrix} q_1^1 & q_1^2 & \cdots & q_1^k \\ q_2^1 & q_2^2 & \cdots & q_2^k \\ \vdots & \vdots & \ddots & \vdots \\ q_m^1 & q_m^2 & \cdots & q_m^k \end{bmatrix} \tag{2.2.2}$$

under conditions

$$\sum_{j=1}^{k} q_i^j = n_i + 1 \qquad \forall i \in \{1, 2, ..., m\} \ , \tag{2.2.3}$$

such that each row represents a projective space $\mathbb{P}^{n_i}_{\mathbb{C}}$ in the ambient product space (where (2.2.3) ensures the Calabi-Yau property), and each column is a polynomial which is homogeneous in all the projective space coordinates for each projective space but with integer degree q_i^j in the $n_i + 1$ coordinates of the $\mathbb{P}^{n_i}_{\mathbb{C}}$ space. As a complete intersection, the resulting CY is of dimension $\sum_{i=1}^{m}(n_i) - k$, as the k polynomials each slice out a dimension of the full ambient space. For the case of the physically interesting CY 3-folds, there are 7890 configuration matrices [79], [80] from which all the topological data can be extracted except explicit Hodge numbers later calculated in [81].

The *second* set of CY constructions generalises the hypersurfaces in projective space in a different way to the complete intersections. An initial alternative generalisation would be to alter the identification relation in (2.2.1) by introducing weights, $w_i \in \mathbb{Z}^+$, such that the identification becomes

$$(z_0, z_1, ..., z_n) \sim (\lambda^{w_0} z_0, \lambda^{w_1} z_1, ..., \lambda^{w_n} z_n) \quad \forall \lambda \in \mathbb{C} \setminus \{0\} \ , \tag{2.2.4}$$

creating weighted projective spaces, where now in an analogous way to the complete intersection CY condition in (2.2.3) a CY hypersurface is a homogeneous polynomial with degree $\sum_i w_i$ in this weighted projective space. For the CY 3-folds, there are 7555 weighted projective space hypersurfaces [74], which is the database of focus in §3.6. Beyond this first step of generalisation one may then consider taking products of these weighted projective spaces [82] as for complete intersections, however a more popular generalisation has been to extend these weighted projective spaces to toric varieties.

Toric varieties are most generally defined with lattice cones, however a nicer (and perhaps slightly more restricted) definition looks similar to our definition of projective spaces. Whereas for projective spaces the original $\mathbb{C}^{n+1}$ space has only the origin removed and is quotiented by a single relation, for toric varieties a larger subset of the original $\mathbb{C}^{n+1}$ space may be removed (at least the origin) and it is then quotiented by a set of relations (also called an algebraic torus). This is denoted

$$(z_0, z_1, ..., z_n) \sim (\lambda^{w_0^a} z_0, \lambda^{w_1^a} z_1, ..., \lambda^{w_n^a} z_n) \quad \forall \lambda \in \mathbb{C} \setminus \{0\} \, , \forall a \ , \tag{2.2.5}$$

for each of the a sets of weights w_i^a. The weights w_i^a form a matrix[9], and interestingly this weight matrix may be thought of as the kernel of the vertex matrix of a reflexive polytope [83], [84].

A reflexive polytope is a lattice polytope whose dual polytope is also reflexive. A lattice polytope, Δ, is a polytope whose vertices[10] $\mathbf{p}_i$ all take position on the integer lattice $\mathbf{p}_i \in \mathbb{Z}^n$, and most often we are interested in convex polytopes where the polytope is the convex hull of the vertex points. A polytope's dual, $\overline{\Delta}$, is the set of all points $\overline{\mathbf{p}}_i \in \mathbb{R}^n$ which satisfy: $\mathbf{p} \cdot \overline{\mathbf{p}} \geq -1$. Reflexive polytopes are thus extremely important in the classification and construction of toric varieties, and the reflexivity condition at the polytope level becomes equivalent to mirror symmetry of the CY hypersurface at the toric variety level [83]. In dimension 1 there is a single reflexive polytope (the point), in dimension 2 there are 16 reflexive polygons, in dimension 3 there are 4319 reflexive polyhedra [85], and in dimension 4 there are 473800776 [86]. The ML of polytopes, their properties, and some properties of their respective toric varieties is the topic of the work summarised in §3.5.

The toric varieties built from reflexive polytopes in this way are Fano (i.e. with positive curvature), have complex dimension equal to the polytope's real dimension, and may have singularities which are at worst Gorenstein in nature. The CY hypersurfaces within these Gorenstein toric Fano ambient space varieties are the anticanonical divisors (a specific type of codimension 1 hypersurface) [83]. Hence reflexive polygons give elliptic curves (CY 1-folds), reflexive polyhedra give K3 surfaces (CY 2-folds), and 4-dimensional reflexive polytopes give CY 3-folds, with the connection continuing for higher dimensional polytopes. Importantly for the topological data that classifies the CY 3-folds, the Hodge data can be extracted directly from the polytope [83], [87] but the second Chern class and triple intersection number cannot be. In their computation the ambient space is required to be smooth which is only the case where the polytope is also 'regular', which happens rarely. Therefore to resolve singularities (related to the lattice points on the polytope facets which are not vertices) the polytope must first be triangulated, which introduces a huge redundancy in the number of resolved toric varieties and hence CY hypersurfaces that can be constructed from each polytope. Polytope triangulation is a popular topic of research for this reason [88], [89]. In a similar vein to the complete intersection CY construction with products of projective spaces, there is further work to use products of toric varieties [90], or to generalise the ambient space used further beyond toric varieties. However these constructions produce relatively few CYs in comparison at this present time.

Beyond this polytope construction for compact Fano toric varieties, the same construction can be used for non-compact varieties also, and in the non-compact case these can be CY directly. The condition for a toric variety to be CY is that the polytope vertices must exist in a codimension 1 hyperplane of the lattice. This can artificially be done by embedding an n-dimensional polytope in an $(n+1)$-dimensional lattice with the last coordinate 1 for all the vertices. Then the fan over this polytope defines a toric variety which is CY, but also explicitly non-compact, as the vertices need to form cones which cover the full lattice for the variety to be compact, here prevented by the cohyperplanar condition [91], [92].

[9]Toric varieties reduce to weighted projective spaces when this matrix has only one row, and reduce again to unweighted projective spaces when it is a row where all entries are 1.

[10]In this notation $\mathbf{p}_i$ represents a single vertex, with the i index running over the dimensions.

The polytopes used in this non-compact construction are called toric diagrams, and for CY 3-folds they are 2-dimensional lattice polygons [93], [94].

The *third* and final method of constructing CYs is through elliptic fibration. Here the Weierstraß form of the $\dim_{\mathbb{C}} = 1$ CY elliptic curve: $z_2 z_1^2 = 4z_0^3 - g_2 z_0 z_2^2 - g_3 z_2^3$ in $\mathbb{P}^2_{\mathbb{C}}$, is lifted such that it becomes a fibre over an appropriate base. This is done by making the g_2, g_3 functions of this base space. For the popular CY 3-folds, the base spaces can only be of a specific types: del Pezzo, Hirzebruch and their blowups, or Enriques surface [95]. It is believed that in many cases this formulation can be mapped by changing coordinates such that the 3-fold is considered again as some hypersurface within a Fano variety, and there are programs of work to perform this for CY 3-folds [96], [97]. Although this construction has some relevance to the dessins d'enfants considered in §3.1 via elliptically fibred K3s, the work in §3.6 and other objects examined in this research is more relevant to the other constructions.

Beyond these 3 constructions for compact CYs, more general non-compact CYs can also be constructed as the representation varieties of quiver gauge theories [98], where this construction can produce any non-compact CY variety. The simplest non-compact CY is $\mathbb{C}^n$ as it is trivially flat (and hence Ricci flat) and has trivial holonomy (hence $\subset SU(n)$). Stepping beyond this case one can create orbifolds which are CY by taking certain quotients of $\mathbb{C}^n$, such that the quotient is by a discrete finite subgroup of $SU(n)$ and noting to resolve the variety singularities appropriately [99]–[101].

Brane webs are a diagrammatic tool used to represent certain classes of supersymmetric quantum field theories coming from string theory [102], [103]. Whilst compactification of M-theories on CY spaces can lead to physically interesting superconformal field theories in 5 dimensions, brane webs provide an alternative route for their consideration as low energy descriptions of systems of Type IIB 5-branes. The 5-branes can be set-up to each end on a 7-brane, and the whole system of intersection is drawn in a 2-dimensional plane [104]. This 2-dimensional plane is parameterised by two integers (p, q) which represent the branes magnetic charges under the RR and NSNS 2-forms respectively.

Supersymmetry in the theory requires the 5-branes to have 4 common dimensions of extension, such that the (p, q) plane contains the only non-trivial information about the branes' set-up. The $(1, 0)$-direction indicates D5-branes, and the $(0, 1)$ direction NS5-branes, with general 5-branes taking any integer (p, q) values, such that the $gcd(p_i, q_i) = n_i$ gives the multiplicity of the brane ($gcd \Rightarrow$ greatest common divisor). The full brane web set-up is then a collection of 5-branes ending on 7-branes which intersect at a point in the (p, q) plane, such that this intersection junction conserves both the p and q charges. An example of a brane web, as well as the two types of equivalence move ($SL(2, \mathbb{Z})$ and Hanany-Witten) is shown in Figure 2.2.1; whilst Table 2.2.1 demonstrates the incident dimensions each of the possible branes are extended in.

However there is redundancy in this representation of the theories. In particular, the $SL(2, \mathbb{Z})$ duality of Type IIB string theory transfers to the brane web in the 2-dimensional plane, as well as another redundancy known as a Hanany-Witten move, where a 5-brane leg can be shortened along its direction of extension in the 2-dimensional plane until it crosses the intersection point of the web and moves onto the other side. After the transition to the other side of the intersection, the 7-brane at the end of the 5-brane is left with an axiodilaton

Brane	Dimension									
	0	1	2	3	4	5	6	7	8	9
D5	x	x	x	x	x	x				
NS5	x	x	x	x	x		x			
(p,q)	x	x	x	x	x	x				
7	x	x	x	x	x			x	x	x

Table 2.2.1: A table demonstrating the dimensions each brane type is extended in for the 5-dimensional brane web construction of Type IIB string theory. An 'x' indicates a brane is incident and hence extended in that respective dimension, where dimensions 5 and 6 are the D5 and NS5 directions in the 2-dimensional plane the web is drawn in; the 7-branes are point-like in this plane.

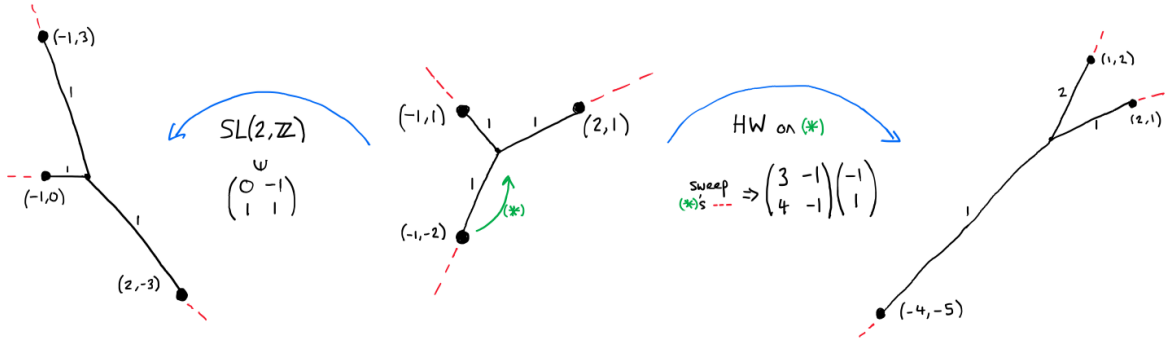


Figure 2.2.1: An example 3-leg brane web. The centre image shows 3 intersecting 5-branes each ending on a respective 7-brane in the (p,q) plane, with the junction of intersection at the origin (0,0) in each case. Each brane has multiplicity 1 labelled on the leg, and axiodilaton branch cuts oriented away from the intersection. The leftmost web is a brane web equivalent to the original under an example $SL(2,\mathbb{Z})$ transformation, rotating and shearing all the legs together. The rightmost web is another equivalent brane web reached via a Hanany-Witten move on the '$(*)$' labelled leg which has its (p,q) values flipped to $(-p,-q)$ and then respective branch cut swept clockwise through the (-1,1) leg causing a monodromy action of: $M_{(p,q)} = \begin{pmatrix} 1+pq & -p^2 \\ q^2 & 1-pq \end{pmatrix} = \begin{pmatrix} 3 & -1 \\ 4 & -1 \end{pmatrix}$, on this leg; finally the moved leg's multiplicity updates to conserve charge at the intersection.

branch cut which passes through the web, to correct this it must be swept around half the web, and appropriate monodromy applied to any other legs that it sweeps through [105].

The classification of 5d superconformal field theories that can be constructed from 3-leg brane webs was the topic of the work in §3.7, and the relative success of the ML techniques employed identified a gap in the current understanding of their invariants.

Amoebae are central objects in the study of tropical geometry, a field of geometry based on the tropical semiring $(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$, i.e. a field with no additive inverse [106]. The semiring's operations of tropical addition, $x \oplus y := max(x,y)$, and tropical multiplication, $x \odot y := x + y$, become manifest under the amoeba projection on complex polynomials, making them natural objects of study in this field. In this projection one first considers a multi-variable polynomial determining a Riemann surface, which we define from a toric diagram.

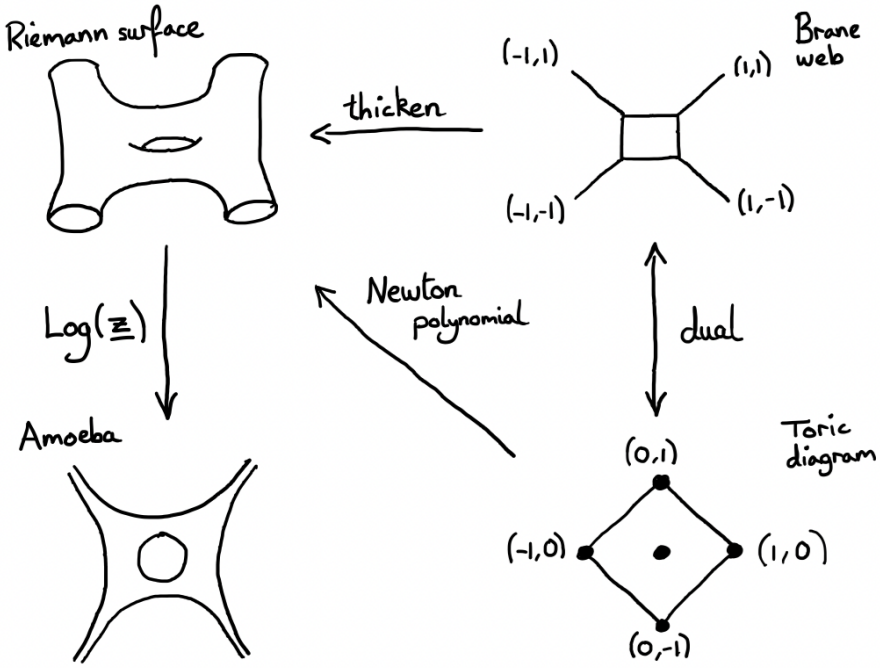


Figure 2.2.2: The relations between an example genus 1 amoeba, its respective Riemann surface (the 0th Hirzebruch surface $F_0 \cong \mathbb{P}^1 \times \mathbb{P}^1$), the brane web the surface is a thickening of, and the toric diagram used to generate them. The toric diagram has one internal vertex (leading to a maximum genus of 1 in the Riemann surface), and 4 external vertices, creating a Newton polynomial: $P(z_1, z_2) = c_0 + c_1 z_1 + c_2 z_2 + \frac{c_3}{z_1} + \frac{c_4}{z_2}$ for coefficients $\{c_i\}$, which the amoeba is the *Log* projection of. The brane web is also the dual graph to the toric diagram and vice versa.

Starting with a convex lattice polytope, Δ, as a toric diagram in n dimensions (dimensions indexed by i) with points indexed by j such that each point $\mathbf{p}_i \in \Delta$, the polytope's Newton polynomial is defined as $P(\mathbf{z}) := \sum_j c_j \mathbf{z}^{\mathbf{p}_i}$ such that the c_j are arbitrary $\mathbb{C}$ coefficients, and the $\mathbf{z}$ represents a $\mathbb{C}^n$ vector space for the n-dimensional polytope Δ. Each lattice point hence produces a monomial term which is a product of each z_i raised to the power p_i for the n dimensions indexed by i (taking p_i as the i-th entry of the lattice point $\mathbf{p}_i$). Each of these monomial terms is then summed to produce the surface's equation $P = 0$. Representing points on the surface as

$$\mathbf{z} = (z_1, z_2, \ldots, z_n) = (e^{(s_1 + i\theta_1)}, e^{(s_2 + i\theta_2)}, \ldots, e^{(s_n + i\theta_n)}) \,, \tag{2.2.6}$$

a Log projection can be taken to the $\mathbf{s}$ coordinates to provide a $\mathbb{R}^n$ representation of the surface known as the amoebae, $\mathcal{A}$:

$$Log: \; (z_1, z_2, \ldots, z_n) \longmapsto (s_1, s_2, \ldots, s_n) = (log|z_1|, log|z_2|, \ldots, log|z_n|) \,. \tag{2.2.7}$$

Equivalently one may consider the projection to θ, known as the algae map [107].

Lopsided-amoebae provide a good approximation to amoebae, and are easier to calculate genus from due to the condition of lopsidedness. The Lopsided amoeba, $\mathcal{LA}$, is the set of all points $\mathbf{a} \in \mathbb{R}^n$ such that $P(\mathbf{a})$ is *not* lopsided. To not be lopsided the requirement on

the Newton polynomial coefficients $\{c_j\}$ is that $\nexists\ j\ s.t.\ |c_j| > \sum_{k\neq j} |c_k|$. Physically this corresponds to there not existing a set of complex phases θ such that the point in the real space can be lifted via Log^{-1} into the complex space to a set of points that satisfy the equation $P = 0$. This follows naturally from the triangle inequality. To further calculate membership for the true amoebae requires considering all coefficient lists for lopsidedness in the amoeba's generating ideal, computed using cyclic resultants [108].

The connection to physical theories is through the toric diagram's dual being the (p, q)-brane web, with the Riemann surface being the thickening of this brane web. This Riemann surface may then be considered as the mirror geometry to the CY 3-fold probed by D3 branes in string theories [107], [109]. Beyond this the Mahler measure may be used for physical interpretation of tropical limits of toric quiver gauge theories depicted with amoebae [110], [111]. The number of interior points of the toric diagram gives the maximum genus of the Riemann surface, and the conditions that lead to these holes and their projection onto the amoeba is the topic of §3.4. An example of an amoeba examined thoroughly in §3.4, along with its toric diagram, brane web, Newton polynomial and Riemann surface is given in Figure 2.2.2.

2.3 Data Science

Data science is an enormously broad field that concerns itself with the generation, processing, analysis, and inference of data. In modern times, this field relies almost exclusively on computational resources for each of these data management techniques.

As discussed in §1, computational work in the research projects included largely relied on the use of `python`, and regularly used HPC resources with parallelisation for big data tasks. As also referenced in the introduction, string theory and its related gauge theories are now at the point large datasets can readily be generated, as has already been done in the case of the Calabi-Yau landscape. These large datasets call for analysis methods that can handle large amounts of data, in particular methods of a statistical nature are needed for research inquiries seeking feasible extraction of big data structure.

Machine learning is the subfield of data science that centres itself on application of techniques from computational statistics. In many forms the models used seek to learn underlying, usually inconspicuous, structure in the data leading to inference on the properties of the objects that make up the dataset. An insightful comparison can be made here to the mathematical process of conjecture formulation. In traditional mathematics one examines a selection of simpler examples, extracts a pattern in their behaviour, and uses this to construct a conjecture which applies beyond the examples considered. This process is a nice parallel to the typical workflow of machine learning. A sample dataset is taken, a technique is applied to learn some structure in the data, this structure is assessed on data outside of the original sample. Therefore, in times where academia has a wealth of resources for large scale data generation it has become especially appropriate to employ techniques from machine learning to extend the limits of pattern recognition and conjecture formulation.

In the following subsections of §2.3.1 machine learning as a vast field in itself is broadly introduced, putting focus on the methods most relevant to the succeeding research work. Then in §2.3.2 some relevant techniques from network science are introduced, as many of the mathematical objects worked with take the form of graphs and networks.

2.3.1 Machine Learning

ML problems generally categorise themselves into: *supervised*, *unsupervised*, and *reinforcement*. Each category has a number of different architectures that can be employed to address the problem at hand; in most cases the input data is formatted as a set of tensors and the architecture seeks to learn some data structure.

In the first case of *supervised* learning, the style most commonly used and often the focus of the following projects, the ML architecture is given data in a pair-like format, one part being the input tensor, and the other the output tensor. The architecture then runs through the training data attempting to reproduce the output tensor from the input tensor for each pair in the full training dataset. Within supervised learning there're two main subcategories:
1) Classification – the outputs take value in a finite set, such that the output tensor is a vector with each entry corresponding to a potential class that the input tensor is in, usually normalised to a probability. The highest valued entry of the vector is taken as the classification.
2) Regression – the outputs take value in a continuous range, such that the output tensor can take any form, populated with floating point numbers, which the architecture is aiming to make as close to the true output tensor (according to some chosen loss function).

The second case of *unsupervised* learning inputs data as a single input tensor, with no output tensor. The architecture then aims to find patterns in the data, simplifying its representation (feature extraction) or sorting it into sets (clustering).

The final case of *reinforcement* learning uses data which is selected from an environment to input into the architecture according to some Markovian process. The goal is to train an agent to know how to sensibly navigate this environment, selecting from the list of available actions in such a way as to maximise cumulative reward. When the agent is at a certain state in the environment, the state data the architecture processes is used to calculated a corresponding reward/punishment according to some function, which educates the architecture about its next selection of an action from those which are available. This style aims to find near-optimum sequences of actions that meet some desired conditions [112].

Each of these categories of ML has seen a variety of applications to high-energy theory. In particular the applications for string theory started with [113]–[117] and since then there has been an explosion of areas of applicability for this wealth of techniques. Beyond a strong program of work using ML methods to numerically find CY metrics [118]–[132] and other CY properties central to examining the string landscape [133]–[162], there has been substantial use of ML in other string theory and mathematics related areas including but not limited to: elliptic curves [163], [164], space-time classification [165], conformal field theories [166]–[169], quantum chromodynamics [170], [171], integrability [172], group theory [173]–[175], number theory [176]–[179], knot theory [180]–[184], graph theory [185], and general mathematics [186], [187]. There is even a program of work to describe neural network evolution with concepts from quantum field theory [188]–[195]. The machine-learning work completed for this thesis has focused on techniques from supervised and unsupervised learning, the techniques, or *architectures*, used are detailed in the following subsections.

2.3.1.1 Supervised Architectures

The most common architecture used in supervised learning is a *neural network* (NN) [196]. These NNs are code structures for non-linear function fitting. Their design was generally inspired by that of a biological brain, being built out of components known as neurons, and they have seen significant success in recent years where computation speed can now compensate for the computational inefficiency of using these networks compared to traditional algorithms. Of particular importance are the theorems of universal approximation, which state that a NN of sufficiently large width [197], or of sufficiently large depth [198]–[200] can approximate any continuous function, with non-polynomial activation [201]. These theorems support the use of NNs in the wide selection of problems they are applied to. A majority of networks used in the subsequent projects were dense and deep, in that they had all neurons fully connected between layers, and there were multiple hidden layers in the network. This NN structure is represented diagrammatically in Figure 2.3.1.

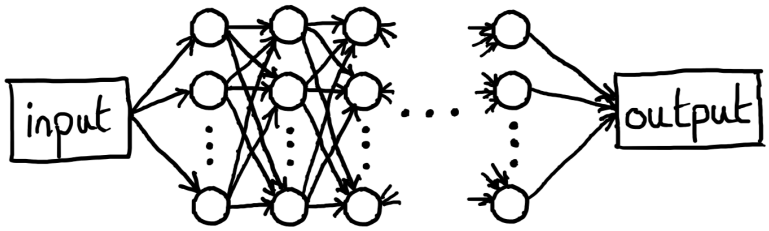


Figure 2.3.1: A diagrammatic representation of a dense feed-forward NN. An input tensor is inputted into all nodes in the first layer, the single number output of each neuron is then sent to all nodes (dense) in the next layer only (feed-forward). Each layer may have a different number of neurons, and the whole NN any number of layers. The final layer's output gives the output tensor.

These NNs are formally 'Multi-Layer Perceptrons', in that they are a combination of perceptron neurons into sets of layers. Each perceptron neuron acts linearly, and then non-linearly on its input tensor (usually just a vector for dense NNs), to give a single number output. The formal representation for this is that for an input vector into any neuron, $\mathbf{x}$, made-up of the number outputs of the previous layer collected together (or just the input vector), the neuron applies the function:

$$\mathbf{x} \longmapsto act(\mathbf{w} \cdot \mathbf{x} + b)\,, \tag{2.3.1}$$

acting with a dot-product on the weight vector, $\mathbf{w}$, and then adding the bias number, b. After this linear action, the neuron acts with a non-linear function called the activation function, here denoted $act(...)$.

Although NNs are the most commonly used architecture for supervised learning, they are not the simplest. Perhaps the most simple of these architectures, designed for the simplest supervised learning problem of binary classification is the *logistic regressor* (LR) [202]. The logistic regressor function takes the form

$$p(x_i) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i x_i)}}\,, \tag{2.3.2}$$

for a vector of n inputs x_i and $i+1$ trainable parameters $\{\beta_0, \beta_i\}$ to give the probability of class 1 (i.e. a positive case) over class 0 (i.e. negative non-case). The functional form is shown for one input in Figure 2.3.2a. Since there are so few trainable parameters the complexity of the classification it can perform is low, however conversely this means often very little data is needed to train it.

The next step in architecture complexity for supervised binary classification is the *support vector machine* (SVM) [203]. Support vector machines consider the input n-dimensional data as points in $\mathbb{R}^n$, and over training they optimise the position of an $(n-1)$-dimensional hyperplane that best separates the datapoints into the two binary classes (optimised by having the largest distance to the nearest point).

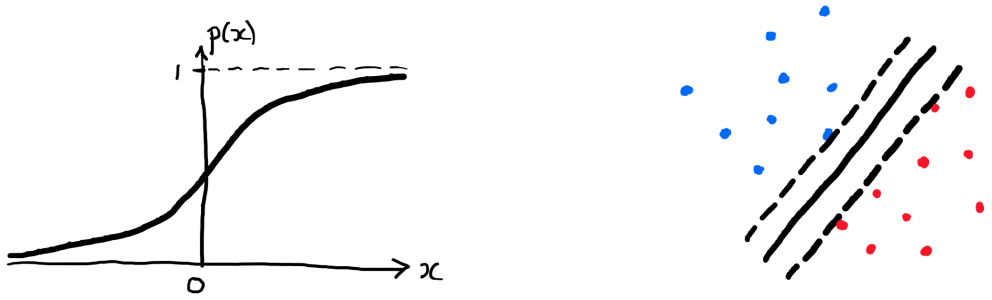
They are designed to draw linear hyperplanes in the space, but can be generalised to produce non-linear hyperplanes by using the kernel trick. The kernel trick effectively introduces non-linearity to the data's feature space by trading the non-linear degrees of freedom for linear degrees of freedom in a higher-dimensional embedding of the original feature space. In ensuing computations for optimisation of the linear hyperplane in this higher-dimensional space only the kernel is used, which is the inner product between these higher-dimensional features. Since this kernel is naturally symmetric and positive semi-definite, the kernel matrix in the higher-dimensional space can be written as just a function of the original features via Mercer's theorem – no computation of each of the higher-dimensional embeddings of the original features or explicitly their inner product is needed, the kernel can be computed directly.

The linear hyperplane (effectively in the higher-dimensional functional space if using the kernel trick) can be defined with the equation: $0 = \sum_i (w_i x_i) - b$ for datapoint vectors x_i and trainable hyperplane parameters of weights w_i and a bias b. The optimum hyperplane is then found by minimising

$$\frac{1}{N}\sum_x \left(max\big(1 - y_i\big(\sum_i (w_i x_i) - b\big)\big)\right) + \lambda\sqrt{\sum_i w_i^2}\,. \tag{2.3.3}$$

The idea is that two hyperplanes with equations $\sum_i (w_i x_i) - b = 1$ or -1 respectively can be drawn to separate the data, and the algorithm desires to make these as far apart as possible (and hence as close as possible to the respective classes). The final hyperplane is then halfway between these and has distance $1/||\mathbf{w}||$ to either bounding hyperplane, which explains the $||\mathbf{w}|| = \sqrt{\sum_i w_i^2}$ term to be minimised. If the data's classes are labelled $y_i \in \{1, -1\}$ then the bounding hyperplanes can be written $1 - y_i(\sum_i (w_i x_i) - b) \geq 0$, and to ignore datapoints which are already in the correct class (the correct side of the hyperplane) the *max* function is introduced. Finally the parameter λ indicates the trade-off between how well the hyperplane separates the classes and how wide the separation to those correctly classified is. An example of a SVM separating two datasets in 2-dimensions is given in Figure 2.3.2b.

For the projects summarised in this thesis, some projects used logistic regressors and support vector machine as starting grounds for classification, however a majority of the results are centred on application of NNs for the learning. The NNs used were sequential in form, meaning they satisfied the simple conditions of single input and output tensors, and sequential layer to layer information flow (feed-forward NNs). All NNs used were *deep*,



(a) Logistic regression function

(b) Support vector machine classification

Figure 2.3.2: The two simple binary classification architectures considered. In (a) the logistic regression function for one input x shows the output probability $p(x)$. In (b) a support vector machine classifies the two classes (blue and red respectively) into their classes using a 1-dimensional hyperplane (the solid black line) in the 2-dimensional space.

this means they contained hidden layers – any layers which are not either the input or output layers. The number of layers indicates the NN depth, and may be correlated to the problems complexity, as it represents the number of times non-linearity acts on the data flowing through the network. The NNs tended to be *dense*, where the outputs of all neurons in one layer become the input for all neurons in the next layer, also called fully-connected. Where the alternative to this is *sparse*, where some neurons are not connected between layers – this idea is utilised in the introduction of dropout layers to avoid overfitting.

NN layer types dictate the style of NN one is using, and should be chosen to match the style of the problem. Keeping focus on the most frequently used architecture (even outside of supervised learning) of the NN, we brefly discuss a few of the common layer structures used to build and delineate them. The most standard NN layer is a dense layer, here all nodes in the layer are connected to all nodes in the previous layer. The structure is simple, where each node of the layer has a weight vector applied, and bias number added, to the vector input from the previous layer. Activation is then applied at each node to give the input vector for the next layer's nodes. Dense layers strictly take the inputs as vectors, however since they are fully connected any multi-dimensional organisation of the input is irrelevant.

Dropout layers are commonly added between regular layers. They take one probabilistic argument (a float in [0,1]), which determines the proportion of vector entries to randomly ignore from the previous layer's output vector. This process reduces the likelihood of overfitting to data, as some connections are ignored and then retrained differently as the training data passes through the network. It is possible to seed the random selection to repeat training processes if desired.

Convolutional layers are used in convolutional neural networks (CNNs). They maintain any multi-dimensional tensor structure of an input, and explicitly limit the connectivity to put a focus on local connections between the input data's coordinates. The input into each neuron is based on a kernel tensor[11], which selects only values of the layer's input tensor which surround the neuron aligned with the current neuron (in the specified kernel shape)

[11]Note the use of 'kernel' here is independent of the use for the kernel trick for non-linear SVMs.

to use in calculation at that neuron, enforcing the local structure used for learning. The same weight, bias, and activation process is applied to the used inputs at each neuron. The stride of the kernel can be specified which dictates how the kernel is moved across the input tensor, it is how many sites across/down the kernel is moved for calculation at the next neuron; ideally this should match the change in the layer sizes. Pooling layers may also be applied between the convolution layers. These are either 'max' or 'mean' in type, such that another kernel this time is taken over the tensor and instead the maximum, or mean, value is taken of all entries in the kernel. This introduces more non-linearity and focuses the NN on the most important local features. In both uses of the kernel, a padding is required. This adds values on the boundary of the tensor such that the image size can be maintained as the kernel runs over it. For max pool the padding is with $-\infty$ values, and for mean pool it is with 0s. If padding is 'valid', then no boundary is added and the layers then decrease in size according to the kernel size and stride. If the padding is 'same' then boundary values are added such that the output tensor is of the same size as the input.

Finally in designing a CNN, a Flatten layer is required after the convolutional layers. This removes the multi-dimensional tensor structure of the input, and outputs a vector which can then be processed by the final dense layers used to output the classification probabilities (or another output format). CNNs were trialled in some the following projects' investigations, but final results showed simpler dense NNs as more suitable, as locality of the data structure was less relevant and the CNN local structure can be learnt from the more general dense NNs if more optimal.

The layer-to-layer connectivity can have a strong impact on the learning performance, and can even change the problem structure. An important example is that of Siamese NNs (SNN), which train multiple identical NN architectures simultaneously such that they learn to map similar data inputs to similar embeddings in the output vector space, a particularly useful NN structure for classification [204].

Activation functions importantly introduce the non-linearity into the NN. After each standard layer the activation function acts on each node's output to give a non-linearly modified output. Importantly the activation function is chosen to be easily differentiable so that it is computationally inexpensive when used in the NN optimiser's gradient descent method of some form.

The most common functions are: ReLU, sigmoid, tanh, softmax. ReLU is the simplest, defined

$$ReLU(x) = \begin{cases} x & x > 0 \\ \alpha \cdot x & otherwise \end{cases}, \tag{2.3.4}$$

for $\alpha = 0$ in the simplest case, however setting $\alpha \in [0, 1]$ (usually ~ 0.05) creates the 'leaky-ReLU' function. This function simply adds priority to more significant parts of the layer's input tensor, whilst being computationally quick to calculate with. There are generalisations of the ReLU function, such as ELU which is smooth at the origin, and may be preferable for gradient calculations. However the simplicity of ReLU means it is often used in preference to the other options. Softmax as an activation, used in classification problems, importantly normalises its output into a probability distribution. It is hence used in the final output layer, where there should be as many neurons as data classes.

Data formatting is important for controlling the training process. During supervised learning, the machine is optimising the matching of inputs and outputs, outputs being floats in regression problems or simple class numbers in classification problems. The data hence has two equal size lists. One is the data form the machine is learning (inputs), and the other the values used in verification of predictions for the architecture parameter updating process (outputs). In contrast in unsupervised learning there is only one list, and the machine is set to determine its own patterns without another list for verifying its success.

Traditionally during supervised learning the full dataset is split into three smaller datasets. The important two are training and testing. Training data is given to the machine during its learning, it uses the data in this input list to optimise its loss function via updating the parameters of the architecture function (i.e. for NNs its neurons' weights $\mathbf{w}$ and biases b). Test data is, importantly, unseen to the architecture, it is used after training to calculate the metric measure of the architecture's success. The final dataset split is the validation dataset, this is a subset of the training dataset, still used in training, but is also used for giving intermediary measures of the performance. Often validation data is not used, as training loss can provide a sufficient measure of performance during the training process. Usually the dataset is split such that 80% of the data is used for training (with 0% of that used for validation), and the final 20% used for testing. Varying these ratios to reduce the training proportion can show how easily the architecture is learning the underlying trend (since less data is being used).

A common additional procedure for evaluating confidence in an architecture's ability to learn, is using k-fold cross-validation. Here the full dataset is split into k equal subsets (or as close to equal as discretely possible). Then in turn k independent architectures (of exactly the same specifications) are trained and tested with two datasets formed from this split. One dataset is $\left(\frac{k-1}{k}\right)$-th of data, and is formed from concatenating (k-1) of the subsets, whilst the other makes up $\left(\frac{1}{k}\right)$-th of data being the final subset. The first of these new datasets is used for training, and the latter for testing. Hence k independent architectures have been trained and tested in k different ways on the full dataset (each with a different test set from the k-partition). Therefore the final metric outputs can be averaged and corresponding confidence calculated. A common choice of k is 5 [205].

The training dataset is usually split into *batches*, often with default of 32 inputs at a time. This determines how many input tensors to run through the architecture before the next iteration of the optimiser is run to update the function's parameters (i.e. NN weights/biases). A large batch size leads to more accurate updating of the values, however as the architecture needs to account for more data in the parameter updating the optimisation process then takes longer, and requires more memory. More frequent passes of the optimiser with smaller batches may then be preferable to speed up computation, but will be more stochastic and less accurate. However there is a trade-off, as the stochastic nature of smaller batches also helps the optimiser avoid local minima in the loss function.

Beyond this the *iterations* variable for the training will dictate how many times to run a batch through the architecture and then apply the optimiser before moving onto consider the next batch. Then in a similar vain, the number of *epochs* is the number of times the entire learning process is repeated on the architecture, running all the data batches through and then starting again. More iterations and epochs will lead to a better trained architecture,

but may lead to overfitting. Usually no iterations are used (value of 1), whilst the number of epochs is larger than 1 (~ 20).

Overfitting is a typical problem in ML, it occurs when the architecture learns how to predict the output from the input by learning all the inputs as opposed to the true underlying data structure. Where overfitting occurs the accuracy and losses during training show very promising values, whilst those on the test data are poor. Overfitting is typically avoided by ensuring the number of iterations and epochs of the training data is not too high such that the architecture is not over-trained. Alternatively *regularisation* methods are introduced. In the case of NNs these include introduction of dropout layers which obstruct the information flow, or adding loss function terms (for chosen layers) which're proportional to either the absolute value of a layers weights (L1), or the square of weights (L2), to penalise complicated connections.

Problems associated to overfitting are well established in the context of *bias* vs *variance*. This standard problem in ML addresses the importance of selecting the appropriate complexity of architecture to describe the dataset. Where a model is too simple such that it cannot well represent the true distribution of the data, the model is described to have high bias, however over-simple models typically have low variance such that they perform consistently on a variety of samples of the dataset. However, over-complex models in contrast tend to overfit to the data's true distribution and are hence described to have low model bias whilst being prone to high variance as they have overfit the trained sample.

Another typical issue of ML, associated primarily to classification problems is bias of the data. For example, where a classification problem dataset is naturally biased, having more occurrences of a particular class, an architecture can shortcut to a good accuracy by never predicting other classes. Although this gives promising learning measures, it is not truly learning the structure of the data, so must be avoided. This bias can be deterred by introducing class weights in the training process, such that the architecture used is told to pay proportionately more attention to smaller classes. Certain learning measures are less susceptible to bias in the architecture, giving a better measure of the architecture's performance.

In general the more data available for an architecture to train and test on, the better the model will be. In particular, higher dimensional data requires exponentially more datapoints for an equivalently effective model to be learnt; this conflict between wanting more technical data to work with (with more features / dimensions) whilst then needing exponentially more data to train well is known as the 'curse of dimensionality'.

Architecture hyperparameters dictate the style of training and ultimately impact the final trained model performance. They are the variables used to define the specific architecture and fitting algorithm used, such as the number of NN layers, layer sizes, training tolerances, etc. In selecting these hyperparameters it is important to also select a relevant loss function, appropriate optimiser, and representative metric.

The *optimiser* of a NN defines the method of optimising the non-linear fitting. It is defined during architecture compilation, and often uses some form of gradient descent in the architecture's parameter space. During the optimisation process (after seeing each data batch during training), the optimiser updates the architecture's parameters (i.e. weights and biases at each of the network's neurons) to better fit the data it has just seen, often with

some statistical variation included. The standard optimiser is stochastic gradient descent (SGD), however the most common is an extension of SGD known as Adam. Adam is an alteration of SGD which allows for variation of the learning rate according to the first and second moments of the parameter upgrading gradients [206].

The *loss function* is also specified during compilation of the chosen architecture. This is the function which the training process is seeking to optimise, there is no necessary range for the evaluation, but it is often a function which the machine is minimising by following the method specified by the network's optimiser choice. The choice of loss function depends on the problem type. For regression problems, a simple mean-squared error (MSE) or mean-absolute error (MAE) function may be used, fitting the architecture's approximate non-linear curve to the data in the standard statistical way. These functions, as well as other common choices, take the form

$$\begin{aligned} MAE &= \frac{1}{N}\sum_x |y_{true} - y_{predicted}| \ , \\ MSE &= \frac{1}{N}\sum_x (y_{true} - y_{predicted})^2 \ , \\ MAPE &= \frac{1}{N}\sum_x \left|\frac{y_{true} - y_{predicted}}{y_{true}}\right| \ , \\ log(cosh) &= \frac{1}{N}\sum_x log(cosh(y_{true} - y_{predicted})) \ , \\ \text{cross-entropy} &= -\sum_x \sum_c y_{true,c}\ log(p_{predicted,c}) \ , \end{aligned} \tag{2.3.5}$$

for N datapoints in the dataset $\{x\}$ each paired with an output y_{true} whose value is predicted by the architecture to be $y_{predicted}$, such that perfect prediction makes these equal and minimises each loss. Mean-absolute-percentage error (MAPE) is defined in an analogous way to MAE with each datapoints contribution being relative to its value (note this can cause blow-ups for data evaluating to 0). Log(cosh) is a smooth generalisation of MAE which approaches MAE in the limit of the output values being large but is smooth at the discontinuity of $y_{true} = 0$. so is better for gradient computations. For classification problems categorical cross-entropy is used, this is based on Kullback-Liebler divergence, and aims to introduce independence between the classes by considering their information entropy. It is defined as in (2.3.5), by summing over all datapoints x and classes c where $y_{true,c}$ now is the binary label ($\{0, 1\}$) of whether the current datapoint is in class c and $p_{predicted,c}$ is the predicted probability that this datapoint is in class c. A common variant used to avoid data output formatting is sparse categorical cross-entropy, this provides the same functionality but doesn't require the data labels in the form of a vector space, instead it can handle simpler integer class labels.

Measures of the performance, or equivalently *metrics*, are essential for justifying the use of ML in supervised problems (where there is output data to evaluate with). These functions are often of the same form as loss functions, providing similar functionality, however the metric is usually chosen to be of a simpler form. For regression problems the metrics are exactly the loss functions, however there is no restriction to use the same function as the

loss and of course they are now evaluated exclusively on the test data. In addition, there is a further traditional regression performance measure known as R^2. This evaluates how well a regressor is performing by comparing the proximity of the predicted output to the mean (which would be the prediction for a null regressor model). For this measure 1 is optimal, 0 means that prediction is not better than just predicting the true mean each time, and < 0 means worse than just predicting the mean. It is defined

$$R^2 = 1 - \frac{\sum (y_{true} - y_{pred})^2}{\sum (y_{true} - y_{truemean})^2} \,, \tag{2.3.6}$$

summing over all predicted, y_{pred}, and true, y_{true}, outputs in the test data. In addition for R^2 the mean of the true values over the test data outputs, $y_{truemean}$, is also used.

The most standard metric for a classifier is accuracy. This is the proportion of correct classifications performed by the classifier on the validation or testing datasets. To ensure the measure is unbiased, it is important these datasets are not used for training whilst still being representative. More technical measures of performance used include precision, recall, specificity, F1 score, and Mathew's correlation coefficient (MCC). These measures take into account Type I and II errors from misclassification. A Type I error being a 'false positive' (FP), where an object not in the considered class is classified into the considered class, and conversely a Type II error is a 'false negative' (FN), where an object is classified as not in its true class by the NN. The F1 score measure gives equal weight to Type I and II errors, it is the harmonic mean of precision and recall, whereas the MCC measure uses variable weights based on the occurrence of true positives and negatives (TP/TN). These factors make MCC a more favourable measure [207].

All metric measures can be summarised as functions over the confusion matrix, defined for binary classification:

$$CM = \begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix} \,, \tag{2.3.7}$$

and which can be generalised to the multiclassification case with:

$$CM = \begin{pmatrix} TP_1 & E_{12} & E_{13} & \cdots & E_{1n} \\ E_{21} & TP_2 & E_{23} & \cdots & E_{2n} \\ E_{31} & E_{32} & TP_3 & \cdots & E_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ E_{n1} & E_{n2} & E_{n3} & \cdots & TP_n \end{pmatrix} \,, \tag{2.3.8}$$

for n classes, where the entry CM_{ij} is the number of objects truly in class i which the classification architecture classifies into class j. Therefore for class k, the TP value is given by the diagonal entry (denoted TP_k), the FN value by $\sum_{i \neq k} E_{ki}$, FP by $\sum_{i \neq k} E_{ik}$ and finally TN by $\sum_{i,j \neq k} E_{ij}$. Such that for each class, or directly in the binary classification case, these

metrics which evaluate performance can be defined via

$$\begin{aligned} \textit{accuracy} &:= \frac{TP+TN}{TP+TN+FP+FN}\,, \\ \textit{precision} &:= \frac{TP}{TP+FP}\,, \\ \textit{recall} &:= \frac{TP}{TP+FN}\,, \\ \textit{specificity} &:= \frac{TN}{TN+FP}\,, \\ \textit{F1 score} &:= \frac{2\cdot TP}{2\cdot TP+FP+FN}\,, \\ \textit{MCC} &:= \frac{TP\cdot TN - FP\cdot FN}{\sqrt{(TP+FP)\cdot(TP+FN)\cdot(TN+FP)\cdot(TN+FN)}}\,. \end{aligned} \tag{2.3.9}$$

In addition, all of these measures have generalised forms for the multiclassification case, however due to their common use both broadly and in these studies the formulas for accuracy and MCC are given by

$$\begin{aligned} \textit{accuracy} &:= \frac{\sum_i CM_{ii}}{\sum_{ij} CM_{ij}}\,, \\ \textit{MCC} &:= \frac{\sum_{ijk}\left(CM_{ii}CM_{jk} - CM_{ij}CM_{ki}\right)}{\sqrt{\left(\sum_i\left(\sum_j CM_{ij}\right)\left(\sum_{k\neq i,l} CM_{kl}\right)\right)\left(\sum_i\left(\sum_j CM_{ji}\right)\left(\sum_{k\neq i,l} CM_{lk}\right)\right)}}\,. \end{aligned} \tag{2.3.10}$$

All measures excluding MCC evaluate in the range $[0,1]$, whilst the MCC measure takes values in $[-1,1]$. A value of 1 for any measure indicates perfect prediction of the model. All these metrics provide useful statistical insight, however accuracy is the industry standard measure, and MCC is typically considered to be the most representative measure of the learning (since it better accounts for varying class sizes, and is less susceptible to bias).

2.3.1.2 Unsupervised Architectures

What characterises data for unsupervised learning is the lack of output data to train with. Given a dataset unsupervised architectures seek to extract structure from this data, and this can be done in two distinct ways. Feature extraction identifies the most dominant parts of the data which best describe it in a simplified representation. Whereas clustering seeks to sort the data in its current form into sets with common behaviour or features.

In the research work carried out techniques from both of these branches were used, as described in the following.

Principal component analysis (PCA) is a method of feature extraction which looks for independent linear combinations of the inputs which have the most variance and hence best describe the distribution of the data [208]. This is practically computed by starting with the the covariance matrix, which for a dataset of n-dimensional vectors x_i takes the form

$$K(x_i, x_j) = E((x_i - E(x_i))(x_j - E(x_j)))\,, \tag{2.3.11}$$

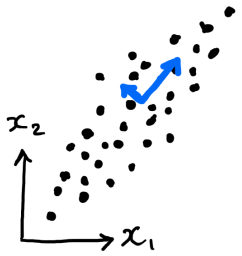


Figure 2.3.3: A PCA decomposition of a 2-dimensional dataset, showing the two principal components. The larger component blue arrow is the first principal component, representing the linear combination of the x_i coordinates which has the most variance across the dataset. The other component is orthogonal to this.

for expected values $E(x_i)$, the mean of that coordinates values. This matrix is positive semi-definite such that its eigenvalues are non-negative, and symmetric such that its eigenvectors are orthogonal. Therefore the eigendecomposition of this matrix produces an eigenbasis of 'principal components' which give the most natural directions to describe the data's variance. These eigenvectors are usually sorted in decreasing order of eigenvalue, and the first $k < n$ are selected as the dimensionally reduced basis to describe the data. This is done by projecting each data vector onto these k principal components using the $k \times n$ matrix built from these normalised eigenvalues. The choice of k can be made by examining the normalised distribution of the eigenvalues, say one wants to represent 80% of the data structure one could choose the first k eigenvectors such that $\sum_{i=1}^{k} \overline{\lambda} > 0.8$ for normalised eigenvectors $\overline{\lambda}$ sorted in decreasing order of size. An example of PCA in 2-dimensions is shown in Figure 2.3.3.

