Chapter 2: Exploring ketosynthase substrate specificity with machine learning.

Author: Max Walmsley
PIs: Rainer Breitling and Eriko Takano
31/05/2024

Contents

Abbreviations and Glossary

Category	Abbreviation	Definition
Polyketides	ACP	Acyl-Carrier Protein
	AT	Acyl-transferase
	DH	Dehydratase
	ER	Enoyl-reductase
	FAS	Fatty Acid Synthase
	KR	Ketoreductase
	KS	Ketosynthase
	mPKS	Modular Polyketide Synthase
	TE	Thioesterase
Protein studies	Cryo-EM	Cryo-Electron Microscopy
	PDB	Protein Database
	SAXS	Small Angle X-ray Scattering
Chemistry	SMILES	Simplified Molecular Input Line Entry
Machine Learning	ADAM	Adaptive Momentum
	AI	Artificial Intelligence
	AUC	Area Under Curve (pertaining to ROC)
	CNN	Convolutional Neural Network
	GAT	Graph Attention Network
	GCN	Graph Convolutional Neural Network
	GNN	Graph Neural Network
	GraphSAGE	Graph Sample and Aggregation
	LLM	Large Language Model
	ML	Machine Learning
	NLP	Natural Language Processing
	OHE	One Hot Encoding
	PCA	Principle Component Analysis
	pLDDT	Predicted Local Distance Difference Test
	ReLU	Rectified Linear Unit
	ROC	Receiver Operating Characteristic
	SNN	Spiking Neural Network
	xAI	Explainable Artificial Intelligence

A note on gene/module naming:

For naming modules within mPKS assembly lines, we use a format of Gene_Mod.# (e.g. NysC_Mod.2), where # is the position of the module within the gene, not the BGC. This has been to keep gene names consistent with the webscraping described in section 2.2.1.

There are two erythromycin pathways used in this chapter. One from *Aeromicrobium erythreum* (MIBIG: BGC0000054) and another from *Saccharopolyspora erythraea* (MIBIG: BGC0000055). These have quite different protein sequences so we may make the distinction in gene names, as AeEry and SeEry, respectively. *S. erythraea*'s erythromycin BGC is a model pathway for mPKSs.