PCA can be extended to non-linear combinations of the inputs by use of the kernel trick (as described in §2.3.1.1 for SVMs) in kernel PCA (KPCA). The higher-dimensional embeddings of the feature vectors have an equivalent covariance matrix whose explicit diagonalisation is bypassed by use of the kernel to give a method of projecting the features onto these non-linear variational principal components. The kernel function is defined through the outer product operation on the higher-dimensional features (expanded into a function), in a parallel manner to the linear covariance case which is the average of the outer product of the centred feature vectors (as can be seen in (2.3.11) where the internal $E(x_i)$, $E(x_j)$ terms remove the means in centring).

As hinted to in the KPCA interpretation, PCA methods tend to perform better on centred data where all the feature vector components are shifted by their mean value across the data. The correlation matrix is a naturally centred object, but if one were to diagonalise the average outer product of the feature vectors directly the resulting eigenvectors would be shifted by a factor proportional to the mean vector. Therefore in this work centring is always performed implicitly to avoid this biasing. In addition to centring, scaling is another common practise prior to PCA. When scaling the components are divided by their variance such that each feature vector component has unit variance. This practise is important when each of the feature vectors have different units of measure and hence a direct comparison has less physical meaning so scaling is used to make the entries dimensionless and sensibly comparable. However in this work all the datasets were directly vectors (with common units) so this practise was not used.

Topological Data Analysis (TDA) is a class of methods used to extract topological features of a dataset. These methods are not always categorised under the machine learning umbrella, however due to their clear use for topological feature extraction we have listed them here.

The most prominent technique within TDA is *persistent homology* [209]. Homology and cohomology within algebraic geometry are concerned with equivalence classes of forms on manifolds, they create the groups which lead to computation of Hodge numbers for complex manifolds, and Betti numbers for real manifolds (although are also defined for complex manifolds via their real interpretation). Betti numbers b_k are the ranks of the respective homology groups H_k, and effectively b_k counts the number of k-dimensional curves which can be removed from a manifold keeping it connected. Betti numbers however also have a simpler interpretation in terms of complexes, which may be considered the discrete equivalents of manifolds, which make them particularly useful for data analysis through persistent homology.

For the n-dimensional datapoints in the dataset, considered as points in $\mathbb{R}^n$, a Vietoris-Rips (VR) complex can be created for all distance scales $\delta \in [0, \infty)$. For a set distance scale δ the VR complex Δ_δ is constructed by considering all subsets of points which are mutually separated by at most the Euclidean distance δ, then for each set of size $(\kappa+1)$ the Δ_δ contains a respective κ-complex (for $\kappa \leq n$, as a s-simplex needs at least s dimensions). A filtration of these complexes is then made as the δ value increases from 0, recording the next step in the filtration at each δ value the complex changes. This filtration can be imagined to be built by starting with $\delta = 0$ and the Δ_0 complex being the start of the filtration consisting of 0-simplices (points) for every datapoint. Then as δ is increased one could image n-balls being drawn centred on each datapoint in the $\mathbb{R}^n$ space of radius δ, and the balls' radii being simultaneously continuously increased. As soon as the first two balls touch a 1-simplex (line) is drawn between the equivalent two 0-simplices whose datapoints produced those balls. This changes the complex and gives the next step in the filtration. This steady increase in δ continues, adding κ-simplices every time κ balls intersect (for $\kappa \leq n$), and producing the next step in the filtration for each δ a change is made to the complex. Eventually δ will reach a value where all possible simplices exist between all datapoints, this is where the filtration terminates. Some VR complexes from the persistent homology filtration for an example 2-dimensional dataset are given in Figure 2.3.4.

The persistent homology is then built on this filtration of VR complexes. The role of persistent homology in the analysis of this filtration is to examine how long cycles of k-simplices last throughout the filtration before they become filled by the $(k+1)$-simplices they bound. Specifically H_k examines how long cycles of k-simplices exist until becoming filled by $(k+1)$-simplices. For n-dimensional space up to H_{n-1} can be considered, however as k increases H_k becomes much more computationally intensive to compute. Therefore in the projects TDA was applied, in §3.6 and §3.7, only H_0 and H_1 were considered and used the `ripser` library for computation [26].

Each class in H_0 is a connected component in Δ_δ. Since there is a homology group for each complex in the filtration, b_0 counts the number of connected components in the VR complex at each stage in the filtration. The persistence of persistent homology then concerns itself with how persistent these features are throughout the filtration, i.e. how

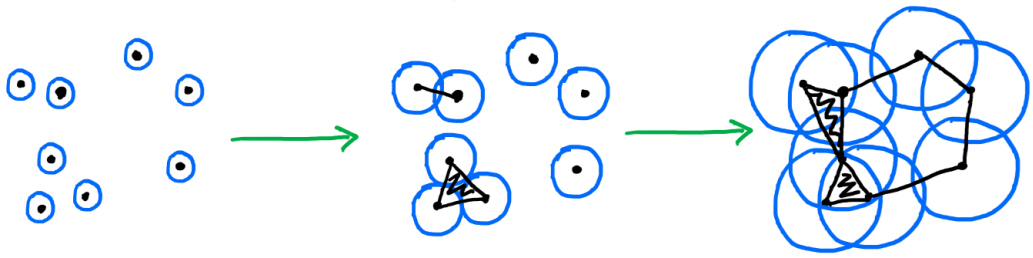


Figure 2.3.4: Selected VR complexes from the persistent homology filtration of an example 2-dimensional dataset. The first image shows the data with a small δ and hence none of the blue 2-dimensional balls (i.e. circles) overlap. All these points are H_0 features and hence $\beta_0 = 8$, whilst $\beta_1 = 0$. Then for larger δ where two balls overlap a 1-simplex (i.e. edge) is drawn between the points, and where three balls mutually overlap a 2-simplex (triangle) is also drawn. Here $\beta_0 = 5$ for the 5 connected components, whilst $\beta_1 = 0$ still. For the final even larger δ value even more balls overlap leading to many more 1-simplices, and an additional 2-simplex. Here there is only one connected component so $\beta_0 = 1$, and importantly there is now a H_1 feature from the ring of 1-simplices not filled by a set of 2-simplices, such that $\beta_1 = 1$.

many stages through the filtration until each H_0 feature b_0 counts disappears. Each H_k feature has its (birth,death) recorded, as the δ values where the feature was created and then destroyed respectively. For the H_0 features this means that the birth is always $\delta = 0$ where each datapoint is a component, whilst the death δ value is each time a 1-simplex is added connecting two different features into one (killing the shorter lived feature by convention). Equivalently the H_1 features are cycles of 1-simplices which are not the boundary of a set of 2-simplices (triangles). These are born at δ values where a sequence of points are distance $\leq \delta$ away from each other, but they are not mutually all within distance δ of each other. They then die when at the δ value within which all the points are mutually $\leq \delta$ away from each other.

Persistence diagrams plot these (birth,death) pairs for each of the H_k features. H_0 features naturally lie on a vertical line (since all the births are 0), and H_1 features indicate a lot of noise usually close to the birth = death diagonal where cycles are quickly filled. The interesting structure persistent homology seeks to find is large gaps in the H_0 persistence diagram line – where there are largely separated clusters of data which persist as connected components for a large δ range without change. While the interesting H_1 features are those far from the diagonal – which are ring-like structures requiring a big step in δ to be filled.

K-Means clustering is a clustering method which partitions the dataset into a specified number of clusters based on squared Euclidean distance [210], [211]. During clustering the K-Means optimiser takes an input pre-decided number of clusters, initialises cluster centre values for each cluster, and then iteratively updates these cluster centres such that the final sum of squared distances from each datapoint to its nearest cluster centre is minimised. The concept is related to that of Voroni cells, and during the iterative process firstly the datapoints are collected into clusters associated to the initialised cluster centre value they are closest to. Then within each cluster the mean is calculated and the cluster centre is updated to that value, then all the datapoints are reclustered with respect to the new cluster centres, with the process repeating until the cluster assignments of the datapoints don't change.

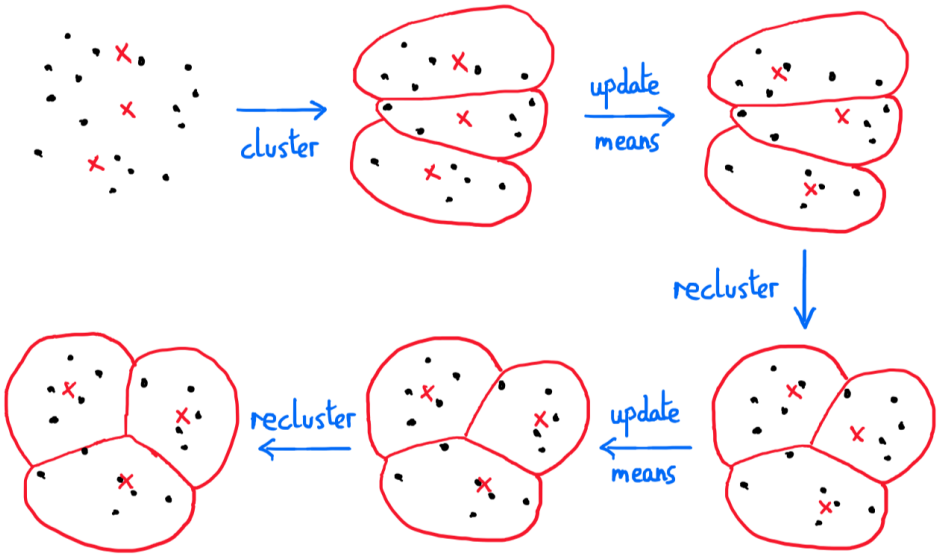


Figure 2.3.5: The iterative process of K-Means clustering is exemplified for this example 2-dimensional dataset. The data (black dots) is plotted in the 2-dimensional plane, and for the input of 3 clusters the cluster centres (red crosses) are randomly initialised. The next step shows the datapoints being sorted into clusters dependent on which centre they are closest to, shown by the red circles enclosing the clusters. For each cluster the centres are then updated according to the mean of the cluster members, moving the red crosses. Then the data is reclustered according to the new means, and the process repeated until convergence where the cluster membership of the datapoints does not change.

The most common measure of clustering performance is *inertia*, defined

$$\mathscr{I} = \sum_{\mathscr{C}} \sum_{x \in \mathscr{C}} \Big(\sum_{i} (\mu_{\mathscr{C},i} - x_i)^2 \Big) \ , \tag{2.3.12}$$

for clusters $\mathscr{C}$, with respective means $\mu_{\mathscr{C},i}$, and all datapoints x_i exclusively in their nearest cluster. The innermost bracket is the standard sum of squares of the difference vector components. Inertia is minimised when the clustering process has converged, but not necessarily to a global minimum as different initialisations can affect the performance. An example K-Means clustering process for a 2-dimensional dataset with 3 clusters specified on initialisation is given in Figure 2.3.5.

As described in the iteration process, the K-Means algorithm requires the number of clusters to use to be specified. This however is often the goal of a clustering task, to identify the optimum number of clusters. To find this optimum number a method known as the elbow method is used. Here the clustering is performed to convergence for a range of integers as the input numbers of clusters, and final inertia recorded. The inertia values, relative to the inertial value for clustering with one cluster (i.e. the total squared distance to the dataset mean for all the data), are plotted against number of clusters, with a weighting factor to penalise the use of too many clusters[12].

[12]The penalisation is needed since more clusters means trivially better clustering, for N datapoints N clusters has optimum inertia 0.

For the work in §3.6, a more novel measure we called 'scaled-max-inertia' was used in the elbow method. This measure identifies the maximum squared-distance any point is from its closest cluster centre, normalises it according to that maximum squared-distance from using only one cluster, and adds this weight factor to penalise using an excessive number of clusters. We define this to be:

$$\mathscr{I}_{max} = \frac{\mathrm{Max}_x\big(\sum_i(\mu_{\mathscr{C},i} - x_i)^2\big)}{\mathrm{Max}_x\big(\sum_i(\mu_1 \ - x_i)^2\big)} + \frac{(k-1)}{100}\,, \tag{2.3.13}$$

where Max_x determines the maximum over all datapoints x_i, examining the squared distance to either the closest cluster's mean $\mu_{\mathscr{C},i}$ or the single cluster's mean μ_1; then weighting by the number of clusters k.

As well as the inertial values for the full dataset, normalised equivalents were also used to measure the final clustering performance, based on a normalisation by the number of datapoints and also by the range of the datapoints' values. These are detailed further with the results in §3.6.

2.3.2 Network Science

Many of the mathematical objects considered in these studies take the form of *graphs*, such as quivers, dessins, and cluster algebra exchange graphs. Graphs are defined with two sets, a set of vertices/nodes $\mathscr{V} = \{\nu_i\}$, and a set of edges $\mathscr{E} = \{\varepsilon_\alpha\}$ which are pairs of vertices $\varepsilon_\alpha = (\nu_i, \nu_j)$ and define the connectivity amongst them [212]. Graphs can be undirected such that the presence of (ν_i, ν_j) implies the presence of (ν_j, ν_i) and a single edge is drawn to represent them, or directed where an arrow labels the direction from first to second vertex in the edge pair. If the vertex set can be split into two subsets such that all the edges are only between these sets then the graph is 'bipartite', the vertices are usually then drawn with black or white colouring respectively. Graph vertices and edges can have labels, and in particular if the edges are assigned numerical labels, known as weights, the graph becomes a weighted graph and is called a *network*.

The embedding of the graph is not important to its structure, all that matters is its content and the combinatorics of its connectivity. The connectivity can be represented by an adjacency matrix $\mathscr{A}_{ij}$, whose entries indicate either the number (graphs) or weight (networks) of the edges connecting vertices ν_i and ν_j. In the undirected case these matrices are naturally symmetric, and for directed graphs without multiple edges between the same vertices they will be antisymmetric. Another important type of edge is a loop, where both vertices in the edge pair are the same (ν_i, ν_i). Many graphs under our consideration are 'simple' such that they have no loops and don't have multiple edges between the same vertex pairs, and also built from only one connected component. An example graph, with a variety of the features discussed is shown in Figure 2.3.6.

When analysing a graph the first property one considers is usually the *degree* distribution. The degree of a vertex is the number of edges incident to it, in the directed case this splits into an in-degree and out-degree for edges ending or starting on the vertex respectively. Next a style of analysis known as *clustering* examines how the distribution of this connectivity leads to clustering of the nodes. The measures for clustering used in this work are global in nature (i.e. functions of the full graph); they were the average triangle and square clustering

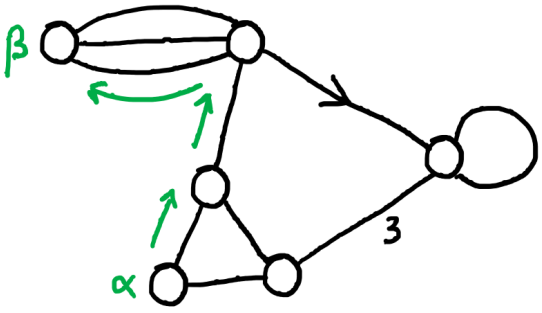


Figure 2.3.6: An example graph with vertices (black circles) and edges (black lines). There is an example of multiple edges in the top left, directed edge in the top right, a loop on the right, a weighted edge on the bottom right, and a cycle between the three bottom vertices. An example shortest path is drawn between the nodes labelled α and β, with the route traced out in green.

coefficients [213]. These measures return the proportion of sets of three/four vertices which are connected in a triangle/square respectively.

Beyond degrees, and related to clustering, another important property of the connectivity is the existence of *cycles*. Firstly a path around a graph is a sequence of edges where each subsequent edge starts at the vertex the previous edge finished at; from there a cycle is a path with the first and last vertices the same. Existence of cycles shows some degeneracy in connectivity as there are multiple different paths which link the same start and end vertices. Trees are graphs without cycles, and spanning trees are subgraphs which are trees including all the vertices in the original graph – these are useful for building a minimum cycle basis.

Considering the cycles in a graph, they can be added via symmetric difference. The symmetric difference of two cycles is the set of all edges that belong to only one of the two cycles (i.e. any edge in both cycles is removed and the cycles are joined into one on the other edges), and all the vertices of both cycles. With this addition rule, and multiplication by 0 to give the null graph of the cycle (just the vertices) or by 1 to give the same cycle, a cycle space can be defined over the finite field $\{0, 1\}$ [212]. In this cycle space each vector entry is one of the possible cycles in a basis, where 1 indicates presence of the cycle and 0 indicates the cycle is not present. Since there are many possible bases for the cycle space, the minimum cycle basis is often chosen (the basis with the smallest total number of edges, or weight if networks) [214]. Interestingly there is a dual to this cycle space, the cut space. The cuts which make up this space are the edges that when removed separate the graph into two connected components.

Further to analysis of degrees and cycles, in graphs one is also often interested in the *shortest paths* between any two vertices. The Wiener index provides a global measure of this shortest path, as the total length of all shortest paths between all distinct pairs of vertices across the vertex set [215]. This measure can then be suitable normalised by the number of pairs ($nC2$). The final common avenue of network analysis is *centrality* analysis. Centrality measures how relatively influential vertices in the graph are. The method focused on in this thesis was eigenvector centrality [216]. This is computed by diagonalisation of the adjacency matrix, and taking the unique vector with the largest eigenvalue. The respective eigenvector then gives the relative centralities of all the vertices in the graph, note if they are all equal

then there is no obvious centre to the graph (it is symmetric).

Therefore when examining a graph or network, as in the work of §3.8, a full analysis considers the degree distribution and techniques from each of: clustering analysis, cycle analysis, shortest path analysis, centrality analysis. Although the exchange graphs in §3.8 are the graphs seeing the most analysis through these methods, the quivers themselves, as multi-digraphs, are valid graphs in their own right (as in §3.2) which are directed and allow multiple edges between vertices. The `python` library `networkx` was used for handling of the graph objects and computation of these network analysis measures [27].

3 Research Work

Throughout the period of my doctoral studies I have been very fortunate to work on a large variety of projects, studying a range of different mathematical objects through the lens of machine learning and data science.

In this section the research work performed by myself in contribution to the respective papers is summarised. The sections are listed by the `arxiv` numbers of the respective papers and the mathematical objects which were the centre of each ML investigation.

3.1 Dessins d'Enfants (`2004.05218`)

Dessins d'enfants, as introduced in §2.1, are bipartite graphs that describe the structure of Riemann surfaces through the combinatorics of their Belyĭ maps, expressing them as covers over the Riemann sphere, $\beta : \mathcal{V} \longmapsto \mathbb{P}^1_{\mathbb{C}}$.

Dessins first saw academic interest for determining combinatorial invariants associated with the action of the absolute Galois group of the rational numbers, $Gal\big(\overline{\mathbb{Q}}\,/\,\mathbb{Q}\big)$, which has no direct description [59]. In defining the Galois group, *Galois extensions* of a base field are considered. These are algebraic field extensions which are both normal and separable[13] [217]. A Galois group is then the group of all automorphisms of a Galois extension which fix the base field. Beyond these the *absolute* Galois group of a field, requires a specific extension of the base field known as the separable closure of the field. Note that here since the base field considered, $\mathbb{Q}$, is perfect, the separable closure is equal to its algebraic closure denoted $\overline{\mathbb{Q}}$.

The important relation which the theory of dessins capitalises on is an equivalence between several categories. The equivalence relates the category of embedded dessin graphs with: finite sets under permutation; field extensions; and certain types of algebraic curves. Since field extensions are directly related to the definition of the absolute Galois group, it is through this categorical equivalence that representations of the Galois group act faithfully on dessins d'enfants; which is what makes dessins so useful in studying the elusive $Gal\big(\overline{\mathbb{Q}}\,/\,\mathbb{Q}\big)$ group [218], [219].

Much of the inspiration for dessins came from Belyĭ's theorem. This states that any non-singular algebraic curve represents a 3-point ramified covering of the Riemann sphere. Therefore the Belyĭ function used to define a dessin is ramified at exactly three points, from

[13] A field extension E/F is normal if every minimal polynomial with roots in E has all its roots in E and separable if it has no repeated roots.

here the three points can be mapped to $\{0, 1, \infty\}$ under a Möbius transformation of the form:

$$z \longmapsto \frac{az+b}{cz+d}\,, \quad \text{for } a, b, c, d \in \mathbb{Z}\,,\ z \in \mathbb{C}\,. \tag{3.1.1}$$

The white and black nodes are associated with the preimages of the points 0 and 1 respectively under the action of the Belyĭ function. In addition the edges are associated to the preimage of the unit interval [0,1]. Finally, the ramification of the ∞ points are then inferred from the graph structure. Since the ramification points represent degeneracy in the Belyĭ function, the graph-theoretic degree of the vertices hence corresponds to the ramification degree of the general Riemann surface cover, as well as the first non-zero order in the map's Taylor expansion.

Dessins in the database considered in this study are 'clean', such that the degree of all vertices in one of the sets is 2. This allows simplification of the representation by removing all of the degree 2 vertices and connecting the edges which were incident to each – producing a standard graph with one type of vertex. Alternatively introducing vertices at the edge midpoints of standard graphs forms clean dessins. Additionally, any dessin can be ascribed a genus associated with the lowest genus surface it can be drawn on. In particular, these dessins are 'planar', as they can be embedded on a genus 0 sphere, or the $\mathbb{R}^2$ plane.

This specific dataset of dessins examined relate to certain subgroups of the modular group, $PSL(2, \mathbb{Z})$. The focus was on those dessins corresponding to specific forms of elliptically fibred K3 surfaces as examined in [220], whose corresponding Seiberg-Witten curves were also computed in this work following [60], and further examined in [221]. To classify these dessins according to the degree of the field extension they are defined over, and hence equivalently the size of their Galois orbits, NNs were used working with tensorial representations of the dessins.

The first of the representation methods takes the adjacency matrix of the dessin as input, whilst the second a list of cycles around each node inspired by the set permutation category equivalence (which we call cyclic edge lists); both are classified according to the degree of the field extension as output. High accuracy for the cyclic edge list learning suggests the possibility of a way to directly understand the Galois orbits of dessins, and thence the absolute Galois group via tensor representations of the dessins' graph information, at least stochastically.

Action of elements of the absolute Galois group may change the subfields of the algebraic completion which the Belyĭ functions are defined over, this maps Belyĭ maps to each other, and hence corresponds to moving around orbits of dessins. The orbit of a dessin is all dessins that the absolute Galois group can transform the original dessin into. The largest orbit in the dataset considered corresponds to a quartic extension of the rationals. The minimal polynomial associated with the quartic root used in the extension hence has four distinct roots (since the extension is also separable). Therefore each dessin in this Galois orbit corresponds to an extension of the rationals by one of these four roots. The root corresponding to each of these dessins is used in defining the coefficients of its Belyĭ map. Importantly the size of the orbit of a dessin corresponds to the degree of the field extension. This degree of field extension is also an invariant under Seiberg duality of the equivalent dimer model, and hence acts as a useful test for QFT duality, the physical interpretation of this degree is further examined in [222].

3.1.1 Modular K3 Surfaces and Dessins

As stated previously, the dessins considered in this paper, arise through analysis of the modular group, $PSL(2,\mathbb{Z})$. It is the group of 2×2 matrices with integer entries and unit determinant. The projective quality of the group arises through the identification of elements $A\sim -A$ through the standard quotient method in the group's definition: $SL(2,\mathbb{Z})/\{\pm\mathbb{1}\}$. This group represents fractional linear transformations on the upper half complex plane, $\mathscr{H}$, acting alike Möbius transformations in (3.1.1).

A principle congruence subgroup is the kernel of a reduction modulo n morphism on the field the matrix elements are defined over, i.e. it is what maps to the identity under $\pi_n : SL(2,\mathbb{Z}) \longmapsto SL(2,\mathbb{Z}/n\mathbb{Z})$. Note the actual matrices aren't necessarily finite order modulo n, just their elements are either 1 (if on-diagonal) or 0 (off-diagonal) modulo n. A congruence subgroup is then any subgroup which contains a principle congruence subgroup for some $n\geq 1$; the smallest n such that the nth principle congruence subgroup is contained is known as the level of the subgroup [223].

More specifically we are interested in torsion-free subgroups, where all elements of the subgroup (except the identity) are of infinite order. Additionally the subgroups considered must be genus zero. The genus of the subgroup is the topological genus of the modular curve formed from the quotient of $\mathscr{H}$ by the subgroup's action (such that parts of $\mathscr{H}$ which map to each other under elements of the subgroup are identified). This is equivalent to saying the modular curve formed from the subgroup is topologically the Riemann sphere [224].

Importantly there are only 33 torsion-free genus zero congruence subgroups of the modular group. The conjugacy class decomposition of the modular group using these subgroups leads to indexes: $\{6,12,24,36,48,60\}$, where the restriction to a multiple of 6 can be shown through use of the Riemann-Hurwitz formula [220]. The action of the subgroups in question can be extended from $\mathscr{H}$ to $\mathscr{H}\times\mathbb{C}$ such that:

$$(\tau,z)\longmapsto\left(\gamma\tau,\frac{z+m\tau+n}{c\tau+d}\right),\quad \text{for }\gamma=\begin{pmatrix}a & b\\ c & d\end{pmatrix},\ m,n\in\mathbb{Z}\,, \tag{3.1.2}$$

where γ denotes action of an element of one of the subgroups with the Möbius action of (3.1.1). Now taking the quotient of $\mathscr{H}\times\mathbb{C}$ by this extended automorphism gives the modular curve (from before the extension) with an elliptic fibration to form the unique 'modular surface' for that subgroup. The index of the subgroup used gives the Euler number of the corresponding modular surface [220].

The modular surfaces formed from the index 24 torsion-free genus zero congruence subgroups are K3 surfaces. These modular surfaces hence take the form of a Weierstraß equation:

$$z_1^2 = 4z_0^3 - g_2(z)z_0 - g_3(z)\,, \tag{3.1.3}$$

where the coefficients g_2 and g_3 are functions of the base Riemann sphere's complex coordinate z, alike in §2.2 for construction of elliptically fibred CYs but using $z_2=1$ to give the non-compact version in $\mathbb{C}^2[z_0,z_1]$ (not $\mathbb{P}^2_{\mathbb{C}}$), and fibred with the new coordinate z.

Weierstraß elliptic equations may be thought of as the relation between the Weierstraß elliptic function, $\mathfrak{P}$, and its derivative $\mathfrak{P}'$ (respectively denoted by z_0 and z_1 in (3.1.3)). These Weierstraß elliptic functions are two-periodic, and hence are functions of some abstract complex plane coordinate, $\mathfrak{z}$, and two periods (which define a lattice structure for the complex

plane in $\mathfrak{z}$). The two periods of this lattice define the 'fundamental parallelogram' in the complex plane, and due to the periodic structure, this parallelogram hence has the topology of a 2-torus; an important property in the definition of this fibration as an elliptic curve [225].

Taking the Laurent expansion of the Weierstraß function about the origin gives a series in increasing powers of two. The coefficients for the $\mathfrak{z}^2$ and $\mathfrak{z}^4$ terms are directly proportional to the g_2 and g_3 coefficients (used in (3.1.3)) respectively. These coefficients are known as invariants of the function. Taking a polynomial equation of these invariants one can form the 'modular discriminant', a modular form of weight 12. This is defined

$$\Delta := g_2(z)^3 - 27g_3(z)^2\,, \tag{3.1.4}$$

and as a discriminant it indicates the degeneracy of roots of the cubic part of the Weierstraß elliptic equation. Modular forms are particularly useful in creating Galois representations, as well as appearing in many other useful areas of mathematics also. In general they are holomorphic functions on $\mathscr{H}$, and satisfy the condition

$$f\left(\frac{az+b}{cz+d}\right) = (cz+d)^k f(z)\,, \tag{3.1.5}$$

for weight k. This condition shows the response of a modular form to action of the modular group on its input. Using the modular discriminant, a weight zero modular form can be defined, this is known as the j-invariant,

$$j(z) := \frac{g_2(z)^3}{\Delta} = \left(\frac{g_2(z)^3}{g_2(z)^3 - 27g_3(z)^2}\right), \tag{3.1.6}$$

which is a modular function (in fact the unique one that generates the field of modular-invariants). Since the form is weight zero it is hence invariant under action of the modular group. These j-invariants for the K3 modular surfaces in consideration can be considered as endomorphisms of the Riemann sphere, which are ramified at $\{0, 1, \infty\}$. Note that some definitions of $j(z)$ may include an additional factor of 1728, requiring a further Möbius transformations to convert the second ramified point from $1728 \mapsto 1$.

The key observation is that due to the ramification structure of (3.1.6), these j-invariants are Belyĭ! This striking property is discussed and proved in [226]. To get a sense for this structure consider $j(z) = 0$, this requires $g_2(z) = 0$, and the cubic dependence on it in the invariant function makes each root into a three-fold ramification point; these are the white nodes of the dessin. Equivalently $j(z) = 1$ requires $g_3(z) = 0$, and the square dependence here makes each root into a two-fold ramification point; the black nodes. Finally the $j(z) = \infty$ singularities require the modular discriminant $\Delta = 0$ (or $z \mapsto \infty$). Thus, each modular surface gives a dessin via its j-invariant.

In particular, the degree of the j-invariant's numerator equals the surface's index (here 24). So that the $z \mapsto \infty$ limit remains a singularity we require the denominator to have a lower leading order, hence the g_2^3 and g_3^2 factors must be of the same degree (both being 24), so the leading power of g_2^3 can be negated. This makes the invariants g_2 and g_3 of degree 8 and 12 respectively, leading to 8 trivalent white nodes, and 12 bivalent black nodes in these dessins.

Henceforth, we will focus on these 'extremal' K3 surfaces where all singular fibres are of Kodaira type I_n and there are exactly 6 of these fibres (further discussed in [226] and [220]). These 6 Kodaira singularities act as a 6-part partition of the mapping's degree (which is 24), giving the passport information. These 6 ramifications induce the 6 faces of the dessins (including the outside face, since the dessin is truly drawn on a sphere). There are 112 elliptic modular K3 surfaces which are extremal (indeed, there are 199 6-partitions of 24 but [226] showed only 112 produce K3s) and we will focus on these in this study[14].

3.1.2 The Dessin Data

In summary, we study 112 extremal K3 elliptic fibrations, where the modular K3 surfaces have index 24. The j-invariant of each is a Belyĭ map with passport

$$\left\{\begin{array}{c} 3^W \\ 2^B \\ n_1^{a_1}, n_2^{a_2}, ..., n_k^{a_k} \end{array}\right\}, \quad \begin{array}{ccc} W & = & \text{number of preimages of 0} \\ B & = & \text{number of preimages of 1} \\ \{n_i^{a_i}\} & = & \text{cusp widths of elliptic modular K3 surface} \end{array} \quad \begin{array}{c} = \ 8 \\ = \ 12 \\ \\ \end{array} \tag{3.1.7}$$

such that

$$\sum_i a_i = 6\,, \quad \sum_i a_i \cdot n_i = 24\,. \tag{3.1.8}$$

Each dessin is planar, trivalent, and clean and corresponds to a particular subgroup of the modular group $PSL(2;\mathbb{Z})$. Of these, 9 are congruence subgroups, including the principal ones $\Gamma(m)$, $m = 3, 4, 5$; whilst the remainder correspond to more general ones.

All such K3 surfaces and associated dessins were classified in [226] and [227], with respective subgroups computed in [220]. All these dessins, along with their field extensions, adjacency matrices, and cyclic edge lists are available as a database in the appendices of [30].

As can be seen in this database, there are actually more than 112 dessins in the dataset, in fact there are 191. Whilst there are 112 distinct passports in this dataset, the dessins sort themselves into 125 orbits of varying degree extension, where the degree of extension is equal to the size of the orbit (leading to 191 dessins). In some cases there are multiple orbits per passport, and this accounts for the extra 13 orbits on top of the 112 passports.

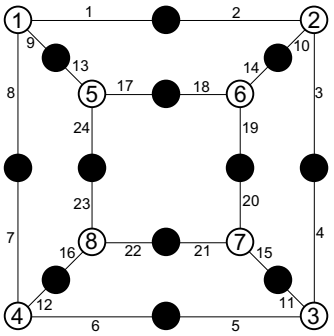
Dessin isomorphisms occur when a Galois orbit contains complex conjugate pairs, such that the Galois action between these conjugate roots leads to a chirality flip of the dessin embedding. Additionally, they may also occur in rare cases between dessins in different orbits. Both these subtleties associated to graph isomorphism are rectified with the cyclic representation method, which is sensitive to the embedding. Removing repeated matrices due to these isomorphisms leads to 152 distinct matrices in the dataset, corresponding to the 152 dessins unique up to this isomorphism within their own orbit (where isomorphisms occur out of orbit both matrices are removed). Whereas for the cyclic edge list representation all 191 lists are independent and can all be used.

Since these dessin objects are drawn as graphs, it is logical to first try learning from the adjacency matrix representation method (the graph-theory standard method of tensorial

[14]Note that if dessins are in the same Galois orbit they necessarily have the same passport (but not vice versa).

representation). Therefore under this representation method, since our dessins are clean, it suffices to only draw the white vertices and the graph is, at least combinatorically, completely captured by adjacency matrices of shape 8×8. The second data input style represents dessins by cyclic lists of the edges surrounding each node to give a white node list set of 3 edges per 8 nodes, and black node list set of 2 edges per 12 nodes such that the full representation has 48 entries. This is a faithful representation method, and hence all 191 dessins are included in this dataset whereas the adjacency matrix representation can only use 152 distinct dessin matrices.

Importantly, for our dataset, the dessin Belyĭ maps are defined over $\mathbb{Q}$, or some extension of $\mathbb{Q}$ involving a square, cubic, or quartic root. This means that the extension degree of all our dessins are 1, 2, 3, or 4 (where 1 means $\mathbb{Q}$ itself). Where the field is an extension of $\mathbb{Q}$, the Galois action maps between the roots of the minimal polynomial defining the extension and in each case this corresponds to a different dessin.



Example: This example dessin, as given in Figure 2.1.3b, is shown again to the right with vertices and edges arbitrarily numbered. This dessin corresponds to ramification passport:

$$\left\{\begin{matrix} 3^8 \\ 2^{12} \\ 4^6 \end{matrix}\right\}, \tag{3.1.9}$$

which can be read off from the dessin due to the 8 valence 3 white vertices, 12 valence 2 black vertices, and 6 faces (including outer face) which are each bordered by 4 white vertices (and 4 black vertices). Using the respective Belyĭ map provided in [228]:

$$\beta(z) = \frac{(z^8-14^4+1)^3}{(-108z^4(z^4+1)^4)}\,, \tag{3.1.10}$$

since the map contains no roots of rational numbers the field is simply $\mathbb{Q}$ with no extension and hence an orbit size of 1 containing only this dessin. Equating the map to (3.1.6), the functions g_2 and g_3 can be extracted:

$$\begin{aligned} g_2(z) &= z^8 - 14z^4 + 1\,, \\ g_3(z) &= \frac{1}{3\sqrt{3}}\big(z^{12} + 33z^8 - 33z^4 - 1\big)\,. \end{aligned} \tag{3.1.11}$$

Using these functions in (3.1.3), the Weierstraß equation for the modular surface is:

$$z_1^2 = 4z_0^3 - z_0\big(z^8 - 14z^4 + 1\big) - \frac{1}{3\sqrt{3}}\big(z^{12} + 33z^8 - 33z^4 - 1\big)\,. \tag{3.1.12}$$

This is the modular surface which corresponds to the extended quotient action of the congruence modular subgroup $\Gamma(4)$ on $\mathscr{H}$, as described in §3.1.1.

Conversely calculating the preimages of $\{0, 1, \infty\}$, the dessin can be reproduced. The preimages of 0 correspond to roots of the Belyĭ map numerator. The preimages of 1 correspond to roots of the polynomial formed from the difference of the Belyĭ map numerator

Images $\beta(z)$	Preimages z	Number of Preimages	Ramification Multiplicity
0	$\sqrt{2\pm\sqrt{3}}\,,-\sqrt{2\pm\sqrt{3}}\,,i\sqrt{2\pm\sqrt{3}}\,,-i\sqrt{2\pm\sqrt{3}}$	8	3
1	$\pm1\,,\pm i\,,$ $\pm\left(1+\frac{1}{\sqrt{2}}+i\sqrt{\frac{1}{2}(3+2\sqrt{2})}\right),\pm\left(1+\frac{1}{\sqrt{2}}-i\sqrt{\frac{1}{2}(3+2\sqrt{2})}\right),$ $\pm\left(1-\frac{1}{\sqrt{2}}+i\sqrt{\frac{1}{2}(3-2\sqrt{2})}\right),\pm\left(1-\frac{1}{\sqrt{2}}+i\sqrt{\frac{1}{2}(3-2\sqrt{2})}\right)$	12	2
∞	$0\,,\pm\frac{1}{\sqrt{2}}(1+i)\,,\pm\frac{1}{\sqrt{2}}(1-i)\,,\infty$	6	4

Table 3.1.1: The preimages of $\{0,1,\infty\}$, along with their multiplicities, for a Belyĭ map corresponding to congruence modular subgroup $\Gamma(4)$.

and denominator. Finally the preimages of ∞ are given by the limit $z\mapsto\infty$, or the roots of the Belyĭ map denominator.

These preimages, along with their ramification multiplicities are listed in Table 3.1.1, calculated using `mathematica` [29]. In each case the number of preimages and the multiplicities agree with this example dessin's passport. Therefore producing the 8 trivalent white vertices, 12 bivalent black vertices, and 6 faces (including the outer face) of the dessin in this example.

This example dessin can be represented in both of the two tensorial forms used in this study. The first through its adjacency matrix, as given in (3.1.13), by absorbing the redundant information of the black vertices (all valence 2 as the dessins are clean) into the edges and only considering the white vertices. Noting also that this representation is only unique up to permutation of the rows/columns corresponding to the relabelling of the vertices. The second representation was as a cyclic edge list, listing the edges surrounding each white/black vertex in a clock/anticlockwise manner, shown in (3.1.14). For this representation there is redundancy associated to cyclic permutation of the edge lists, reordering of the edge lists, and relabelling of the edges.

$$M_{8\times8}=\begin{pmatrix}0&1&0&1&1&0&0&0\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\1&0&1&0&0&0&0&1\\1&0&0&0&0&1&0&1\\0&1&0&0&1&0&1&0\\0&0&1&0&0&1&0&1\\0&0&0&1&1&0&1&0\end{pmatrix}\,, \tag{3.1.13}$$

$$\begin{aligned}V_{48}=\{&\{\{8,1,9\},\{2,3,10\},\{4,5,11\},\{12,6,7\},\{18,14,19\},\\&\{22,16,23\},\{13,24,17\},\{21,20,15\}\},\\&\{\{1,2\},\{9,13\},\{10,14\},\{3,4\},\{15,11\},\{24,23\},\{17,18\},\\&\{8,7\},\{5,6\},\{12,16\},\{21,22\},\{20,19\}\}\}\,.\end{aligned} \tag{3.1.14}$$

Therefore in each of the ML investigations the NN architecture is seeking to learn the relationship

$$\{M_{8\times8},V_{48}\}\longrightarrow\text{Extension Degree over }\mathbb{Q}\,, \tag{3.1.15}$$

Data Input Type		Averaged Measures	
		Accuracy	MCC
$M_{8\times 8}$ (matrix)		0.54 $\pm$ 0.00	0.18 $\pm$ 0.01
V_{48} (edge list)	1000:0:0	0.92 $\pm$ 0.03	0.88 $\pm$ 0.04
	0:1000:0	0.38 $\pm$ 0.02	*nan* $\pm$ *nan*
	0:0:1000	0.55 $\pm$ 0.01	0.28 $\pm$ 0.02
	10:10:10	0.42 $\pm$ 0.02	*nan* $\pm$ *nan*

Table 3.1.2: ML results showing the learning measures of Accuracy and MCC, each averaged with standard error over the 5 NNs trained in each investigation for the 5-fold cross-validation performed. The 5 investigations constitute 2 data formats, the first the adjacency matrix format, the second the cyclic edge list format. For the cyclic edge list format the number of permutations generated under each redundancy are also given in the form {Edge Cycles : Node Reordering : Edge Relabelling}. The *nan* return value (not a number) constitutes where the MCC could not be computed as one class was never predicted in one of the cross-validation runs.

via multiclassification across the possible extension degrees $\{1, 2, 3, 4\}$.

Each respective dataset of tensorial representations was augmented to increase its size using the respective redundancies. For the matrix representation, the 152 distinct adjacency matrices were permuted through vertex relabelling to produce $\sim 200,000$ matrices, ignoring any repeated matrices (i.e. that were invariant under the chosen permutation). For the edge list representation, the 192 dessin lists were permuted in each of the 3 redundancy reordering/relabelling methods (both individually and together) to produce again $\sim 200,000$ vectors (flattened versions of the lists). The proportions of the datasets with extension degree $\{1, 2, 3, 4\}$ were $\{0.49, 0.31, 0.19, 0.01\}$ and $\{0.40, 0.35, 0.24, 0.02\}$ respectively for matrix and edge list representations. These proportions were approximately the same after data augmentation, and were used as class weights in training the NNs to avoid biasing.

3.1.3 ML Results

NNs built from 4 layers of 512 Leaky-ReLU neurons (with 0.25 dropout) and an output softmax layer of 4 neurons, were trained with an Adam optimiser on cross-entropy loss for 20 epochs with batches of size 32. The 5-fold cross-validation results are shown in Table 3.1.2.

The results show mediocre success for learning the adjacency matrix representations, since an accuracy of 0.25, and MCC of 0, denotes true random guessing amongst the 4 categories. This lesser performance here may be perhaps attributed to the subtleties in using this representation method for dessins associated to isomorphisms occasionally occurring outside of Galois orbits. For the cyclic edge list representation, the latter 3 experiments were not especially successful, for some NNs classes were completely ignored in classification, leading to MCC being incalculable (shown by the value '*nan*'). However the cyclic edge list

representation with dataset augmented using the cyclic reordering of edges within each node list was highly successful, exceeding 0.9 accuracy and with MCC 0.88 (indicating non-bias learning across the classes). This result is exceptionally strong, and shows the importance of the cyclic property in this representation, and suggests a promisingly strong link between this cyclic edge representation of dessins and their Galois orbit size / extension degree.

Conclusion
This study showed the success of ML methods in predicting the degree of the field extension over the rationals which a dessin d'enfant graph's Belyĭ map is defined over. The simplicity of the networks used indicates that perhaps there is a simpler description undermining the dessin structure which allows the extension degree to be extracted. Since the field extension is closely related to the orbit structure of the dessins, perhaps this indicates a direction for further research, and supports the use of dessins in examining the Galois group structure of $Gal\big(\,\overline{\mathbb{Q}}\,/\,\mathbb{Q}\,\big)$.

3.2 Quiver Mutation (2006.10783)

Quivers, as introduced in §2.1, diagrammatically represent supersymmetric gauge theories using graphs from graph theory. Within these general supersymmetric gauge theories, those from quivers which represent 4-dimensional $\mathcal{N} = 1$ theories exhibit an electric-magnetic duality known as Seiberg duality [49] which connects theories with the same low-energy IR description under renormalisation group flow. The quivers describing these theories are connected through a graph mutation process, as depicted in Figure 2.1.2a.

In this process the node mutated has its gauge group swapped to the dual group (by changing the rank as described in the figure), and incident fields to the node are swapped to their conjugate representations in this dual group. A meson field is then introduced as an edge connecting any 2-path through the node into a 3-cycle, and finally where introduced edges lead to a 2-cycle both edges are removed, as these correspond to two chiral-multiplets in opposite bifundamental representations combining to form a massive $\mathcal{N} = 2$ hypermultiplet which can be integrated out of the theory as focus is on the low-energy effective theory where these massive fields are not present. The dualising process for the gauge group rank update is defined in the figure using incoming arrows, however could also be equally well defined with outgoing arrows and thus leads to ambiguity for anomalous quivers where $\sum_{\ell\mapsto j} a_{\ell j}N_\ell \neq \sum_{j\mapsto\ell} a_{j\ell}N_\ell$ for mutated node j. Hence in this study, the rank information is only considered in the anomaly-free cases. In a sense, one can think of anomaly-free theories as existing in the kernel of a quiver's adjacency matrix, for their gauge group rank selection.

In conjunction to the quiver, when considering a quiver gauge theory it is also important to consider the theory's superpotential, which determines the theory's interactions through its Lagrangian description. The fields corresponding to the quiver arrows arise in the superpotential as gauge invariant operators, which are combinations of traces and products of fields corresponding to selected cycles in the quiver [229]. Under Seiberg duality, the dual theory's superpotential is equivalent to the original with products of fields relating to 2-paths through the dualised node replaced with meson fields (i.e. $X_{ij}X_{jk} \mapsto M_{ik}$); and then meson fields replaced by cubic interaction terms between the meson and the two new dual fields relating to the new reversed arrows of the 2-path (i.e. $M_{ik} \mapsto M_{ik}\tilde{X}_{kj}\tilde{X}_{ji}$).

As also discussed in §2.1, the Seiberg duality process is mathematically equivalent to the process of cluster algebra mutation [230]–[232]. Cluster algebras are built from a subset of generators under the cluster mutation process, dictated by quivers (specifically $\mathcal{N} = 1$ quivers with no loops or 2-cycles). Dropping the cluster information from each seed gives only the mutating quivers, and respectively the Seiberg duality process. Cluster algebras are further discussed in §3.8, where this work [31] is extended to analysis and ML of cluster algebras directly in [37].

Repeated application of the duality process on all the quiver's nodes produces a duality tree (or quiver exchange graph in the cluster algebra language). Where interestingly the node ranks are constrained to always obey a classifying Diophantine equation [51], [233]. The structure of these duality trees is dictated by the equivalent cluster algebra type, as described in §2.1, being finite or infinite.

3.2.1 The Quiver Data

Beyond the desire for both finite and infinite types, the selection of the quivers used in this investigation was somewhat arbitrary. However, those with fewer numbers of nodes were preferable to make computation of the mutation more feasible.

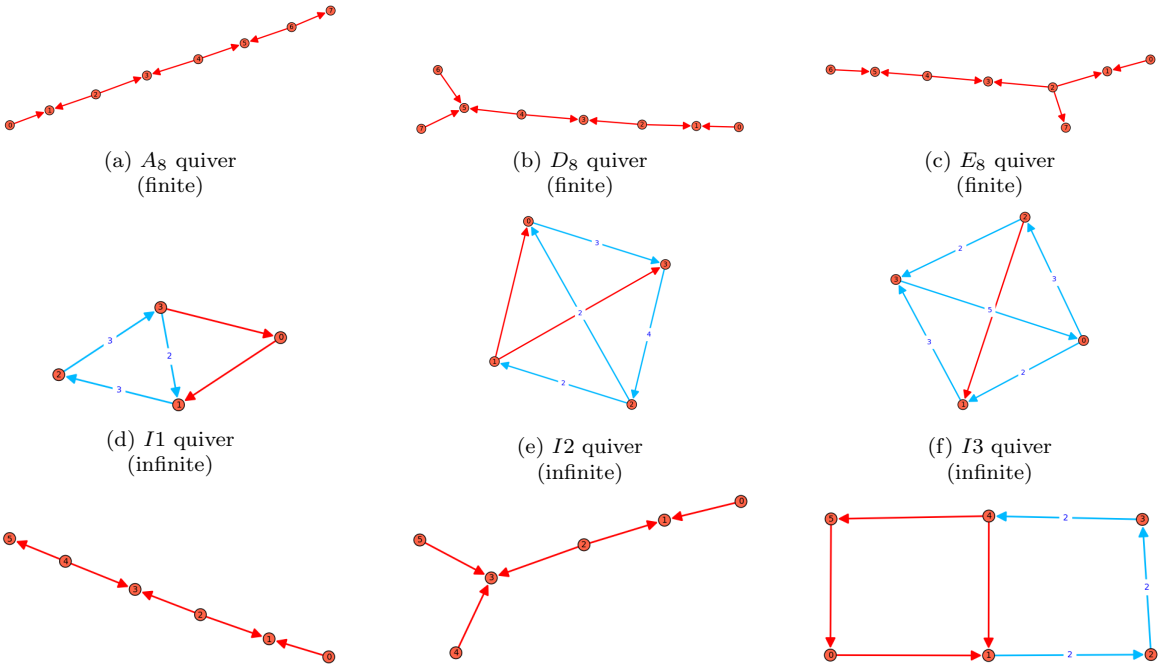
The quivers were represented as adjacency matrices, highlighting here that since the quivers were oriented the matrices were anti-symmetric. Hence an edge $i \mapsto j$ means the quiver adjacency matrix has entries $a_{ij} = 1 = -a_{ji}$. Choosing this antisymmetric representation for adjacency matrices simplified processing of the ranks. Whereby the quiver node ranks were represented by an n-vector, N, whose entries were the corresponding ranks of the nodes listed in the same order as the matrix rows, N_i for node with gauge group $U(N_i)$. With this representation, the vector produced by $a \cdot N \equiv A$, which we call the 'anomaly vector' is preserved under the Seiberg duality operation. In particular, if the quiver gauge theory in consideration was anomaly-free, the anomaly vector was the zero vector, and the dual rank vector remained in the dual adjacency matrix's kernel under Seiberg duality. This we relate to the fact that anomaly-free theories remain anomaly-free under Seiberg duality, as does the style of anomaly in anomalous theories.

These quivers, and their adjacency matrices were generated and manipulated using the `sagemath` package [234]. The rank vectors were inserted manually, and dualised externally to the package. Since no quiver nodes had loops, in all cases the matrix diagonal entries were all zeros. Since this is as many entries as the rank vector, when the rank information was included it was placed in the adjacency matrix as the diagonal entries (efficiently storing the matrix and rank information together in a tensor for the ML architecture to process).

The specific quivers used in generation of the duality-trees / mutation classes for use in the ML investigations discussed here are shown in Figure 3.2.1. These give a range of types, with consistent sizes within the types (so adjacency matrices are the same shape), which produce a sizeable amount of data for the architecture to learn from when generating each duality tree to depth 5.

Six investigations were carried out, classifying: 3 finite type classes, 3 infinite type classes, 2 finite and 1 infinite type classes; each for the case of classification with and without rank information.

In the finite type investigation 3 initial quivers based on the respective A8, D8, and



(a) A_8 quiver (finite)
(b) D_8 quiver (finite)
(c) E_8 quiver (finite)
(d) $I1$ quiver (infinite)
(e) $I2$ quiver (infinite)
(f) $I3$ quiver (infinite)
(g) A_6 quiver (finite)
(h) D_6 quiver (finite)
(i) $I6$ quiver (infinite)

Figure 3.2.1: Quivers used to generate the duality trees via repeated exhaustive application of quiver mutation to all nodes up to depth 5. Each row corresponds to an investigation amongst: (a)-(c) finite types, (d)-(f) infinite types, (g)-(i) mixed types. Quivers are shown with labelled nodes and randomly oriented red directed edges, where blue edges then indicate more than one edge (with the number given by the edge label).

E8 Dynkin diagrams were used to generate the 3 mutation classes for learning. In the infinite type investigation the 3 initial quivers were anomaly-free and chosen to be those denoted $I1, I2,$ & $I3$. Finally, the mixed mutation type investigation used a mix of the two types, notably 2 finite type quivers based on the A6 and D6 Dynkin diagrams, and another anomaly-free infinite type denoted $I6$. Each of these quivers were used to generate a respective mutation class, mutating up to a depth of 5, to contribute to that investigation's dataset. Where rank information was included the 1-vector (a vector with all entries 1) of size equal to the quiver's number of nodes was used as the rank of the initial quiver.

Additional quivers, and hence duality trees, were also considered for further investigations as discussed in [31]. However, focus in this thesis is placed on my contributing work as summarised in the subsequent investigation results.

Example: The NN input was the flattened adjacency matrix of the quiver in question, outputting a 3 vector which dictated the relative probabilities the input quiver was in each of the classes being multiclassified, with the largest / most probable entry taken as the predicted classification. To exemplify the input data for the quiver we consider the adjacency

matrix for the quiver $I1$ given in Figure 3.2.1d. This has adjacency

$$a = \begin{pmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 3 & -2 \\ 0 & -3 & 0 & 3 \\ 1 & 2 & -3 & 0 \end{pmatrix} , \tag{3.2.1}$$

flattened to give the 16 entry input vector, which when an example rank vector $N = [1, 1, 1, 1]$ is used gives the vector input as

$$[1, 1, 0, -1, -1, 1, 3, -2, 0, -3, 1, 3, 1, 2, -3, 1] , \tag{3.2.2}$$

with anomaly vector $A = a \cdot N = [0, 0, 0, 0]$, indicating the $I1$ theory with these ranks is anomaly-free as N is in the kernel of the $I1$ adjacency matrix.

3.2.2 ML Results

The primary scope of the application of ML in this study was to see if the architectures can learn the mutation process, by identifying the duality tree class structure.

The ML problem considered hence mapped a randomly ordered dataset of quiver matrices (either all with or all without rank information) to one of 3 mutation classes which the data was generated from. Each NN's goal was to correctly classify these quiver matrices into their correct mutation class, with learning measures averaged over the 5-fold cross-validation. Each NN consisted of 4 64-neuron layers, with Leaky-ReLU ($\alpha = 0.01$) activation, and dropout (0.2 dropout factor) between them. These layers were followed by a final dense layer with as many neurons as classes (3), and softmax activation. The softmax activation normalises the outputs to probabilities such that the most probable class can be selected as the NN's predicted class for an input quiver matrix. The NNs were trained in batches of 16 over the dataset for 20 epochs, using Adam optimiser to minimise the sparse-categorical cross-entropy loss function.

The results are shown in Table 3.2.1. They show the NNs average accuracy, macro-F1 (mean of the 5 cross-validation F1 scores), and MCC scores for each investigation, with the standard deviation across the 5 NNs trained. These results show strong success for these simple NNs in differentiating between finite type quivers, and between finite and infinite types also. Rank information had negligible impact, but did slightly improve learning in the finite vs infinite type investigation, where information on the anomaly is more important and as expected aided the classification. The NNs did not perform well in the case of classifying infinite types, since these represent anomaly-free physical theories this was disappointing. Results could perhaps be improved with a larger dataset, but overall indicate that finite type quivers have more characteristic mutation under the Seiberg duality process.

Further results in the paper saw accuracies and MCC values reaching 1 for binary classification between and amongst types. Enhancement of the data by including the rank information, or variables inspired by the Diophantine equations associated to the duality trees did not improve learning. There was however reasonable success in identifying the correct duality tree on data at depths beyond that trained on, and the architectures performed very well when classifying against random antisymmetric matrices.

Investigation		Learning Measures		
		Accuracy	Macro-F1	MCC
A8 & D8 & E8 (4801 matrices)	No Rank	0.956 ± 0.004	0.956 ± 0.011	0.935 ± 0.006
	Rank	0.952 ± 0.004	0.951 ± 0.011	0.928 ± 0.006
I1 & I2 & I3 (1357 matrices)	No Rank	0.426 ± 0.016	0.419 ± 0.044	0.145 ± 0.028
	Rank	0.425 ± 0.017	0.420 ± 0.054	0.143 ± 0.026
A6 & D6 & I6 (2660 matrices)	No Rank	0.915 ± 0.004	0.860 ± 0.020	0.850 ± 0.007
	Rank	0.920 ± 0.002	0.870 ± 0.011	0.860 ± 0.003

Table 3.2.1: Results of classification of quiver matrices into their mutation classes. Number of matrices in each investigation's full dataset listed, each chosen such that it is approximately balanced between the classes. Results show the 3 averaged learning measures with standard error for learning on the given quivers with and without rank information included in the matrices.

In these additional investigations (whose results are not listed here) the data was set up such that inputs were concatenated pairs of matrices, with binary label of whether the quiver matrices belonged to the same duality tree or not. This data set-up allowed multiclassification problems to be rephrased as binary classification, and the combinatorics of selecting pairs hugely inflated the dataset for learning. Along with NN architectures, many of the good results used a Naive Bayes architecture. This architecture is built out of Bayes theorem, looking at conditional probabilities of the binary outcomes given the training data information [235], since this reflects the independence between the pairwise data structure it is expedient that this architecture did so well. More information about these investigations can be found in [31].

Conclusion
This study showed success when using NNs to differentiate quivers according to their mutation classes / duality trees. It supports the idea of simple invariants under the mutation process being extractable from the adjacency matrix representation and motivates the further use of ML in the field of quiver and cluster mutation, as performed and summarised in §3.8.

3.3 Hilbert Series (2103.13436)

Hilbert series, as introduced in §2.1, provide a means of counting regular functions on varieties. Physically, they are used in a range of scenarios, including to count BPS operators [67], [68], [236]–[239], determine moduli spaces [240]–[243], find standard model invariants [244]–[246], deal with string compactifications [247], as well as for a range of other quiver-related computations [65], [248]–[254].

In our use we consider them as encoding complex varieties embedded in $\mathbb{P}^n_{\mathbb{C}}$, more specifically encoding the dimensions of the graded pieces of the coordinate ring which defines the variety. In the closed forms the Hilbert series are functions of geometric properties of the variety and the embedding space, as detailed in (2.1.3), (2.1.4), (2.1.5). The cones over these varieties physically correspond to moduli spaces of supersymmetric gauge theories from a

string interpretation. Often one wishes to compute the moduli space from the matter and superpotential information, and in certain (particularly non-toric) scenarios this must be done by counting gauge invariant operators order by order. An example of this process is given for the quiver gauge theory of Figure 2.1.3a at the end of §2.1.

To bypass this lengthy computation we examine the performance of ML techniques in learning the properties of the variety and its embedding, defining the moduli space, directly from the Hilbert series Taylor expansion coefficients.

3.3.1 The Hilbert Series Data

Hilbert series data used for this work was retrieved from the Graded Ring Database (GRDB) [255]–[257], consisting of Hilbert series associated to three-dimensional $\mathbb{Q}$-Fano varieties with Fano index one [258], [259]. Investigations also used generated 'fake' data, created through sampling the parameters of the equations (2.1.4) and (2.1.5) from distributions fitted to the 'real' GRDB data. The distributions are available in the appendices of [32]. The sampling discarded parameter combinations which did not satisfy certain physical conditions[15], encouraging the data to be more representative; and also discarded any repeats of GRDB data.

For both Hilbert series styles of input data mentioned, which we call 'real' and 'fake' respectively, the Hilbert series closed forms were Taylor expanded and expansion coefficients saved. These lists of coefficients form vectors of integers, and make up the ML input for the investigations carried out. Two types of coefficient vector were used as inputs, one from the start of the series (coefficients 0-100) and one from deeper into it (coefficients 1000-1009). The reason to also learn with coefficients sampled from deeper in the series was to provide intuition on the importance of the variety's orbifold points, which have a greater relative significance than the initial part at higher orders.

Output data depended on the investigation. For the regression of embedding weights (each p_i repeated q_i times), these were sorted 3-vectors of integers (sampled in the range [1,10]). For the multi-classification the outputs were single integers in the range [1,5], which were the values of either the variety's Gorenstein index ($\mathcal{J}$) or its dimension (dim) respectively. The final binary classifications just outputted 0 or 1 dependent on whether the Gorenstein (palindromic numerator) or complete intersection properties were respectively satisfied, or whether the data came from the GRDB or was generated by us. Whereas the plethystic logarithm can be used to confirm the complete intersection property, in this work a specific factorisation of the numerator was used in generating complete intersection Hilbert series, as detailed in [32].

[15]Hilbert series closed forms which lead to Taylor expansions with negative coefficients were discarded, as the respective graded pieces cannot have negative dimension. Additionally series with a denominator degree less that or equal to the numerator degree were discarded, since Hilbert series are required to have an infinite expansion.

Example: Considering an example Fano variety (number 11122 in the GRDB) embedded in $\mathbb{P}_{\mathbb{C}}(1^3, 2^2, 3^2)$, it has Hilbert series

$$H(t;\mathcal{V}) = \frac{1-2t^4-2t^5+2t^7+2t^8-t^{12}}{(1-t)^3(1-t^2)^2(1-t^3)^2} \tag{3.3.1}$$

$$= \frac{\tilde{P}(t)}{(1-t^6)^4}\ , \tag{3.3.2}$$

$$\begin{aligned}\text{... for } \tilde{P}(t) = {} & 1+3t+8t^2+18t^3+34t^4+58t^5+89t^6+127t^7+167t^8+203t^9+232t^{10} \\ & +248t^{11}+248t^{12}+232t^{13}+203t^{14}+167t^{15}+127t^{16}+89t^{17}+58t^{18}+34t^{19} \\ & +18t^{20}+8t^{21}+3t^{22}+t^{23}\ ,\end{aligned}$$

in both respective forms of (2.1.4) and (2.1.5). The vector of weights is simply [1,1,1,2,2,3,3] from (3.3.1)'s denominator powers, then from (3.3.2)'s denominator the Gorenstein index is $\mathcal{J} = 6$ and the dimension $dim = 3$. This variety is a 'real' example since it comes from the GRDB, it is Gorenstein as the numerator of (3.3.2) is palindromic, and it is not of the complete intersection form used in this work (described further in [32]). The input vector (in the 0-100 coefficients case shown here) is computed from its Taylor expansion as:

$$\begin{aligned}[1, 3, 8, 18, 34, 58, 93, 139, 199, 275, 368, 480, 614, 770, 951, 1159, 1395, 1661, 1960, 2292, 2660, \\ 3066, 3511, 3997, 4527, 5101, 5722, 6392, 7112, 7884, 8711, 9593, 10533, 11533, 12594, 13718, \\ 14908, 16164, 17489, 18885, 20353, 21895, 23514, 25210, 26986, 28844, 30785, 32811, 34925, \\ 37127, 39420, 41806, 44286, 46862, 49537, 52311, 55187, 58167, 61252, 64444, 67746, 71158, \\ 74683, 78323, 82079, 85953, 89948, 94064, 98304, 102670, 107163, 111785, 116539, 121425, \\ 126446, 131604, 136900, 142336, 147915, 153637, 159505, 165521, 171686, 178002, 184472, \\ 191096, 197877, 204817, 211917, 219179, 226606, 234198, 241958, 249888, 257989, 266263, \\ 274713, 283339, 292144, 301130, 310298]\ .\end{aligned}$$

3.3.2 ML Results

In performing the learning, NN architecture was used for the binary classification of the Gorenstein, complete intersection, and real/fake properties, as well as for multiclassification of the varieties' Gorenstein index and dimension, and for regression of the embedding weights.

The NNs used were built of 4 dense layers of 1024 neurons, all with ReLU activation and 0.05 dropout factor, and final layer reflecting the output tensor size[16]. Training was in batches of 32 for 20 epochs using the Adam optimiser to minimise either the regression log(cosh) loss function or the classification cross-entropy loss function. 5-fold cross-validation provided confidence on the metrics used to evaluate the learning via averaging and standard error.

To analyse the binary classification ML results, the unsupervised technique of PCA (with linear kernel) was also implemented to examine whether there was simple linear structure

[16]For binary classification the final layer had 2 neurons with softmax activation, for multiclassification there were 5 neurons also with softmax for each possible $\mathcal{J}$ / dim value, for regression there were 3 neurons with no activation for the 3 weights.

Investigation		Embedding weights	Gorenstein index	Dimension	Gorenstein property	Complete Intersection	GRDB
Output Ranges		3 x [1,10]	[1, 5]	[1, 5]	Binary	Binary	Binary
Measure		MAE	MCC	MCC	MCC	MCC	MCC
Orders	0 - 100	1.94 ± 0.11	0.916 ± 0.010	0.993 ± 0.006	0.717 ± 0.155	0.910 ± 0.022	0.717 ± 0.155
	1000 - 1009	1.04 ± 0.12	0.727 ± 0.022	0.822 ± 0.031	0.919 ± 0.073	-	0.919 ± 0.073

Table 3.3.1: ML results for each of the Hilbert series investigations. NNs learnt geometric properties associated to an algebraic variety from coefficients in their Hilbert series expansions. Regression was assessed with mean absolute error (MAE) and classification with Matthew's correlation coefficient (MCC), training using 5-fold cross-validation to provide a standard error on the averaged measures.

in the coefficient vectors which the NNs could take advantage of in the learning. The 0-100 coefficient vectors produced PCA eigenvectors of size 101, hence to enable visualisation of these components only the two most significant principal components were plotted (i.e. projections of the coefficient vectors onto the first two eigenvectors) to provide a means for examining the data's clustering.

NN Results

In the regression investigation, MAE evaluated learning by computing the average absolute difference between each true embedding weight and the NN predicted embedding weight. This measure evaluates in the range $[0, \infty)$ with 0 indicating perfect learning, where the embedding weights are always correctly predicted. In the classification investigations, MCC evaluated the learning. This measure evaluates in the range $[-1, 1]$ with 1 indicating perfect learning.

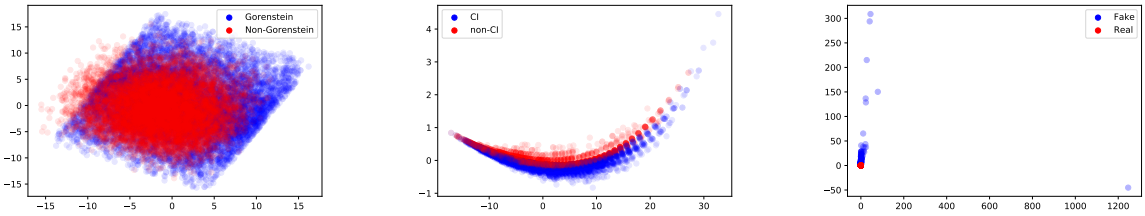
ML results are provided in Table 3.3.1 for each of these 5 investigations. Learning of the varieties' ambient space embedding weights predicted each weight within an average range of 1 from the correct values when using higher order coefficients. Additionally the geometric parameters of equation (2.1.5) were learnt exceptionally well, but now better from the lower order coefficients.

Each of the binary classifications also evaluated with MCC values exceeding 0.9, demonstrating strong learning. Both the Gorenstein and 'fake' (non-GRDB) properties were better identified with higher order coefficients, whilst the complete intersection property was only learnt from lower orders but performed equally well.

PCA Results

Whilst the learning so far uses NNs from supervised ML, PCA provides another avenue for analysis of this data from the conjugate field of unsupervised ML. These PCA plots can be seen for the three binary classification investigations in Figure 3.3.1.

Here, 2-dimensional PCA was performed for each of the binary classification investigations, and the datapoints in each class plotted separately in different colours. The largest 2 components respectively represented proportions $\sim 0.6, 0.98, 0.94$ of the variances for the Gorenstein, complete intersection, and real/fake investigations, generally motivating 2-dimensional plotting. Whilst there is no clear separation for the Gorenstein data, the complete intersection and 'real'/'fake' data do show some separation, indicating there is some



(a) Gorenstein (b) Complete Intersection (c) 'real' / 'fake'

Figure 3.3.1: 2d principal component analysis plots of the Hilbert series 0-100 coefficient data, labelled by each of the binary classification investigations respectively.

linear structure that the NNs can take advantage of for classification. In particular the occurrence of this for the 'fake' data demonstrates the difficulty of generating representative Hilbert series.

Conclusion

The learning results demonstrate the success of NNs in learning geometric properties of a variety from Hilbert series. Their application allows extraction of information about a supersymmetric gauge theory's moduli space (for analysis of the theory) from counts of the numbers of single-trace BPS operators at lower orders, effectively allowing extrapolation to the counts for higher orders without direct computation.

The PCA results indicate that the complete intersection property can be somewhat identified from linear structure in the coefficient data, however the Gorenstein property is much harder to extract and hence learn. Finally the generation of representative 'fake' data is particularly difficult, and techniques beyond direct modelling and sampling of the GRDB data are motivated for further investigation.

3.4 Amoebae (2106.03695)

Amoebae, as introduced in §2.2 find their primary uses in tropical geometry, with a fortuitous connection to brane webs as duals of toric diagrams through their Newton polynomial Riemann surfaces. The central focus of the work in [33], examined methods for ML amoebae genus from the abstract space of amoebae coefficients; sampling the vectors of n coefficients from varying bounds within the lattices: $(\mathbb{Z}^+)^n$, $(\mathbb{R}^+)^n$, $\mathbb{R}^n$. Balanced datasets were created for NNs to multiclassify amongst all the possible genus values for each of the amoeba considered. These amoeba included the prototypical F_0, but also $L^{3,3,2}$, $\mathcal{C}/(\mathbb{Z}_2 \times \mathbb{Z}_4)$, and $K^{4,5,3,2}$ which had many more interior and boundary points respectively. Accuracies often exceeded 0.9, performing better on the more restricted lattices, and also on the smaller toric diagrams (and hence smaller coefficient vectors). Different nonlinear projection methods (such as multidimensional scaling and isomap from the `yellowbrick` library [260]) could reproduce the boundary conditions in the coefficient space where the genus changed, agreeing with derived boundaries from lopsidedness arguments.

However, my contribution focused on assessing the success of ML on the amoebae images directly. For these investigations the F_0 Riemann surface was used to generate amoebae,

as the prototypical example shown in Figure 2.2.2, with the database consisting of point-sampling generated images of $n = 1$ lopsided amoebae.

CNNs find their most common uses in image related tasks, due to their convolutional action over the local structure of the images. Since this is the behaviour used in collecting sampled points into a full amoeba, this ML architecture was the most natural choice. To fully investigate their success in identifying genus from these images, varying image resolutions were used for the CNN input. Determining how the learning measures varied as the amoebae images varied in quality was the main focus of this investigation.

3.4.1 The Amoeba Image Data

In this work, the database consisted of point-sampling generated *images* of the F_0 amoebae for varying coefficients such that the full database consisted of 1000 genus 0 amoebae, and 1000 genus 1 amoebae. The amoebae genus, g, labelling used the $n = 1$ lopsided amoeba approximation

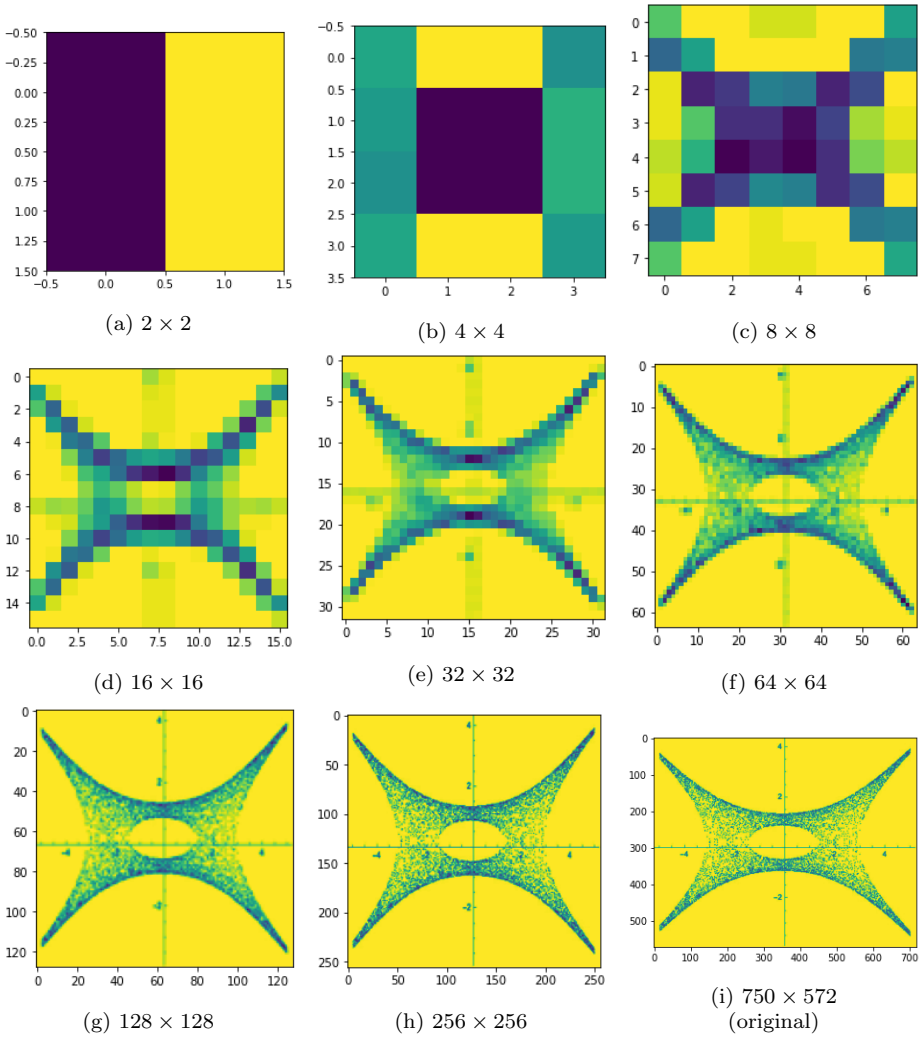
$$g = \begin{cases} 0, & |c_5| \leq 2|c_1 c_3|^{1/2} + 2|c_2 c_4|^{1/2} \\ 1, & |c_5| > 2|c_1 c_3|^{1/2} + 2|c_2 c_4|^{1/2} \end{cases} , \tag{3.4.1}$$

for the F_0 amoebae with Newton Polynomial: $P(z_1, z_2) = c_1 z_1 + c_2 z_2 + c_3 z_1^{-1} + c_4 z_2^{-1} + c_5$. Here the genus condition is analytically derived, using the asymptotic behaviour of the amoebae spines to identify their intersection point – the amoeba centre. Since the F_0 case has at most one hole, due to the toric diagram polygon's singular interior point, identifying the lopsidedness behaviour at the centre dictates whether the centre is contained within the amoebae, thus giving the amoebae genus.

Point-sampling image generation is commonly used for more complicated amoebae where analytical expressions are hard to compute and then plot. In this process the analytic bounds between an amoeba and its complement in the $\mathbb{R}^2$ space are had to determine through projection of the Riemann surface equation. Instead the point-sampling method is implemented whereby points are randomly sampled from the complex surface and each point projected down onto the amoeba. This sample of amoeba points in $\mathbb{R}^2$ can then be used to stochastically infer the amoeba bounds, and in particular the existence of hole to determine the amoeba genus. Therefore examining the success of a CNN in identifying genus from these plots is a particularly relevant task. We now undertake this task of the CNN ML of the labelled dataset

$$\text{Image}(\mathcal{A}_{P(F_0)}) \longrightarrow \text{genus } \{0 \text{ or } 1\} .$$

The amoebae images in the dataset are of varying shape, to create consistency across the dataset (as needed for the CNN tensor inputs), the images are resized such that there is always an equal number of pixels in each dimension. The images contained the real plane axes since the identification of a plot's origin is useful in determining genus for F0 amoebae by eye, hence giving the CNN realistic information for its learning. Images are reformatted to greyscale such that inclusion in the amoeba is the only relevant data at each resolution. The resolutions learnt on varied logarithmically with base 2, choosing this base such that the computational learning algorithms worked most efficiently.



(a) 2×2 (b) 4×4 (c) 8×8

(d) 16×16 (e) 32×32 (f) 64×64

(g) 128×128 (h) 256×256 (i) 750×572 (original)

Figure 3.4.1: An example F_0 amoeba image of genus 1 at varying resolutions. Each subcaption denotes the respective number of pixels in the resizing, the original image is reformatted to a square shape for consistency across all amoebae.

Example: An example of a genus 1 amoeba from the dataset is given in Figure 3.4.1, this amoebae had randomly generated $\mathbb{R}^+$ coefficients:

$$(c_1, c_2, c_3, c_4, c_5) \sim \{0.6104, 1.8940, 0.4989, 2.9777, 6.9871\} \, . \tag{3.4.2}$$

In the figure the image at the varying resolutions considered is shown, the reshaping to square from the original is clear, and the binary nature of the data is emphasised by the greyscaling (here plotted with a blue-yellow colour scheme).

3.4.2 ML Results

The CNNs trained had a consistent layer structure: 3 2-dimensional convolutional layers of size matching the image input size, each followed by a 2-dimensional max-pooling layer and a dropout layer (factor 0.01); following these is a flattening layer, then a dense layer of size

Learning Measures	Image Resolution							
	2×2	4×4	8×8	16×16	32×32	64×64	128×128	256×256
Accuracy	0.484	0.485	0.746	0.972	0.987	0.986	0.803	0.754
	0.005	0.006	0.064	0.008	0.005	0.004	0.116	0.055
MCC	nan	nan	0.620*	0.944	0.974	0.971	0.639	0.646*
	nan	nan	0.012*	0.017	0.010	0.008	0.212	0.032*

Table 3.4.1: Learning results for CNN binary classification of F_0 $n = 1$ lopsided amoebae images, determining genus 0 or 1 over balanced dataset of 2000 images at varying image resolutions. Resolution is given in terms of the number of pixels. Measures are averaged accuracies and MCCs over the 5-fold cross-validations runs, with standard errors. MCCs calculated over less than 5 of the cross-validations are denoted with a '$*$', for those completely incalculable 'nan' is given.

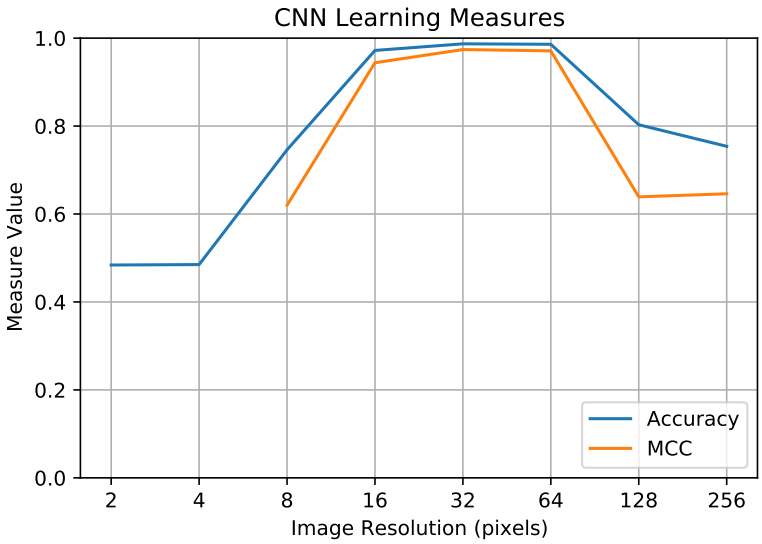


Figure 3.4.2: Averaged accuracy and MCC learning measures for the CNNs trained over F_0 amoebae images at varying resolutions (denoted by number of pixels in one dimension of the square images). MCC values were incalculable at certain resolutions, and may be artificially high at resolutions $\{8, 256\}$.

equal to the number of pixels in one dimension of the input image, and then a final output dense layer with 1 neuron. All layers used a LeakyReLU activation (with $\alpha = 0.01$), except the final output Dense layer which used sigmoid activation to better map to the required binary classification. The convolutional and max-pooling layers used the 'same' padding regime, and the convolutional layers used a 3×3 kernel size.

The CNN architecture used for the investigation at each image resolution is trained over 20 epochs of the data on batches of 32, in a 5-fold cross-validation scheme. The Adam optimiser is used to minimise the binary cross-entropy loss function for predicting genus 0 or genus 1 respectively. The learning is measured using the metrics: accuracy and MCC, which are averaged over the 5 runs in the 5-fold cross-validation.

CNN Results

Results for the CNN ML over varying image resolutions are given in Table 3.4.1. The

accuracies increase as image resolution improves up to some optimal value, around 32×32 pixels where it then falls off; this behaviour is further shown in Figure 3.4.2. The high accuracies at optimal resolution (> 0.98) show the success of CNNs in identifying genus from point-sampling generated amoebae images.

The change in accuracy as image resolution decreases from the largest size considered may be explained by initial resolution loss causing averaging over the generated points to produce a connected amoeba component with a clearer hole structure in genus 1 amoebae (Figure 3.4.1e). Then further decrease in resolution loses more of the image information combining any holes into the amoebae themselves until there is no sensible information in the image remaining (Figure 3.4.1a).

Similar behaviour can be seen over the MCC values where calculable. Although MCC as a measure is less susceptible to bias in the data making it generally preferential, there are issues of incalculability where learning fails and the same class is predicted for all test data. Where this happened for some of the 5 runs in the 5-fold cross-validation, the MCC average and standard error is calculated over the calculable values, and are denoted with a '$*$' (note this inflates the measures' value as the cases of no learning are ignored). Where the MCC is incalculable for all runs an averaged MCC is then incalculable also, these scenarios are denoted with 'nan' in the results in Table 3.4.1.

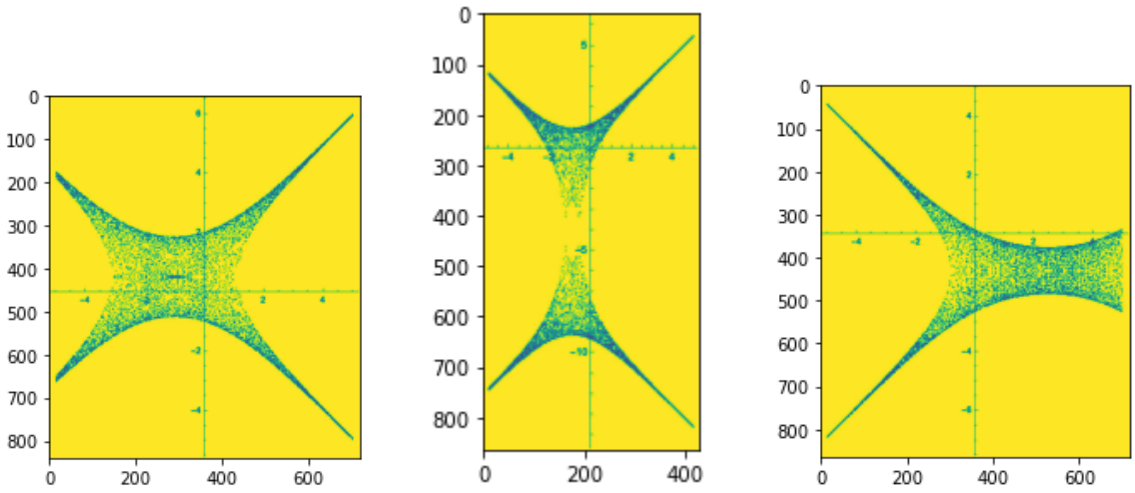
CNN Misclassifications

Examining the image resolution 32×32 which led to the most successful CNN classification, the misclassifications where the CNN disagreed with the images labelled genus provide interesting further insight into the learning. In one of the 5-fold cross-validation runs 3 of the 400 images in the testing dataset were misclassified such that the predicted amoeba genus differed from the labelled genus. These images are shown at full resolution as well as the 32×32 resolution in Figure 3.4.3.

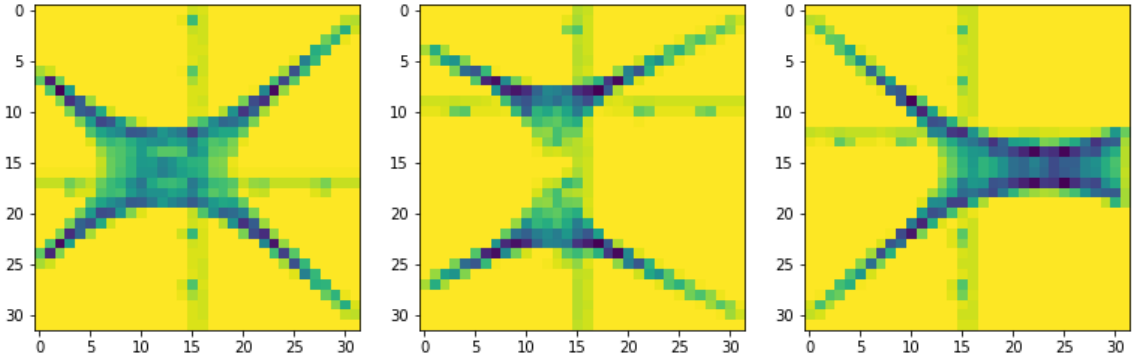
The first image misclassified, shown in full and lowered resolutions in Figures 3.4.3a & 3.4.3d respectively, is labelled as genus 0, but misclassified to genus 1. The original image shows the true genus 0 amoeba, however the sampling of amoeba points shows a particularly low density in the amoeba's centre, this leads to a poorer collation of points as the resolution drops, shown by fainter parts of the amoebas centre in the lower resolution image perhaps misleading the CNN to predict a non-zero genus.

The second misclassified image, shown in full and lowered resolutions in Figures 3.4.3b & 3.4.3e respectively, is labelled as genus 1, but misclassified to genus 0. In the original image in particular, the issues with the point-sampling image generation become clear, the amoeba looks disconnected into two parts, where the hole in the centre dominates the amoeba. This leads to part of the hole's boundary being lost in the lower resolution image, making the amoeba appear to be genus 0, misleading the CNN.

The third and final misclassified image in this run, shown in full and lowered resolutions in Figures 3.4.3c & 3.4.3f respectively, is also labelled as genus 1, but misclassified to genus 0. Here the original image shows a genus 0 amoeba, interestingly this scenario is a rare occurrence where the $n = 1$ lopsided amoeba (used to produce the genus labels for the dataset as in 3.4.1) has a different genus to the true amoeba. Therefore although this is considered by the CNN as a misclassification, the CNN has actually managed to predict



(a) Case 1: 840 × 720 (b) Case 2: 864 × 426 (c) Case 3: 864 × 716



(d) Case 1: 32 × 32 (e) Case 2: 32 × 32 (f) Case 3: 32 × 32

Figure 3.4.3: Example CNN misclassifications during model testing, images show the original image and the image at the 32 × 32 resolution considered. Case 1, (images a & d) were labelled as genus 0 with the CNN predicting genus 1; cases 2 & 3 (images b,e & c,f) were labelled genus 1 with CNN predictions of genus 0.

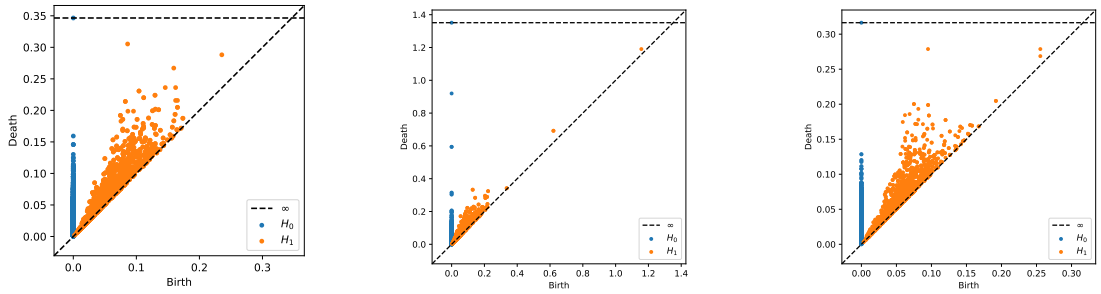
the true amoeba genus from the image, whilst at the same time highlight a case where the match-up to lopsided amoebae fails.

TDA Genus Identification

TDA provides an alternative technique for describing the topology of data manifolds. Within this field, identifying genus of surfaces is often well performed by the technique of persistent homology as described in §2.3.1.2.

In the persistent homology computation of this image classification problem, the 2-dimensional sampled amoeba points would have 2-dimensional discs of radius δ drawn about each of them, with δ varied to build a filtration of VR complexes from the disc intersections. Emphasising here that TDA is performed on the point-sampling generated amoebae directly and not on the varying resolution images used by the CNNs.

To identify the genus, one must analyse the occurrence of 2-dimensional holes; the H_1 homology should then have persistent features for each hole contributing to the genus count. For these examples where genus is 1 there should hence be a single persistent feature which



(a) Case 1 (b) Case 2 (c) Case 3

Figure 3.4.4: Persistence diagrams for the 3 misclassification cases considered, showing the H_0 and H_1 persistent homology of the point-sampled amoebae points.

is far from the persistence diagram diagonal, corresponding to a hole whose boundary is connected quickly and fills in much later. Conversely for 0 genus there should be no such feature.

Additionally, the H_0 homology keeps track of the number of connected components in the complex, for well sampled amoeba points all the features should hence 'die' quickly as all the amoeba points become connected to nearby points and form the amoeba as a single component. The persistence diagrams for each of these three misclassification cases considered are plotted in Figure 3.4.4.

The H_0 homology of cases 1 and 3 show the amoeba is connected quickly, with no gaps in the H_0 feature line. However for case 2 there are gaps in the line, indicating the sampling is poor and the amoeba appears as disconnected components, agreeing with previous analysis.

For the H_1 homology all cases have no significant features far from the diagonal, indicative of a 0 genus amoebae. Any significant features should be a distance from the diagonal comparable to the scale of the amoeba. Since the amoeba centres have size ~ 2 and all H_1 features die for $r \lesssim 0.3$, despite there being some points slightly further from the diagonal they are not persistent enough to indicate a significant hole and genus 1. For case 1 the persistent homology now correctly predicts the genus 0, outperforming the CNN. However for case 2 where the genus is truly 1 the homology prediction is incorrect, likely due to the same reasons as the CNN, that poor sampling means the case 2 boundary isn't connected properly. Finally for case 3 the true amoeba genus of 1 is correctly identified (as for the CNN) where the lopsided approximation breaks down giving an incorrect label of genus 1.

Conclusion

The CNN learning of the amoeba genus directly from point-sampling generated amoeba images performed better as image resolution was dropped to average over the points. The exceptionally high classification performance supports the use of ML in determining genera of amoeba in an automated fashion. Applying persistent homology to the 3 misclassified images produces H_0 and H_1 data which corroborated these CNN results. However, the persistent homology technique did perform better with case 1, correctly identifying the genus of 0.

These three misclassification examples from the run analysed demonstrate the subtleties of this point-sampling image generation for amoeba, as well as the issues with relying on the lopsided amoeba approximation for genus prediction.

3.5 Polytopes (2109.09602)

As discussed in §2.2, lattice polytopes can be used to construct toric varieties, within which CY hypersurfaces can be taken. In particular if the lattice polytope, Δ, contains the origin in its strict interior and all the vertices, $\mathbf{p}_i$, of the polytope are primitive such that $gcd(\mathbf{p}_i) = \pm 1 \ \forall$ vertices, then the toric variety is Fano. They are further called canonical Fano if the origin is the only interior point. Note here that the Fano condition is slightly more general than the reflexive condition which also requires the bounding facets to be a distance 1 from the origin[17] [257], [261]–[264].

In the work [34] summarised here, properties of lattice polytopes and their respective toric varieties in dimensions 2 and 3 were learnt. In dimension 3 the polyhedra considered generated canonical Fano 3-folds and NN regressors learnt the polytope volume, dual polytope volume, and reflexivity. The volume was learnt to an MAE of 1.68, dual volume to 2.59, and reflexivity to accuracy 0.81. However the focus of the my contribution was the ML of polygons in dimension 2.

3.5.1 The Polygon Data

In general there is a large redundancy in the matrix representation of lattice polytopes. Any polytope is traditionally represented by a matrix of its vertices $M_{ij} = (\mathbf{p}_i)_j$, where i indexes the lattice dimension and j indexes the vertices. However any rotation or reflection of the polytope which keeps it on the integer lattice clearly preserves the polytope structure. In fact, this generalises further, and any $GL(n, \mathbb{Z})$ preserves the combinatorics such that it generates the same n-dimensional toric variety [265].

However the redundancy of this $GL(n, \mathbb{Z})$ action can be avoided by instead representing the polytopes by their Plücker coordinates. Plücker coordinates are the maximal minors of the integer kernel of the vertex matrix, and are naturally $GL(n, \mathbb{Z})$ invariant. They hence vastly reduce the redundancy in representation, except for the ordering of these Plücker coordinates in the NN vector inputs due to the reordering of the vertices in the matrix, a redundancy taken advantage of for data augmentation. Further to this, to ensure the Plücker coordinates uniquely determine the polytope, only polytopes whose vertices generate the integer lattice $\mathbb{Z}^n$ are considered (avoiding the need to consider quotient gradings) [266], [267].

The properties machine learnt in this work included the polytope volume, defined in a dimensionally covariant way for an n-simplex as $Vol(\Delta_n) = \frac{1}{n!}\prod_{i=1}^{n} h_i$ [268], such that the polytope volume $Vol(\Delta)$ is then the sum over the volumes of the simplices that make up any triangulation of Δ multiplied by $n!$ to normalise it. The h_i values are defined for any ordering of the $(n+1)$ n-dimensional n-simplex vertices $((\mathbf{p}_i)_1, (\mathbf{p}_i)_2, ..., (\mathbf{p}_i)_{n+1})$ such that h_1 is the distance between $(\mathbf{p}_i)_1$ and $(\mathbf{p}_i)_2$, h_2 is the height of $(\mathbf{p}_i)_3$ above the line connecting $(\mathbf{p}_i)_1$ and $(\mathbf{p}_i)_2$, h_3 is the height of $(\mathbf{p}_i)_4$ above the plane containing $(\mathbf{p}_i)_1$, $(\mathbf{p}_i)_2$, and $(\mathbf{p}_i)_3$, etc. Defined equivalently for dual volume as the volume of the polytope's dual $\overline{\Delta}$.

[17] This distance 1 idea comes from the intersecting hyperplane definition of a lattice polytope $\Delta = \{\mathbf{p}_i \mid \mathbf{A} \cdot \mathbf{p} \geq \mathbf{c}\}$, for integer matrix $\mathbf{A}$ made up of primitive elements of the dual integer lattice, and integer vector $\mathbf{c}$. For interior point $\tilde{\mathbf{p}}_i$ the reflexivity condition is then $\mathbf{A} \cdot \tilde{\mathbf{p}} - \mathbf{c} = \mathbb{1}$, i.e. translating the polytope such that the interior point is the origin (setting $\tilde{\mathbf{p}} = 0$) then simply means $\mathbf{c} = -\mathbb{1}$.

Whilst these two properties are for the specific polytope under consideration, two properties of the respective toric variety were also learnt. The first was the Gorenstein index $\mathcal{J}$, as studied in §3.3, which in the polytope context is the smallest positive integer $\mathcal{J}$ such that the $\mathcal{J}$-dilation of the polytope's dual is lattice. Note this is trivially 1 where the polytope is reflexive (such that the dual is automatically lattice), and is more useful for non-reflexive polytopes whose duals have vertices in $\mathbb{Q}^n$. Codimension, C, was also considered as the dimensional difference between the toric variety and the weighted projective space it is embedded in. This is technically computed from the size of the Hilbert basis (HB) over the lattice points in the cone of the dual polytope when embedded in the $z_n = 1$ hyperplane in $\mathbb{R}^{n+1}$, which is equivalent to the number of points in and on the dual polytope. To get the codimension $C = |HB| - dim(\Delta) - 1$.

The specific database of polytopes considered were of dimension 2, these polygons consisted of $m \in \{3, 4, 5, 6\}$ vertices with $\{277, 7041, 16637, 3003\}$ respectively generated, such that there was a total of 26958 polygons. For each polygon the properties: $\{Vol(\Delta), Vol\big(\overline{\Delta}\big), \mathcal{J}, C\}$ were computed, with distributions in the respective ranges $[3, 514], (0.21, 15.37), [1, 30], [2, 42]$. Note that the properties were all integer-valued except for dual volume which was rational, and since there are only 16 reflexive polygons reflexivity wasn't learnt.

For each polygon with m vertices there are $\binom{m}{m-2}$ Plücker coordinates with a redundant ordering. Therefore each investigation focused on a specific m and used 3 distinct permutations of the Plücker coordinates to augment the data. This led to 80874 input-output pairs for the NN architectures to learn: Plücker(Δ) $\longmapsto$ Property(Δ).

Example: An example of a lattice polygon and its dual is given in Figure 3.5.1. The dual polytope is also lattice, such that both polytopes are reflexive with Gorenstein index 1. Since both also only have the origin in their strict interior they are both also canonical Fano. Also since the original polygon (a) has no lattice points on its facets (i.e. edges in 2-dimensions), the variety is smooth, however the dual (b) has many boundary lattice points so is not, these singularities would need resolution for use in physics.

To exemplify computation of the learnt parameters in the ML investigations, we compute each of $\{Vol(\Delta), Vol\big(\overline{\Delta}\big), \mathcal{J}, C\}$ for the original lattice polytope in (a).

• $Vol(\Delta)$: To compute volume the polytope must be first triangulated into simplices, as shown also in (c). Each of the triangles (2-simplices) in this triangulation can have the origin and one of the top right two points selected as $(\mathbf{p}_i)_1$ and $(\mathbf{p}_i)_2$ to make $h_1 = 1$, then the final point $(\mathbf{p}_i)_3$ is a perpendicular distance of $h_2 = 1$ away in each case. Therefore each simplex volume is $\frac{1}{2!}\prod_{i=1}^{2} h_i = \frac{1}{2}(1 \cdot 1) = \frac{1}{2}$. The total polytope volume is then $(2!)\sum_{simplices} Vol(\Delta_n) = 2 \cdot (\frac{1}{2} + \frac{1}{2} + \frac{1}{2}) = 3$.

• $Vol\big(\overline{\Delta}\big)$: The computation of the dual volume is the equivalent process on the dual polytope, triangulated in (d). Within this triangulation the two lower-left-most triangles can have the longest (external) side selected to be $h_1 = 3$, and the origin point is then $(\mathbf{p}_i)_3$ in both cases and a perpendicular distance of $h_2 = 1$ away. Therefore for these two simplices the volume is $\frac{1}{2!}\prod_{i=1}^{2} h_i = \frac{1}{2}(3 \cdot 1) = \frac{3}{2}$. For the final upper-right triangle, taking the longest side as $h_1 = \sqrt{3^2 + 3^2} = 3\sqrt{2}$, then the origin as $(\mathbf{p}_i)_3$ is a perpendicular distance of $h_2 = \sqrt{\frac{1^2}{2}} = \frac{1}{\sqrt{2}}$ as half the diagonal of a lattice grid. This final simplex hence has volume $\frac{1}{2!}\prod_{i=1}^{2} h_i =$

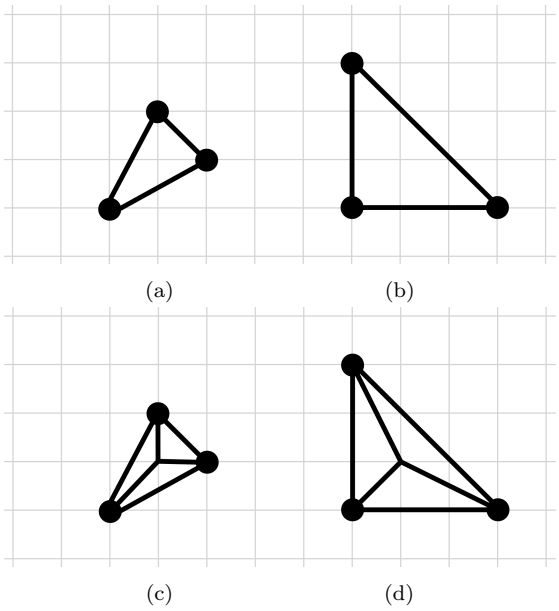


Figure 3.5.1: An example of a lattice polygon (a) with vertices $\{(1,0),(0,1),(-1,-1)\}$, corresponding to the compact del Pezzo surface $dP_0 = \mathbb{C}^3/\mathbb{Z}_3$ where identification is with weights (1,1,1) for the $\mathbb{Z}_3$ such that it is the familiar $\mathbb{P}^2_{\mathbb{C}}$; and its dual polytope (b) with vertices $\{(2,-1),(-1,2),(-1,-1)\}$, corresponding to the surface $\mathbb{C}^3/\mathbb{Z}_3 \times \mathbb{Z}_3$ where identification is with weights (1,0,2), (0,1,2) for each $\mathbb{Z}_3$ respectively. Below shows the triangulations of these polygons ((c), (d) respectively) using the single interior point, into 2-simplices for volume computation.

$\frac{1}{2}(3\sqrt{2}\cdot\frac{1}{\sqrt{2}}) = \frac{3}{2}$. The overall polytope volume is therefore $(2!)\sum_{simplices} Vol(\Delta_n) = 2\cdot(\frac{3}{2} + \frac{3}{2} + \frac{3}{2}) = 9$.

• $\mathcal{J}$: As stated before since the polytope is reflexive the dual polytope is already lattice and hence needs no dilation, making $\mathcal{J} = 1$.

• C : To compute the codimension the cardinality of the Hilbert basis is first needed, which can be found from the number of lattice points on and in the dual polytope, for the polytope (a) the dual (b) has 10 points (1 interior, 6 on the edges, and 3 vertices) making this Hilbert basis size 10. The codimension is then equal to: $|HB| - dim(\Delta) - 1 = 10 - 2 - 1 = 7$.

For completeness, the dual polytope in (b) has parameters: $\{Vol(\Delta), Vol(\overline{\Delta}),\ \mathcal{J}, C\} = \{9,3,1,1\}$. Since it is the dual the volume and dual volume are switched. As it is still reflexive the dual is lattice so the Gorenstein index is 1. Finally the codimension is computed using the size of the Hilbert basis as the number of points on and in (a) = 4 (1 interior, 0 boundary, and 3 vertices), such that $C = 4 - 2 - 1 = 1$.

For the original polygon in (a) its Plücker coordinates can be computed from its vertex matrix

$$V_{(a)} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix} \implies ker(V_{(a)}) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \tag{3.5.1}$$

giving a 1-dimensional kernel, geometrically indicating the proportions to add the vertex coordinates in to produce the origin. Since this polygon has 3 vertices there are $\binom{3}{1} = 3$ Plücker coordinates, which are each of the 3 1×1 minors of the kernel, trivially just the entries. Therefore the example polytope in (a) has Plücker coordinates (1,1,1).

The dual polytope in (b), can also have it Plücker coordinates computed from the vertex matrix as

$$V_{(b)} = \begin{pmatrix} 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix} \implies ker(V_{(b)}) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \tag{3.5.2}$$

Again there are $\binom{3}{1} = 3$ Plücker coordinates, which are trivially again just (1,1,1). Here both the polytope and its dual have the same kernel, and for (a) the kernel is the same as the weights used in the identification to define the variety [93], [269]. However since the polygon's vertices in (b) do not span the $\mathbb{Z}^2$ lattice, additional quotient gradings are needed to define the variety, explaining the different weights [266].

3.5.2 ML Results

NN regressors with 4 layers of 64 neurons and Leaky-ReLU activation ($\alpha = 0.01$) and then a single output neuron were trained with Adam optimiser in batches of 32 for 20 epochs to minimise a log(cosh) loss function. Performance was measured with MAE as a regression measure, and also accuracy bins were used to determine what proportion of the test data was predicted to be within a specific range of the true value. These bin sizes were either 1 (i.e. can it predict the true value correctly to the nearest integer), or a proportion of the range of the output parameter learnt across the dataset (proportions being 0.025 or 0.05 to be within 5% or 10% of the true value).

As a first test volume was learnt for the polygons (stratified by number of vertices) from the vertex matrix input (flattened) and the Plücker coordinates input. As shown from the results in Table 3.5.1, the Plücker coordinate representation outperformed the vertex representation every time. This motivates the use of Plücker coordinates in the following investigations, with superior success thought to be attributed to both the Plücker representation and parameter values being preserved under the rotational/reflectional invariance of the polytope. Additionally, both volume and Plücker coordinates come from determinant calculations on the vertex matrix, so it is not surprising they are intimately linked.

ML results for the parameter properties considered are given in Table 3.5.2, where the volume learning results from Plücker coordinates are repeated from Table 3.5.1 . The ranges used in calculating the accuracy bin widths are: 511, 15.16, 29, 40; for volume, dual volume, Gorenstein index, and codimension respectively. The results show that volume can be learnt especially well for all number of vertices. In addition dual volume can be learnt well, and codimension with some success. Conversely Gorenstein index could not be learnt with this NN architecture.

Whereas volume is a determinant and likely a simpler function of Plücker coordinates to learn, hence the good performance. Dual volume is less simple to compute and surprisingly still is learnt well, hinting at some simpler function that connects these inputs and outputs that shortcuts computation of the dual polytope. The lesser performance for codimension and Gorenstein index indicate they are likely incredibly complex, if not even non-existent, functions of the Plücker coordinates (and all performed even worse when using the vertices as input). In general, learning is more successful for lower numbers of vertices, perhaps due to the smaller Plücker vector inputs, and the larger datasets available for these polygons.

Number of Vertices	Representation	MAE	Accuracy		
			± 0.5	$\pm 0.025 \times$ range	$\pm 0.05 \times$ range
3	Vertices	4.941	0.302	0.891	0.945
	Plücker	0.209	0.827	1.000	1.000
4	Vertices	10.012	0.072	0.891	0.945
	Plücker	0.625	1.000	1.000	1.000
5	Vertices	8.640	0.060	0.777	0.926
	Plücker	1.051	0.451	1.000	1.000
6	Vertices	14.947	0.036	0.603	0.826
	Plücker	3.359	0.139	0.969	0.997

Table 3.5.1: ML polygon volume from polygon representations: flattened list of vertices, or Plücker coordinates. Investigations carried out for each subset of polygons with each number of vertices. Learning is measured using MAEs, and accuracies (to 3d.p.) of test set predicted volumes being within some bin centred on the true value. The first bin has width 1, the second and third have widths of 5% or 10% $\times$ range respectively, where "range" is the difference between the maximum and minimum volumes in the full dataset (for volume the range is 511).

Property	Number of Vertices	MAE	Accuracy		
			± 0.5	$\pm 0.025 \times$ range	$\pm 0.05 \times$ range
Volume $Vol(\Delta)$	3	0.209	0.826	1.000	1.000
	4	0.615	0.625	1.000	1.000
	5	1.051	0.452	1.000	1.000
	6	3.359	0.139	0.969	0.997
Dual Volume $Vol(\overline{\Delta})$	3	1.181	0.370	0.370	0.501
	4	0.642	0.634	0.634	0.754
	5	0.818	0.496	0.496	0.638
	6	0.941	0.405	0.405	0.557
Gorenstein index $\mathcal{J}$	3	5.710	0.039	0.064	0.132
	4	5.002	0.069	0.101	0.196
	5	4.632	0.071	0.102	0.202
	6	5.343	0.056	0.080	0.150
Codimension C	3	1.897	0.192	0.361	0.615
	4	2.726	0.140	0.268	0.496
	5	3.182	0.103	0.210	0.404
	6	2.884	0.126	0.251	0.468

Table 3.5.2: ML results for each of the properties considered, using input Plücker coordinates representing the polygons in each subset of the full dataset based on the number of polygon vertices. Learning is measured with MAE and accuracies based on test set predictions being within some bin centred on the true value. The bin widths are based on the range of values each property can take in the full dataset. Results show volume and dual volume learnt well, codimension learnt to some extent, and Gorenstein index could not be learnt. For reference the respective ranges are: 511, 15, 29, 40 for volume, dual volume, Gorenstein index, and codimension respectively.

Presumably differing dataset sizes also explains the occasional lower performances learning with triangles compared to quadrilaterals and pentagons.

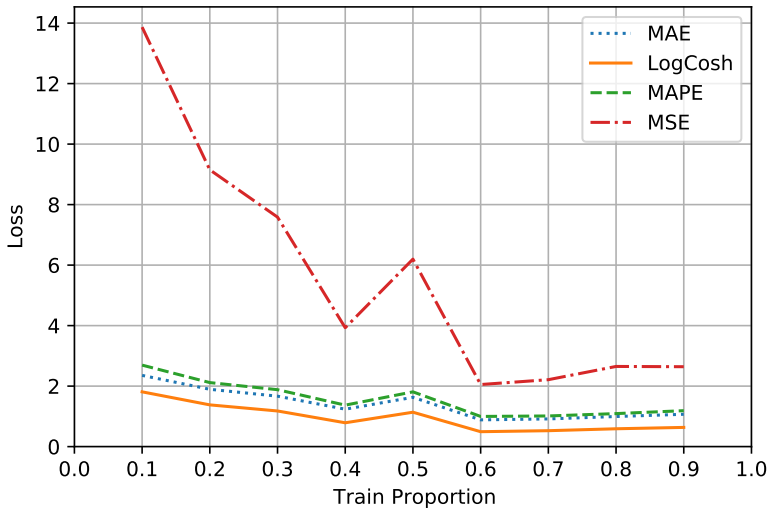


Figure 3.5.2: NN regressor measures for learning *pentagon* volume from Plücker coordinates, at varying proportions of the dataset used for the training data. The results show strong learning for the standard MAE measure used such that good predictions can be achieved quickly; whilst the improvement for MSE shows that more training data does lead to better learning.

Augmenting the input Plücker coordinates by also concatenating the *gcd*s between their values did not improve learning as may be anticipated from the potential function form.

Finally to investigate the extent to which the architectures need large amounts of data to fit the well performing NN functions when learning volume, learning performance was plotted for varying train:test proportions on the pentagon data (the largest subdataset). The test data results are plotted for the MAE, log(cosh), MSE, and MAPE measures (when trained consistently with log(cosh)) in Figure 3.5.2. Learning is still consistently good for the linear measures at lower train proportions, emphasising that the function fitted for the volume from Plücker coordinates is likely of a simple form such that not much data is needed to approximate it well. However the MSE value does blow-up for low amounts of training data, indicating that the function becomes worse at fitting to all the data and focuses on the bulk, such that outliers whose difference is averaged out in the linear measures have significantly larger contributions for this squared measure.

Conclusion

This study advocated for the use of Plücker coordinates as a $GL(n,\mathbb{Z})$-invariant representation of polytopes over the standard vertex matrices. This was strongly supported by the ML learning results for a variety of polytope properties. Focus was on lattice polygons which generate toric Fano varieties, and NN regressors managed to learn volume and dual volume with surprising success, indicative of a simple relation which can be extracted from this representation. There was some learning for codimension, and negligible learning for Gorenstein index. As more complicated parameters this was less surprising, and motivates development of the study to tackle this again with more technical architecture.

Beyond this study, investigation into Plücker representation, and general ML, of higher-

dimensional polytopes is also prudent, in particular the KS dataset of 4-dimensional reflexive polytopes useful for constructing CY 3-folds [86], [156].

3.6 Calabi-Yaus (2112.06350)

As discussed in §2.2, there are many different ways to construct compact Calabi-Yau manifolds. The most abundant source being as codimension 1 hypersurfaces in a toric variety created from a reflexive polytope. The prototypical subset of these are the CYs coming from hypersurfaces in weighted projective spaces. For CY 3-folds there are exactly 7555 weighted projective spaces which admit these CY 3-folds as hypersurfaces, as found in [74]. These 4-dimensional projective spaces are defined with a set of 5 positive coprime integer weights as in (2.2.4), which along with the non-trivial CY Hodge numbers make up the data used in this study. The vanishing first Chern class for these hypersurfaces reduces nicely to the condition that the polynomial homogeneous degree $d = \sum_i w_i$ in the case of weighted projective spaces. However only in the special 7555 cases are these hypersurfaces CY.

The Hodge numbers for any complex variety count the number of classes of forms with both holomorphic and anti-holomorphic indices at each level. The restriction on the form levels due to the manifold dimension inspires a symmetrical layout in a Hodge diamond, as can be seen in

$$\begin{array}{ccccccccccc}
 & & & & h^{0,0} & & & & & & \\
 & & & h^{1,0} & & h^{0,1} & & & & & \\
 & & h^{2,0} & & h^{1,1} & & h^{0,2} & & & & \\
 & \iddots & & \iddots & & \ddots & & \ddots & & & \\
h^{n,0} & & \cdots & & \cdots & & \cdots & & h^{0,n} & & \\
 & \ddots & & \ddots & & \iddots & & \iddots & & & \\
 & & h^{n,n-2} & & h^{n-1,n-1} & & h^{n-2,n} & & & & \\
 & & & h^{n,n-1} & & h^{n-1,n} & & & & & \\
 & & & & h^{n,n} & & & & & &
\end{array} \tag{3.6.1}$$

for a complex n-fold. In the Hodge diamond of the CY 3-folds, only the $h^{1,1}, h^{2,1}$ entries are non-trivial positive integers. The other entries are 0 or 1, due to a series of relations. These relations include use of complex conjugation implying $h^{\alpha,\beta} = h^{\beta,\alpha}$ and Serre duality implying $h^{\alpha,\beta} = h^{n-\alpha,n-\beta}$. Generally, the relations rely on there being a unique $(n,0)$-form from the CY n-fold definitions such that $h^{n,0} = h^{0,n} = 1$ (hence also a unique $(0,n)$-form by conjugation), and this $(n,0)$-form can be contracted with any $(0,\alpha)$-form to produce a (n,α)-form which by Serre duality gives a $(0,n-\alpha)$-form, and means $h^{0,\alpha} = h^{0,n-\alpha}$. Also as the CYs are simply connected then the first fundamental group is zero and so too is the first homology group trivial implying $h^{1,0} = h^{0,1} = 0$.

For the CY 3-fold Hodge diamond the corners are hence $1 = h^{3,0} = h^{0,3} = h^{0,0} = h^{3,3}$, with the latter two coming from $h^{0,\alpha} = h^{0,n-\alpha}$ and the conjugate relation. The sides are $0 = h^{1,0} = h^{0,1} = h^{2,3} = h^{3,2} = h^{2,0} = h^{0,2} = h^{1,3} = h^{3,1}$, with the second two coming from Serre duality of the first two, the third two from $h^{0,\alpha} = h^{0,n-\alpha}$ and the conjugate relation, and the final two from both these relations applied together. This just leaves the final two

non-trivial Hodge numbers $h^{1,1}$ (which equals $h^{2,2}$ from Serre duality), and $h^{2,1}$ (which equals $h^{1,2}$ under complex conjugation or Serre duality). The final Hodge diamond for these CY 3-folds is hence

$$\begin{array}{ccccccc} & & & 1 & & & \\ & & 0 & & 0 & & \\ & 0 & & h^{1,1} & & 0 & \\ 1 & & h^{2,1} & & h^{2,1} & & 1 \\ & 0 & & h^{1,1} & & 0 & \\ & & 0 & & 0 & & \\ & & & 1 & & & \end{array} \tag{3.6.2}$$

where $h^{1,1}$ are dubbed the Kähler parameters, and $h^{2,1}$ are the complex parameters.

Since CY manifolds are Kähler, there is a relation that the sum of each row in the Hodge diamond is the respective Betti number as $b_i = \sum_{\alpha+\beta=i} h^{\alpha,\beta}$. Which then leads to computation of the Euler number $\chi = \sum_i (-1)^i b_i$. In this CY 3-fold case, hence $\chi = 2(h^{1,1} - h^{2,1})$.

These topological parameters can be extracted from the weights of the ambient projective space from formulas in [270]–[273]. Specifically these are

$$Q(u,v) = \frac{1}{uv} \sum_{l=0}^{\sum_i (w_i)} \left[\prod_{\tilde{\theta}_i(l) \in \mathbb{Z}} \frac{(uv)^{q_i} - uv}{1 - (uv)^{q_i}} \right]_{int} \left(v^{size(l)} \left(\frac{u}{v} \right)^{age(l)} \right), \tag{3.6.3}$$

$$\chi = \frac{1}{\sum_i (w_i)} \sum_{l,r=0}^{\sum_i (w_i) - 1} \left[\prod_{i | l q_i \& r q_i \in \mathbb{Z}} \left(1 - \frac{1}{q_i} \right) \right], \tag{3.6.4}$$

for weights w_i, normalised weights $q_i = w_i / \sum_i (w_i)$, and u, v are the dummy variables of the Poincaré polynomial. $Q(u,v) := \sum_{p,q} h^{p,q} u^p v^q$ is the Poincaré polynomial as the generating function of the Hodge numbers. Further notation includes $\tilde{\theta}_i(l)$ as the canonical representative of lq_i in $(\mathbb{R}/\mathbb{Z})^5$, $age(l) = \sum_{i=0}^{4} \tilde{\theta}_i(l)$ and $size(l) = age(l) + age(\sum_i (w_i) - l)$. Note also for χ, where $\forall\, i$ (lq_i or $rq_i \notin \mathbb{Z}$) then the product takes value 1 [273].

Both formulas require significant computation, involving many non-trivial steps. Even if we realize this dataset in the language of the toric geometry of [86], [274], the formulae involve non-trivial sums over faces of various dimension. Therefore as well as learning the CY property directly, it is consequently also interesting to examine the performance of supervised NN methods (as in [114]) in learning these topological parameters from the weights, guided by insight from unsupervised analysis.

3.6.1 The Calabi-Yau Data

The projective space weights make up a list of 7555 5-vectors of positive integers $(w_1, w_2, w_3, w_4, w_5)$. In parallel to this the non-trivial Hodge data[18] for the CY 3-fold hypersurface is also given $(h^{1,1}, h^{2,1})$.

[18]These CY topological parameters can be computed directly from the ambient toric variety (and are not dependent on triangulation) so are the same for all CY hypersurfaces from the same polytope, here defined

In defining a weighted projective space, the weights must be positive integers. Beyond this there is a desire to remove redundancy where multiple weight systems produce the same space by requiring the set of integers to also be coprime. The identifications that are used in constructing the projective space lead to singular sets, which the hypersurfaces can intersect with suitable resolution. To be consistently defined over these singular sets another property of the polynomial is required: *transversity*. The transversity property implies that the polynomial equation and its derivative share no common solutions, and this condition translates into a condition on the projective space weights:

$$\forall w_i \; \exists w_j \; s.t. \; \frac{\sum_k (w_k) - w_j}{w_i} \in \mathbb{Z}^+. \tag{3.6.5}$$

However as described in [74], this condition is necessary but not sufficient for the surface to be CY. It is therefore of interest to consider the extent to which each of these 5-vector weights properties contribute to determine the very special CY property; and it is this question we look to probe with new tools from data analysis, and ML.

Therefore, in addition to the CY dataset which forms the central focus of this study, some auxiliary datasets that help in assessing the learning of the CY property were constructed. These are equivalent datasets of 5-vectors that possess fewer of the necessary properties required to meet the Calabi-Yau property.

The 4 datasets (including the original CY dataset) are:

(a) 7555 5-vectors of positive random integers,

(b) 7555 5-vectors of positive random coprime integers,

(c) 7555 transverse 5-vectors of positive random coprime integers,

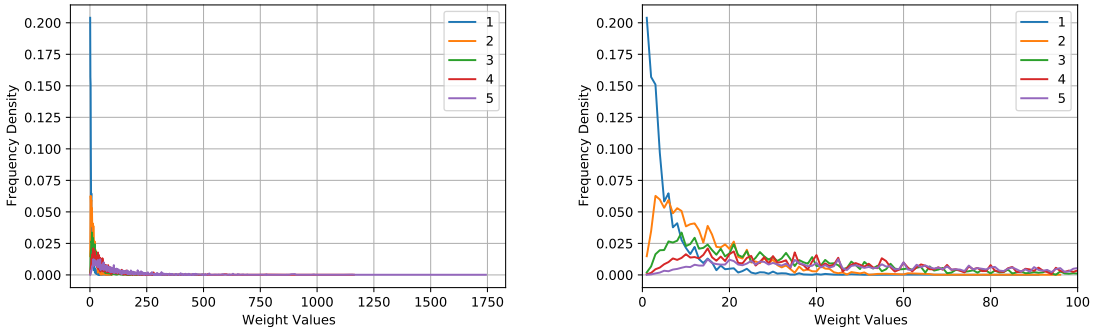
(d) 7555 Calabi-Yau 5-vectors.

These datasets were specifically constructed so as not to form a filtration, therefore at each stage the dataset generated was ensured to not include data which satisfies the additional conditions at the next level. To clarify, each 5-vector in set (a) had weights which shared a common factor, in set (b) all 5-vectors did not satisfy condition (3.6.1), and those in set (c) where not in the CY list of (d).

To introduce a consistency across the datasets, all the 5-vectors entries are sorted in increasing order. Initially the weights for each of the datasets (a-c) were sampled using a discretised uniform distribution, $U(1, 2000)$, bound above by 2000 to mimic the highest value in the CY dataset of 1743. However as shown in Figure 3.6.1 the weights follow a distribution far from that of a uniform distribution. Therefore to make the generated data more representative, an exponential distribution was fitted to the histogram of all weights in the CY dataset, as shown in Figure 3.6.2, using the `scipy` library. This exponential distribution was then used to sample weights in each case, causing the frequency distributions of the weights for each of the artificial datasets to align much closer to that of the CY data.

---

just by one set of weights. Whereas the other CY topological parameters, $[c_2]_\alpha, d_{\alpha,\beta,\gamma}$, are specific to the surface.



(a) (b)

Figure 3.6.1: Frequency distribution of each of the CY 5-vector weights, w_i (labelled by $i : 1 - 5$). Figure (b) shows the same data as (a), but restricted to lower entries so as to highlight the low value behaviour, due to the entry sorting.

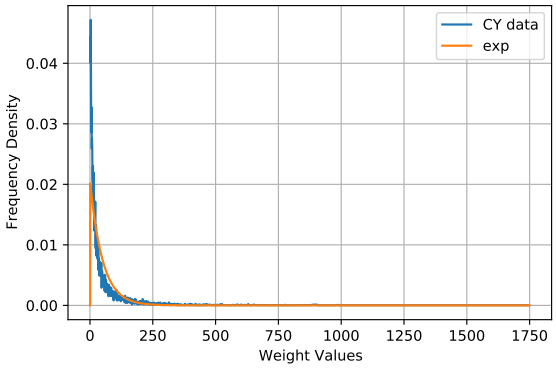


Figure 3.6.2: Frequency distribution for *all* weights occurring across all 5-vectors in the CY dataset. Plot also shows the fitted exponential distribution, with scale parameter 49.536 (to 3 decimal places).

Coprimality: It is interesting to note that the probability of k randomly chosen integers being coprime is: $1/\zeta(k)$; via the Riemann zeta function. Hence the probability of a random 5-vector of integers being coprime is ~ 0.964, and therefore the dataset (b) is relatively more common than the dataset (a). Effectively it is easy to randomly generate consistently defined weighted projective spaces.

Example: The archetypal example of a Calabi-Yau 3-fold in weighted projective space is the quintic, embedded in $\mathbb{P}^4_{\mathbb{C}}$ such that the weights are simply $(1, 1, 1, 1, 1)$. The CY hypersurface drawn inside the projective space is formed from degree 5 monomials in the projective space coordinates, and since all these are weight 1 then this is all combinations of 5 coordinates. The most general hypersurface hence has equation:

$$0 = \sum_{\substack{\nu_0,\nu_1,\nu_2,\nu_3,\nu_4 \\ s.t.\ \nu_0+\nu_1+\nu_2+\nu_3+\nu_4=5}} c_{\nu_0\nu_1\nu_2\nu_3\nu_4} z_0^{\nu_0} z_1^{\nu_1} z_2^{\nu_2} z_3^{\nu_3} z_4^{\nu_4} \ , \tag{3.6.6}$$

for any $c_{\nu_0\nu_1\nu_2\nu_3\nu_4} \in \mathbb{C}$. For all these choices the Hodge numbers are the same as $h^{1,1} = 1$, $h^{2,1} = 101$, leading to Euler number $\chi = 2(h^{1,1} - h^{2,1}) = -200$.

In the learning the inputs were the weight 5-vector, here $(1, 1, 1, 1, 1)$, and the output a topological parameter (say 1, 101, or -200), or binary classification to say whether this set of weights admits a CY hypersurface.

3.6.2 ML Results

In order to identify structure in the weight dataset that enabled the respective projective space to admit a CY hypersurface (the 'CY property'), techniques from unsupervised and supervised ML were used. On these 4 datasets PCA was applied to the weight vectors to visualise dominant variation and how this changed as extra necessary properties were added. TDA was applied to look for higher-dimensional structure in these 5-dimensional datasets. Then K-Means clustering was used to analyse the unanticipated linear structure found. In vain of [114], NN regressors were used to learn the hypersurface topological properties from the weights, however the more interesting supervised ML investigation looked at NN classification between the CY dataset of weights and the other 3 artificial datasets; further using the Hodge information to analyse the performance.

The regressor NNs had layer sizes of 32, 64, 32 with ReLU activation, and a final layer with as many neurons as the output, training on batches of 200 for up to 200 epochs (or until a tolerance update of the loss less than 0.0001 was reached) to minimise a MSE loss with an Adam optimiser. The classifier NNs had the same hyperparameters (but with cross-entropy loss) to perform either multiclassification between the datasets, or binary classification between each artificial dataset and the CY dataset. Further to NNs, LRs and SVMs were also trialled as simpler classifying architectures with comparable performance on this data. The LRs had a tolerance of 1 for learning the weight behaviour, a C-value of 100 such that there was a low amount of regularisation, and used Newtons method for solving, such that multiclassification could also be performed. The SVMs used a simple linear kernel, and here a higher regularisation due to a C-value of 1. All architectures were trained and tested over 5-fold cross-validation runs to provide standard errors on the performance measures.

PCA

Each of the 4 datasets of weights consists of weight vectors in $(\mathbb{Z}^+)^5$ which can be considered to be in $\mathbb{R}^5$. Performing PCA with a linear kernel on each of these datasets produced variance eigenvalues which in each case were at least 5 times larger for the first principal component compared to the others. In particular for the CY dataset the first principal component was 2 orders of magnitude larger than the others. This indicates that much of the variation, and hence data structure, is dominated by a single dimension.

Noting that in this process scaling of the data was not used as the entries are all dimensionless and their relative difference is important. As the data is not scaled one may think that the latter weights of each vector would dominate the behaviour (since the weights are ordered). This would lead the covariance matrix to be near-diagonal, and the principal components would align closely to the original vector entries. However, as shown by the covariance matrix for the CY dataset in (3.6.7), the matrix is not diagonal and the eigenvectors have significant contribution from multiple components.

$$K_{CY} = \begin{pmatrix} 41 & 43 & 109 & 250 & 404 \\ 43 & 119 & 278 & 642 & 1017 \\ 109 & 278 & 1795 & 3626 & 5562 \\ 250 & 642 & 3626 & 8588 & 12941 \\ 404 & 1017 & 5562 & 12941 & 20018 \end{pmatrix},$$
$$\varepsilon_{CY} = \begin{pmatrix} 0.016 & 0.041 & 0.229 & 0.531 & 0.815 \\ 0.021 & 0.036 & -0.973 & 0.100 & 0.205 \\ 0.120 & 0.206 & 0.034 & -0.823 & 0.514 \\ 0.417 & 0.875 & 0.023 & 0.173 & -0.172 \\ 0.900 & -0.435 & 0.003 & 0.018 & -0.008 \end{pmatrix}, \ \lambda_{CY} = \begin{pmatrix} 30071 \\ 233 \\ 161 \\ 74 \\ 21 \end{pmatrix}, \tag{3.6.7}$$

for eigenvectors as rows of ε_{CY} with respective eigenvalues in λ_{CY}; where covariance and eigenvalue entries are given to the nearest integer, and eigenvector entries to 3 decimal places. This implies that the PCA structure is more subtle than a trivial projection.

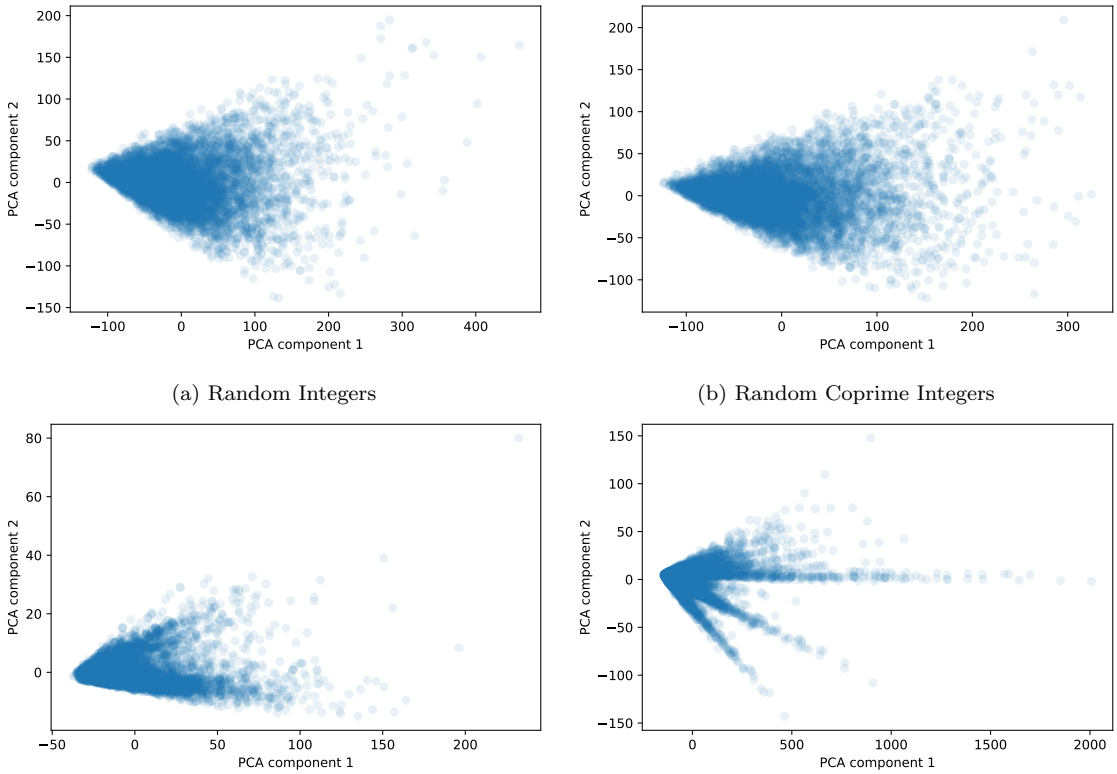
To relatively compare the datasets' PCAs, the normalised vectors of eigenvalues are given in (3.6.8), for the random 'R', coprime 'C', transverse 'T', and Calabi-Yau 'CY' datasets respectively. They show that the first component significantly dominates, and hence lower dimensional representation of the data through PCA will usefully depict the data's underlying linear structure.

$$\lambda_R = \begin{pmatrix} 0.75534 \\ 0.16297 \\ 0.05274 \\ 0.02059 \\ 0.00837 \end{pmatrix}, \quad \lambda_C = \begin{pmatrix} 0.74845 \\ 0.16856 \\ 0.05417 \\ 0.01997 \\ 0.00885 \end{pmatrix}, \quad \lambda_T = \begin{pmatrix} 0.91388 \\ 0.04211 \\ 0.02578 \\ 0.01334 \\ 0.00489 \end{pmatrix}, \quad \lambda_{CY} = \begin{pmatrix} 0.98399 \\ 0.00764 \\ 0.00525 \\ 0.00242 \\ 0.00070 \end{pmatrix}. \tag{3.6.8}$$

Hence for the sake of visualisation, the first 2 components of each datapoint's principal component projection are plotted as a 2-dimensional scatter diagram for each dataset. These components show the directions with the most variation, and hence display the underlying structure most clearly. The 2-dimensional PCA plots are given in Figure 3.6.3, for each of the 4 datasets considered; the PCA information (additional to that in (3.6.7)) is available in §A.1.

The cone-like bounding structure of all plots shows the effects of the weight ordering. This is simply that as the first component's value increases (most correlated to the largest, and hence last, weight in the 5-vector) the range of values the second component (roughly correlated to the second-largest / second-last weight) can take increases. Or put more simply, the second-last weight takes values up to the size of the last weight and so this places cone-like bounds on the plots. All plots also show higher densities at lower values of the principal components which is also related to this effect.

The PCA plots show that as more of the necessary conditions are added to the datasets, more structure is apparent in the projected outputs. First note the coprime condition causes a negligible change to the distribution of weights. The transverse condition however has a significant effect. The second components become much more limited and the data begins to separate into approximately two forks. Most exciting, is the jump to the full CY data. Now the PCA shows a clear clustering of the 5-vectors at higher values of the first principal component. This distinct separation into clear lines of datapoints shows an intriguing structure to the weights of Calabi-Yau projective spaces, which is **not** present for spaces with just the transverse condition. The reasons for this separation are unclear, however we



(a) Random Integers

(b) Random Coprime Integers

(c) Random Transverse Coprime Integers

(d) CY Weights

Figure 3.6.3: 2-dimensional PCA plots for the 4 considered datasets. As more of the conditions are added, more structure appears, in particular there is some form of distinct class separation for the CY weights.

make conjectural statements about a potential relation to the spaces' Hodge numbers due to a similar structural separation when performing the clustering analysis.

A final note is that the PCA used here was explicitly linear, and hence probes the simplest kind of implicit structure. Kernel PCA methods were also used to analyse these datasets, for a variety of traditional kernels (including Gaussian, sigmoid, and an array of polynomial kernels); however, none of these methods produced as distinct a clustering separation as that for the linear kernel, indicating, that surprisingly, the most prominent implicit structure of the Calabi-Yau weights takes a linear form.

TDA

To examine if this linear separation in the CY data is an artefact of the projection of a higher-dimensional structure, topological data analysis was used through persistent homology in the $\mathbb{R}^5$ space.

This persistent homology for the CY data was computed for H_0 and H_1 (higher H_n up to $n = 4$ can be computed in 5-dimensional space but are incredibly computationally expensive in terms of memory for $n \geq 2$). The persistence diagram for this analysis is shown in Figure 3.6.4, where the diagram plots all members of H_0 and H_1 as points with their respective δ values of birth (cycle creation) and death (cycle filling).

As can be seen from the diagram all the members of H_0 are blue points born at $\delta = 0$,

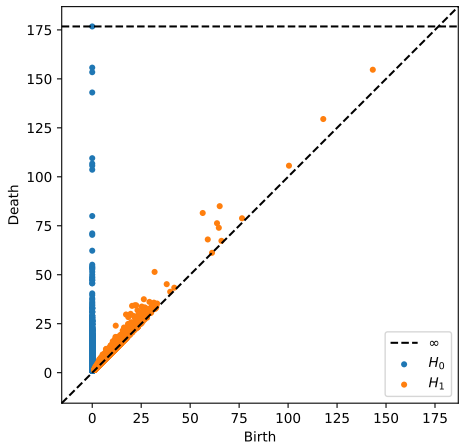


Figure 3.6.4: Persistent diagram for the H_0 and H_1 homology groups of the CY data's Vietoris-Rips complex filtration.

these are each of the 0-cycles (i.e. 0-simplices / datapoints) that exist until they are connected by an edge (1-simplex) to any other datapoint. The behaviour shows that there are some datapoints that are significantly further away from the rest of the data and hence join/die much later in the filtration. These points are those with large weight values such that they are far from the origin in the $\mathbb{R}^5$ embedding.

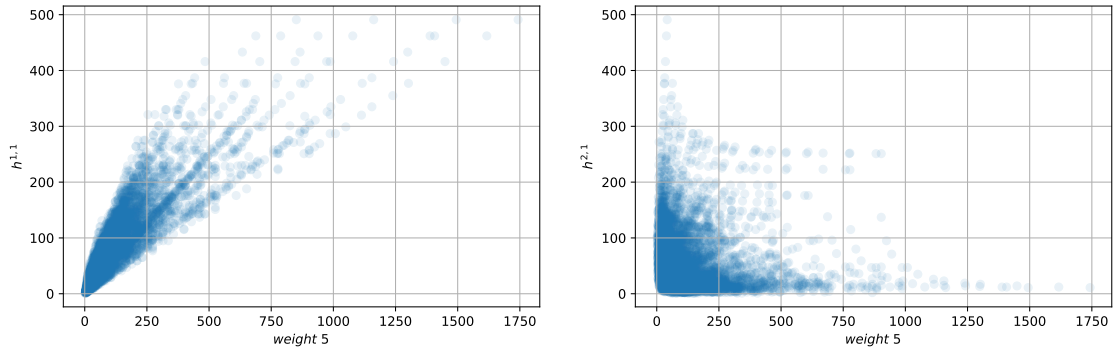
Conversely all members of H_1 are points in orange, and as expected all these cycles of 1-simplices/edges which are not boundaries of 2-simplices/triangles lie close to the diagonal line in the persistence diagram. This behaviour indicates a short life of each cycle, a behaviour typical of noise in the dataset. Since traditionally it is only points far from the diagonal that indicate significant persistent structure, there is hence not higher dimensional structure formation or non-trivial topology in the data which would deter from the linear clustering behaviour seen through the PCA.

K-Means

Following intuition from the success of NNs predicting the topological data from the weight vectors (as studied in [114] and repeated below), plots are made to examine the correlations between the sorted weight entries and the Hodge numbers. The observed correlation behaviour is best exemplified plotting against the largest weight, as shown in Figure 3.6.5.

The behaviour in Figure 3.6.5a shows a similar form of fork-like splitting of the datapoints as in the PCA of Figure 3.6.3d, even with a central fork particularly more dominant than the others. This seemingly linear behaviour between final weight and $h^{1,1}$ is quite surprising, and here again the CY hypersurfaces appear to be separating themselves into classes, according to the ratio of $h^{1,1}$ to the final weight, w_5. On the contrary, the behaviour in Figure 3.6.5b, follows the familiar mirror symmetry plot [74], complimenting the linear behaviour with $h^{1,1}$ such that their combination will preserve this structure. Similar behaviour also occurs for the other weights in the 5-vectors, as can be seen in §A.2, despite less obvious clustering.

To further examine this clustering phenomena we plot a histogram of the ratio $h^{1,1}/w_5$ in Figure 3.6.6. Note for this plot only datapoints with $w_5 > 250$ were used since this was where the class separation was more prominent such that the cluster identification would be



(a) (b)

Figure 3.6.5: Distribution of Calabi-Yau weighted projective spaces, according to their final (and largest) weight and (a) $h^{1,1}$ or (b) $h^{2,1}$ respectively.

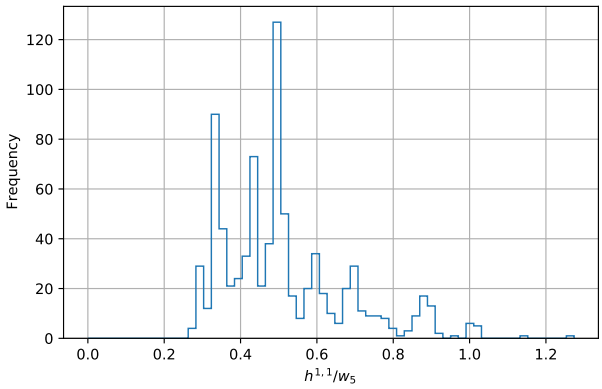


Figure 3.6.6: Frequency of the ratio between $h^{1,1}$ and the largest weight, w_5, for the CY data with $w_5 > 250$ (where structure more prominent). Peaks indicate a natural clustering.

improved. As can be seen from the peaks in the figure, there is a clear clustering behaviour. Therefore we reexamine this data of ratios with the use of K-Means clustering.

As motivated by the formation of a set of linear relationships between w_5 and $h^{1,1}$ shown in Figure 3.6.5a, and the peak occurrence in the histogram of ratios in Figure 3.6.6, unsupervised clustering methods were used to examine this behaviour.

The 'outer' ratio data used to produce the histogram plot, where clustering was more prominent, provides a very suitable database for 1-dimensional K-Means clustering using 'scaled-max-inertia' as in (2.3.13). An elbow plot of scaled-max-inertia against number of clusters identifies an optimum of 10 clusters, as shown in Figure 3.6.7.

Using the optimal number of 10 clusters, the separation matches up exceptionally for the outer data, as shown by plots of the cluster bounds in Figure 3.6.8. The clusters sizes for the clusters moving anticlockwise about the plot, for increasing ratio, are:
$[103, 354, 454, 734, 626, 623, 643, 895, 1419, 1704]$, highlighting that there is a greater density of points at low w_5 as expected, since this was why 'outer' data was focused on for clustering.

To measure the clustering performance we use the standard inertia measure over the *full* dataset, however normalised by the number of ratios across the dataset, $\hat{\mathscr{I}}$, and an

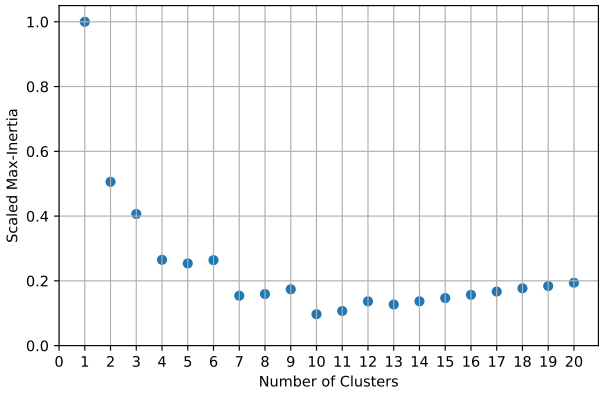


Figure 3.6.7: Plot of scaled-max-inertia as the number of clusters used for K-Means clustering varies. The minimum identifies an optimum number of clusters: 10.

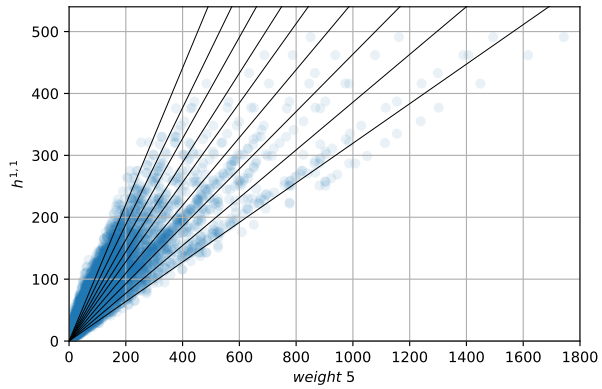


Figure 3.6.8: Plot of the bounds of the 10 clusters produced on the outer data ($w_5 > 250$) via K-Means clustering.

equivalent measure also normalised by the range of the ratios:

$$\hat{\mathscr{I}} = 0.0266\,, \qquad \frac{\hat{\mathscr{I}}}{max(h^{1,1}/w_5) - min(h^{1,1}/w_5)} = 0.00084\,, \tag{3.6.9}$$

These values show that clustering performed exceptionally well, as each ratio in the full CY dataset was less than 0.1% of the ratio-range away from its nearest cluster. Therefore confirming the distinct *linear* behaviour observed, as well as the class separation. The class partitioned data is made available with the other datasets on the papers respective Github.

NN Regression

The topological parameters can be extracted for the weight information following the highly non-trivial formulas of (3.6.3) and (3.6.4).

Both of these formulas require *gcd* computations throughout their evaluation. ML methods famously perform badly when approximating these styles of equations and so one would expect the simple NN architecture used here to not be particularly successful. These results

Measure	Property			
	$h^{1,1}$	$h^{2,1}$	$[h^{1,1}, h^{2,1}]$	χ
R^2	0.9630 ± 0.0015	0.9450 ± 0.0133	0.9470 ± 0.0041	0.9510 ± 0.0023
MAPE	0.1493 ± 0.0027	0.2519 ± 0.0152	0.2375 ± 0.018	-
MSE	166.9 ± 10.0	147.0 ± 35.6	186.9 ±13.9	1746.1 ± 82.4

Table 3.6.1: Learning each of the CY topological parameters from the 5-vectors of weights (dataset (d)). Note the final column is Euler number $\chi = 2(h^{1,1} - h^{2,1})$, and since it can evaluate to 0 its MAPE value is not defined. Measurement of learning performance uses 5-fold cross-validation to provide an average and standard error on each measure's value.

for learning the CY topological parameters from the weights which admit CY hypersurfaces, with the previously described NN regressor architecture are given in Table 3.6.1.

The results show a surprisingly successful predictive ability for the Hodge numbers and Euler number, particularly with R^2 values exceeding 0.9. The MAPE values show the Hodge numbers are consistently predicted to be only around a proportion of 0.2 off from their true values, whilst the MSE values provide a less physical measure of learning but are included for reference since they were used as the regressor loss.

Considering the complexity of the equation forms in (3.6.3) and (3.6.4), it is impressive the NNs can learn any correlating behaviour for computation of Hodge numbers or Euler number from the weights alone. Therefore the relatively better performance in learning $h^{1,1}$ is likely due to the apparent linear relationship to the weights as exemplified by the K-Means clustering.

NN Classification

The conditions for a 5-vector of weights to represent a weighted projective space which can admit a CY hypersurface are particularly special. To probe the necessary conditions of coprimality and transversity equivalent datasets were generated as discussed in §3.6.1, with which the CY dataset was compared.

Due to the exponentially-fitted generation distribution making these weights more representative, differentiating which dataset a 5-vector belongs to is not possible by eye. Therefore it is natural to wish to consider the effectiveness of ML to this classification problem: learning the Calabi-Yau nature.

Three classification architectures (with hyperparameters as previously described) were used to learn to differentiate the CY weights from each of the other datasets: random integers, coprime random integers, and transverse coprime random integers in binary classification problems. Furthermore they were also used to differentiate all 4 datasets in a multiclassification problem.

Results for this learning are given in Table 3.6.2. Measures show that NNs can well differentiate the CY weights from each of the other datasets. As expected there is minimal difference due to introduction of coprimality, since the nature of primes is famously elusive. However, once transversity was included into the dataset the binary classification

Architecture	Measure	Dataset			
		Random	Coprime	Transverse	All
Logistic Regressor	Accuracy	0.7152 ± 0.0035	0.7199 ± 0.0037	0.7430 ± 0.0065	0.4825 ± 0.0035
	MCC	0.4352 ± 0.0065	0.4467 ± 0.0073	0.5003 ± 0.0121	0.3141 ± 0.0043
Support Vector Machine	Accuracy	0.7253 ± 0.0029	0.7116 ± 0.0029	0.7464 ± 0.0014	0.4732 ± 0.0070
	MCC	0.4605 ± 0.0054	0.4374 ± 0.0054	0.5174 ± 0.0029	0.3060 ± 0.0078
Neural Network	Accuracy	0.9189 ± 0.0037	0.9178 ± 0.0030	0.7575 ± 0.0024	0.5881 ± 0.0048
	MCC	0.8380 ± 0.0073	0.8377 ± 0.0056	0.5306 ± 0.0059	0.4615 ± 0.0072

Table 3.6.2: ML results for three different architectures performing binary classification between the CY data and each specified dataset; and in addition multiclassification across all 4 datasets (labelled 'All'). Learning is measured using Accuracy and MCC with 5-fold cross-validation to provide an average and standard error on each measure's value.

performance dropped, but was still surprisingly good.

A further surprise was the equally good performance of the LRs and SVMs. These simple architectures could accurately classify approximately three-quarters of the data even without using transversity (where this condition was in both CY and compared dataset).

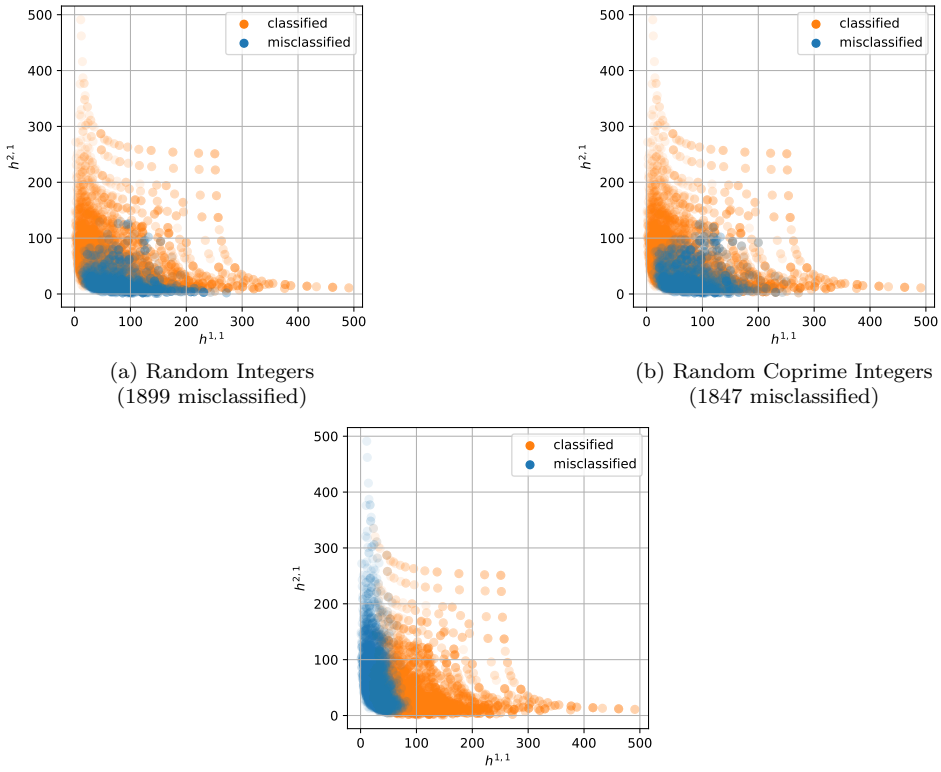
Multiclassification of all datasets was not as strong. However within these measures the identification of the CY data was considerably better, with most of the performance reduction due to misclassifying between random, coprimality, and transversity. To exemplify this we give a sample normalised confusion matrix for the multiclassification with a LR:

$$CM_{LR} = \begin{pmatrix} 0.116 & 0.013 & 0.029 & 0.091 \\ 0.076 & 0.083 & 0.074 & 0.020 \\ 0.074 & 0.078 & 0.062 & 0.019 \\ 0.026 & 0.004 & 0.008 & 0.228 \end{pmatrix}, \tag{3.6.10}$$

where row indicates true class and column predicted class for each of: random, coprime, transverse, CY respectively. The final entry shows nearly all the CY data is correctly classified (0.25 indicates the full quarter of the accumulated datasets). Therefore measures will indicate lower performance where the other conditions cannot be differentiated, and it is likely that these conditions are not the most prominent conditions to indicate the CY property.

To further examine the learning performance we next look explicitly at the misclassifications of the CY data, using again links to the Hodge numbers to identify areas of difficulty. Since the LR performed comparably to the other architectures, and is a significantly simpler architecture than the NN, its use for misclassification analysis seemed the most appropriate.

Due to the simple structure, only 50 5-vectors in each non-CY dataset were used to train the regressor with another 50 CY 5-vectors. The regressor was then used to predict the class of all the CY data (i.e. that it was CY), producing accuracies of: 0.78, 0.81, 0.61 when



(a) Random Integers
(1899 misclassified)

(b) Random Coprime Integers
(1847 misclassified)

(c) Random Transverse Coprime Integers
(2739 misclassified)

Figure 3.6.9: A LR, trained on 50 CY 5-vectors and 50 non-CY 5-vectors, predicts whether all of the CY 5-vectors are CY or not. The plot shows the distribution of the CY surfaces according to their Hodge numbers. Those in blue are misclassified as non-CY, those in orange are correctly classified to be CY. The non-CY vectors come from datasets of Random, Coprime, or Transverse 5-vectors respectively.

trained with each of the random, coprime and transverse datasets respectively.

Perhaps more curious is the distribution of these CY misclassifications with respect to their Hodge numbers, plotted in Figure 3.6.9. Training random and coprime datasets in both cases leads to perfect classification of CY spaces with high $h^{2,1}$, whereas training with transverse data leads to perfect classification with high $h^{1,1}$. Note also that investigations showed both other architectures had similar performance with respect to Hodge numbers, as shown by figures in §A.3.

To further investigate the dependence of the learning performance on the Hodge numbers, the CY dataset was binned in two independent ways. The first was according to $h^{2,1}$, and the second according to $h^{1,1}$. The bin bounds were optimised such that an approximately consistent number of CYs had Hodge numbers within each bin's bounds, with a preset number of 50 bins used (selected to have a suitable bin size > 100). Plots of these bin frequencies are given in Figures 3.6.10a and 3.6.10b.

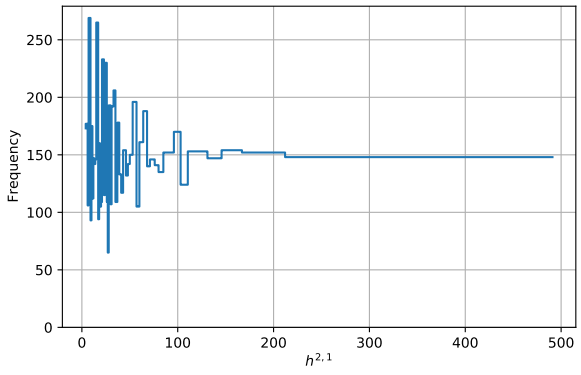
This produced a CY dataset associated to each bin, with which a non-CY 5-vector dataset was randomly sampled. For the $h^{2,1}$ partition the random dataset was used to sample an equal number of non-CY 5-vectors for each bin, such that the datasets were balanced. As training behaviour for the random and coprime datasets was so similar, only the random

dataset was used in this investigation. Conversely, for the $h^{1,1}$ partition the transverse dataset was used. These choices of non-CY datasets used for training were selected such that they aligned with the predicted behaviour demonstrated in Figure 3.6.9, where random-training improves high-$h^{2,1}$ performance, and transverse-training improves high-$h^{1,1}$ performance.

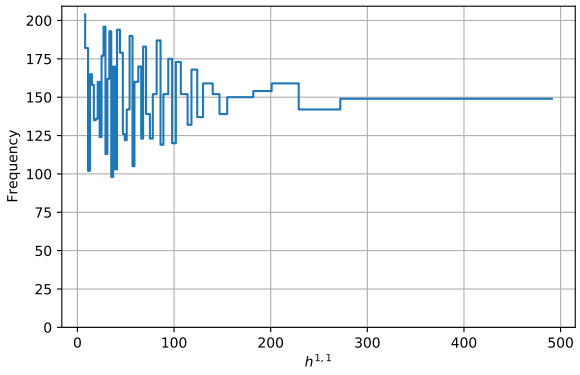
For each bin's now balanced dataset an independent LR (with architecture as before) was initialised, trained and tested. A random 80% sample of the bin data was used for training, with testing on the remaining 20% complement. For each bin, the initialisation, training, and testing was repeated 20 times, such that variances on the measures could be calculated. Accuracies were recorded for each bin regressor, as well as the final 5 weights used to define the trained LR.

Accuracies across the bins for both partitions are given in Figures 3.6.10c and 3.6.10d, with their respective accuracy variances in 3.6.10e and 3.6.10f. There are near-perfect predictions at the upper ends of these partitions, with relatively very small variances. Determination of the CY property is hence considerably easier for surfaces whose Hodge numbers take extreme values, and pre-training against data with or without the transverse condition can significantly aid learning depending on what values the Hodge numbers take.

Finally, the 5 averaged LR weights are plotted for each bin (with respective variances surrounding them) in Figures 3.6.10g and 3.6.10h. As can be seen by comparing the relative weight sizes, in both cases at the higher ends of the partitions the first two weights particularly dominate the regression[19]. Since each LR weight aligns with the projective space weight, this indicates at these extremes where learning is particular strong, only the first two (i.e. lowest) weights are needed to identify whether the weighted projective space admits a CY hypersurface. Where only the CY dataset has the transversity property (i.e. training against random) the first weight is the most significant, whilst where transversity is in both datasets (i.e. training against transverse) the second weight is the most significant.

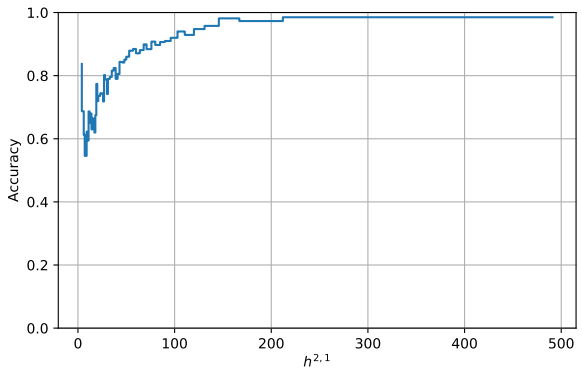


(a) Bin frequencies for $h^{2,1}$ partition

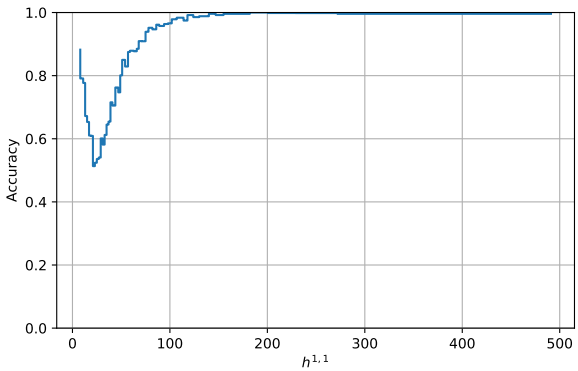


(b) Bin frequencies for $h^{1,1}$ partition

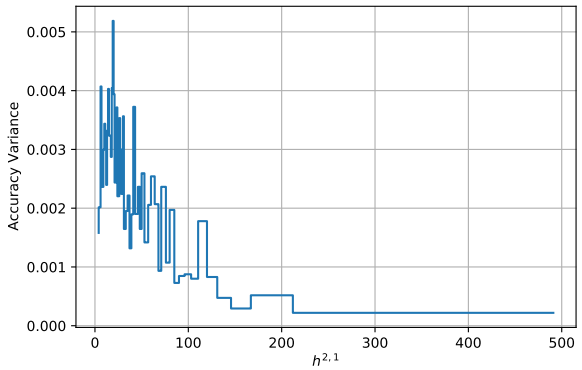
[19]Note that in the LR architecture the intercept is set to 0 such that optimisation must focus on the relative sizes of the inputs.



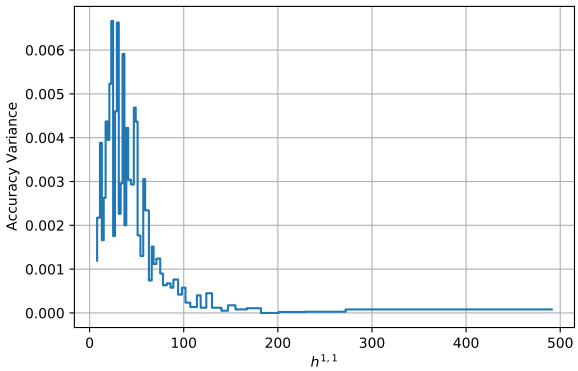
(c) LR (Random-trained) Accuracies for $h^{2,1}$ partition



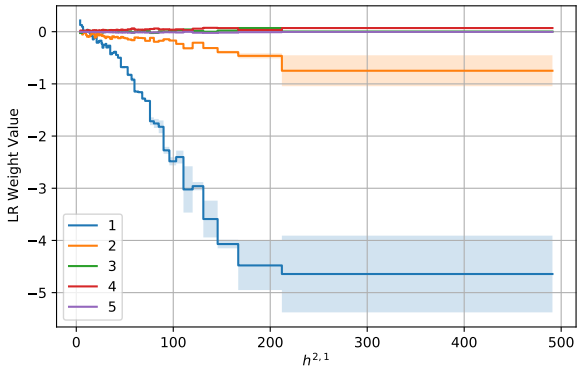
(d) LR (Transverse-trained) Accuracies for $h^{1,1}$ partition



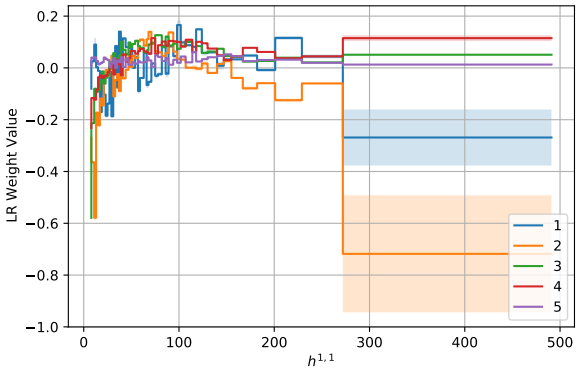
(e) Variances of the LR (Random-trained) accuracies for $h^{2,1}$ partition



(f) Variances of the LR (Transverse-trained) accuracies for $h^{1,1}$ partition



(g) LR (Random-trained) weights for $h^{2,1}$ partition, plotted with variance bars



(h) LR (Transverse-trained) weights for $h^{1,1}$ partition, plotted with variance bars

Figure 3.6.10: Relevant plots for LR learning of the 5-vectors being CY or non-CY. Where the non-CY data was the Random data then binning was according to $h^{2,1}$, where it was Transverse data then according to $h^{1,1}$. The CY data was binned according to either $h^{1,1}$ or $h^{2,1}$ Figures (a) & (b) show the number of CYs in each Hodge partition bin (half the dataset used in each case as non-CYs cannot be plotted without known Hodge numbers). Figures (c) & (d) show the average accuracies for the LR learning in each case, with (e) & (f) the respective variances (very small comparatively). Finally, figures (g) & (h) show the averaged trained LR weights, plotted with their variances as bands about the average values.

Conclusion
Through the use of unsupervised ML methods a linear clustering structure of the weighted projective spaces that admit CY hypersurfaces was identified. This structure was first observed through PCA, corroborated with TDA, and then observed again due to relations with the hypersurface's Hodge numbers confirmed with strong clustering measures.

Supervised ML methods then learnt to predict the CY topological parameters from the weights directly to a surprisingly exceptional accuracy, perhaps making use of this simple structure. In addition, simple classifier architecture could detect whether a generic weighted-$\mathbb{P}^4$ admitted a CY hypersurface from the weights alone, and with specific pre-training could reach perfect performance at certain extremes of Hodge numbers.

Further analysis into this CY clustering behaviour for weighted-$\mathbb{P}^4$s would hope to uncover its source, simultaneously explaining the success of ML techniques on this dataset.

3.7 Brane Webs (2202.05845)

As introduced in §2.2, and exemplified in Figure 2.2.1, brane webs are a diagrammatic tool to represent 5-dimensional superconformal field theories coming from Type IIB string theory [102], [275]. The (p, q) plane gives the 5-brane charges dictating their orientation and subsequent intersection in these intermediate dimensions.

A brane web (with a single intersection/junction taken to be the origin) can be represented using a web matrix

$$W_L = \begin{pmatrix} n_1 p_1 & n_2 p_2 & \cdots & n_L p_L \\ n_1 q_1 & n_2 q_2 & \cdots & n_L q_L \end{pmatrix} , \tag{3.7.1}$$

for a web with L legs. To standardise the convention all brane charges are taken to be ingoing, the legs are sorted anticlockwise from the negative p axis, and the 5-brane multiplicity of a leg can be taken to be $gcd(p_i, q_i) = n_i$ when the n_i factors are absorbed into the p_i, q_i terms.

Brane webs provide a method to classify 5-dimensional superconformal field theories which arise form this construction according to the $SL(2, \mathbb{Z})$ and Hanany-Witten (HW) equivalences [276]–[279]. In this study the focus of the classification is restricted to webs of 3 legs connected at a single junction, where each of the legs ends on a 7-brane (as described in Table 2.2.1). This choice simplifies considerations avoiding the need deal with irreducible junctions or multiple webs existing for each 7-brane set.

In string theory, the introduced 7-branes are a source of the axiodilaton $\propto log(x^5 + ix^6)$, leading to a branch cut traditionally oriented away from the junction but can have any orientation provided the appropriate monodromy action is applied to any other leg when the branch cut sweeps through it. For the 7-branes to preserve supersymmetry there is an extra condition on the self-intersection, defined

$$\mathcal{I} = \left| \sum_{i<j} det \begin{pmatrix} p_i & p_j \\ q_i & q_j \end{pmatrix} \right| - \sum_{i=1}^{L} gcd(p_i, q_i)^2 , \tag{3.7.2}$$

over the L legs of the brane web, requiring $\mathcal{I} \geq -2$ since it is equivalent to the Coulomb branch dimension $d_{CB} = \frac{\mathcal{I}+2}{2} \geq 0$. Beyond the self-intersection condition for the 7-branes,

the brane web must also satisfy charge conservation at the junction such that $0 = \sum_i p_i = \sum_i q_i$, and also a consistency in removing the 5-brane multiplicity n_i for all legs such that $gcd(p_i, q_i) = 1$ when removed.

Consistent webs (satisfying these requirements) can be classified into equivalence classes where they describe the same superconformal field theory. The classes consist of webs connected by any number/combination of $SL(2,\mathbb{Z})$ and HW transitions, as depicted in Figure 2.2.1 for 3 example webs in the same class. For the web matrices the $SL(2,\mathbb{Z})$ action, transforms all legs as

$$\begin{pmatrix} p_i \\ q_i \end{pmatrix} \mapsto \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p_i \\ q_i \end{pmatrix} , \tag{3.7.3}$$

with $a, b, c, d \in \mathbb{Z}$ and $ad - cb = 1$. Whilst the HW transition acts firstly on a chosen leg (p_j, q_j) by moving it to the other side of the junction to $(-p_j, -q_j)$ (since leg length is not a parameter of the theory), leaving a branch cut which now passes through the junction. Sweeping this branch cut clockwise induces a monodromy action on the leg the branch cut is swept through (p_i, q_i) via

$$\begin{pmatrix} p_i \\ q_i \end{pmatrix} \mapsto M_{(p_j,q_j)} \begin{pmatrix} p_i \\ q_i \end{pmatrix} = \begin{pmatrix} 1 + p_j q_j & -p_j^2 \\ q_j^2 & 1 - p_j q_j \end{pmatrix} \begin{pmatrix} p_i \\ q_i \end{pmatrix} , \tag{3.7.4}$$

or alternatively using $M^{-1}_{(p_j,q_j)}$ for an anticlockwise sweep. Finally the moved leg's multiplicity is updated to preserve charge at the junction $n_j = \frac{1}{p_j} \sum_{i \neq j} p_i = \frac{1}{q_j} \sum_{i \neq j} q_i$.

Since the focus is on 3-leg brane webs, the classification is equivalent to that of 7-brane sets (since each set of 3 7-branes has a unique 5-brane web). These sets of 3 7-branes have been conjectured to be partitioned into classes according to the three parameters: total monodromy M_{total}, asymptotic charge invariant ℓ, and rank (i.e. dimension of the Coulomb branch) d_{CB} [280], as defined for W_3 webs

$$\begin{aligned} M_{total} &= M_{(p_1,q_1)} M_{(p_2,q_2)} M_{(p_3,q_3)} , \\ \ell &= gcd\left(det \begin{pmatrix} p_1 & p_2 \\ q_1 & q_2 \end{pmatrix}, \; det \begin{pmatrix} p_1 & p_3 \\ q_1 & q_3 \end{pmatrix}, \; det \begin{pmatrix} p_2 & p_3 \\ q_2 & q_3 \end{pmatrix} \right) , \\ d_{CB} &= \frac{\mathcal{I} + 2}{2} . \end{aligned} \tag{3.7.5}$$

To probe the validity of this classification with techniques from ML a notion of *strong*, and *weak* equivalence was defined:
$\sim$ *Strong* equivalence $\implies$ webs have been explicitly transformed into each other through a series of $SL(2,\mathbb{Z})$ and HW moves.
$\sim$ *Weak* equivalence $\implies$ webs have the same invariants as in (3.7.5), and number of legs.

With these definitions, strong equivalence naturally implies weak equivalence but not vice versa, as the equality of these 3 invariants between two webs is a necessary but not sufficient condition for *strong* equivalence. ML results then indicate the incompleteness of this classification via *weak* equivalence, as demonstrated by explicit examples in [36], [281].

3.7.1 The Brane Web Data

The brane web data was set up as web matrices, and since focus was on 3-leg webs the W_3 matrices were 2×3, which were flattened for NN input.

Balanced datasets were generated for both *strong* and *weak* equivalences. Initially for the *weak* equivalent data, denoted $\mathbf{X}$, all consistent webs with $p, q \in [-3, 3]$ and $n \in [1, 3]$ were exhaustively generated, for each web their 3 invariants of (3.7.5) computed, the webs were then collected into their 14 weak equivalence classes and 48 randomly selected from each such that the dataset of 672 webs was balanced. For the *strong* equivalent data, denoted $\mathbf{Y}$, a single web was taken from each *weak* equivalence class (these are naturally *strong* inequivalent) and a series of $SL(2, \mathbb{Z})$, HW, and shuffle[20] moves were performed on the web to produce 48 webs, and again a balanced dataset of 672.

This process was also expanded to perform an exhaustive web data generation. Made available on the respective GitHub is a database of all consistent webs (with entries ≤ 100) which can be generated from initial webs with $p, q \leq 5$ via 2 iterations of an $SL(2, \mathbb{Z})$ and HW move (considering all $SL(2, \mathbb{Z})$ matrices with entries ≤ 5, and HW moves on all legs). This database is partitioned into the respective *strong* equivalence classes, and each of these classes partitioned into the subsequent *weak* equivalence classes, for ease of searching. There is a complementary notebook with functionality to compute invariants, compare equivalences, and generate random equivalent webs from this data.

Example: A candidate brane web considered in the learning had web matrix

$$\begin{pmatrix} -4 & 3 & 1 \\ -2 & -1 & 3 \end{pmatrix} , \tag{3.7.6}$$

such that the *gcd* multiplicities were 2,1,1 respectively for each leg (column). Following the computations in (3.7.5), this web has invariants

$$\begin{aligned} M_{total} &= \begin{pmatrix} -67 & 76 \\ -149 & 169 \end{pmatrix} , \\ \ell &= 5 , \\ d_{CB} &= 3 . \end{aligned} \tag{3.7.7}$$

Therefore the flattened web matrix as a 6-vector (-4,3,1,-2,-1,3) made up the input for the SNN embedding, or TDA directly.

3.7.2 ML Results

The primary goal of this paper was to use ML architectures to aid in the partitioning of webs into their true equivalence classes. To perform this a SNN architecture was used. The aim of SNNs is to train a NN that can effectively embed input vectors into some abstract space such that those which are similar are close in Euclidean distance in the embedding space, and those which are not similar are far away. In this context this equates to webs in the same class being close in the SNN embedding space.

The SNN set-up used trained 3 identical NNs (with the same hyperparameters and parameters) in an identical manner according to a triplet loss. This triplet loss was minimised

[20]Note that shuffle moves are an additional redundancy in the representation according to the leg ordering in the web matrix, acting with the permutation group on the matrix columns shuffles this representation trivially giving the same web.

for each triplet set, which contained an 'anchor' web, along with a 'positive' web which it was equivalent to and a 'negative' web it was not equivalent to. Minimisation of the loss would map the anchor and positive webs to the same point in the embedding space, and the negative web to a point at least the threshold distance away. Once trained, the embedding of any two input webs should exceed the threshold value if they are not equivalent, and this can be used for classification. In this work, despite there only being 672 webs per dataset, the grouping into triplet sets for training, and pairs for testing significantly inflates the amount of data for learning.

SNNs embedded the 6-dimensional web matrix data (W_3 flattened and imagined in $\mathbb{R}^6$) into $\mathbb{R}^{10}$, and could well identify whether a pair of webs was in the same *weak* equivalence class or not: (accuracy, MCC) of (1, 1) for two *weak* classes, and (0.77, 0.55) between all 14. However the SNNs had negligible success for the *strong* equivalence classes: (accuracy, MCC) of (0.5, 0) between two classes or all 14 (i.e. no learning). This motivated analytic work to check the compatibility between *strong* and *weak* equivalences and ultimately lead to explicit examples which highlighted the gaps in the classification via invariants.

Visualisation of the web data and SNN embedding was performed using non-linear $t-SNE$ methods to reduce both the $\mathbb{R}^6$ and $\mathbb{R}^{10}$ spaces to $\mathbb{R}^2$ for plotting. However, my predominant contribution was the higher-dimensional direct analysis of these raw and SNN embedded datasets through TDA's persistent homology.

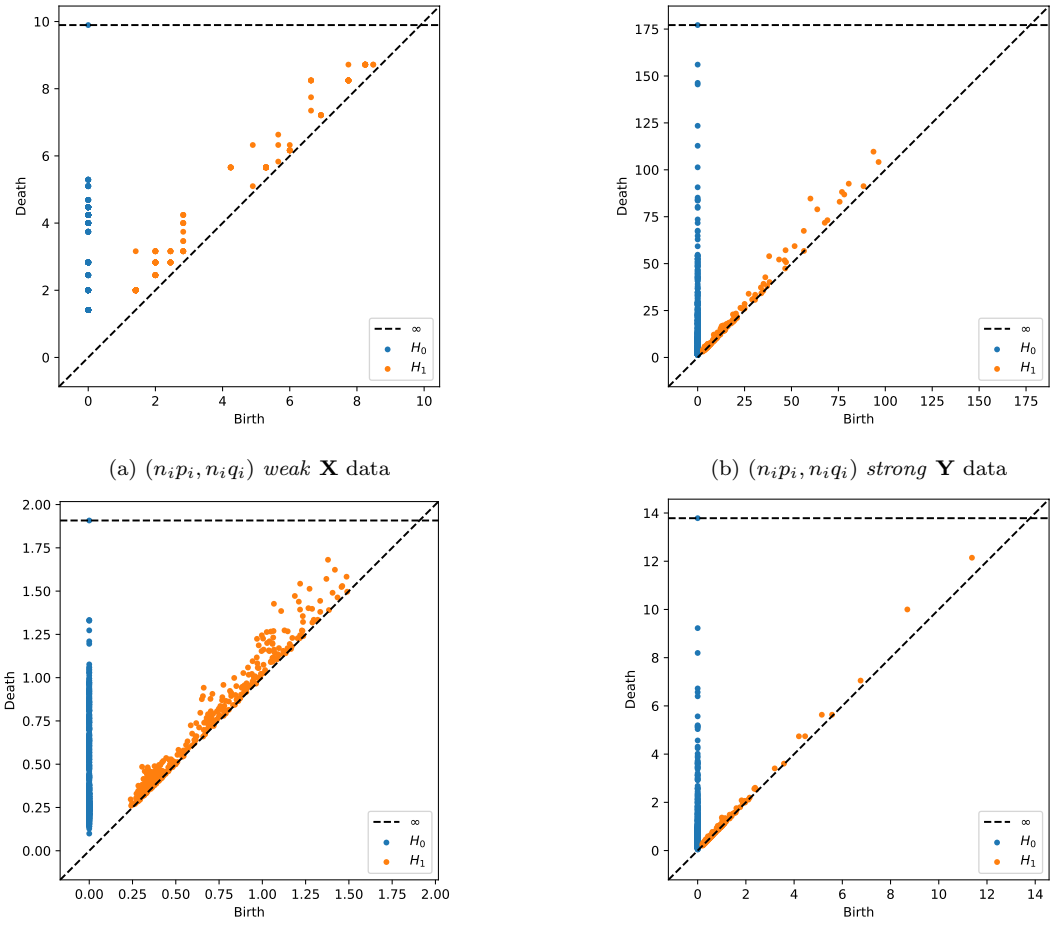
Raw Web Data: Each web in the raw web datasets $\mathbf{X}$ and $\mathbf{Y}$ amounts to the contents of the web matrices W_3. These 6 integers were plotted in $\mathbb{R}^6$ for the persistent homology analysis. Noting that, since datapoints are restricted to the integer lattice this translates to a grid-like distribution of features in the persistence diagrams, as only at specific radii can balls intersect and change the VR complex[21].

The analysis was performed for both datasets of 672 web datapoints, $\mathbf{X}$ in Figure 3.7.1a, and $\mathbf{Y}$ in Figure 3.7.1b. The HW moves in the generation process for $\mathbf{Y}$ lead to much larger web matrix entries, as reflected in the larger scales and finer grid structure in 3.7.1b. For the $\mathbf{X}$ data the roughly uniform distribution of H_0 features indicates points are uniformly distributed in the space, as expected from the exhaustive generation procedure over the search space. Conversely the $\mathbf{Y}$ data has a slightly larger gap in the line ($\sim$125-145, note the much larger scale here) indicating there are clusters of points further from the main cluster, likely a result of some HW moves jumping datapoints away from the bulk.

Both datasets have H_1 features close to the diagonal, behaviour typical of noise; since there are no features far from the diagonal there is not a significant loop structure in the data which would otherwise indicate regions either omitted from the sampling or not physically plausible.

SNN Embedded Web Data: The embedded web data is the result of the SNN's base model NN mapping all the 6-parameter web matrix data into the $\mathbb{R}^{10}$ embedding space, where the aim of the SNN is to create an embedding model which separates inequivalent webs into their respective clusters.

[21]Note also that this grid-like behaviour occurs for the projective space weight data in §3.6.2 as well, but there the larger range of integers in the data make the grid effect negligible.



(a) $(n_i p_i, n_i q_i)$ *weak* $\mathbf{X}$ data

(b) $(n_i p_i, n_i q_i)$ *strong* $\mathbf{Y}$ data

(c) SNN embedded *weak* $\mathbf{X}$ data

(d) SNN embedded *strong* $\mathbf{Y}$ data

Figure 3.7.1: Persistence diagrams for H_0 and H_1 on the web $(n_i p_i, n_i q_i)$ data and SNN embedded data, for datasets $\mathbf{X}$ and $\mathbf{Y}$ respectively.

The H_0 analysis for the embedded $\mathbf{X}$ data in Figure 3.7.1c shows a continuous stream of features indicating points merging together to form independent simplices, then the gap ($\sim$1.10-1.20) followed by a collection of features closer together indicates the separate clusters for each of the web classes combining together to finish the filtration. Since the features are relatively close they represent more symmetrically distributed clusters, nice behaviour since a priori no equivalence classes should be especially more related than others. This behaviour supports the success of the SNN embedding to separate the webs into clusters based on the equivalence used for training. The H_0 features for the embedded $\mathbf{Y}$ data, in Figure 3.7.1d, has similar behaviour, however the cluster separation is less uniform, indicated by the less consistent line where clusters aren't combining smoothly and then only a couple of features are separated from the main line (> 8), therefore some of the classes' clusters merge earlier, and are not as well separated as in the embedded $\mathbf{X}$ data, making this classification worse, and hence supporting the poorer learning results observed.

For both embedded datasets the H_1 features lie close to the diagonal, again indicating a lack of significant loop structures in the data clouds such that the clustering performs

as expected. The embedded **X** data has many more higher birth features (relative to the respective H_0 scales), likely a consequence of the better separated clusters combining later in the filtration.

Conclusion
The relative performance of supervised ML methods indicated the need to refine the classification of 3-leg brane webs (and equivalently sets of 3 7-branes) using the invariants of (3.7.5). SNNs learnt to well separate the web data according to the invariants, but could not well separate the classes of webs connected by explicit $SL(2,\mathbb{Z})$ and HW moves.

Persistent homology of the datasets highlighted the success of the embedding for the *weak* equivalent data, as well as the significance of the HW moves in more substantially changing the range of consideration for the leg sizes.

3.8 Cluster Algebras (2203.13847)

As introduced in §2.1, cluster algebras are a mathematical framework with intriguing combinatorial structure. Central to this is their generation process built from a single seed, containing a cluster of r cluster variables and an $r \times r$ exchange matrix (interpretable as a quiver adjacency matrix). Mutation on any of the r cluster variables (and the respective quiver node) updates that cluster variable to produce a new cluster (with one variable different) and also a new quiver; as dictated in (2.1.1) and (2.1.2), and exemplified in Figure 2.1.2b.

Exhaustive application of this mutation action on all variables in a seed's cluster, and then on all variables in the new seeds' clusters, and repeating exhaustively produces a (potentially infinite) set of seeds. The union of all the clusters from this set gives the full list of unique cluster variables, as generators of the underlying algebra. In a sense, the variables in the initial seed (with the quiver) generate the full list of generators for the algebra, and this condensed representation for the algebra, which can effectively be represented exclusively by an initial quiver (as the initial cluster is often just $\{x_1, x_2, ..., x_r\}$), is a defining feature of these algebras. The subsequent cluster variables produced from repeated mutation all obey a Laurent phenomenon, in that they can each be expressed as a Laurent polynomial in the initial cluster's variables (i.e. a rational function with a single monomial denominator). In particular since the mutation process in (2.1.2b) only involves addition there is a preserved positivity such that all monomial coefficients are positive integers [52].

The algebra types are classified into 3 categories: finite, finite-mutation, infinite. The finite type algebras (built using initial quivers as directed Dynkin diagrams) have a finite number of cluster variables which generate them, and hence a finite number of seeds that can be produced via mutation. The finite-mutation algebras have a finite number of exchange matrices (i.e. quivers) that can be produced via mutation, but may have infinitely many clusters and cluster variables; such that they expand the finite type to include more examples (as described in §2.1). The infinite types then are the complement to finite-mutation with infinitely many exchange matrices, and hence also infinitely many clusters, cluster variables, and seeds.

For any cluster algebra the combinatorics of its generation via mutation can be represented by an exchange graph. Each seed is represented by a vertex of the graph and an

edge connects two seeds' vertices if the seeds can mutate into each other[22]. Starting from an initial seed (at depth 0), all of the r variables can be mutated to give up to r new seeds (at depth 1), then each of these new seeds can be mutated on each of their $r-1$ variables to give up to $r(r-1)$ new seeds (depth 2), etc. In the limiting case all mutations give new seeds and the exchange graph is a tree, however mutations may produce seeds already seen and when this occurs the introduced edges start to form cycles in the graph. For these exchange graphs it may be possible to always generate new seeds via mutation (infinite and finite-mutation types) such that the graph is infinite, or there may be a maximum depth where no new seeds are generated (finite type) leading to a closed graph.

One may also construct another graph representing connectivity under mutation by considering only the quiver part of the seeds, i.e. the exchange matrix. This exchange graph represents how all the possible exchange matrices[23] are connected via mutation action. This in turn is called the *quiver exchange graph*, and is often significantly smaller than the exchange graph with cluster information at any depth. This is because many seeds may have the same exchange matrix but different clusters, making them the same vertex in the quiver exchange graph but different vertices in the exchange graph. To make clear the differentiation between these types of exchange graph we will call the graph where the seeds include the cluster information the *seed exchange graph*. These quiver exchange graphs may be infinite as more quivers can also be mutated to (infinite type) or closed where only finitely many quivers can be generated (finite-mutation and finite type).

The finiteness of these seed and quiver exchange graphs is exactly what specifies the algebra type. Analysing the combinatorics of these graphs and the frequency of repeated seed generation leading to cycles, techniques from network analysis and ML can start to probe the structure of *how* finite or infinite certain algebras are. In this respect, the finite-mutation types which are not finite act as an interesting bounding case between the finite and infinite types, where their quiver exchange graph is finite but their seed exchange graph is infinite.

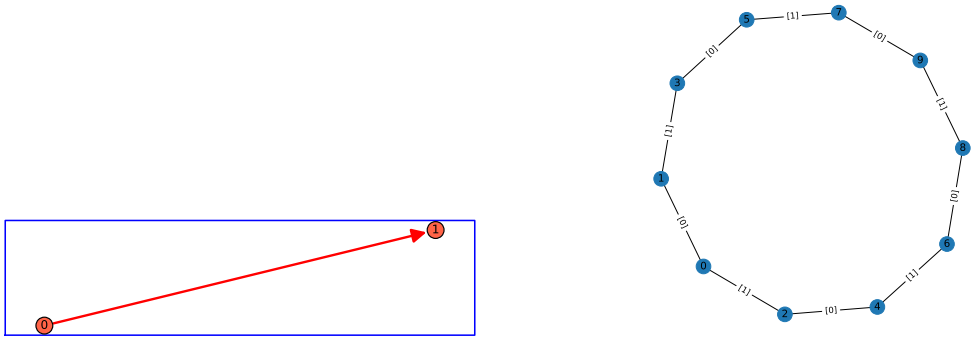
Cluster algebras have a range of applications in physics: from an interpretation of the exchange graph mutation as quiver Seiberg duality [49], [282]–[284] (as examined in §3.2), to duality extensions to incorporate more general symmetries [285], to Toda field theories [286], [287], to wall crossing [288], [289], and to scattering amplitudes [290]–[293]. Additionally, finite-mutation type cluster algebras can be related to complete quantum field theories [294]. Therefore the analysis of cluster algebras opens doors to new theories and computational efficiencies alike, providing some physical motivation for this study.

Cluster Algebra Example: The prototypical example for cluster algebras is the finite type rank 2 algebra: A_2. This has quiver given by the directed A_2 Dynkin diagram shown in Figure 3.8.1a.

Starting with the initial seed $\{x_1, x_2\}$ and the A_2 quiver, one can mutate about either of the quiver vertices. With each mutation, since the rank is too small for the quiver to include

[22]Note that as mutation is an involution the edges are all undirected.

[23]Note the repetition of 'exchange' in the nomenclature: exchange matrices define the mutation of a specific seed, and their graph is the seed's quiver; whereas the exchange graph dictates the entire generation process of all seeds of the cluster algebra.



(a) A2 quiver

(b) A2 seed exchange graph

Figure 3.8.1: The quiver for the A_2 example cluster algebra (a), as well as its exchange graph (b) where permutation equivalence is not applied. The respective clusters for each vertex in the exchange graph are: $\{0 : [x_1, x_2],\ 1 : [(x_2+1)/x_1, x_2],\ 2 : [x_1, (x_1+1)/x_2],\ 3 : [(x_2+1)/x_1, (x_1+x_2+1)/(x_1x_2)],\ 4 : [(x_1+x_2+1)/(x_1x_2), (x_1+1)/x_2],\ 5 : [(x_1+1)/x_2, (x_1+x_2+1)/(x_1x_2)],\ 6 : [(x_1+x_2+1)/(x_1x_2), (x_2+1)/x_1],\ 7 : [(x_1+1)/x_2, x_1],\ 8 : [x_2, (x_2+1)/x_1,\ 9 : [x_2, x_1]\}$, and the quiver node mutated on to connect each seed is given as the respective edge feature.

2-paths, the only change to the quiver is a reversal of the single edge's orientation. Therefore in this simple A_2 case the quiver exchange graph is just 2 vertices connected by an undirected edge to represent both these quivers (which we consider not equivalent via permutation[24]), the second quiver exchange matrix being just the transpose of the first. Furthermore in the seed exchange graph, Figure 3.8.1b, the quiver alternates between these two forms around the exchange graph's loop.

To exemplify the mutation process, consider for this A_2 algebra the mutation of the seed associated to vertex 4 in the exchange graph on the quiver vertex labelled 1 (associated to the second variable), which mutates the seed to the seed labelled 6 in the seed exchange graph. The quiver associated to this seed is the same as for the initial seed, as shown in Figure 3.8.1a. Therefore the exchange matrix is $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, which updates to its transpose under the process in (2.1.2). The cluster $[(x_1+x_2+1)/(x_1x_2), (x_1+1)/x_2]$ is respectively mutated on its second variable following the process in (2.1.1) with $k = 2$. The process hence keeps the first variable unchanged and updates the second variable where the first numerator term in (2.1.1) is just the first variable (as $EM_{12} = 1$) and second numerator term is just 1 (since there are no negative entries in the second column). Therefore $\frac{(x_1+1)}{x_2} \longmapsto \frac{(x_1+x_2+1)/(x_1x_2)+1}{(x_1+1)/x_2} = \frac{(x_2+1)}{x_1}$, matching the expected cluster for seed 6.

3.8.1 The Cluster Algebra Data

In this study, focus was on the exchange graphs of cluster algebras, considering a selection that spans the three possible types. How these graphs take shape as they are generated was analysed and application of ML techniques to study their respective cluster algebras introduced. In spirit this extends the work in [31] summarised in §3.2 where ML was applied

[24]It is common in cluster algebra studies to consider clusters which are permutations of each other to be the same, we however consider them to be independent as is later motivated.

to quiver exchange graphs to learn the underlying Seiberg duality. The `sagemath` 'Cluster Algebra and Quiver' package [234] was used to perform mutation and handle cluster algebra seeds.

The cluster algebras considered in these investigations were denoted by the respective initial seeds used to generate them. These were hence defined by a choice of exchange matrix, each paired with the initial cluster $\{x_1, x_2, ..., x_r\}$. For the purposes of considering a consistent rank with enough interesting structure, but not too large so as to require excessive computational resources, all these algebras were of rank $r = 4$. Across the 3 types, 7 specific algebras were selected. These included 3 finite type algebras, generated from orientations of the A_4, D_4, and F_4 Dynkin diagrams (denoted A4, D4, F4 respectively). Note the F_4 algebra does not lie in the skew-symmetric classification, but is a skew-symmetrisable finite type[25]. Next, 2 orientations of the affine $\tilde{A}_3$ type were used as finite-mutation type algebras which are specifically *not* finite type; such that one had 1 anticlockwise 3-path and 1 clockwise arrow (denoted A13), whilst the other had a 2-path in each direction (denoted A22)[26]. Lastly, 2 infinite type algebras (denoted $\mathcal{I}1$ and $\mathcal{I}2$ respectively) were generated from the exchange matrices

$$EM_{\mathcal{I}1} = \begin{pmatrix} 0 & 2 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}, \qquad EM_{\mathcal{I}2} = \begin{pmatrix} 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 1 \\ 2 & 0 & -1 & 0 \end{pmatrix}. \tag{3.8.1}$$

The quivers for each of these initial seed exchange matrices are shown in Figure B.1.1, available in §B.1.

Beyond the choice of Dynkin or affine Dynkin type, an orientation to each quiver must be prescribed. The `sagemath` package initiates the finite type quivers with bipartite orientations, such that each node is either a source or a sink. Whilst the ambiguity to select an orientation may appear to lose generality, where the quiver's underlying graph is a tree (as for our finite types) any orientation is mutation equivalent to any other orientation [53]. Therefore for finite types, a choice of orientation is effectively a choice of initial point to expand around in the same exchange graph. Note this does not apply for the other mutation types, hence choosing any 2 different orientations produces 2 different cluster algebras and 2 different exchange graphs (as exemplified by the 2 chosen orientations of $\tilde{A}_3$ giving the 2 different A13 & A22 algebras). These cluster algebras were chosen such that under mutation similar Laurent polynomial styles (in particular monomial coefficients) were occurring so it was non-trivial to differentiate the seed representations by eye ahead of ML.

Due to the significant growth of complexity in the infinite type Laurent polynomials with depth, especially $\mathcal{I}2$ where depth 5 could not be computed in a feasible time, the core focus of the exchange graph analysis and subsequent ML was chosen to be up to and including depth 4. In building the seed exchange graphs, clusters were not considered equivalent if their variables were the same but in a different order. As we see in later analysis identifying

[25]The skew-symmetrisable (but not skew-symmetric) exchange matrices can be represented by quivers still but require a double weighting on the respective edge to denote the exchange matrix non-symmetric entries. In the skew-symmetric case these entries are the same in magnitude, giving the edge weighting.

[26]We emphasise here the notation A22 (A13) does not relate to A_{22} (A_{13}), but an orientation of affine $\tilde{A}_3$.

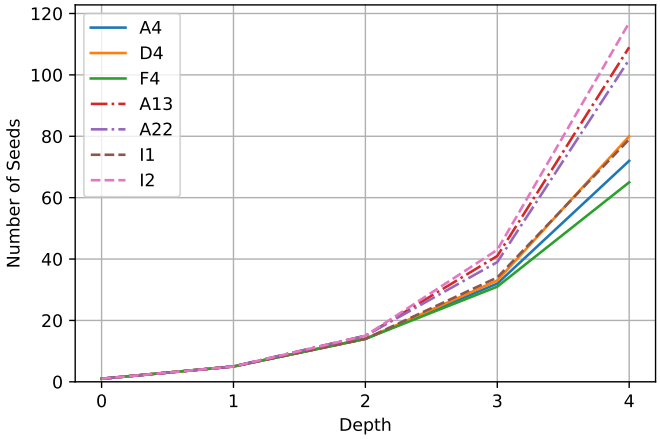


Figure 3.8.2: The number of seeds in the seed exchange graphs as depth varies for each of the considered cluster algebras, labelled by their respective initial seeds. Each type is depicted with a different linestyle.

by this permutation equivalence loses some elegant symmetric structure in the exchange graphs. We account for the possibility of this causing some degeneracy in cycles on a case by case basis and mention where that happens explicitly. Also to further motivate this choice, when defining an algebra only the cluster variables are important and hence taking the union of all clusters still produces the same generating set with or without application of this equivalence. Exchange graphs up to depth 4 are given for each of the considered cluster algebras in Figure B.2.1, available in §B.2.

Prior to any thorough data analysis of these exchange graphs, the types can begin to be distinguished by the number of seeds at each depth, as shown in Figure 3.8.2. Although it can be seen already that this information is not sufficient for classification, as more seeds are generated the finite type algebras are more likely to be reproducing previous seeds. We believe that there is a similar behaviour where finite-mutation types (that are not finite) are also more likely to reproduce seeds than infinite types, as they are on the boundary between finite and infinite.

Example: To give an example of typical seed information we arbitrarily select seed 30 from the A4 seed exchange graph in Figure B.2.1a, whose exchange matrix and cluster take the form:

$$EM_{A4:30} = \begin{pmatrix} 0 & -1 & 1 & 0 \\ 1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} , \tag{3.8.2}$$

$$\left\{ x_1, \frac{(x_1x_3^2 + x_1x_3 + x_2x_4 + x_3 + 1)}{x_2x_3x_4}, \frac{(x_2x_4 + x_3 + 1)}{x_3x_4}, \frac{(x_3 + 1)}{x_4} \right\} . \tag{3.8.3}$$

Cluster Algebra	Seed Exchange Graph Analysis (depth 4)					
	Number of Vertices	Density	Clustering (tri, squ)	Wiener Index (full, norm)	Centrality (centre, diff)	Min cycle basis ([length, freq])
A4	72	0.034	(0, 0.058)	(13968, 5.46)	(0, 0.029)	[4,17]
D4	80	0.029	(0, 0.037)	(17941, 5.68)	(0, 0.037)	[4,13]
F4	65	0.040	(0, 0.064)	(10700, 5.14)	(0, 0.031)	[4,17], [6,3]
A13	109	0.020	(0, 0.034)	(35284, 5.99)	(0, 0.054)	[4,12]
A22	105	0.021	(0, 0.016)	(32664, 5.98)	(0, 0.061)	[4,8]
$\mathcal{I}1$	79	0.031	(0, 0.065)	(17174, 5.57)	(0, 0.015)	[4,18]
$\mathcal{I}2$	117	0.019	(0, 0.037)	(41160, 6.07)	(0, 0.063)	[4,12]

Table 3.8.1: Network analysis of the seed exchange graphs (EGs) generated to depth 4 for the 7 cluster algebras considered. The first 3 algebras are finite type, the latter 2 are infinite type, and the remaining 2 are finite-mutation type but not finite type (hence having infinite seed EGs). The analysis lists: the number of vertices in the EG; the density of the EG; the triangle and square average clustering coefficients; the Wiener index (both full form and normalised form); the eigenvector centrality analysis listing the central vertex and then the size of the smallest difference in centrality from the initial seed "0" to the clusters at depth 1; and finally the information on the minimum cycle basis showing the length of the basis cycles and the frequency of those lengths in the basis.

3.8.2 Network Science Results

The intricate graph structure of the exchange graphs exemplified by Fig. B.2.1 is suggestive for an examination with network science. For each of the cluster algebras considered the seed exchange graphs generated to depth 4 were thus compared with a variety of network analysis techniques. The results of the analysis are provided in Table 3.8.1, and use a range of assessment techniques across the core themes of network analysis, including: clustering analysis, shortest path analysis, centrality analysis, and cycle basis analysis.

For each seed exchange graph (sometimes denoted EG) the number of distinct seeds (i.e. vertices in the exchange graph) up to depth 4 is given. As expected the infinite cluster algebras (including A13 and A22 which are distinctly not finite) have more total vertices, except for $\mathcal{I}1$ which is surprisingly low. This is due to the finite type algebras usually reproducing more previously generated seeds as mutation continues to higher depths.

Further vertex analysis usually considers the degree distribution, but due to the construction process for seed exchange graphs all vertices will have degree 4, except those truncated from further mutation by the depth limit. Therefore this would provide little insightful analysis for this graph style.

The exchange graph *density* then considers the number of edges, as opposed to the number of vertices. Here, the total number of edges in the graph is divided by the total number of possible edges (i.e. the number of edges in a complete graph with the same number of vertices). Due to there being 4 edges per vertex in the depths up to 3, as each cluster has 4 variables to mutate, these density scores correlate with the number of vertices. The finite types have higher densities as they are more tightly-knit graphs and are further from the more tree-like structure of the infinite types, as can be seen in the graphs of Figure B.2.1. In addition, the finite-mutation types with infinite exchange graphs have a slightly higher density than most infinite types (better represented by $\mathcal{I}2$), supporting that they are on this border of the seed exchange graphs closing up from infinite to finite. Although $\mathcal{I}1$ has a higher density, closer to the finite types at this depth, as can be seen in its seed exchange

graph, it does show the typical tree-like substructure of infinite types, with an unusual star-pattern of lines of 4-cycles forming a net-like structure between the tree subgraphs coming off them.

Clustering Analysis: *Clustering coefficients* give information about how vertices cluster within graphs. The two styles of coefficient considered here are both global in nature, and consider the number of triangles (3-cycles) or squares (4-cycles) that exist in the graph relative to the total number of possible triangles or squares that could exist.

For all the algebras there were no triangles in the seed exchange graphs. This is expected since it would require seeds to either jump mutation depths or connect the same depth. Both these scenarios are not possible since any 2-path of connected vertices either spans 3 depths so closing this 2-path into a 3-cycle would require the mutation that this closing edge represents to jump a depth (i.e. mutating two variables simultaneously), or the 2-path has the initial seed as its centre vertex and hence the vertices which need to connect would both be depth 1 and unable to mutate to each other, as they are both a different mutated-variable from the initial seed, hence again requiring a double mutation.

This idea generalises to all odd size cycles. Since each mutation changes the depth of seed under consideration, for any sequence of mutations to close into a cycle the number of mutations increasing depth must equal the number reducing the depth. Therefore all cycles must be of even length for these seed exchange graphs.

However all the algebras have a non-zero square clustering coefficient. Squares, or 4-cycles, in a seed exchange graph indicate the scenario where two mutation actions commute; i.e. a seed can have two variables mutated in either order to produce the same seed. The frequency of this commutative action interestingly does not appear to correlate with the algebra type, exemplified by D4 having the same coefficient as $\mathcal{I}2$ and A13 having over double the coefficient value of A22.

Shortest Path Analysis: The shortest path analysis carried out comes in the form of the *Wiener index*. This index computes the sum of the shortest paths between all pairs of vertices (the full form), which we also normalise by the number of pairs of vertices, $nC2$, to give an average shortest path between vertices. The normalised form is more useful for comparison and indicates that infinite types (excluding $\mathcal{I}1$) have vertices further separated on average. This measure provides an indication of the frequency of cycles, and also the placement of them. If they are more spread over the graph (as for F4) they are more useful in shortening the shortest paths between outer vertices.

Centrality Analysis: Centrality of a network determines the natural centre vertex. Due to the generation process being from an initial seed to some depth, this would make the initial seed (labelled 0) the logical choice. However, experimentation with some lower rank algebras showed this was not always the case so it is good to confirm for these algebras considered. The centrality measure used was *eigenvector centrality*, which performs eigendecomposition of the graph's adjacency matrix, then taking the eigenvector with the largest eigenvalue, whose entries indicate the relative importance of each vertex in the network. In addition to the computed centre vertex (highest respective entry), the difference in centrality between

the centre initial seed and the most central seed at depth 1 is given. Where this difference is large, as for most of the infinite types, the initial seed is a more obvious centre. However, the difference is remarkably small for $\mathcal{I}1$, which we believe to be attributed to the symmetry of 4-cycles surrounding the centre cycle with vertices labelled from [0,4,7,1].

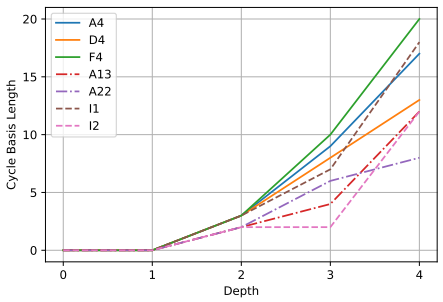
Cycle Basis Analysis: Where the exchange graphs diverge from the limiting case of an infinite tree, cycles are introduced to the graph as seeds are reproduced through mutation. As more cycles are added the graph begins to close, becoming 'more finite' in the process with less new seeds produced per mutation. It is hence this cycle structure we believe to be key to what dictates the cluster algebra behaviour, correlating loosely with the algebra type. Not only the frequency of cycles, but their distribution of sizes is an area of particular interest.

To analyse the cycle structure of each seed exchange graph, the *minimum cycle basis* for each was computed. The minimum cycle basis puts emphasis on selecting basis cycles with the lowest lengths, which can then be summed by symmetric difference in the cycle vector space to produce any cycle in the graph. Since the symmetric difference of any 2 even cycles is also even (s-cycle + t-cycle with u overlapping edges = $(s + t - 2u)$-cycle), this corroborates the idea that all cycles being even is self-consistent. A corollary of this is that all cycles in any seed exchange graph basis will be even also. In Table 3.8.1, the structure of the minimum cycle bases for each algebra is given. Since each cycle space can have many legitimate bases, the actual basis content was not of focus; instead the number of cycles required ('freq') and of what sizes ('length') provide the interesting information for analysis.

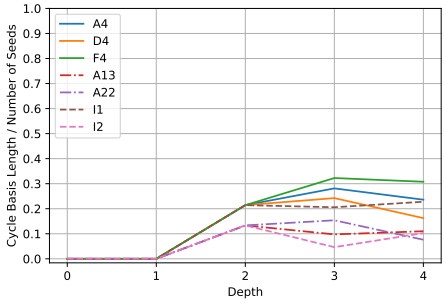
It turns out for most of these algebras, that the cycle spaces at this depth can be constructed from exclusively 4-cycles (this does not hold in general at higher depths). 4-cycles are the natural lower bound of cycle sizes in seed exchange graphs. This is because any mutation sequence (longer than the trivial involution 2-cycle for all edges) has to mutate on any vertex at least twice to change it back, hence 4-cycles being the minimum where two mutations on different vertices commute. However, F4 requires 3 6-cycles to define the cycle space up to depth 4, implying that its exchange graph structure is distinct from the others with a more subtle design. This may be related to the skew-symmetrisable vs skew-symmetric style of the exchange matrices, but more investigation into this would be required for a range of skew-symmetrisable matrices before concluding any particular pattern.

Interestingly the finite-mutation types A13 and A22 have the smallest frequency of 4-cycles in their basis, expectedly less than the finite types whose graphs are compact, however also less than the infinite types. This may be due to a more systematic occurrence of cycles, which are further apart and hence fewer are visible at this depth; whereas the infinite graphs have a skewed distribution of cycles where there is some iterative commutativity via 4-cycles.

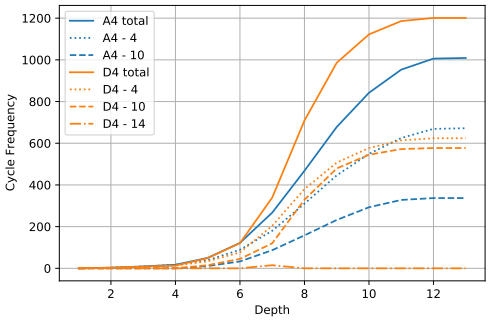
How the lengths of the cycle bases vary with depth (up to the computable depth 4 for infinite type, and beyond for the finite types) is given in Figure 3.8.3. Since all basis cycles were of length 4 except for 3 6-cycles in the F4 case, the total number of cycles in the basis is plotted instead of differentiating these cycle lengths. It shows that there is some noise as basis size grows at these lower depths for the algebras, and thus particularly for the infinite types it would be useful to find more efficient ways of computing the seed exchange graphs to probe higher depths. The basis size with respect to the number of seeds is also shown;



(a) Cycle basis size



(b) Relative cycle basis size



(c) A4 & D4 cycle bases sizes

Figure 3.8.3: The size of the minimum cycle basis for each of the considered cluster algebra's seed exchange graphs as depth varies is plotted directly in (a), and the cycle basis size relative to the number of seeds is plotted in (b). Moreover, (c) focuses on the finite type A4 and D4 algebras with seed exchange graphs generated to their maximum depths; beyond depth 4 10-cycles are introduced so each cycle length frequency is denoted separately, as well as the total frequencies.

since behaviour seems to show a plateau in the ratio growths, this indicates there is likely a limiting fixed proportion of new seed introduction versus new cycle introduction as more depths are generated.

Interestingly, the A4 and D4 finite type algebras have similar behaviour up to depth 5. This is the depth at which 10-cycles are permitted, and beyond this depth the behaviour diverges with D4 having consistently more total cycles due to the considerably larger number of 10-cycles. This indicates that these algebras have fundamentally different construction; A4's more significant structural reliance on 4-cycles may be due to more mutation operations commuting which in turn is likely due to the lower connectivity of the original A4 quiver, translating to subsequent quivers also. We note there is an anomalous 15 14-cycles occurring for D4 at depth 7: an artefact of the depth truncation that immediately splits into 4- and 10-cycles at the next depth.

Additionally, dual to the cycle vector space is the cut vector space; and although an interesting alternative method of network analysis, its application to exchange graphs is left to future work.

3.8.2.1 Generalised Associahedra

The cluster variables for the infinite type algebras quickly become highly complex with mutation – this prevents generation of the seed exchange graphs to large depths. However

Cluster Algebra	Generalised Associahedra Analysis					
	Number of Vertices	Density	Clustering (tri, squ)	Wiener Index (full, norm)	Centrality (centre vertex)	Min cycle basis ([length, freq])
A4	1008 (13)	0.0040	(0, 0.080)	(3881976, 7.65)	*no centre*	[4,672], [10,337]
B4	420 (10)	0.0095	(0, 0.077)	(542400, 6.16)	*no centre*	[4,270], [6,60], [10,91]
C4	420 (10)	0.0095	(0, 0.077)	(542400, 6.16)	*no centre*	[4,270], [6,60], [10,91]
D4	1200 (12)	0.0033	(0, 0.072)	(5150592, 7.16)	*no centre*	[4,624], [10,577]
F4	420 (10)	0.0095	(0, 0.072)	(536816, 6.10)	*no centre*	[4,252], [6,111], [10,58]

Table 3.8.2: Network analysis of the generalised associahedra for the finite type cluster algebras, labelled by their respective initial seeds. The analysis lists: the number of vertices in the exchange graph (EG) – with depth to generate these in brackets; the density of the EG; the triangle and square average clustering coefficients; the Wiener index (both full form and normalised form); the eigenvector centrality analysis listing the central vertex (or lack of); and finally the information on the minimum cycle basis showing the length of the basis cycles and the frequency of those lengths in the basis.

the finite type algebras do not suffer as severely from this behaviour, and due to their finite-ness their entire seed exchange graphs ('generalised associahedra') of all seeds can be completely generated.

Therefore for the rank 4 finite type cases we generate their generalised associahedra, and perform similar network analysis on them all. Beyond the A4, D4, F4 cases considered previously, we also introduce the B4 and C4 cluster algebras. These are also finite type, and as they arise from non-simply laced Dynkin orientations, they are not skew-symmetric but skew-symmetrisable. This Dynkin terminology means simply-laced edges have weight one, whilst the non-simply laced edges have a double weighting given by the non-skew-symmetric exchange matrix components. Their respective quivers are shown in Figure B.1.2 in §B.1, and their network analysis along with A4, D4, & F4 in Table 3.8.2.

Interestingly, all the skew-symmetrisable quivers generated algebras with the same number of seeds and density at the same depth. Further to this, B4 and C4 have identical analysis values for all measures, indicating the swap of quiver edge weighting separating them has no effect on the seed exchange graph. F4 has some differences, with slightly fewer squares but a better connectivity (via Wiener index), supported by more 6-cycles and fewer 4-cycles in the basis. The B4 and C4 algebras actually have the same seed exchange graph; however the Laurent polynomials of the variables take different values and hence the seeds at each vertex are very different [234], [295].

Conversely, the A4 and D4 algebras have similar numbers of clusters and densities, despite D4 having far fewer 4-cycles and many more 10-cycles in its basis. Since D4's fundamental structure relies more on 10-cycles, these provide quicker paths between components of the generalised associahedra leading to a smaller normalised Wiener index.

Gratifyingly, all these generalised associahedra have no discernible centre via eigenvector centrality, such that the dominant eigenvector has all its entries equal for each algebra. This supports that any seed may be used to generate each algebra symmetrically, as these generalised associahedra form complicated polytopes built from 4- and 10-cycles; and in the skew-symmetrisable cases 6-cycles become a necessity too.

Seed Equivalence: Since only the cluster variables are required to define a cluster algebra, once all variables have been generated the quivers and clusters themselves become superfluous

Cluster Algebra	A4	B4	C4	D4	F4
N	42	70	70	50	105
N'	1008	420	420	1200	420
N'/N	24	6	6	24	4

Table 3.8.3: The rank 4 finite type cluster algebras considered whose generalised associahedra are generated in full. N is the number of seeds up to identification by the permutation equivalence, N' the number of seeds generated without the permutation equivalence, and the final row the factor between them.

information. Therefore usually when one considers clusters they consider them equivalent up to permutation of the variables within the cluster, where the exchange matrix (and hence quiver) must be permuted in the same way. However, we find from some testing that different numbers (i.e. $\leq r!$) of permutations of clusters are produced within each algebra. Therefore by identifying under the full permutation group some combinatorial structure of the generation process is lost.

For example, for the generalised associahedra considered here, which the entire seed exchange graphs can be generated for, one expects the number of clusters (up to permutation equivalence), N, to follow the relation

$$N = \prod_{i=1}^{n} \frac{e_i + h + 1}{e_i + 1} = \prod_{i=1}^{n} \frac{d_i + h}{d_i} \ , \tag{3.8.4}$$

for e_i the exponents and d_i the degrees (of polynomial invariants) of the considered Dynkin type's root system (of which there are n), and h the Coxeter number [296]–[298]. For each of the rank 4 finite type algebras considered, the number of clusters up to permutation equivalence, N, is given in Table 3.8.3. One may then naively expect all permutations of the seeds to occur in the exchange graphs generated when this permutation equivalence is not identified by. However, although the A4 and D4 algebras do have all $4! = 24$ permutations, the non-simply laced types do not. The number of seeds without identification by permutation equivalence that we generate in the full generalised associahedra, denoted N', are repeated in Table 3.8.3 also for reference.

As exemplified by the rank 4 cases, the non-simply laced cluster algebras (B4, C4, F4) do not generate all permutations of seeds from mutation about an initial seed. In fact all permutations are allowed amongst the simply-laced components, where components are defined to be the sets of quiver nodes (and hence cluster positions) only connected by simply-laced edges, i.e. different components are connected by non-simply laced edges. Therefore any cluster $\{x_1, x_2, x_3, x_4\}$ will be mutation equivalent to $\{x_1, x_3, x_2, x_4\}, \{x_2, x_1, x_3, x_4\}, \{x_2, x_3, x_1, x_4\}$, $\{x_3, x_1, x_2, x_4\}, \{x_3, x_2, x_1, x_4\}$ for the B4 and C4 algebras; whilst mutation equivalent to $\{x_1, x_2, x_4, x_3\}, \{x_2, x_1, x_3, x_4\}, \{x_2, x_1, x_4, x_3\}$ for the F4 algebra. We emphasise here that this behaviour holds for all clusters of any variables (i.e. including those which are Laurent polynomials of the initial variables), hence occurring with permutation frequencies given by the relevant factor.

Beyond the rank 4 cases considered we find through experimentation up to rank 5 that the factor N'/N is $r!$ for the A_r and D_r types, $(r-1)!$ for the B_r and C_r types (anticipating these to also hold $\forall\, r$), and 1 for G_2 (where all 8 clusters are different combinations of cluster

variables). We therefore predict due to the lack of non-simply laced edges that the E_r types have an $r!$ factor too (for $r \in \{6, 7, 8\}$), despite them being too large to computationally generate in full.

This factor for the non-simply laced types is related to their skew-symmetrisable exchange matrices. Since the skew-symmetriser diagonal matrix in the skew-symmetrisation process will have a non-unit factor associated to the non-simply laced edge, and since the skew-symmetriser matrix is preserved under mutation [299], any mutation involving a non-simply laced edge will cause the variables crossing it to pick up some non-trivial power that cannot be cancelled in this other component. Therefore any cluster variable is restricted in its current form to only appear in its current component (separated by a non-simply laced edge to other components). We also note that whilst mutation may introduce more non-simply laced edges, these will only ever connect nodes across the different components, and hence the above reasoning still applies for permutations of clusters within the components, which are dictated by the initial seed.

This subtle combinatorial structure of cluster mutation and the algebra generation process is lost under the identification by permutation equivalence. Since our primary focus here is the exchange graphs built from these clusters, we think it best to consider the full structure for each algebra without the permutation equivalence applied, with the idea that once this full exchange graph is generated one can then still take all independent variables from all clusters (which already have a lot of overlap) to retrieve the algebra's generators. This viewpoint also reveals other unanticipated structure in the exchange graphs as detailed in the following section.

3.8.2.2 Quiver Exchange Graphs

Whilst the previous subsection analysed the seed exchange graphs, this subsection shifts focus to the smaller quiver exchange graphs. For these graphs the combinatorics of the generation is less limited by the cluster mutation rule. For comparison the same network analysis methods were applied to the considered algebras' quiver exchange graphs up to depth 4, as shown in Table 3.8.4.

We can see that for all algebras there are fewer vertices and a higher density (despite $\mathcal{I}_2$ being very similar) in these quiver exchange graphs relative to the seed exchange graphs of Table 3.8.1; this represents the expected behaviour from multiple seeds with different clusters having the same quivers. Importantly, whereas where clusters are present no triangle 3-cycles can occur, when considering only quivers these triangles are possible, shown by a non-zero triangle clustering coefficient for A22. Additionally there are more 4-cycles shown by consistently higher square clustering coefficients, which are likely used to provide alternative routes and cause the consistently smaller Wiener indices. It is worth emphasising also that D4 has a considerably higher square clustering coefficient.

Most interestingly about the centrality analysis is that for the $\mathcal{I}_1$ algebra, the centre is no longer the initial seed! This indicates that the extra quiver identification when the cluster information is omitted is not symmetric about the centre, and this behaviour is reflected in some of the other algebras too where the smallest difference to depth 1 is usually lower without the cluster information in the quiver exchange graphs.

Finally, the minimum cycle bases have quite non-trivial changes. Most notably, the cycles

Cluster Algebra	Quiver Exchange Graph Analysis (depth 4)					
	Number of Vertices	Density	Clustering (tri, squ)	Wiener Index (full, norm)	Centrality (centre, diff)	Min cycle basis ([length, freq])
A4	52	0.048	(0, 0.066)	(6870, 5.18)	(0, 0.036)	[4,13]
D4	41	0.071	(0, 0.251)	(3463, 4.22)	(0, 0.001)	[4,15], [7,3]
F4	40	0.072	(0, 0.098)	(3334, 4.27)	(0, 0.030)	[4,14], [6,2], [8,1]
A13	70	0.036	(0, 0.041)	(12826, 5.31)	(0, 0.020)	[4,9], [6,8]
A22	50	0.067	(0.080, 0.108)	(4780, 3.90)	(0, 0.029)	[3,8], [4,15], [7,2], [8,8]
$\mathcal{I}1$	61	0.044	(0, 0.134)	(9456, 5.17)	(1, -)	[4,18], [6,2]
$\mathcal{I}2$	107	0.020	(0, 0.040)	(33900, 5.98)	(0, 0.061)	[4,10]

Table 3.8.4: Network analysis of the quiver exchange graphs generated to depth 4 for the 7 cluster algebras considered, labelled by their respective initial seeds. The analysis lists: the number of vertices in the EG; the density of the EG; the triangle and square average clustering coefficients; the Wiener index (both full form and normalised form); the eigenvector centrality analysis listing the central vertex and then the size of the smallest difference in centrality from the initial quiver "0" to the quivers at depth 1 (when the initial seed is the centre); and finally the information on the quiver exchange graph minimum cycle basis showing the length of the basis cycles and the frequency of those lengths in the basis.

are no longer always even, as there is extra redundancy where quivers can mutate to other quivers in the same depth. In some cases the cycle bases have more cycles (D4, A13, A22, $\mathcal{I}_1$), and sometimes fewer (A4, F4, $\mathcal{I}_2$). This behaviour is curious and highlights the subtleties of the quiver exchange graph's embedding in the seed exchange graph. Something we now analyse more explicitly.

3.8.2.3 Exchange Graph Embedding

To truly probe the relationship between quiver mutation as used in Seiberg duality and cluster algebras, one needs to know the behaviour of how the mutation process reduces for each algebra as the cluster information is dropped from the seed. This is therefore intrinsically linked to how each algebra's quiver exchange graph embeds in its respective seed exchange graph.

Any embedding fundamentally depends on the number of vertices in each graph. Therefore, a first look at this embedding considers the vertex embedding of the quiver exchange graphs into the seed exchange graphs for all the types, generated up to depth 4. To examine this, the number of vertices in the quiver exchange graph relative to the number in the respective seed exchange graph is plotted for each considered algebra in Figure 3.8.4. The figure shows that up to depth one there is an isomorphism between each considered algebra's quiver/seed exchange graphs, since all initial mutations will change a different variable in the cluster; they hence here also always change a different node for the exchange matrix such that there are 5 distinct seeds (with distinct quivers) up to depth 1 in each case. To qualify this, since all vertices are connected, the mutated vertex will always have an orientation flip of its connected arrows, hence always changing the exchange matrix by this at least.

Beyond depth 1 the ratios drop off below 1, with the pure infinite cases dropping off the least. These infinite cases therefore have far more distinct quivers, as may be expected since different quivers lead to different mutation processes and are hence more likely to produce new cluster variables, a defining feature of infinite types.

To probe this further we wish to lose the limiting behaviour of the depth truncation, and

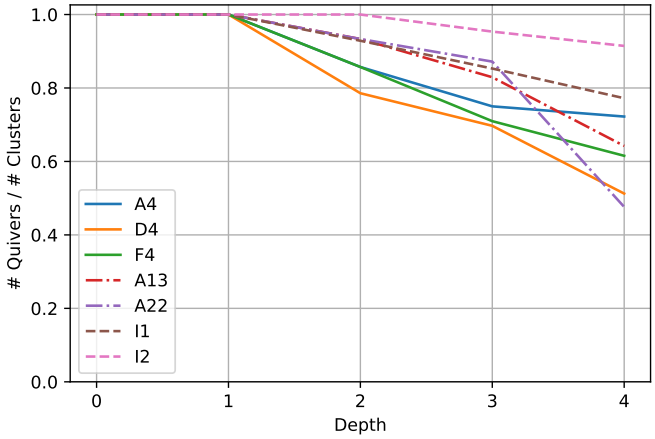


Figure 3.8.4: The relative number of quivers to clusters in the respective exchange graphs as depth varies up to depth 4 for each of the considered cluster algebras, labelled by their respective initial seeds. Each type is depicted with a different linestyle.

for that we return to the finite type generalised associahedra where both seed and quiver exchange graphs are finite and can hence be generated in their entirety. Therefore after the quiver generalised associahedra were generated completely for all the rank 4 finite type algebras (as in §3.8.2.1), the connection to the seed exchange graphs was analysed through the embedding of vertices and cycles.

Generalised Associahedra Vertex Embedding: To examine the embedding of the vertices we look at all the seeds in the seed exchange graph (each a vertex) which have the same quiver, hence matching to a single vertex in the quiver exchange graph. This is initially considered through the ratio of number of vertices between each algebra's seed and quiver generalised associahedra. For these ratios, when identifying by the permutation equivalence there is no consistent pattern, however when instead one does not identify in this way, some beautiful structure emerges. To illustrate this we show the number of vertices in the seed and quiver generalised associahedra, as well as the ratios between these numbers, for the rank 4 finite type algebras in Table 3.8.5.

A priori, one may not expect all these ratios to be integer. When each seed in the seed generalised associahedra has its cluster information removed to leave just the quiver, there are additional identifications to be made amongst vertices where seeds with different clusters have the same quiver. However, there is no obvious requirement for all the quivers to occur the same number of times (which is what we see and leads to the concurrent identification of each set of the 'ratio' number of quivers and hence an integer ratio overall).

These ratios take somewhat surprising and perhaps unintuitive numbers, with no clear foundation in the Dynkin construction. To further probe this behaviour, these ratios were also computed for the finite type algebras (arising from the 4 Dynkin series) for all ranks up to rank 5 in Table 3.8.6, and show signs of further alluring structure. Conjectured natural continuations of these observed ratios are provided for higher ranks also, noting that D_4 appears to be anomalous in its series, likely related to triality of its initial quiver.

Cluster Algebra	A4	B4	C4	D4	F4
Number of Quivers	144	84	84	50	60
Number of Seeds	1008	420	420	1200	420
Ratio	7	5	5	24	7

Table 3.8.5: The number of vertices in the quiver exchange graph and seed exchange graph for each of the rank 4 finite type algebras considered. Their ratios are also listed, all taking integer values.

Cluster	Rank					
Algebra	1	2	3	4	5	$r \geq 6$
A_r	2	5	6	7	8	$r+3$
B_r	-	3	4	5	6	$r+1$
C_r	-	3	4	5	6	$r+1$
D_r	-	4	6	24	10	$2r$

Table 3.8.6: The ratios between number of seeds in the seed exchange graph and number of quivers in the quiver exchange graph for the finite type cluster algebras, not applying permutation equivalence between seeds/quivers. Conjectured relationships are shown for higher ranks, r, beyond current feasible computation.

For completeness, we provide the G_2 ratio: 4, reiterate that the F_4 ratio is 7, and note that the remaining E_6, E_7, E_8 exceptional cases are too high a rank to be feasibly computed with current resources (leaving this to future work).

We finally reemphasise that if one identifies via the permutation equivalence some of this intriguing behaviour is lost. If one does identify in this way, some integer ratios do occur sporadically for the A_r and D_r series ($A_4 : 7, D_5 : 7$) beyond the trivial rank 2 cases where there is only one quiver. We believe this to be a probabilistic artefact occasionally carried over through the permutation identification, where there are few factors to choose from when dividing the number of seeds in these smaller rank algebras under permutation equivalence identification, so maintaining the integer ratio is more likely. The F_4 and G_2 ratios remain 7 and 4 respectively after the permutation equivalence identification; and for the B_r and C_r series all the ratios stay the same!

3.8.2.4 Generalised Associahedra Cycle Embedding: Already the vertex embedding of the quiver exchange graph into the seed exchange graph reveals an extraordinary structure. As motivated when examining all algebra types at depths up to 4, the cycle space also acts as a foundation of the graph structure and an important tool for analysing these embeddings.

Let us consider an s-cycle of quivers in a quiver exchange graph, i.e. there is a sequence of s mutations connecting each quiver exchange graph vertex to the next, eventually reproducing the quiver started with. Then considering the same algebra's seed exchange graph and taking the subgraph of seeds which have the same quivers as in the s-cycle produces a subgraph built out of q t-cycles, where $t = ps$ and $p, q, s, t \in \mathbb{Z}_+$. We call p the scale factor, as it describes how the size of the cycle scales, and q the copy factor, as it dictates how many copies of the cycle are produced.

Looking at all the algebras we consider, the values of p and q change depending on the

Cluster Algebra	QEG MCB [[len,freq]]	Cycle scale factor p [[value,freq]]	Cycle copy factor q [[value,freq]]
A4	[[4,108],[6,8],[10,29]]	[[1,90],[7,55]]	[[1,55],[7,90]]
B4=C4	[[4,60],[6,15],[8,6],[10,4]]	[[1,49],[5,36]]	[[1,36],[5,49]]
D4	[[4,33],[7,12],[8,6]]	[[1,15],[2,2],[4,25],[6,6],[12,3]]	[[2,3],[4,6],[6,25],[12,2],[24,15]]
F4	[[4,42],[6,14],[8,4],[10,1]]	[[1,31],[7,30]]	[[1,30],[7,31]]
A3=D3	[[3,6],[8,2]]	[[3,2],[6,6]]	[[1,6],[2,2]]
B3=C3	[[3,4],[5,2]]	[[4,6]]	[[1,6]]

Table 3.8.7: The embedding of the quiver exchange graph minimum cycle basis (QEG MCB) into the respective seed exchange graph for the rank 3 & 4 finite type cluster algebras. The embedding information is listed as the p & q values and frequencies that dictate how each cycle scales in size and copies respectively.

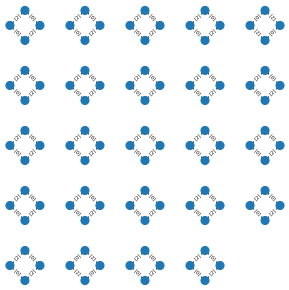
cycle in the quiver exchange graph considered. To best illustrate this we again focus on the generalised associahedra of the finite type rank 4 algebras, where the entire subgraph corresponding to the chosen cycle can be computed, and the value of pq remains constant, equalling the respective 'ratio' values calculated previously.

The embedding of each quiver exchange graph cycle can be considered by taking the subgraph of all seeds in the seed exchange graph that have a quiver from the cycle under consideration. For all the rank 3 and 4 finite type cluster algebras the distribution of p and q values for the cycles in each algebra's minimum cycle basis are given in Table 3.8.7. The minimum cycle basis was used here as it is a sensible set of independent cycles of different sizes to probe the embedding behaviour.

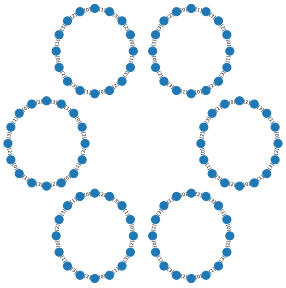
As can be seen, the p and q values are not the same for all cycles in each algebra, but matching up the frequencies (and computationally confirmed explicitly) shows that the pq ratio is always constant at the value listed in Table 3.8.6. Due to the requirement that all cycles in the seed exchange graph are even, if the quiver exchange graph cycle being embedded is odd then p has to be even, which is well exemplified with the rank 3 cases where for A3 the 6 3-cycles have $p = 6$, and all the B3 cycles have $p = 4$ (i.e. $p = 1$ cannot occur). Note this breaks the symmetry of the ratio being split into its factors where p and q can take either factor's value, as seen for the even cycles in the table.

To provide some explicit examples of the cycle embedding we focus on the D4 algebra. The embedding of 3 different quiver exchange graph cycles in the seed exchange graph are given in Figure 3.8.5. As can be seen, the two 4-cycles have different (p, q) values, and the larger 7-cycle has an even p value.

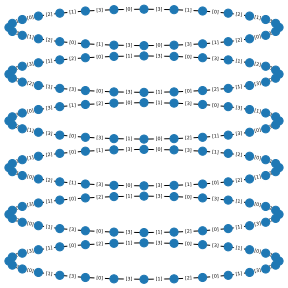
Of particular note is the 4-cycle with seed exchange graph embedding shown in Figure 3.8.5a. This quiver exchange graph 4-cycle comes from commuting action of mutation on two different vertices, which in the quiver happens when the vertices mutated about are not connected. When any 4-cycle (in any cluster algebra) is built from commuting action on disconnected vertices, the respective embedding in the seed exchange graph always has $p = 1$, such that the mutation remains a commuting relation with the cluster information added for all clusters with those quivers. Since the two mutated quiver vertices are not connected, then no edge connecting them is introduced, and in the cluster each new variable from a mutation will not include the variable from the unconnected vertex. This does not hold for all generic 4-cycles, as exemplified in Figure 3.8.5b where the quiver 4-cycle comes from mutation on different vertices and has $p > 1$; therefore only when the cycle is from



(a) QEG 4-cycle ($p = 1, q = 24$)



(b) QEG 4-cycle ($p = 4, q = 6$)



(c) QEG 7-cycle ($p = 4, q = 6$)

Figure 3.8.5: The seed exchange graph embedding of 3 selected cycles from the quiver exchange graph (QEG) for the D4 cluster algebra. (a) & (b) are 4-cycles, which when embedded become 24 4-cycles and 6 16-cycles respectively; whilst (c) is a 7-cycle which becomes 6 28-cycles. Each embedded cycle subgraph is shown with edge labels indicating the respective mutating quiver vertex, which connects the seeds, forming multiples of the quiver 4 and 7-cycles respectively.

commuting action will $p = 1$.

In addition, whilst the finite type cases have constant pq, putting an upper-bound on the p and q values for all cycles in the quiver exchange graph, in the finite-mutation but not finite, and infinite types, there is no such upper bound. In fact, particularly for the finite-mutation but not finite cases, since there are finitely many quivers but infinitely many seeds, for some cycles either (or both) of the p or q values must be infinite.

From examining these finite type cluster algebras we have seen that the pq value remains constant for all cycles; where the quiver cycle is odd then p must be even, and where a quiver exchange graph 4-cycle is from commuting action of mutation then $p = 1$. How the p and q values are determined more generally for each cycle in each algebra, particularly finite-mutation and infinite types, is left open for future exploratory work.

3.8.3 ML Results

Given two random seeds it is unclear as to whether a sequence of mutations exists which connects them, i.e. whether they belong to the same cluster algebra. Beyond simple checks for the mutation type (a necessary but not sufficient condition), brute force computation of all mutations is the usual method for checking this equivalence. However, brute force mutation is extremely computationally expensive, as particularly emphasised by the infeasibility of

computing the exchange graph for $\mathcal{I}2$ beyond depth 4.

To speed up this process of checking mutation equivalence, here ML methods are applied to this problem, with the idea they may be able to find invariants in the variables' Laurent polynomial structure under mutation and use it for this speeding up of equivalence evaluation.

In this work, the NNs used had 3 layers of 256 neurons with ReLU activation and a single sigmoid output neuron, learning the binary cross-entropy loss with the Adam optimiser on these binary classification problems. Learning performance was measured with the accuracy and MCC metrics, and 5-fold cross-validation was performed such that the metrics could be averaged and standard error calculated to provide confidence in the results.

Each cluster must be represented with a tensor for NN input, and within that each cluster variable Laurent polynomial must be represented. A 'coo' inspired style of representation was enacted for each monomial in the numerator and denominator of the variable's polynomial, represented as a 5-vector of entries $[c, \alpha, \beta, \gamma, \delta]$ for each monomial $cx_1^\alpha x_2^\beta x_3^\gamma x_4^\delta$. The 5-vectors are then concatenated, with the denominator single monomial 5-vector at the end. This representation style had improved sparsity compared to other more physical representation styles tried.

After all tensors were generated for each variable in a cluster, they were flattened and concatenated across the cluster with the flattened exchange matrix also, to produce a single data vector per seed. Note that sometimes the exchange matrix was omitted from the representation in order to examine the ML performance based on only the clusters, i.e. can the NNs distinguish the cluster algebra from just sets of generators (or is the exchange matrix needed as well in order to learn the full generator set structure). Importantly though, the initial seed's cluster is the same for all the algebras, along with further repetition of variables in other clusters between algebras; this leads to some redundancy in information where the same tensor may be affiliated to multiple algebras. This *should* make the learning noticeably harder without the exchange matrix information, as some data will mislead the learning.

Since different variables have different numbers of monomial terms, the length of the seeds' vectors varies substantially. To create a consistent input length for all vectors in an investigation the tensors were post-padded with zeros such that all vectors were the same length as the longest in that investigation. As the infinite type mutations lead to more complex Laurent polynomials with far higher degrees, the respective vectors for these higher-depth outer seeds are much longer than all others across all the algebras. Therefore in order to stop this dilution of the information for ML comparison between non-infinite algebras, the investigations were designed to be binary classifications.

For all ML investigations, seeds in the same algebra were not considered identical if they were related via a permutation of variables in the cluster. Due to the systematic vector generation procedure this led to different vectors for different permutations, and one may consider this process as the common practice of data augmentation on the permutation invariant algebra seeds if preference is to consider these over those where different permutations are unique.

Example: The example A4 cluster considered in (3.8.3) would hence have vectors corresponding to each cluster variable as

$$\begin{aligned} x_1 &\longmapsto [1,1,0,0,0|1,0,0,0,0]\ , \\ \frac{(x_1x_3^2+x_1x_3+x_2x_4+x_3+1)}{x_2x_3x_4} &\longmapsto [1,1,0,2,0|1,1,0,1,0|1,0,1,0,1|1,0,0,1,0|1,0,0,0,0|1,0,1,1,1]\ , \\ \frac{(x_2x_4+x_3+1)}{x_3x_4} &\longmapsto [1,0,1,0,1|1,0,0,1,0|1,0,0,0,0|1,0,0,1,1]\ , \\ \frac{(x_3+1)}{x_4} &\longmapsto [1,0,0,1,0|1,0,0,0,0|1,0,0,0,1]\ , \end{aligned} \tag{3.8.5}$$

noting the final set of 5 in each vector is the denominator. These would all be concatenated, perhaps with the flattened exchange matrix in (3.8.2), and padded to form the vector input for the NN.

3.8.3.1 Distinguishing Cluster Algebra Types

The first investigation uses the above described NN architecture to classify seeds coming from different cluster algebras. The selected algebras considered throughout were 3 finite type, 2 finite-mutation but not finite type, and 2 infinite type. Where these numbers were selected for the purpose of ML, such that binary classifications could be performed for all combinations of types, and in particular the additional F4 finite type was introduced, as its tensor data is closer in form to the infinite types (with more larger-than-unit entries, due to the initial quiver's non-simply laced edge double multiplicity). Furthermore, as there are many infinite type initial seeds, $\mathcal{I}1$ was also specifically chosen due to its lower quiver edge multiplicities, making it more similar to the finite type cases.

These selections of the algebras considered were all made such that the tensor representations of each algebra could not be distinguished by eye; one may look at this data directly in the respective GitHub. Therefore any learning results would be non-trivial and constitute some true learning of the algebra structure.

For all pairs selected for binary classification, both algebras were generated to depth 4, their seeds converted to vectors, both sets of vectors shuffled together, and the 5-fold cross-validation ML performed. Learning results for each of these investigations are provided in Table 3.8.8, where each investigation is repeated both with and without the exchange matrix information (removal reducing the vector length by 16 each time).

For each investigation the algebras considered are labelled by their initial seeds; the respective class sizes are given, with the full tensor length into which all the seeds are embedded (based on the largest seed in that investigation). In order to compare investigations the average tensor sparsity is also given. As can be seen from this meta-data, the dataset sizes for training are very small relative to usual ML investigations, and especially with the low proportion of non-zero terms there is little information for a NN to learn any relationship between seeds from the same cluster algebra.

It is therefore evermore surprising that the architecture learns so well in all investigations, with accuracies and MCC scores > 0.9 in some investigations. It can therefore be confidently concluded that ML can identify structure inherent to each algebra, and learn the cluster mutation process.

Interestingly, the inclusion of the exchange matrix information only improves learning for classifications between algebras of different type. This may be due to the quivers for algebras

Investigation	Class Sizes	Tensor Length	Tensor Sparsity	ML Performance			
				with EM		no EM	
				Accuracy	MCC	Accuracy	MCC
A4 vs D4	(72, 80)	180	0.120	0.867 ± 0.021	0.741 ± 0.036	0.893 ± 0.026	0.788 ± 0.053
A4 vs A13	(72, 109)	280	0.088	0.944 ± 0.011	0.886 ± 0.023	0.878 ± 0.022	0.743 ± 0.047
F4 vs $\mathcal{I}1$	(65, 79)	2320	0.015	0.950 ± 0.013	0.903 ± 0.024	0.936 ± 0.012	0.875 ± 0.024
A13 vs A22	(109, 105)	280	0.091	0.810 ± 0.028	0.630 0.050	0.810 ± 0.024	0.633 ± 0.049
A13 vs $\mathcal{I}1$	(109, 79)	2320	0.015	0.930 ± 0.021	0.855 ± 0.048	0.914 ± 0.021	0.801 ± 0.059
$\mathcal{I}1$ vs $\mathcal{I}2$	(79, 117)	94280	0.008	0.918 ± 0.023	0.830 ± 0.047	0.923 ± 0.021	0.840 ± 0.043

Table 3.8.8: ML results for NN binary classification between clusters generated by the respective initial seeds. The sizes of each class are listed, along with the tensor length used to represent them, and the average sparsity of those tensors (proportion of non-zero entries). The investigations are carried out with and without the exchange matrix (EM) information for each cluster. The performance is measured by accuracy and MCC with 5-fold cross-validation to provide standard error confidence on the measures.

of the same type looking more similar and therefore diluting the already sparse information in the tensors, hindering the learning performance. This is also surprising since removal of the exchange matrix information makes some seeds in different algebras identical (for example the initial seeds); hence one would expect performance to always be worse without it. It is therefore likely the case that the dilution of relevant information is a more substantial factor than occasional misleading of the learning for classifications between algebras of the same type.

3.8.3.2 Distinguishing Generalised Associahedra Whereas the previous investigations focused on all types with data generated to depth 4, here we learn only the finite types – but generated to their maximum depth to include all seeds in the algebras.

Since these all have similar complexity cluster variables we embed them all in the same size tensors of length 1576 (with the exchange matrix information), and perform multiclassification between all of them (including B4 and C4 also). The NN architecture is the same except now cross-entropy loss must be used instead of binary cross-entropy, and the output layer is 5 neurons with softmax activation.

The 5-fold cross-validation results give averaged performance measures:

$$Accuracy = \quad 0.989 \pm 0.003\,, \tag{3.8.6}$$

$$MCC = \quad 0.985 \pm 0.004\,, \tag{3.8.7}$$

$$CM = \begin{pmatrix} 0.289 & 0.003 & 0.000 & 0.001 & 0.000 \\ 0.001 & 0.120 & 0.000 & 0.000 & 0.000 \\ 0.002 & 0.000 & 0.119 & 0.000 & 0.000 \\ 0.002 & 0.000 & 0.000 & 0.344 & 0.000 \\ 0.004 & 0.001 & 0.000 & 0.000 & 0.117 \end{pmatrix}. \tag{3.8.8}$$

Depth	1	2	3	4	5	6	7	8	9	10	11	12	13
Class Sizes	5	14	32	72	151	283	462	653	815	927	988	1007	1008
	5	14	33	80	180	372	658	928	1091	1167	1195	1200	1200
Lengths	76	96	136	196	196	196	196	196	196	196	196	196	196

Table 3.8.9: Data information for the binary classification between A4 clusters and D4 clusters generated for depths 1-13 (such that all clusters were generated). The class sizes for A4 are shown above those for D4 respectively, as well as the lengths of the flattened tensors that the clusters are embedded in.

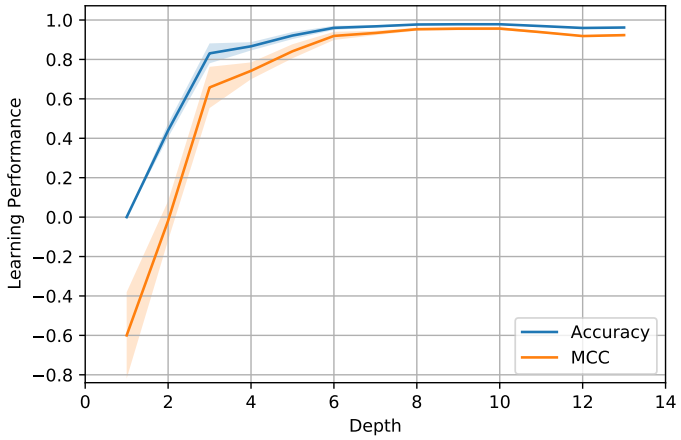


Figure 3.8.6: ML results for binary classification between A4 and D4 clusters with data generated from the respective initial seeds up to the given depths. Performance is measured with accuracy and MCC, and over the 5-fold cross-validation the measures are averaged and standard error calculated giving the shown error bounds.

This learning performance is exceptionally strong, despite now demanding multiclassification from the architecture.

The averaged confusion matrix, CM, shows the proportion of truly class A4:B4:C4:D4:F4 (given by the row) classified into class A4:B4:C4:D4:F4 (given by the column). Perfect learning produces a diagonal matrix, and here the learning is very close to that, with off-diagonal components two orders of magnitude smaller than the diagonal components. The larger diagonal entries of A4 and D4 reflect their larger frequencies in the dataset. The most frequent non-zero off-diagonals occur where the other algebras are more likely to misclassify as A4 (larger first column entries), potentially due to lower depth seeds all being exceptionally similar across the algebras which the NNs then arbitrarily assume to all be A4. Surprisingly, the matrix shows that the architecture can distinguish well between the B4 and C4 architectures, despite the analysis showing that they have identical generalised associahedra structure (despite different cluster variables).

3.8.3.3 Learning at Varying Depths

In order to connect these results of learning at depth 4 and at the maximum depth for the finite type full algebras, we examine the A4:D4 binary classification performance as depth increases from the minimum possible depth 1 (such that there is enough data to train &

test) up to depth 13 (where both algebras have all their seeds generated). Here we consider the ML investigation without the exchange matrix information, as the results in the §3.8.3.1 suggested it was in some sense superfluous for the learning.

As the depth increases not only do the cluster variables become more complex and hence represented by longer and sparser vectors, there are more variables to train with too. This investigation aims to probe these competing effects of longer, more complicated vectors to learn from, against the benefit of more data to learn with. The information regarding the tensor length needed for embedding, and the respective A4 and D4 class sizes at each depth are given in Table 3.8.9.

The cross-validation learning results are this time plotted as depth varies in Figure 3.8.6, with the performance measures' standard errors given as error bounds. It can be clearly seen that as depth increases the architectures perform better in the classification, even at lower depths where the tensor length has not yet stabilised.

3.8.3.4 Identifying Cluster Algebras

Whereas preceding investigations used ML to differentiate which cluster algebra a specific seed generates, it is also interesting to see if NNs can identify tensor representations which represent sensible seeds altogether. An easy example would be representations whose exchange matrices are not skew-symmetrizable, or have diagonal elements; but beyond these there are many ways the encoding could define nonsensical seeds (especially if adding non-zeroes deep into the padding).

In order to explore this learning we first generate suitable fake data. To ensure the fake data is representative, and not trivially distinguishable by eye, the true seed data for each cluster algebra is analysed. Each algebra is generated on its own to depth 4, reformulated as a vector, and padded to the maximum vector length for that algebra. The set of vectors for each algebra is then assessed to give a discrete distribution of frequencies of all integer entries that occur in all seed vectors across the algebra.

Then as many fake vectors are generated as there are true vectors, generated to be the same lengths as the true, with each entry's value drawn from the respective discrete distribution of possible values. All the fake vectors were checked to not overlap with the true vectors, despite a highly improbable chance of this occurring. For each algebra considered the datasets of true and fake vectors were shuffled and binary classification performed, with results shown in Table 3.8.10.

The results show perfect classification for all except the infinite types. This indicates that the NNs can learn some non-trivial structure in the finite and finite-mutation types which it can use to effectively differentiate from fake data. However, for the infinite types the poorer performance suggests that the tensor structure is perhaps more erratic and hence harder to differentiate from the simple random uniform model for its fake data. This infinite data may be expected to span a larger proportion of the possible tensors generated since there are infinitely many of them in the algebra, and hence it may also be the case that these fake tensors are related to seeds at higher depths.

Performance Measure	Cluster Algebra						
	A4 (196)	D4 (136)	F4 (336)	A13 (296)	A22 (176)	$\mathcal{I}1$ (2336)	$\mathcal{I}2$ (94296)
Accuracy	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.819 ± 0.0488	0.800 ± 0.028
MCC	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.671 ± 0.078	0.640 ± 0.044

Table 3.8.10: Binary classification results for differentiating tensors representing cluster algebras generated to depth 4 from the respective listed initial seeds, against fake tensors generated to mimic them. The respective tensor lengths are listed beneath the initial seeds in brackets. Performance is measured with accuracy and MCC across the 5-fold cross-validation runs.

Conclusion

Network analysis methods uncovered patterns in the cluster algebra exchange graphs unique to each type. In particular, a symmetric behaviour for quiver exchange graph embedding in the seed exchange graph showed constant integer ratios between respective numbers of graph vertices, which we conjecture for Dynkin types of any rank. This behaviour is made manifest by omitting the permutation equivalence identification in the exchange graphs, since certain established sets of permutations are not mutation equivalent to the initial seed.

Simple ML architectures could successfully learn to differentiate cluster algebras from their seeds. Interestingly performing better within types when the exchange matrix information was omitted, and better between types when it was included (especially for generalised associahedra).

4 Outlook

String and gauge theories attempt to describe and explain our natural world through the language of mathematics and symmetry. In the process of designing and refining these theories the desire to align them with physical phenomena continually inspires the creation of new mathematical objects, often with uses far beyond their initial intention.

To truly explore the potential of these objects their bulk generation is required, producing datasets for analytic and computational analysis. Then where processing of these databases becomes computationally infeasible statistical methods become essential to this workflow. In the work for this thesis, data science, network science, and machine learning methods have been implemented to create, study, and infer from databases of a variety of objects relevant to high-energy theory.

In `2004.05218` dessins d'enfants, as objects central to Galois theory, were generated from modular subgroups and the size of their respective Galois orbits learnt. Results highlighted the importance of faithful representation of data and indicated combinatoric Galois invariants likely exist in the dessin structure that the architecture could learn. Developments should consider larger datasets of dessins d'enfants with a greater range of orbit sizes, and classify dessins into orbits directly.

In `2006.10783` the quiver mutation process dictated by Seiberg duality between gauge theories was learnt. Success in identifying IR equivalence between theories varied depen-

dent on theory mutation type, however the performance did suggest some combinatoric consistency under mutation which could be learnt. Further steps would aim to expand this consideration to triality in 2-dimensional $\mathcal{N} = (2, 0)$ theories [300], [301] and quadrality in 0-dimensional $\mathcal{N} = 1$ theories [302], as well as to more general quivers with superpotential information.

In 2103.13436 Hilbert series were used to learn information about the variety they represent from the first coefficients in their Taylor expansions. The learning efficacy supports the use of these statistical methods to predict geometric properties from simpler operator counts performed by hand. There are many more geometric properties that can be learnt in future work, particularly for more complicated varieties, as well as further investigation into the Ehrhart series interpretation [303].

In 2106.03695 the genus of amoebae were learnt from point-sampling generated images, as well as the Newton polynomial coefficients. The success of supervised and unsupervised methods encourage exploration into further use cases of these techniques within tropical geometry. Current work in progress extends this investigation to consider higher-dimensional amoebae, as well as their connection to the Mahler measure [304].

In 2109.09602 lattice polytopes had both their own geometric properties as well as properties of their respective toric variety learnt from a Plücker representation of their vertices. The relative success compared to the vertex input motivates the use of this $GL(n, \mathbb{Z})$-invariant representation in further study, as well as work to identify explicit formulas to directly compute polytope volume. The ubiquitous use of lattice polytopes in toric geometry encourages work into examining the efficacy of the Plücker representation in streamlining and interpreting variety construction.

In 2112.06350 Calabi-Yau manifolds constructed as hypersurfaces in weighted projective spaces were analysed to uncover new structure connecting their embedding directly to their topological features via an unforeseen partitioning. Subsequent work may consider how this partitioning correlates with other topological parameters, and how it is manifest in the more general toric variety construction.

In 2202.05845 equivalence classes of brane webs were learnt to uncover subtle gaps in the current classification of 3-leg webs and sets of 3 7-branes. The relative ease of clustered NN embedding for different levels of equivalence revealed an inconsistency using the current invariants to establish equivalence without direct computation. Developing this conjectured classification to incorporate further necessary invariants or relations are sensible next steps, as well as examination of webs with more legs, particularly in reducible scenarios.

In 2203.13847 cluster algebra exchange graphs were analysed with network science and ML methods across the classification of possible mutation types. Embedding analysis motivated omitting permutation equivalence during the generation process, where an unexpected symmetry emerged. Further examination of this extracted behaviour at higher ranks and in different algebras is necessary to test the conjectures made, and ML in these more general scenarios will hopefully help guide this investigation. Current work takes advantage of these techniques in building cluster algebra databases for Grassmannians [305].

Across the projects discussed in this thesis, as well as the countless others referenced, methods from computational statistics have proved themselves as a new means to perform theoretical research. How this interconnection between computational science with theoretical physics and mathematics develops I look forward to witnessing and participating in.

Extrapolating the current trajectory of their application I would not be surprised if similar computational statistical techniques eventually became a common feature in the workflow of mathematical physicists, offering high-degree-of-freedom level insights into theory behaviour that academics can use to guide their paths to discovery.

Appendices

A Calabi-Yau Analysis

This appendix contains further learning information for architectures beyond the prototypical LR focused on in §3.6.2, as well as Hodge number correlations for smaller weights.

A.1 Additional PCA Information

Further to the PCA information provided for the CY dataset in (3.6.7) in §3.6.2, the covariance matrices, eigenvectors, and eigenvalues are given for the other three datasets here. They are respectively labelled 'R' for the random dataset, 'C' for coprime dataset, and 'T' for transverse dataset. The covariance matrices, K, and eigenvalues, λ, are given to the nearest integer, whilst eigenvectors (as rows of ε) are given to 3 decimal places.

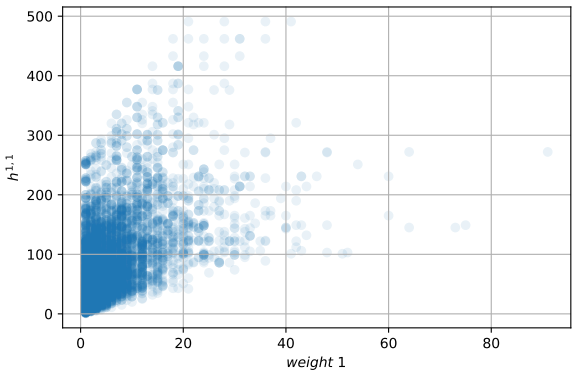
$$K_R = \begin{pmatrix} 97 & 98 & 98 & 96 & 107 \\ 98 & 251 & 250 & 245 & 255 \\ 98 & 250 & 530 & 514 & 542 \\ 96 & 245 & 514 & 1122 & 1157 \\ 107 & 255 & 542 & 1157 & 3614 \end{pmatrix}, \quad \varepsilon_R = \begin{pmatrix} 0.039 & 0.094 & 0.191 & 0.375 & 0.902 \\ -0.121 & -0.298 & -0.519 & -0.669 & 0.424 \\ -0.253 & -0.517 & -0.520 & 0.626 & -0.085 \\ -0.469 & -0.591 & 0.640 & -0.145 & 0.006 \\ -0.837 & 0.535 & -0.117 & 0.006 & 0.003 \end{pmatrix}, \quad \lambda_R = \begin{pmatrix} 4241 \\ 915 \\ 296 \\ 116 \\ 47 \end{pmatrix}, \tag{A.1.1}$$

$$K_C = \begin{pmatrix} 100 & 100 & 101 & 91 & 89 \\ 100 & 254 & 255 & 254 & 249 \\ 101 & 255 & 527 & 534 & 527 \\ 91 & 254 & 534 & 1166 & 1163 \\ 89 & 249 & 527 & 1163 & 3418 \end{pmatrix}, \quad \varepsilon_C = \begin{pmatrix} 0.036 & 0.098 & 0.199 & 0.400 & 0.889 \\ -0.124 & -0.297 & -0.514 & -0.657 & 0.448 \\ -0.284 & -0.532 & -0.497 & 0.617 & -0.095 \\ -0.457 & -0.570 & 0.662 & -0.168 & 0.009 \\ -0.833 & 0.543 & -0.109 & -0.003 & 0.000 \end{pmatrix}, \quad \lambda_C = \begin{pmatrix} 4091 \\ 921 \\ 296 \\ 109 \\ 48 \end{pmatrix}, \tag{A.1.2}$$

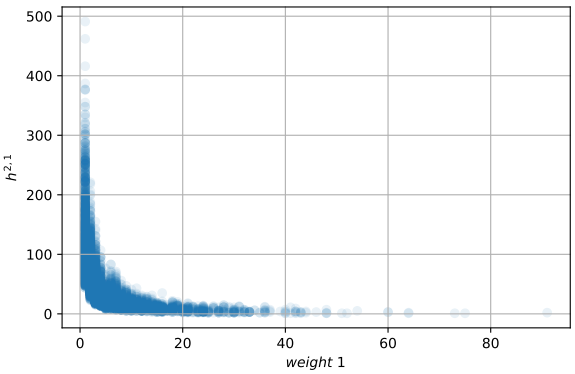
$$K_T = \begin{pmatrix} 6 & 7 & 8 & 12 & 19 \\ 7 & 20 & 25 & 35 & 55 \\ 8 & 25 & 62 & 85 & 125 \\ 12 & 35 & 85 & 173 & 246 \\ 19 & 55 & 125 & 246 & 417 \end{pmatrix}, \quad \varepsilon_T = \begin{pmatrix} 0.040 & 0.114 & 0.264 & 0.507 & 0.812 \\ 0.102 & 0.332 & 0.712 & 0.349 & -0.501 \\ 0.198 & 0.467 & 0.321 & -0.746 & 0.286 \\ -0.428 & -0.660 & 0.556 & -0.253 & 0.091 \\ -0.875 & 0.473 & -0.105 & 0.018 & -0.001 \end{pmatrix}, \quad \lambda_T = \begin{pmatrix} 620 \\ 29 \\ 17 \\ 9 \\ 3 \end{pmatrix}. \tag{A.1.3}$$

A.2 Additional Hodge Plots

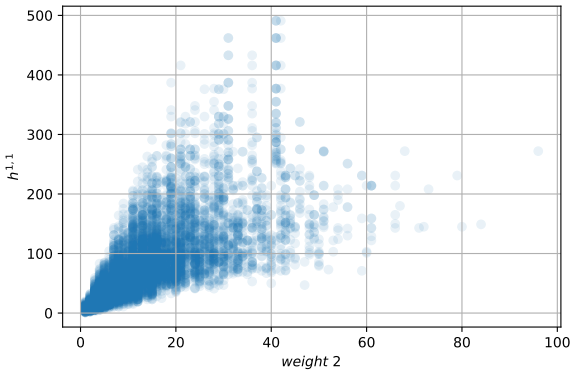
Further to the plots of the two non-trivial Hodge numbers of the CY surfaces, $\{h^{1,1}, h^{2,1}\}$, against the largest of the 5-vector weights (w_5) in Figure 3.6.5 in §3.6.2, additional plots of these Hodge numbers against the other weights are given here in Figure A.2.1 for reference.


$h^{1,1}$
weight 1

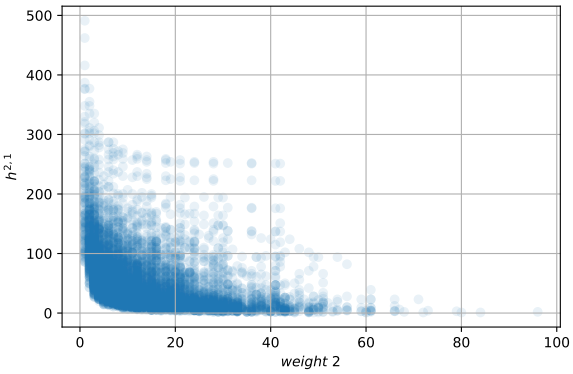
(a)


$h^{2,1}$
weight 1

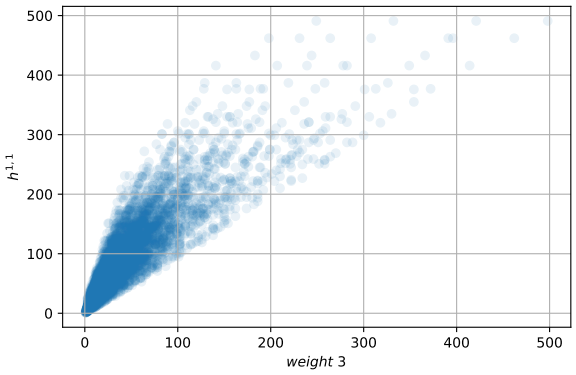
(b)


$h^{1,1}$
weight 2

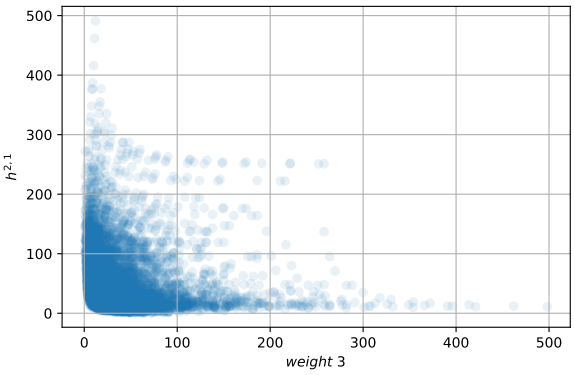
(c)


$h^{2,1}$
weight 2

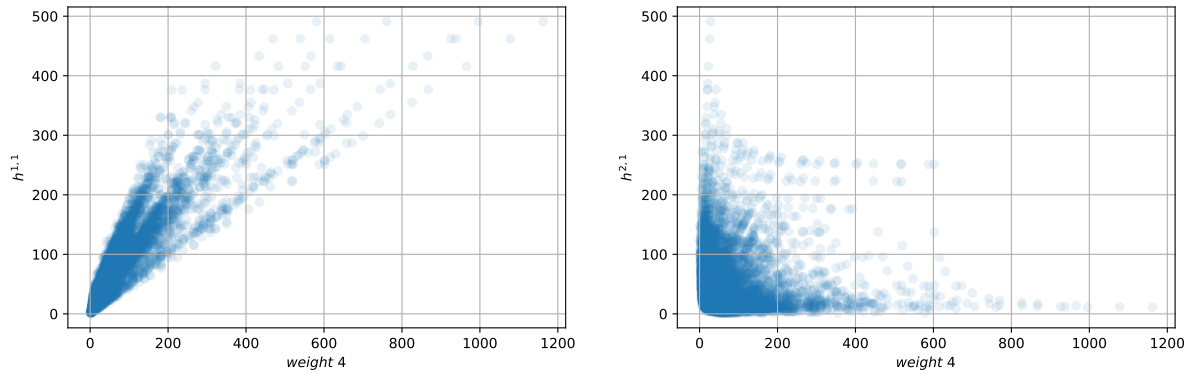
(d)


$h^{1,1}$
weight 3

(e)


$h^{2,1}$
weight 3

(f)



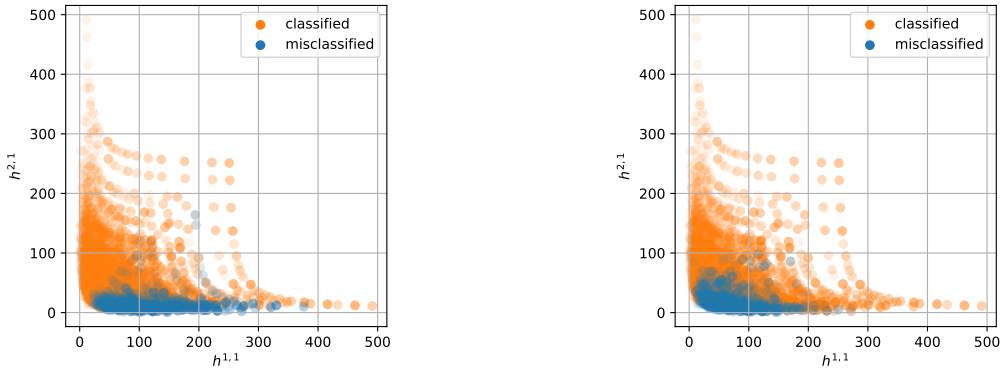
(g) (h)

Figure A.2.1: Plots of the non-trivial Hodge numbers $\{h^{1,1}, h^{2,1}\}$ against each of the first 4 weights in the CY 5-vectors. Behaviour is similar to that with the final weight, showing a linear relationship to $h^{1,1}$ and a relationship preserving the mirror symmetry structure for $h^{2,1}$.

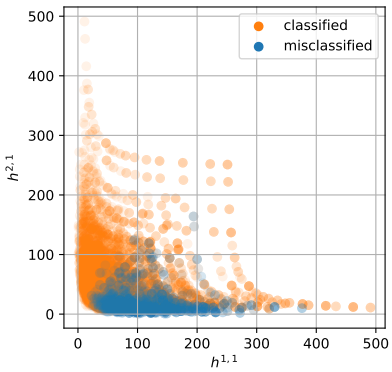
A.3 Additional Misclassification Analysis

Distributions of correctly and incorrectly classified CY 5-vectors for each of the other architectures (SVM and NN), trained on 50 CY and 50 non-CY 5-vectors, are given in Figure A.3.1; complementary to the plots in Figure 3.6.9. Note the architectures had the same hyperparameters as in previous investigations of §3.6.2.

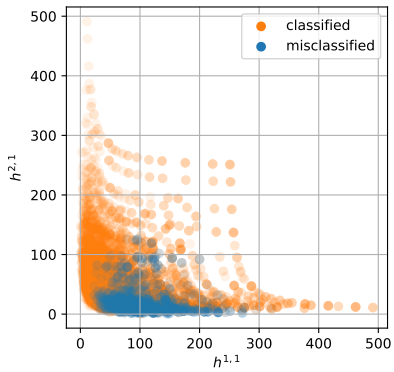
The behaviour is similar to that for the LR, where training with Random 5-vectors improves determination for high $h^{2,1}$, whilst training with Transverse 5-vectors improves determination for high $h^{1,1}$.



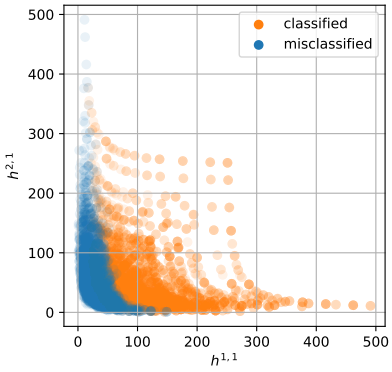
(a) SVM trained with Random (b) NN trained with Random



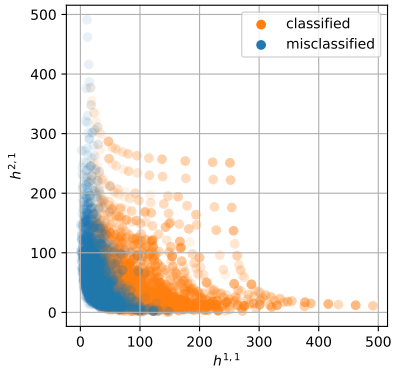
(c) SVM trained with Coprime



(d) NN trained with Coprime



(e) SVM trained with Transverse



(f) NN trained with Transverse

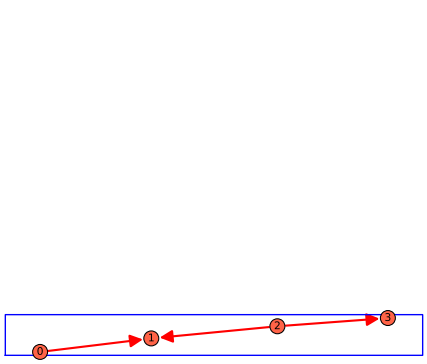
Figure A.3.1: Classified and misclassified CY 5-vectors plotted with respect to Hodge numbers, where prediction was performed by either of the architectures: Support Vector Machine (SVM), or Neural Network (NN); trained with each of the non-CY datasets respectively.

B Cluster Algebra Data

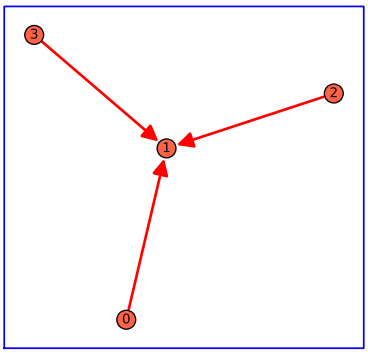
This appendix contains relevant diagrams of quivers and exchange graphs for the cluster algebras considered in the work of §3.8 [37].

B.1 Oriented Quivers

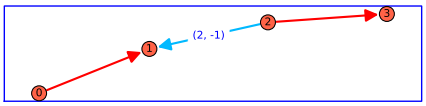
The quivers used to generate the considered rank 4 cluster algebras used in the general network analysis are shown in Figure B.1.1. Then, the additional finite type non-simply laced oriented quivers, used for an exhaustive analysis of the rank 4 finite type generalised associahedra, in Figure B.1.2. The blue boxes bounding the quivers are an artefact of the drawing and may be ignored.



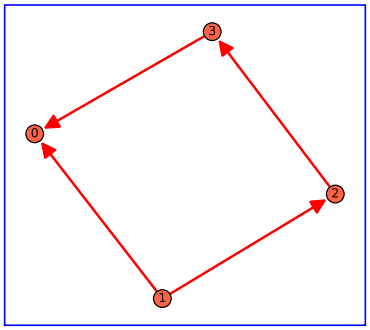
(a) A4 (finite)



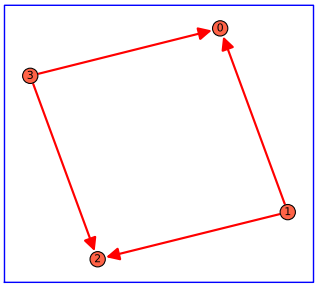
(b) D4 (finite)



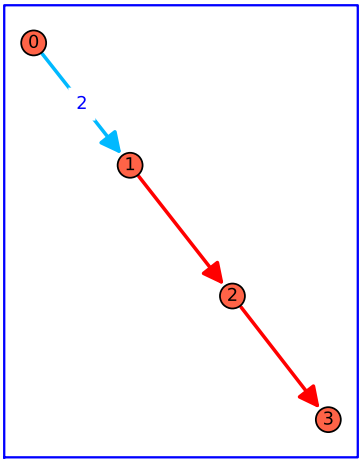
(c) F4 (finite)



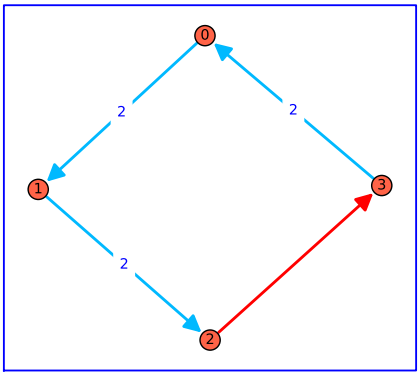
(d) A13 (finite-mutation)



(e) A22 (finite-mutation)

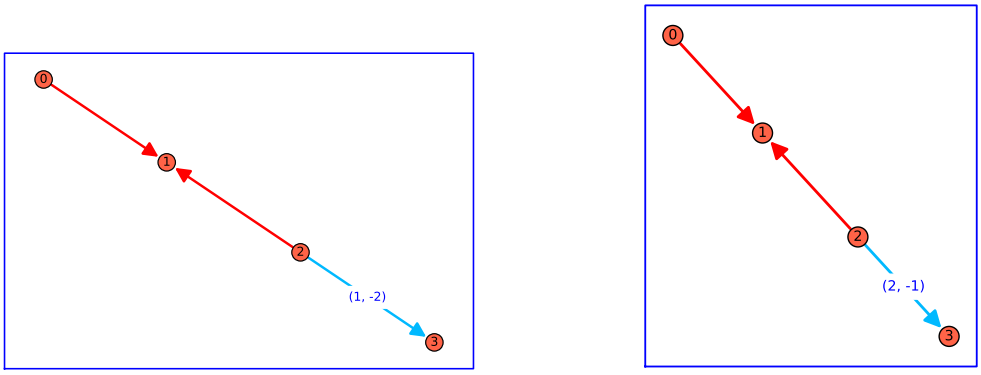


(f) $\mathcal{I}1$ (infinite)



(g) $\mathcal{I}2$ (infinite)

Figure B.1.1: Quivers defining the exchange matrices for the initial seeds. They are all rank 4, and generate cluster algebras of finite type (a), (b), (c); finite-mutation type that are not finite type (d), (e); and infinite type (f), (g). Vertices are labelled with respect to the row/column number in the exchange matrix; the double edge multiplicity in F4 indicates it is not skew-symmetric.

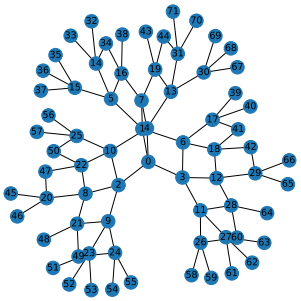


(a) B4 (b) C4

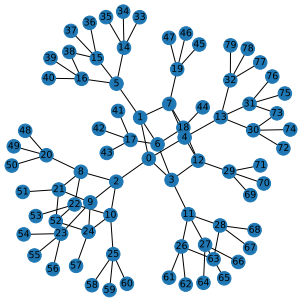
Figure B.1.2: Quivers defining the exchange matrices for the remaining rank 4 finite type cluster algebra initial seeds. Both (a) B4 and (b) C4 are skew-symmetrisable, shown by the non-simply laced, double weighted, edges of opposite weighting.

B.2 Seed Exchange Graphs

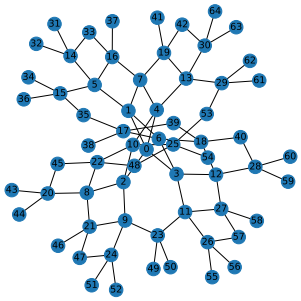
The seed exchange graphs for the considered cluster algebras, analysed in §3.8.2, computed up to depth 4.



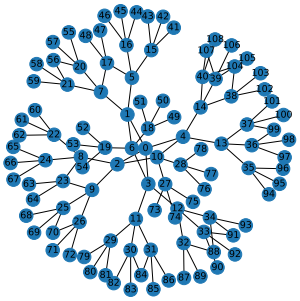
(a) A4 (finite)



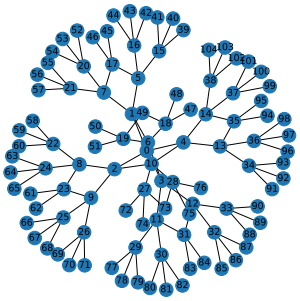
(b) D4 (finite)



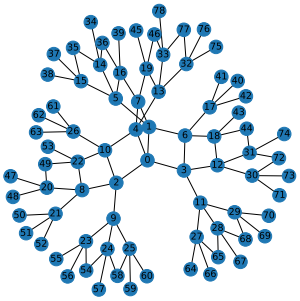
(c) F4 (finite)



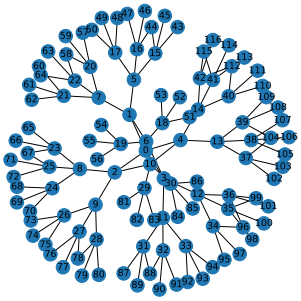
(d) A13 (finite-mutation)



(e) A22 (finite-mutation)



(f) $\mathcal{I}1$ (infinite)



(g) $\mathcal{I}2$ (infinite)

Figure B.2.1: The seed exchange graphs generated to depth 4 for each of the considered cluster algebras. Types are labelled, where finite-mutation are specifically not finite type so are infinite for these seed exchange graphs but finite for the respective quiver exchange graphs (not shown). Vertices are labelled in the order they are generated starting from the initial seed '0'.

References

[1] A. Einstein, "The Foundation of the General Theory of Relativity," *Annalen Phys.*, vol. 49, no. 7, J.-P. Hsu and D. Fine, Eds., pp. 769–822, 1916. DOI: 10.1002/andp.200590044.

[2] H. Weyl, "Elektron und gravitation. I," *Zeitschrift für Physik*, vol. 56, no. 5, pp. 330–352, May 1929, ISSN: 0044-3328. DOI: 10.1007/BF01339504.

[3] C.-N. Yang and R. L. Mills, "Conservation of Isotopic Spin and Isotopic Gauge Invariance," *Phys. Rev.*, vol. 96, J.-P. Hsu and D. Fine, Eds., pp. 191–195, 1954. DOI: 10.1103/PhysRev.96.191.

[4] P. W. Higgs, "Broken Symmetries and the Masses of Gauge Bosons," *Phys. Rev. Lett.*, vol. 13, J. C. Taylor, Ed., pp. 508–509, 1964. DOI: 10.1103/PhysRevLett.13.508.

[5] T. W. B. Kibble, "Symmetry breaking in nonAbelian gauge theories," *Phys. Rev.*, vol. 155, J. C. Taylor, Ed., pp. 1554–1561, 1967. DOI: 10.1103/PhysRev.155.1554.

[6] R. P. Feynman, "Space-time approach to nonrelativistic quantum mechanics," *Rev. Mod. Phys.*, vol. 20, pp. 367–387, 1948. DOI: 10.1103/RevModPhys.20.367.

[7] J. Dai, R. G. Leigh, and J. Polchinski, "New Connections Between String Theories," *Mod. Phys. Lett. A*, vol. 4, pp. 2073–2083, 1989. DOI: 10.1142/S0217732389002331.

[8] L. Susskind, "Harmonic-oscillator analogy for the Veneziano model," *Phys. Rev. Lett.*, vol. 23, pp. 545–547, 1969. DOI: 10.1103/PhysRevLett.23.545.

[9] L. Susskind, "Structure of hadrons implied by duality," *Phys. Rev. D*, vol. 1, pp. 1182–1186, 1970. DOI: 10.1103/PhysRevD.1.1182.

[10] P. Ramond, "Dual theory for free fermions," *Phys. Rev. D*, vol. 3, pp. 2415–2418, 10 May 1971. DOI: 10.1103/PhysRevD.3.2415.

[11] A. Neveu and J. H. Schwarz, "Tachyon-free dual model with a positive-intercept trajectory," *Phys. Lett. B*, vol. 34, pp. 517–518, 1971. DOI: 10.1016/0370-2693(71)90669-1.

[12] J. Scherk and J. H. Schwarz, "Dual models for non-hadrons," *Nuclear Physics B*, vol. 81, no. 1, pp. 118–144, 1974, ISSN: 0550-3213. DOI: 10.1016/0550-3213(74)90010-8.

[13] T. Yoneya, "Connection of Dual Models to Electrodynamics and Gravidynamics," *Prog. Theor. Phys.*, vol. 51, pp. 1907–1920, 1974. DOI: 10.1143/PTP.51.1907.

[14] F. Gliozzi, J. Scherk, and D. Olive, "Supersymmetry, supergravity theories and the dual spinor model," *Nuclear Physics B*, vol. 122, no. 2, pp. 253–290, 1977, ISSN: 0550-3213. DOI: 10.1016/0550-3213(77)90206-1.

[15] M. B. Green and J. H. Schwarz, "Anomaly Cancellation in Supersymmetric D=10 Gauge Theory and Superstring Theory," *Phys. Lett. B*, vol. 149, pp. 117–122, 1984. DOI: 10.1016/0370-2693(84)91565-X.

[16] M. B. Green and J. H. Schwarz, "Supersymmetrical string theories," *Physics Letters B*, vol. 109, no. 6, pp. 444–448, 1982, ISSN: 0370-2693. DOI: 10.1016/0370-2693(82)91110-8.

[17] D. J. Gross, J. A. Harvey, E. Martinec, and R. Rohm, "Heterotic string," *Phys. Rev. Lett.*, vol. 54, pp. 502–505, 6 Feb. 1985. DOI: 10.1103/PhysRevLett.54.502.

[18] C. M. Hull and P. K. Townsend, "Unity of superstring dualities," *Nucl. Phys. B*, vol. 438, pp. 109–137, 1995. DOI: 10.1016/0550-3213(94)00559-W. arXiv: hep-th/9410167.

[19] E. Witten, "String theory dynamics in various dimensions," *Nucl. Phys. B*, vol. 443, pp. 85–126, 1995. DOI: 10.1016/0550-3213(95)00158-O. arXiv: hep-th/9503124.

[20] P. Candelas, G. T. Horowitz, A. Strominger, and E. Witten, "Vacuum Configurations for Superstrings," *Nucl. Phys. B*, vol. 258, pp. 46–74, 1985. DOI: 10.1016/0550-3213(85)90602-9.

[21] M. R. Douglas, "The Statistics of string / M theory vacua," *JHEP*, vol. 05, p. 046, 2003. DOI: 10.1088/1126-6708/2003/05/046. arXiv: hep-th/0303194.

[22] A. Bayucan, R. L. Henderson, C. Lesiak, B. Mann, T. Proett, and D. Tweten, "Portable batch system," MRJ Technology Solutions., 1999.

[23] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60, ISBN: 978-3-540-39727-4.

[24] M. Abadi, A. Agarwal, P. Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] C. Tralie, N. Saul, and R. Bar-On, "Ripser.py: A lean persistent homology library for python," *The Journal of Open Source Software*, vol. 3, no. 29, p. 925, Sep. 2018. DOI: 10.21105/joss.00925.

[27] A. A. Hagberg, D. A. Schult, and P. Swart, "Exploring network structure, dynamics, and function using networkx," 2008.

[28] The Sage Developers, *Sagemath, the Sage Mathematics Software System (Version 9.4)*, 2020. [Online]. Available: https://www.sagemath.org.

[29] W. R. Inc., *Mathematica, Version 13.1*, Champaign, IL, 2022. [Online]. Available: https://www.wolfram.com/mathematica.

[30] Y.-H. He, E. Hirst, and T. Peterken, "Machine-learning dessins d'enfants: explorations via modular and Seiberg-Witten curves," *J. Phys. A*, vol. 54, no. 7, p. 075 401, 2021. DOI: 10.1088/1751-8121/abbc4f. arXiv: 2004.05218 [hep-th].

[31] J. Bao, S. Franco, Y.-H. He, E. Hirst, G. Musiker, and Y. Xiao, "Quiver Mutations, Seiberg Duality and Machine Learning," *Phys. Rev. D*, vol. 102, no. 8, p. 086 013, 2020. DOI: 10.1103/PhysRevD.102.086013. arXiv: 2006.10783 [hep-th].

[32] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, "Hilbert series, machine learning, and applications to physics," *Phys. Lett. B*, vol. 827, p. 136 966, 2022. DOI: 10.1016/j.physletb.2022.136966. arXiv: 2103.13436 [hep-th].

[33] J. Bao, Y.-H. He, and E. Hirst, "Neurons on Amoebae," Jun. 2021. arXiv: 2106.03695 [math.AG].

[34] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, "Polytopes and Machine Learning," Sep. 2021. arXiv: 2109.09602 [math.CO].

[35] D. S. Berman, Y.-H. He, and E. Hirst, "Machine learning Calabi-Yau hypersurfaces," *Phys. Rev. D*, vol. 105, no. 6, p. 066 002, 2022. DOI: 10.1103/PhysRevD.105.066002. arXiv: 2112.06350 [hep-th].

[36] G. Arias-Tamargo, Y.-H. He, E. Heyes, E. Hirst, and D. Rodriguez-Gomez, "Brain webs for brane webs," *Phys. Lett. B*, vol. 833, p. 137 376, 2022. DOI: 10.1016/j.physletb.2022.137376. arXiv: 2202.05845 [hep-th].

[37] P.-P. Dechant, Y.-H. He, E. Heyes, and E. Hirst, "Cluster Algebras: Network Science and Machine Learning," Mar. 2022. arXiv: 2203.13847 [math.CO].

[38] J. Bao, Y.-H. He, E. Hirst, and S. Pietromonaco, "Lectures on the Calabi-Yau Landscape," Jan. 2020. arXiv: 2001.01212 [hep-th].

[39] J. Bao, O. Foda, Y.-H. He, *et al.*, "Dessins d'enfants, Seiberg-Witten curves and conformal blocks," *JHEP*, vol. 05, p. 065, 2021. DOI: 10.1007/JHEP05(2021)065. arXiv: 2101.08843 [hep-th].

[40] J. Bao, A. Hanany, Y.-H. He, and E. Hirst, "Some Open Questions in Quiver Gauge Theory," Aug. 2021. DOI: 10.22199/issn.0717-6279-5274. arXiv: 2108.05167 [hep-th].

[41] E. Hirst, "Machine Learning for Hilbert Series," in *Nankai Symposium on Mathematical Dialogues: In celebration of S.S.Chern's 110th anniversary*, Mar. 2022. arXiv: 2203.06073 [hep-th].

[42] J. Bao, Y.-H. He, E. Heyes, and E. Hirst, "Machine Learning Algebraic Geometry for Physics," Apr. 2022. arXiv: 2204.10334 [hep-th].

[43] S. Lie, "Theorie der transformationsgruppen I," *Mathematische Annalen*, vol. 16, no. 4, pp. 441–528, Dec. 1880, ISSN: 1432-1807. DOI: 10.1007/BF01446218.

[44] E. B. Dynkin, "The structure of semi-simple algebras," *Uspekhi Mat. Nauk*, vol. 2, pp. 59–127, 4(20) 1947.

[45] H. Nakajima, "Instantons on ALE spaces, quiver varieties, and Kac-Moody algebras," *Duke Math. J.*, vol. 76, no. 2, pp. 365–416, 1994. DOI: 10.1215/S0012-7094-94-07613-8.

[46] M. R. Douglas and G. W. Moore, "D-branes, quivers, and ALE instantons," Mar. 1996. arXiv: hep-th/9603167.

[47] F. J. Dyson, "The Radiation theories of Tomonaga, Schwinger, and Feynman," *Phys. Rev.*, vol. 75, pp. 486–502, 1949. DOI: 10.1103/PhysRev.75.486.

[48] K. G. Wilson, "The Renormalization Group: Critical Phenomena and the Kondo Problem," *Rev. Mod. Phys.*, vol. 47, p. 773, 1975. DOI: 10.1103/RevModPhys.47.773.

[49] N. Seiberg, "Electric - magnetic duality in supersymmetric Non-Abelian gauge theories," *Nucl. Phys.*, vol. B435, pp. 129–146, 1995. DOI: 10.1016/0550-3213(94)00023-8. arXiv: hep-th/9411149 [hep-th].

[50] I. R. Klebanov and M. J. Strassler, "Supergravity and a confining gauge theory: Duality cascades and chi SB resolution of naked singularities," *JHEP*, vol. 08, p. 052, 2000. DOI: 10.1088/1126-6708/2000/08/052. arXiv: hep-th/0007191.

[51] S. Franco, A. Hanany, Y.-H. He, and P. Kazakopoulos, "Duality walls, duality trees and fractional branes," 2003. arXiv: hep-th/0306092 [hep-th].

[52] S. Fomin and A. Zelevinsky, "Cluster algebras I: Foundations," *Journal of the American Mathematical Society*, vol. 15, no. 2, pp. 497–529, 2002, ISSN: 08940347, 10886834.

[53] S. Fomin and A. Zelevinsky, "Cluster algebras II: Finite type classification," *Inventiones mathematicae*, vol. 154, pp. 63–121, Jan. 2003. DOI: 10.1007/s00222-003-0302-y.

[54] P. Gabriel, "Unzerlegbare Darstellungen. I," *Manuscripta Math.*, vol. 6, 71–103; correction, ibid. 6 (1972), 309, 1972, ISSN: 0025-2611. DOI: 10.1007/BF01298413.

[55] A. Felikson, M. Shapiro, and P. Tumarkin, "Skew-symmetric cluster algebras of finite mutation type," *Journal of the European Mathematical Society*, 1135–1180, 2012, ISSN: 1435-9855. DOI: 10.4171/jems/329.

[56] H. Derksen and T. Owen, "New graphs of finite mutation type," *Electron. J. Combin.*, vol. 15, no. 1, Research Paper 139, 15, 2008.

[57] A. Felikson, M. Shapiro, and P. Tumarkin, "Cluster algebras of finite mutation type via unfoldings," *International Mathematics Research Notices*, vol. 2012, Jun. 2010. DOI: 10.1093/imrn/rnr072.

[58] G. V. Belyĭ, "On Galois extensions of a maximal cyclotomic field," *Mathematics of the USSR-Izvestiya*, vol. 14, no. 2, pp. 247–256, Apr. 1980. DOI: 10.1070/im1980v014n02abeh001096.

[59] A. Grothendieck, "Esquisse d'un programme," 1984. [Online]. Available: https://webusers.imj-prg.fr/~leila.schneps/grothendieckcircle/EsquisseFr.pdf.

[60] S. Ashok, F. Cachazo, and E. Dell'Aquila, "Children's drawings from Seiberg-Witten curves," *Communications in Number Theory and Physics*, vol. 1, Dec. 2006. DOI: 10.4310/CNTP.2007.v1.n2.a1.

[61] N. Seiberg and E. Witten, "Electric - magnetic duality, monopole condensation, and confinement in N=2 supersymmetric Yang-Mills theory," *Nucl. Phys. B*, vol. 426, pp. 19–52, 1994, [Erratum: Nucl.Phys.B 430, 485–486 (1994)]. DOI: 10.1016/0550-3213(94)90124-4. arXiv: hep-th/9407087.

[62] V. Jejjala, S. Ramgoolam, and D. Rodriguez-Gomez, "Toric CFTs, Permutation Triples and Belyi Pairs," *JHEP*, vol. 03, p. 065, 2011. DOI: 10.1007/JHEP03(2011)065. arXiv: 1012.2351 [hep-th].

[63] A. Hanany, Y.-H. He, V. Jejjala, J. Pasukonis, S. Ramgoolam, and D. Rodriguez-Gomez, "The Beta Ansatz: A Tale of Two Complex Structures," *JHEP*, vol. 06, p. 056, 2011. DOI: 10.1007/JHEP06(2011)056. arXiv: 1104.5490 [hep-th].

[64] Y.-H. He, "Bipartita: Physics, Geometry & Number Theory," in *29th International Colloquium on Group-Theoretical Methods in Physics*, Oct. 2012. arXiv: 1210.4388 [hep-th].

[65] S. Franco, A. Hanany, D. Martelli, J. Sparks, D. Vegh, and B. Wecht, "Gauge theories from toric geometry and brane tilings," *JHEP*, vol. 01, p. 128, 2006. DOI: 10.1088/1126-6708/2006/01/128. arXiv: hep-th/0505211.

[66] S. Franco, A. Hanany, K. D. Kennaway, D. Vegh, and B. Wecht, "Brane dimers and quiver gauge theories," *JHEP*, vol. 01, p. 096, 2006. DOI: 10.1088/1126-6708/2006/01/096. arXiv: hep-th/0504110.

[67] S. Benvenuti, B. Feng, A. Hanany, and Y.-H. He, "Counting BPS Operators in Gauge Theories: Quivers, Syzygies and Plethystics," *JHEP*, vol. 11, p. 050, 2007. DOI: 10.1088/1126-6708/2007/11/050. arXiv: hep-th/0608050.

[68] B. Feng, A. Hanany, and Y.-H. He, "Counting gauge invariants: The Plethystic program," *JHEP*, vol. 03, p. 090, 2007. DOI: 10.1088/1126-6708/2007/03/090. arXiv: hep-th/0701063.

[69] M. F. Atiyah and I. G. Macdonald, *Introduction to commutative algebra*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills Ont., 1969, pp. ix+128.

[70] A. Strominger, S.-T. Yau, and E. Zaslow, "Mirror symmetry is T duality," *Nucl. Phys. B*, vol. 479, pp. 243–259, 1996. DOI: 10.1016/0550-3213(96)00434-8. arXiv: hep-th/9606040.

[71] J. M. Maldacena, "The Large N limit of superconformal field theories and supergravity," *Adv. Theor. Math. Phys.*, vol. 2, pp. 231–252, 1998. DOI: 10.1023/A:1026654312961. arXiv: hep-th/9711200.

[72] L. J. Dixon, "Some world sheet properties of Superstring compactifications, on Orbifolds and otherwise," in *Summer Workshop in High-energy Physics and Cosmology*, Oct. 1987.

[73] W. Lerche, C. Vafa, and N. P. Warner, "Chiral Rings in N=2 Superconformal Theories," *Nucl. Phys. B*, vol. 324, pp. 427–474, 1989. DOI: 10.1016/0550-3213(89)90474-4.

[74] P. Candelas, M. Lynker, and R. Schimmrigk, "Calabi-Yau Manifolds in Weighted P(4)," *Nucl. Phys. B*, vol. 341, pp. 383–402, 1990. DOI: 10.1016/0550-3213(90)90185-G.

[75] E. Calabi, "On Kähler manifolds with vanishing canonical class," 1957.

[76] S.-T. Yau, "Calabi's Conjecture and some new results in algebraic geometry," *Proc. Nat. Acad. Sci.*, vol. 74, pp. 1798–1799, 1977. DOI: 10.1073/pnas.74.5.1798.

[77] S.-T. Yau, "On the ricci curvature of a compact kahler manifold and the complex monge-ampere equation, I*," *Communications on Pure and Applied Mathematics*, vol. 31, pp. 339–411, 1978.

[78] Y.-H. He, *The Calabi-Yau Landscape: From Geometry, to Physics, to Machine Learning* (Lecture Notes in Mathematics). May 2021, ISBN: 978-3-030-77561-2, 978-3-030-77562-9. DOI: 10.1007/978-3-030-77562-9. arXiv: 1812.02893 [hep-th].

[79] P. Candelas, A. M. Dale, C. A. Lutken, and R. Schimmrigk, "Complete Intersection Calabi-Yau Manifolds," *Nucl. Phys. B*, vol. 298, p. 493, 1988. DOI: 10.1016/0550-3213(88)90352-5.

[80] M. Gagnon and Q. Ho-Kim, "An Exhaustive list of complete intersection Calabi-Yau manifolds," *Mod. Phys. Lett. A*, vol. 9, pp. 2235–2243, 1994. DOI: 10.1142/S0217732394002094.

[81] P. S. Green, T. Hubsch, and C. A. Lutken, "All Hodge Numbers of All Complete Intersection Calabi-Yau Manifolds," *Class. Quant. Grav.*, vol. 6, pp. 105–124, 1989. DOI: 10.1088/0264-9381/6/2/006.

[82] J. K. Kim, C. J. Park, and Y. Yoon, "Calabi-Yau Manifolds From Complete Intersections in Products of Weighted Complex Projective Spaces," *Phys. Lett. B*, vol. 224, pp. 108–114, 1989. DOI: 10.1016/0370-2693(89)91058-7.

[83] V. V. Batyrev, "Dual polyhedra and mirror symmetry for Calabi-Yau hypersurfaces in toric varieties," *J. Alg. Geom.*, vol. 3, pp. 493–545, 1994. arXiv: alg-geom/9310003.

[84] M. Kreuzer and H. Skarke, "Reflexive polyhedra, weights and toric Calabi-Yau fibrations," *Rev. Math. Phys.*, vol. 14, pp. 343–374, 2002. DOI: 10.1142/S0129055X0200120X. arXiv: math/0001106.

[85] M. Kreuzer and H. Skarke, "Classification of reflexive polyhedra in three-dimensions," *Adv. Theor. Math. Phys.*, vol. 2, pp. 853–871, 1998. DOI: 10.4310/ATMP.1998.v2.n4.a5. arXiv: hep-th/9805190.

[86] M. Kreuzer and H. Skarke, "Complete classification of reflexive polyhedra in four-dimensions," *Adv. Theor. Math. Phys.*, vol. 4, pp. 1209–1230, 2000. DOI: 10.4310/ATMP.2000.v4.n6.a2. arXiv: hep-th/0002240.

[87] R. Altman, J. Gray, Y.-H. He, V. Jejjala, and B. D. Nelson, "A Calabi-Yau Database: Threefolds Constructed from the Kreuzer-Skarke List," *JHEP*, vol. 02, p. 158, 2015. DOI: 10.1007/JHEP02(2015)158. arXiv: 1411.1418 [hep-th].

[88] R. Altman, J. Carifio, J. Halverson, and B. D. Nelson, "Estimating Calabi-Yau Hypersurface and Triangulation Counts with Equation Learners," *JHEP*, vol. 03, p. 186, 2019. DOI: 10.1007/JHEP03(2019)186. arXiv: 1811.06490 [hep-th].

[89] M. Demirtas, L. McAllister, and A. Rios-Tascon, "Bounding the Kreuzer-Skarke Landscape," *Fortsch. Phys.*, vol. 68, p. 2 000 086, 2020. DOI: 10.1002/prop.202000086. arXiv: 2008.01730 [hep-th].

[90] V. V. Batyrev and L. A. Borisov, "On Calabi-Yau complete intersections in toric varieties," Dec. 1994. arXiv: alg-geom/9412017.

[91] W. Fulton, *Introduction to Toric Varieties. (AM-131)*. Princeton University Press, 1993, ISBN: 9780691000497.

[92] J. Hausen, D. A. Cox, J. B. Little, and S. H. K., "Toric varieties," *Jahresbericht der Deutschen Mathematiker-Vereinigung*, vol. 114, no. 3, pp. 171–175, Sep. 2012, ISSN: 1869-7135. DOI: 10.1365/s13291-012-0048-9.

[93] Y.-H. He, R.-K. Seong, and S.-T. Yau, "Calabi–Yau Volumes and Reflexive Polytopes," *Commun. Math. Phys.*, vol. 361, no. 1, pp. 155–204, 2018. DOI: 10.1007/s00220-018-3128-6. arXiv: 1704.03462 [hep-th].

[94] J. Bao, G. B. Colverd, and Y.-H. He, "Quiver Gauge Theories: Beyond Reflexivity," *JHEP*, vol. 20, p. 161, 2020. DOI: 10.1007/JHEP06(2020)161. arXiv: 2004.05295 [hep-th].

[95] D. R. Morrison and C. Vafa, "Compactifications of F theory on Calabi-Yau threefolds. 2.," *Nucl. Phys. B*, vol. 476, pp. 437–469, 1996. DOI: 10.1016/0550-3213(96)00369-0. arXiv: hep-th/9603161.

[96] L. B. Anderson, X. Gao, J. Gray, and S.-J. Lee, "Fibrations in CICY Threefolds," *JHEP*, vol. 10, p. 077, 2017. DOI: 10.1007/JHEP10(2017)077. arXiv: 1708.07907 [hep-th].

[97] Y.-C. Huang and W. Taylor, "On the prevalence of elliptic and genus one fibrations among toric hypersurface Calabi-Yau threefolds," *JHEP*, vol. 03, p. 014, 2019. DOI: 10.1007/JHEP03(2019)014. arXiv: 1809.05160 [hep-th].

[98] A. D. King, "Moduli of Representations of Finite Dimensional Algebras," *The Quarterly Journal of Mathematics*, vol. 45, no. 4, pp. 515–530, Dec. 1994, ISSN: 0033-5606. DOI: 10.1093/qmath/45.4.515.

[99] P. Du Val, "On isolated singularities of surfaces which do not affect the conditions of adjunction (part I, II, III)," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 30, no. 4, 453–459, 1934. DOI: 10.1017/S030500410001269X.

[100] T. Bridgeland, A. King, and M. Reid, "The McKay correspondence as an equivalence of derived categories," *J. Am. Math. Soc.*, vol. 14, pp. 535–554, 2001. DOI: 10.1090/S0894-0347-01-00368-X.

[101] Y.-H. He, "On algebraic singularities, finite graphs and D-brane gauge theories: A String theoretic perspective," Other thesis, Sep. 2002. arXiv: hep-th/0209230.

[102] O. Aharony and A. Hanany, "Branes, superpotentials and superconformal fixed points," *Nucl. Phys. B*, vol. 504, pp. 239–271, 1997. DOI: 10.1016/S0550-3213(97)00472-0. arXiv: hep-th/9704170.

[103] O. Aharony, A. Hanany, and B. Kol, "Webs of (p,q) five-branes, five-dimensional field theories and grid diagrams," *JHEP*, vol. 01, p. 002, 1998. DOI: 10.1088/1126-6708/1998/01/002. arXiv: hep-th/9710116.

[104] O. DeWolfe, A. Hanany, A. Iqbal, and E. Katz, "Five-branes, seven-branes and five-dimensional E(n) field theories," *JHEP*, vol. 03, p. 006, 1999. DOI: 10.1088/1126-6708/1999/03/006. arXiv: hep-th/9902179.

[105] A. Hanany and E. Witten, "Type IIB superstrings, BPS monopoles, and three-dimensional gauge dynamics," *Nucl. Phys. B*, vol. 492, pp. 152–190, 1997. DOI: 10.1016/S0550-3213(97)00157-0. arXiv: hep-th/9611230.

[106] V. P. Maslov, "On a new principle of superposition for optimization problems," *Russian Mathematical Surveys*, vol. 42, no. 3, pp. 43–54, Jun. 1987. DOI: 10.1070/rm1987v042n03abeh001439.

[107] B. Feng, Y.-H. He, K. D. Kennaway, and C. Vafa, "Dimer models from mirror symmetry and quivering amoebae," *Adv. Theor. Math. Phys.*, vol. 12, no. 3, pp. 489–545, 2008. DOI: 10.4310/ATMP.2008.v12.n3.a2. arXiv: hep-th/0511287.

[108] K. Purbhoo, "A nullstellensatz for amoebas," 2006. arXiv: math/0603201.

[109] A. Zahabi, "Quiver asymptotics and amoeba: Instantons on toric Calabi-Yau divisors," *Phys. Rev. D*, vol. 103, no. 8, p. 086 024, 2021. DOI: 10.1103/PhysRevD.103.086024. arXiv: 2006.14041 [hep-th].

[110] J. Bao, Y.-H. He, and A. Zahabi, "Mahler Measure for a Quiver Symphony," *Commun. Math. Phys.*, vol. 394, no. 2, pp. 573–624, 2022. DOI: 10.1007/s00220-022-04404-y. arXiv: 2108.13903 [hep-th].

[111] J. Bao, Y.-H. He, and A. Zahabi, "Reflexions on Mahler: Dessins, Modularity and Gauge Theories," Nov. 2021. arXiv: 2111.03655 [hep-th].

[112] S. Raschka and V. Mirjalili, *Python machine learning*. Packt Publishing Ltd, 2017.

[113] S. Abel and J. Rizos, "Genetic Algorithms and the Search for Viable String Vacua," *JHEP*, vol. 08, p. 010, 2014. DOI: 10.1007/JHEP08(2014)010. arXiv: 1404.7359 [hep-th].

[114] Y.-H. He, "Deep-Learning the Landscape," Jun. 2017. arXiv: 1706.02714 [hep-th].

[115] J. Carifio, J. Halverson, D. Krioukov, and B. D. Nelson, "Machine Learning in the String Landscape," *JHEP*, vol. 09, p. 157, 2017. DOI: 10.1007/JHEP09(2017)157. arXiv: 1707.00655 [hep-th].

[116] D. Krefl and R.-K. Seong, "Machine Learning of Calabi-Yau Volumes," *Phys. Rev. D*, vol. 96, no. 6, p. 066 014, 2017. arXiv: 1706.03346 [hep-th].

[117] F. Ruehle, "Evolving neural networks with genetic algorithms to study the String Landscape," *JHEP*, vol. 08, p. 038, 2017. arXiv: 1706.07024 [hep-th].

[118] S. Donaldson, "Scalar Curvature and Projective Embeddings, I," *Journal of Differential Geometry*, vol. 59, no. 3, pp. 479 –522, 2001. DOI: 10.4310/jdg/1090349449.

[119] S. Donaldson, *Scalar curvature and projective embeddings, II*, 2004. DOI: 10.48550/ARXIV.MATH/0407534.

[120] S. K. Donaldson, *Some numerical results in complex differential geometry*, 2005. DOI: 10.48550/ARXIV.MATH/0512625.

[121] M. Headrick and T. Wiseman, "Numerical Ricci-flat metrics on K3," *Class. Quant. Grav.*, vol. 22, pp. 4931–4960, 2005. DOI: 10.1088/0264-9381/22/23/002. arXiv: hep-th/0506129.

[122] M. R. Douglas, R. L. Karp, S. Lukic, and R. Reinbacher, "Numerical Calabi-Yau metrics," *J. Math. Phys.*, vol. 49, p. 032 302, 2008. DOI: 10.1063/1.2888403. arXiv: hep-th/0612075.

[123] V. Braun, T. Brelidze, M. R. Douglas, and B. A. Ovrut, "Calabi-Yau Metrics for Quotients and Complete Intersections," *JHEP*, vol. 05, p. 080, 2008. DOI: 10.1088/1126-6708/2008/05/080. arXiv: 0712.3563 [hep-th].

[124] A. Ashmore, Y.-H. He, and B. A. Ovrut, "Machine Learning Calabi–Yau Metrics," *Fortsch. Phys.*, vol. 68, no. 9, p. 2 000 068, 2020. DOI: 10.1002/prop.202000068. arXiv: 1910.08605 [hep-th].

[125] W. Cui and J. Gray, "Numerical Metrics, Curvature Expansions and Calabi-Yau Manifolds," *JHEP*, vol. 05, p. 044, 2020. DOI: 10.1007/JHEP05(2020)044. arXiv: 1912.11068 [hep-th].

[126] J. Halverson and C. Long, "Statistical Predictions in String Theory and Deep Generative Models," *Fortsch. Phys.*, vol. 68, no. 5, p. 2 000 005, 2020. DOI: 10.1002/prop.202000005. arXiv: 2001.00555 [hep-th].

[127] L. B. Anderson, M. Gerdes, J. Gray, S. Krippendorf, N. Raghuram, and F. Ruehle, "Moduli-dependent Calabi-Yau and SU(3)-structure metrics from Machine Learning," *JHEP*, vol. 05, p. 013, 2021. DOI: 10.1007/JHEP05(2021)013. arXiv: 2012.04656 [hep-th].

[128] M. R. Douglas, S. Lakshminarasimhan, and Y. Qi, "Numerical Calabi-Yau metrics from holomorphic networks," Dec. 2020. arXiv: 2012.04797 [hep-th].

[129] V. Jejjala, D. K. Mayorga Pena, and C. Mishra, "Neural network approximations for Calabi-Yau metrics," *JHEP*, vol. 08, p. 105, 2022. DOI: 10.1007/JHEP08(2022)105. arXiv: 2012.15821 [hep-th].

[130] M. Larfors, A. Lukas, F. Ruehle, and R. Schneider, "Learning Size and Shape of Calabi-Yau Spaces," Nov. 2021. arXiv: 2111.01436 [hep-th].

[131] A. Ashmore, L. Calmon, Y.-H. He, and B. A. Ovrut, "Calabi-Yau Metrics, Energy Functionals and Machine-Learning," Dec. 2021. DOI: 10.1142/S2810939222500034. arXiv: 2112.10872 [hep-th].

[132] A. Ashmore, "Calabi-Yau metrics, CFTs and random matrices," in *Nankai Symposium on Mathematical Dialogues: In celebration of S.S.Chern's 110th anniversary*, Feb. 2022. arXiv: 2202.05896 [hep-th].

[133] D. Mehta, Y.-H. He, and J. D. Hauenstein, "Numerical Algebraic Geometry: A New Perspective on String and Gauge Theories," *JHEP*, vol. 07, p. 018, 2012. DOI: 10.1007/JHEP07(2012)018. arXiv: 1203.4235 [hep-th].

[134] M. Cirafici, "Persistent Homology and String Vacua," *JHEP*, vol. 03, p. 045, 2016. DOI: 10.1007/JHEP03(2016)045. arXiv: 1512.01170 [hep-th].

[135] K. Bull, Y.-H. He, V. Jejjala, and C. Mishra, "Machine Learning CICY Threefolds," *Phys. Lett. B*, vol. 785, pp. 65–72, 2018. DOI: 10.1016/j.physletb.2018.08.008. arXiv: 1806.03121 [hep-th].

[136] H. Erbin and S. Krippendorf, "GANs for generating EFT models," *Phys. Lett. B*, vol. 810, p. 135 798, 2020. DOI: 10.1016/j.physletb.2020.135798. arXiv: 1809.02612 [cs.LG].

[137] A. Cole and G. Shiu, "Topological Data Analysis for the String Landscape," *JHEP*, vol. 03, p. 054, 2019. DOI: 10.1007/JHEP03(2019)054. arXiv: 1812.06960 [hep-th].

[138] K. Bull, Y.-H. He, V. Jejjala, and C. Mishra, "Getting CICY High," *Phys. Lett. B*, vol. 795, pp. 700–706, 2019. DOI: 10.1016/j.physletb.2019.06.067. arXiv: 1903.03113 [hep-th].

[139] J. Halverson, B. Nelson, and F. Ruehle, "Branes with Brains: Exploring String Vacua with Deep Reinforcement Learning," *JHEP*, vol. 06, p. 003, 2019. DOI: 10.1007/JHEP06(2019)003. arXiv: 1903.11616 [hep-th].

[140] Y.-H. He and S.-J. Lee, "Distinguishing elliptic fibrations with AI," *Phys. Lett. B*, vol. 798, p. 134 889, 2019. DOI: 10.1016/j.physletb.2019.134889. arXiv: 1904.08530 [hep-th].

[141] C. R. Brodie, A. Constantin, R. Deen, and A. Lukas, "Machine Learning Line Bundle Cohomology," *Fortsch. Phys.*, vol. 68, no. 1, p. 1 900 087, 2020. DOI: 10.1002/prop.201900087. arXiv: 1906.08730 [hep-th].

[142] A. Cole, A. Schachner, and G. Shiu, "Searching the Landscape of Flux Vacua with Genetic Algorithms," *JHEP*, vol. 11, p. 045, 2019. DOI: 10.1007/JHEP11(2019)045. arXiv: 1907.10072 [hep-th].

[143] F. Ruehle, "Data science applications to string theory," *Phys. Rept.*, vol. 839, pp. 1–117, 2020. DOI: 10.1016/j.physrep.2019.09.005.

[144] Y.-H. He, "Calabi-Yau Spaces in the String Landscape," Jun. 2020. arXiv: 2006.16623 [hep-th].

[145] M. Bies, M. Cvetič, R. Donagi, L. Lin, M. Liu, and F. Ruehle, "Machine Learning and Algebraic Approaches towards Complete Matter Spectra in 4d F-theory," *JHEP*, vol. 01, p. 196, 2021. DOI: 10.1007/JHEP01(2021)196. arXiv: 2007.00009 [hep-th].

[146] R. Deen, Y.-H. He, S.-J. Lee, and A. Lukas, "Machine learning string standard models," *Phys. Rev. D*, vol. 105, no. 4, p. 046 001, 2022. DOI: 10.1103/PhysRevD.105.046001. arXiv: 2003.13339 [hep-th].

[147] Y.-H. He and A. Lukas, "Machine Learning Calabi-Yau Four-folds," *Phys. Lett. B*, vol. 815, p. 136 139, 2021. DOI: 10.1016/j.physletb.2021.136139. arXiv: 2009.02544 [hep-th].

[148] A. Constantin, T. R. Harvey, and A. Lukas, "Heterotic String Model Building with Monad Bundles and Reinforcement Learning," Aug. 2021. DOI: 10.1002/prop.202100186. arXiv: 2108.07316 [hep-th].

[149] S. Abel, A. Constantin, T. R. Harvey, and A. Lukas, "Evolving Heterotic Gauge Backgrounds: Genetic Algorithms versus Reinforcement Learning," *Fortsch. Phys.*, vol. 70, no. 5, p. 2 200 034, 2022. DOI: 10.1002/prop.202200034. arXiv: 2110.14029 [hep-th].

[150] A. Ashmore, R. Deen, Y.-H. He, and B. A. Ovrut, "Machine learning line bundle connections," *Phys. Lett. B*, vol. 827, p. 136 972, 2022. DOI: 10.1016/j.physletb.2022.136972. arXiv: 2110.12483 [hep-th].

[151] Y.-H. He, S. Lal, and M. Z. Zaz, "The World in a Grain of Sand: Condensing the String Vacuum Degeneracy," Nov. 2021. arXiv: 2111.04761 [hep-th].

[152] S. Abel, A. Constantin, T. R. Harvey, and A. Lukas, "String Model Building, Reinforcement Learning and Genetic Algorithms," in *Nankai Symposium on Mathematical Dialogues: In celebration of S.S.Chern's 110th anniversary*, Nov. 2021. arXiv: 2111.07333 [hep-th].

[153] A. Cole, S. Krippendorf, A. Schachner, and G. Shiu, "Probing the Structure of String Theory Vacua with Genetic Algorithms and Reinforcement Learning," in *35th Conference on Neural Information Processing Systems*, Nov. 2021. arXiv: 2111.11466 [hep-th].

[154] X. Gao and H. Zou, "Applying machine learning to the Calabi-Yau orientifolds with string vacua," *Phys. Rev. D*, vol. 105, no. 4, p. 046 017, 2022. DOI: 10.1103/PhysRevD.105.046017. arXiv: 2112.04950 [hep-th].

[155] G. J. Loges and G. Shiu, "Breeding Realistic D-Brane Models," *Fortsch. Phys.*, vol. 70, no. 5, p. 2 200 038, 2022. DOI: 10.1002/prop.202200038. arXiv: 2112.08391 [hep-th].

[156] P. Berglund, B. Campbell, and V. Jejjala, "Machine Learning Kreuzer-Skarke Calabi-Yau Threefolds," Dec. 2021. arXiv: 2112.09117 [hep-th].

[157] D. Berman, T. Fischbacher, G. Inverso, and B. Scellier, "Vacua of ω-deformed SO(8) supergravity," *JHEP*, vol. 06, p. 133, 2022. DOI: 10.1007/JHEP06(2022)133. arXiv: 2201.04173 [hep-th].

[158] V. Jejjala, W. Taylor, and A. Turner, "Identifying equivalent Calabi–Yau topologies: A discrete challenge from math and physics for machine learning," in *Nankai Symposium on Mathematical Dialogues: In celebration of S.S.Chern's 110th anniversary*, Feb. 2022. arXiv: 2202.07590 [hep-th].

[159] A. Constantin, "Intelligent Explorations of the String Theory Landscape," Apr. 2022. arXiv: 2204.08073 [hep-th].

[160] G. J. Loges and G. Shiu, "215 Billion Intersecting Brane Models," Jun. 2022. arXiv: 2206.03506 [hep-th].

[161] W. Cui, X. Gao, and J. Wang, "Machine Learning on generalized Complete Intersection Calabi-Yau Manifolds," Sep. 2022. arXiv: 2209.10157 [hep-th].

[162] M. Manko, "An Upper Bound on the Critical Volume in a Class of Toric Sasaki-Einstein Manifolds," Sep. 2022. arXiv: 2209.14029 [hep-th].

[163] Y.-H. He, K.-H. Lee, and T. Oliver, "Machine-Learning the Sato–Tate Conjecture," Oct. 2020. arXiv: 2010.01213 [math.NT].

[164] Y.-H. He, K.-H. Lee, T. Oliver, and A. Pozdnyakov, "Murmurations of elliptic curves," Apr. 2022. arXiv: 2204.10140 [math.NT].

[165] Y.-H. He and J. M. Pérez Ipiña, "Machine-learning the classification of spacetimes," *Phys. Lett. B*, vol. 832, p. 137 213, 2022. DOI: 10.1016/j.physletb.2022.137213. arXiv: 2201.01644 [gr-qc].

[166] H.-Y. Chen, Y.-H. He, S. Lal, and M. Z. Zaz, "Machine Learning Etudes in Conformal Field Theories," Jun. 2020. arXiv: 2006.16114 [hep-th].

[167] G. Kántor, V. Niarchos, and C. Papageorgakis, "Solving Conformal Field Theories with Artificial Intelligence," *Phys. Rev. Lett.*, vol. 128, no. 4, p. 041 601, 2022. DOI: 10.1103/PhysRevLett.128.041601. arXiv: 2108.08859 [hep-th].

[168] G. Kántor, V. Niarchos, and C. Papageorgakis, "Conformal bootstrap with reinforcement learning," *Phys. Rev. D*, vol. 105, no. 2, p. 025 018, 2022. DOI: 10.1103/PhysRevD.105.025018. arXiv: 2108.09330 [hep-th].

[169] G. Kántor, V. Niarchos, C. Papageorgakis, and P. Richmond, "6D (2,0) Bootstrap with soft-Actor-Critic," Sep. 2022. arXiv: 2209.02801 [hep-th].

[170] T. R. Harvey and A. Lukas, "Quark Mass Models and Reinforcement Learning," *JHEP*, vol. 08, p. 161, 2021. DOI: 10.1007/JHEP08(2021)161. arXiv: 2103.04759 [hep-th].

[171] Y. Gal, V. Jejjala, D. K. Mayorga Peña, and C. Mishra, "Baryons from Mesons: A Machine Learning Perspective," *Int. J. Mod. Phys. A*, vol. 37, no. 06, p. 2 250 031, 2022. DOI: 10.1142/S0217751X22500312. arXiv: 2003.10445 [hep-ph].

[172] S. Krippendorf, D. Lust, and M. Syvaeri, "Integrability Ex Machina," *Fortsch. Phys.*, vol. 69, no. 7, p. 2 100 057, 2021. DOI: 10.1002/prop.202100057. arXiv: 2103.07475 [nlin.SI].

[173] Y.-H. He and M. Kim, "Learning Algebraic Structures: Preliminary Investigations," May 2019. arXiv: 1905.02263 [cs.LG].

[174] S. Krippendorf and M. Syvaeri, "Detecting Symmetries with Neural Networks," Mar. 2020. arXiv: 2003.13679 [physics.comp-ph].

[175] H.-Y. Chen, Y.-H. He, S. Lal, and S. Majumder, "Machine learning Lie structures & applications to physics," *Phys. Lett. B*, vol. 817, p. 136 297, 2021. DOI: 10.1016/j.physletb.2021.136297. arXiv: 2011.00871 [hep-th].

[176] L. Alessandretti, A. Baronchelli, and Y.-H. He, "Machine Learning meets Number Theory: The Data Science of Birch-Swinnerton-Dyer," Nov. 2019. arXiv: 1911.02008 [math.NT].

[177] Y.-H. He, K.-H. Lee, and T. Oliver, "Machine-Learning Number Fields," Nov. 2020. arXiv: 2011.08958 [math.NT].

[178] Y.-H. He, K.-H. Lee, and T. Oliver, "Machine-Learning Arithmetic Curves," Dec. 2020. arXiv: 2012.04084 [math.NT].

[179] M. Amir, Y.-H. He, K.-H. Lee, T. Oliver, and E. Sultanow, "Machine Learning Class Numbers of Real Quadratic Fields," Sep. 2022. arXiv: 2209.09283 [math.NT].

[180] V. Jejjala, A. Kar, and O. Parrikar, "Deep Learning the Hyperbolic Volume of a Knot," *Phys. Lett. B*, vol. 799, p. 135 033, 2019. DOI: 10.1016/j.physletb.2019.135033. arXiv: 1902.05547 [hep-th].

[181] S. Gukov, J. Halverson, F. Ruehle, and P. Sułkowski, "Learning to Unknot," *Mach. Learn. Sci. Tech.*, vol. 2, no. 2, p. 025 035, 2021. DOI: 10.1088/2632-2153/abe91f. arXiv: 2010.16263 [math.GT].

[182] J. Craven, V. Jejjala, and A. Kar, "Disentangling a deep learned volume formula," *JHEP*, vol. 06, p. 040, 2021. DOI: 10.1007/JHEP06(2021)040. arXiv: 2012.03955 [hep-th].

[183] J. Craven, M. Hughes, V. Jejjala, and A. Kar, "Learning knot invariants across dimensions," Nov. 2021. arXiv: 2112.00016 [hep-th].

[184] J. Craven, M. Hughes, V. Jejjala, and A. Kar, "(K)not machine learning," in *Nankai Symposium on Mathematical Dialogues: In celebration of S.S.Chern's 110th anniversary*, Jan. 2022. arXiv: 2201.08846 [hep-th].

[185] Y.-H. He and S.-T. Yau, "Graph Laplacians, Riemannian Manifolds and their Machine-Learning," Jun. 2020. arXiv: 2006.16619 [math.CO].

[186] Y.-H. He, "Universes as big data," *Int. J. Mod. Phys. A*, vol. 36, no. 29, p. 2 130 017, 2021. DOI: 10.1142/S0217751X21300179. arXiv: 2011.14442 [hep-th].

[187] Y.-H. He, "Machine-Learning Mathematical Structures," Jan. 2021. arXiv: 2101.06317 [cs.LG].

[188] P. Betzler and S. Krippendorf, "Connecting Dualities and Machine Learning," *Fortsch. Phys.*, vol. 68, no. 5, p. 2 000 022, 2020. DOI: 10.1002/prop.202000022. arXiv: 2002.05169 [physics.comp-ph].

[189] J. Halverson, A. Maiti, and K. Stoner, "Neural Networks and Quantum Field Theory," *Mach. Learn. Sci. Tech.*, vol. 2, no. 3, p. 035 002, 2021. DOI: 10.1088/2632-2153/abeca3. arXiv: 2008.08601 [cs.LG].

[190] A. Maiti, K. Stoner, and J. Halverson, "Symmetry-via-Duality: Invariant Neural Network Densities from Parameter-Space Correlators," Jun. 2021. arXiv: 2106.00694 [cs.LG].

[191] D. Luo and J. Halverson, "Infinite Neural Network Quantum States," Dec. 2021. arXiv: 2112.00723 [quant-ph].

[192] J. Halverson, "Building Quantum Field Theories Out of Neurons," Dec. 2021. arXiv: 2112.04527 [hep-th].

[193] S. Krippendorf and M. Spannowsky, "A duality connecting neural network and cosmological dynamics," *Mach. Learn. Sci. Tech.*, vol. 3, no. 3, p. 035 011, 2022. DOI: 10.1088/2632-2153/ac87e9. arXiv: 2202.11104 [gr-qc].

[194] J. Cotler and S. Rezchikov, "Renormalization Group Flow as Optimal Transport," Feb. 2022. arXiv: 2202.11737 [hep-th].

[195] D. S. Berman, J. J. Heckman, and M. Klinger, "On the Dynamics of Inference and Learning," Apr. 2022. arXiv: 2204.12939 [cond-mat.dis-nn].

[196] J. A. Anderson, *An introduction to neural networks*. MIT press, 1995.

[197] K. Hornik, M. B. Stinchcombe, and H. L. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[198] D. Yarotsky, "Error bounds for approximations with deep ReLU networks," *Neural Networks*, vol. 94, pp. 103–114, 2017, ISSN: 0893-6080. DOI: doi.org/10.1016/j.neunet.2017.07.002.

[199] B. Hanin, "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations," *Mathematics*, vol. 7, no. 10, 2019, ISSN: 2227-7390. DOI: 10.3390/math7100992.

[200] P. Kidger and T. Lyons, *Universal approximation with deep narrow networks*, 2019. DOI: 10.48550/ARXIV.1905.08539.

[201] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993, ISSN: 0893-6080. DOI: doi.org/10.1016/S0893-6080(05)80131-5.

[202] J. Cramer, "The Origins of Logistic Regression," 2002. DOI: doi.org/10.2139/ssrn.360300. [Online]. Available: https://ssrn.com/abstract=360300.

[203] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 1573-0565. DOI: 10.1007/BF00994018.

[204] J. Bromley, J. W. Bentz, L. Bottou, *et al.*, "Signature verification using a siamese time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, 1993, ISSN: 669-688.

[205] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, Montreal, Canada, vol. 14, 1995, pp. 1137–1145.

[206] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[207] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, p. 6, 2020.

[208] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, 1065 2016. DOI: doi.org/10.1098/rsta.2015.0202.

[209] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 1, p. 17, Aug. 2017, ISSN: 2193-1127. DOI: 10.1140/epjds/s13688-017-0109-5.

[210] H. Steinhaus, "Sur la division des corps matériels en parties," *Bulletin L'Académie Polonaise des Science*, vol. 4, pp. 801–804, 1957.

[211] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010, Award winning papers from the 19th International Conference on Pattern Recognition (ICPR), ISSN: 0167-8655. DOI: doi.org/10.1016/j.patrec.2009.09.011.

[212] J. L. Gross and J. Yellen, *Graph Theory and Its Applications, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2005, ISBN: 158488505X.

[213] R. D. Luce and A. D. Perry, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, Jun. 1949, ISSN: 1860-0980. DOI: 10.1007/BF02289146.

[214] T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch, "An $\tilde{O}(m^2n)$ algorithm for minimum cycle basis of graphs," *Algorithmica*, vol. 52, no. 3, pp. 333–349, Nov. 2008, ISSN: 1432-0541. DOI: 10.1007/s00453-007-9064-z.

[215] H. Wiener, "Structural determination of paraffin boiling points," *Journal of the American Chemical Society*, vol. 69, no. 1, pp. 17–20, Jan. 1947, ISSN: 0002-7863. DOI: 10.1021/ja01193a005.

[216] P. Bonacich, "Power and centrality: A family of measures," *American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987, ISSN: 00029602, 15375390. (visited on 09/26/2022).

[217] D. A. Cox, *Galois theory*. John Wiley & Sons, 2011, vol. 61.

[218] P. Guillot, *An elementary approach to dessins d'enfants and the grothendieck-teichmüller group*, 2014. arXiv: 1309.1968 [math.GR].

[219] L. Zapponi, "What is a dessin d'enfant," *Notices of the AMS*, vol. 50, no. 7, 2003.

[220] Y.-H. He, J. McKay, and J. Read, "Modular Subgroups, Dessins d'Enfants and Elliptic K3 Surfaces," *J. Comput. Math.*, vol. 16, pp. 271–318, 2013. DOI: 10.1112/S1461157013000119. arXiv: 1211.1931 [math.AG].

[221] H. Magureanu, "Seiberg-Witten geometry, modular rational elliptic surfaces and BPS quivers," *JHEP*, vol. 05, p. 163, 2022. DOI: 10.1007/JHEP05(2022)163. arXiv: 2203.03755 [hep-th].

[222] A. Hanany, Y.-H. He, V. Jejjala, J. Pasukonis, S. Ramgoolam, and D. Rodriguez-Gomez, "Invariants of Toric Seiberg Duality," *Int. J. Mod. Phys. A*, vol. 27, p. 1 250 002, 2012. DOI: 10.1142/S0217751X12500029. arXiv: 1107.4101 [hep-th].

[223] G. Shimura, *Introduction to the arithmetic theory of automorphic functions*. Princeton university press, 1971, vol. 1.

[224] A. Sebbar, "Classification of torsion-free genus zero congruence groups," *Proceedings of the American Mathematical Society*, pp. 2517–2527, 2001.

[225] N. I. Akhiezer, *Elements of the theory of elliptic functions*. American Mathematical Soc., 1990, vol. 79.

[226] R. Miranda and U. Persson, "Configurations of In fibers on elliptic K3 surfaces," *Mathematische Zeitschrift*, vol. 201, no. 3, pp. 339–361, 1989.

[227] U. Persson and R. Miranda, *Dessin d'Enfant Table*, 1989. [Online]. Available: https://webspace.science.uu.nl/~beuke106/mirandapersson/Dessins.html (visited on 10/23/2020).

[228] F. Beukers, *Montanus List*. [Online]. Available: https://webspace.science.uu.nl/~beuke106/mirandapersson/montanuslist.txt (visited on 10/23/2020).

[229] Y.-H. He, "Calabi-Yau Varieties: from Quiver Representations to Dessins d'Enfants," Nov. 2016. arXiv: 1611.09398 [math.AG].

[230] B. Feng, A. Hanany, and Y.-H. He, "D-brane gauge theories from toric singularities and toric duality," *Nucl. Phys. B*, vol. 595, pp. 165–200, 2001. DOI: 10.1016/S0550-3213(00)00699-4. arXiv: hep-th/0003085.

[231] B. Feng, A. Hanany, Y.-H. He, and A. M. Uranga, "Toric duality as Seiberg duality and brane diamonds," *JHEP*, vol. 12, p. 035, 2001. DOI: 10.1088/1126-6708/2001/12/035. arXiv: hep-th/0109063.

[232] F. Cachazo, B. Fiol, K. A. Intriligator, S. Katz, and C. Vafa, "A Geometric unification of dualities," *Nucl. Phys. B*, vol. 628, pp. 3–78, 2002. DOI: 10.1016/S0550-3213(02)00078-0. arXiv: hep-th/0110028.

[233] Y.-H. He, "Lectures on D-branes, gauge theories and Calabi-Yau singularities," in *1st Hangzhou-Beijing International Summer School*, Aug. 2004. arXiv: hep-th/0408142.

[234] G. Musiker and C. Stump, "A compendium on the cluster algebra and quiver package in sage," Feb. 2011. arXiv: 1102.4844 [math.CO].

[235] H. Zhang, "The Optimality of Naive Bayes," in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, vol. 2, Jan. 2004.

[236] J. Gray, Y.-H. He, A. Hanany, N. Mekareeya, and V. Jejjala, "SQCD: A geometric aperçu," *Journal of High Energy Physics*, vol. 2008, no. 05, pp. 099–099, May 2008.

[237] A. Hanany, N. Mekareeya, and G. Torri, "The Hilbert series of adjoint SQCD," *Nuclear Phys. B*, vol. 825, no. 1-2, pp. 52–97, 2010, ISSN: 0550-3213.

[238] Y. Chen and N. Mekareeya, "The Hilbert series of U/SU SQCD and Toeplitz determinants," *Nuclear Phys. B*, vol. 850, no. 3, pp. 553–593, 2011, ISSN: 0550-3213.

[239] N. Jokela, M. Järvinen, and E. Keski-Vakkuri, "New results for the SQCD Hilbert series," *J. High Energy Phys.*, no. 3, 048, front matter+30, 2012, ISSN: 1126-6708.

[240] S. Benvenuti, A. Hanany, and N. Mekareeya, "The Hilbert series of the one instanton moduli space," *J. High Energy Phys.*, no. 6, pp. 100, 40, 2010, ISSN: 1126-6708.

[241] A. Hanany, N. Mekareeya, and S. S. Razamat, "Hilbert series for moduli spaces of two instantons," *J. High Energy Phys.*, no. 1, 070, front matter + 48, 2013, ISSN: 1126-6708.

[242] E. I. Buchbinder, A. Lukas, B. A. Ovrut, and F. Ruehle, "Instantons and Hilbert Functions," *Phys. Rev. D*, vol. 102, no. 2, p. 026 019, 2020. DOI: 10.1103/PhysRevD.102.026019. arXiv: 1912.08358 [hep-th].

[243] D. Forcella, A. Hanany, and A. Zaffaroni, "Master Space, Hilbert Series and Seiberg Duality," *JHEP*, vol. 07, p. 018, 2009. DOI: 10.1088/1126-6708/2009/07/018. arXiv: 0810.4519 [hep-th].

[244] A. Hanany, E. E. Jenkins, A. V. Manohar, and G. Torri, "Hilbert series for flavor invariants of the Standard Model," *J. High Energy Phys.*, no. 3, pp. 096, 7, 2011, ISSN: 1126-6708.

[245] L. Lehman and A. Martin, "Low-derivative operators of the Standard Model effective field theory via Hilbert series methods," *Journal of High Energy Physics*, vol. 02, no. 81, 2016.

[246] Y. Xiao, Y.-H. He, and C. Matti, "Standard model plethystics," *Phys. Rev. D*, vol. 100, p. 076 001, 7 Oct. 2019.

[247] V. Braun, "Counting points and Hilbert series in string theory," in *Strings, gauge fields, and the geometry behind*, World Sci. Publ., Hackensack, NJ, 2013, pp. 225–235.

[248] S. Cremonesi, A. Hanany, and A. Zaffaroni, "Monopole operators and Hilbert series of Coulomb branches of 3d $\mathcal{N}= 4$ gauge theories," *Journal of High Energy Physics*, vol. 5, 2014.

[249] A. Bourget, J. F. Grimminger, A. Hanany, R. Kalveks, M. Sperling, and Z. Zhong, "Magnetic Lattices for Orthosymplectic Quivers," *JHEP*, vol. 12, p. 092, 2020. DOI: 10.1007/JHEP12(2020)092. arXiv: 2007.04667 [hep-th].

[250] A. Bourget, J. F. Grimminger, A. Hanany, M. Sperling, and Z. Zhong, "Branes, Quivers, and the Affine Grassmannian," Feb. 2021. arXiv: 2102.06190 [hep-th].

[251] A. Bourget, S. Cabrera, J. F. Grimminger, A. Hanany, and Z. Zhong, "Brane Webs and Magnetic Quivers for SQCD," *JHEP*, vol. 03, p. 176, 2020. DOI: 10.1007/JHEP03(2020)176. arXiv: 1909.00667 [hep-th].

[252] A. Bourget, J. F. Grimminger, A. Hanany, R. Kalveks, M. Sperling, and Z. Zhong, "Folding orthosymplectic quivers," *JHEP*, vol. 12, p. 070, 2021. DOI: 10.1007/JHEP12(2021)070. arXiv: 2107.00754 [hep-th].

[253] C. Closset, S. Schafer-Nameki, and Y.-N. Wang, "Coulomb and Higgs Branches from Canonical Singularities: Part 0," *JHEP*, vol. 02, p. 003, 2021. DOI: 10.1007/JHEP02(2021)003. arXiv: 2007.15600 [hep-th].

[254] A. Dey, A. Hanany, N. Mekareeya, D. Rodríguez-Gómez, and R.-K. Seong, "Hilbert Series for Moduli Spaces of Instantons on C^2/Z_n," *JHEP*, vol. 01, p. 182, 2014. DOI: 10.1007/JHEP01(2014)182. arXiv: 1309.0812 [hep-th].

[255] G. Brown and A. M. Kasprzyk, *The Graded Ring Database*, Online. [Online]. Available: http://www.grdb.co.uk/.

[256] G. Brown and A. M. Kasprzyk, *The Fano 3-fold database*, *Zenodo* https://doi.org/10.5281/zenodo.5820338, 2022.

[257] A. M. Kasprzyk, "Canonical toric fano threefolds," *Canadian Journal of Mathematics*, vol. 62, no. 6, pp. 1293–1309, 2010. arXiv: 0806.2604 [math.AG].

[258] S. Altınok, G. Brown, and M. Reid, "Fano 3-folds, $K3$ surfaces and graded rings," in *Topology and geometry: commemorating SISTAG*, ser. Contemp. Math. Vol. 314, Amer. Math. Soc., Providence, RI, 2002, pp. 25–53.

[259] G. Brown and A. M. Kasprzyk, "Kawamata boundedness for Fano threefolds and the Graded Ring Database," arXiv:2201.07178 [math.AG], 2022.

[260] B. Bengfort and R. Bilbro, "Yellowbrick: Visualizing the Scikit-Learn Model Selection Process," *The Journal of Open Source Software*, 1075th ser., vol. 4, no. 35, 2019. DOI: 10.21105/joss.01075.

[261] A. M. Kasprzyk and B. Nill, "Fano polytopes," in *Strings, gauge fields, and the geometry behind*, World Sci. Publ., Hackensack, NJ, 2013, pp. 349–364.

[262] C. Haase and I. V. Melnikov, "The Reflexive Dimension of a Lattice Polytope," *Annals of Combinatorics*, vol. 10, no. 2, pp. 211–217, 2006, ISSN: 0219-3094. DOI: 10.1007/s00026-006-0283-9.

[263] G. Balletti, A. M. Kasprzyk, and B. Nill, "On the maximum dual volume of a canonical Fano polytope," 2016. arXiv: 1611.02455 [math.CO].

[264] V. Batyrev, A. Kasprzyk, and K. Schaller, "On the Fine Interior of Three-dimensional Canonical Fano Polytopes," 2019. arXiv: 1911.12048 [math.AG].

[265] A. A. Borisov and L. A. Borisov, "Singular toric Fano varieties," *Matematicheskii Sbornik*, vol. 183, no. 2, pp. 134–141, 1992.

[266] A. M. Kasprzyk, "Bounds on fake weighted projective space," *Kodai Mathematical Journal*, vol. 32, no. 2, pp. 197 –208, 2009. DOI: 10.2996/kmj/1245982903.

[267] G. Averkov, A. Kasprzyk, M. Lehmann, and B. Nill, *Sharp bounds on fake weighted projective spaces with canonical singularities*, 2021. DOI: 10.48550/ARXIV.2105.09635.

[268] D. Sommerville, *Introduction to the Geometry of N Dimensions* (Dover Books on Mathematics). Dover Publications, ISBN: 9780486842486.

[269] A. Hanany and R.-K. Seong, "Brane Tilings and Reflexive Polygons," *Fortsch. Phys.*, vol. 60, pp. 695–803, 2012. DOI: 10.1002/prop.201200008. arXiv: 1201.2614 [hep-th].

[270] C. Vafa, "String Vacua and Orbifoldized L-G Models," *Mod. Phys. Lett. A*, vol. 4, p. 1169, 1989. DOI: 10.1142/S0217732389001350.

[271] M. Kreuzer and H. Skarke, "No mirror symmetry in Landau-Ginzburg spectra!" *Nuclear Physics B*, vol. 388, no. 1, 113–130, Dec. 1992, ISSN: 0550-3213. DOI: 10.1016/0550-3213(92)90547-o.

[272] A Klemm, B Lian, S.-S Roan, and S.-T Yau, "Calabi-Yau four-folds for M- and F-theory compactifications," *Nuclear Physics B*, vol. 518, no. 3, 515–574, May 1998, ISSN: 0550-3213. DOI: 10.1016/s0550-3213(97)00798-0.

[273] V. V. Batyrev, "On the stringy Hodge numbers of mirrors of quasi-smooth Calabi-Yau hypersurfaces," Jun. 2020. arXiv: 2006.15825 [math.AG].

[274] V. V. Batyrev and L. A. Borisov, "On Calabi-Yau complete intersections in toric varieties," in *Higher dimensional complex varieties*, de Gruyter, 2011, pp. 39–66.

[275] N. Seiberg, "Five-dimensional SUSY field theories, nontrivial fixed points and string dynamics," *Phys. Lett. B*, vol. 388, pp. 753–760, 1996. DOI: 10.1016/S0370-2693(96)01215-4. arXiv: hep-th/9608111.

[276] P. Jefferson, H.-C. Kim, C. Vafa, and G. Zafrir, "Towards Classification of 5d SCFTs: Single Gauge Node," May 2017. arXiv: 1705.05836 [hep-th].

[277] L. Bhardwaj and G. Zafrir, "Classification of 5d $\mathcal{N} = 1$ gauge theories," *JHEP*, vol. 12, p. 099, 2020. DOI: 10.1007/JHEP12(2020)099. arXiv: 2003.04333 [hep-th].

[278] F. Apruzzi, C. Lawrie, L. Lin, S. Schäfer-Nameki, and Y.-N. Wang, "Fibers add Flavor, Part II: 5d SCFTs, Gauge Theories, and Dualities," *JHEP*, vol. 03, p. 052, 2020. DOI: 10.1007/JHEP03(2020)052. arXiv: 1909.09128 [hep-th].

[279] F. Apruzzi, C. Lawrie, L. Lin, S. Schäfer-Nameki, and Y.-N. Wang, "Fibers add Flavor, Part I: Classification of 5d SCFTs, Flavor Symmetries and BPS States," *JHEP*, vol. 11, p. 068, 2019. DOI: 10.1007/JHEP11(2019)068. arXiv: 1907.05404 [hep-th].

[280] O. DeWolfe, T. Hauer, A. Iqbal, and B. Zwiebach, "Uncovering the symmetries on [p,q] seven-branes: Beyond the Kodaira classification," *Adv. Theor. Math. Phys.*, vol. 3, pp. 1785–1833, 1999. DOI: 10.4310/ATMP.1999.v3.n6.a5. arXiv: hep-th/9812028.

[281] V. Saxena, "Rank-two 5d SCFTs from M-theory at isolated toric singularities: a systematic study," *JHEP*, vol. 04, p. 198, 2020. DOI: 10.1007/JHEP04(2020)198. arXiv: 1911.09574 [hep-th].

[282] J. Vitoria, "Mutations Vs. Seiberg duality," *J. Algebra*, vol. 321, pp. 816–828, 2009. DOI: 10.1016/j.jalgebra.2008.11.012. arXiv: 0709.3939 [math.RA].

[283] A. P. Fordy and R. J. Marsh, "Cluster mutation-periodic quivers and associated Laurent sequences," *J. Algebr. Comb.*, vol. 34, pp. 19–66, 2011. DOI: 10.1007/s10801-010-0262-4. arXiv: 0904.0200 [math.CO].

[284] F. Benini, D. S. Park, and P. Zhao, "Cluster Algebras from Dualities of 2d $\mathcal{N} = (2, 2)$ Quiver Gauge Theories," *Commun. Math. Phys.*, vol. 340, pp. 47–104, 2015. DOI: 10.1007/s00220-015-2452-3. arXiv: 1406.2699 [hep-th].

[285] S. Franco and G. Musiker, "Higher Cluster Categories and QFT Dualities," *Phys. Rev. D*, vol. 98, no. 4, p. 046 021, 2018. DOI: 10.1103/PhysRevD.98.046021. arXiv: 1711.01270 [hep-th].

[286] A. B. Goncharov and R. Kenyon, "Dimers and cluster integrable systems," Jul. 2011. arXiv: 1107.5588 [math.AG].

[287] H. Williams, "Toda Systems, Cluster Characters, and Spectral Networks," *Commun. Math. Phys.*, vol. 348, no. 1, pp. 145–184, 2016. DOI: 10.1007/s00220-016-2692-x. arXiv: 1411.3692 [math.RT].

[288] M. Kontsevich and Y. Soibelman, "Stability structures, motivic Donaldson-Thomas invariants and cluster transformations," Nov. 2008. arXiv: 0811.2435 [math.AG].

[289] D. Gaiotto, G. W. Moore, and A. Neitzke, "Wall-crossing, Hitchin Systems, and the WKB Approximation," Jul. 2009. arXiv: 0907.3987 [hep-th].

[290] N. Arkani-Hamed, J. L. Bourjaily, F. Cachazo, A. B. Goncharov, A. Postnikov, and J. Trnka, *Grassmannian Geometry of Scattering Amplitudes*. Cambridge University Press, Apr. 2016, ISBN: 978-1-107-08658-6, 978-1-316-57296-2. DOI: 10.1017/CBO9781316091548. arXiv: 1212.5605 [hep-th].

[291] J. Golden, A. B. Goncharov, M. Spradlin, C. Vergu, and A. Volovich, "Motivic Amplitudes and Cluster Coordinates," *JHEP*, vol. 01, p. 091, 2014. DOI: 10.1007/JHEP01(2014)091. arXiv: 1305.1617 [hep-th].

[292] N. Arkani-Hamed and J. Trnka, "The Amplituhedron," *JHEP*, vol. 10, p. 030, 2014. DOI: 10.1007/JHEP10(2014)030. arXiv: 1312.2007 [hep-th].

[293] S. He, Z. Li, and Q. Yang, "Notes on cluster algebras and some all-loop Feynman integrals," *Journal of High Energy Physics*, vol. 2021, Jun. 2021. DOI: 10.1007/JHEP06(2021)119.

[294] M. Alim, S. Cecotti, C. Córdova, S. Espahbodi, A. Rastogi, and C. Vafa, "BPS quivers and spectra of complete $N = 2$ quantum field theories," *Comm. Math. Phys.*, vol. 323, no. 3, pp. 1185–1227, 2013, ISSN: 0010-3616. DOI: 10.1007/s00220-013-1789-8.

[295] A. Felikson, M. Shapiro, and P. Tumarkin, "Cluster algebras of finite mutation type via unfoldings," *International Mathematics Research Notices*, vol. 2012, Jun. 2010. DOI: 10.1093/imrn/rnr072.

[296] S. Fomin and A. Zelevinsky, "Y-systems and generalized associahedra," *Annals of Mathematics*, vol. 158, no. 3, pp. 977–1018, 2003.

[297] S. Fomin and N. Reading, *Root systems and generalized associahedra*, 2008. arXiv: math/0505518 [math.CO].

[298] P.-P. Dechant, "From the Trinity (A_3, B_3, H_3) to an ADE correspondence," *Proceedings of the Royal Society A*, vol. 474, no. 2220, p. 20 180 034, 2018.

[299] T. Nakanishi, "Cluster Algebras and Scattering Diagrams, Part I. Basics in Cluster Algebras," 2022. arXiv: 2201.11371 [math.CO].

[300] A. Gadde, S. Gukov, and P. Putrov, "(0, 2) trialities," *JHEP*, vol. 03, p. 076, 2014. DOI: 10.1007/JHEP03(2014)076. arXiv: 1310.0818 [hep-th].

[301] S. Franco, S. Lee, and R.-K. Seong, "Brane brick models and 2d (0, 2) triality," *JHEP*, vol. 05, p. 020, 2016. DOI: 10.1007/JHEP05(2016)020. arXiv: 1602.01834 [hep-th].

[302] S. Franco, S. Lee, R.-K. Seong, and C. Vafa, "Quadrality for Supersymmetric Matrix Models," *JHEP*, vol. 07, p. 053, 2017. DOI: 10.1007/JHEP07(2017)053. arXiv: 1612.06859 [hep-th].

[303] T. Coates, J. Hofscheier, and A. Kasprzyk, *Machine learning the dimension of a polytope*, 2022. DOI: 10.48550/ARXIV.2207.07717.

[304] S. Chen, Y.-H. He, E. Hirst, A. Nestor, and A. Zahabi, "Mahler Measuring the Genetic Code of Amoebae," Dec. 2022. arXiv: 2212.06553 [hep-th].

[305] M.-W. Cheung, P.-P. Dechant, Y.-H. He, E. Heyes, E. Hirst, and J.-R. Li, "Clustering Cluster Algebras with Clusters," Dec. 2022. arXiv: 2212.09771 [hep-th].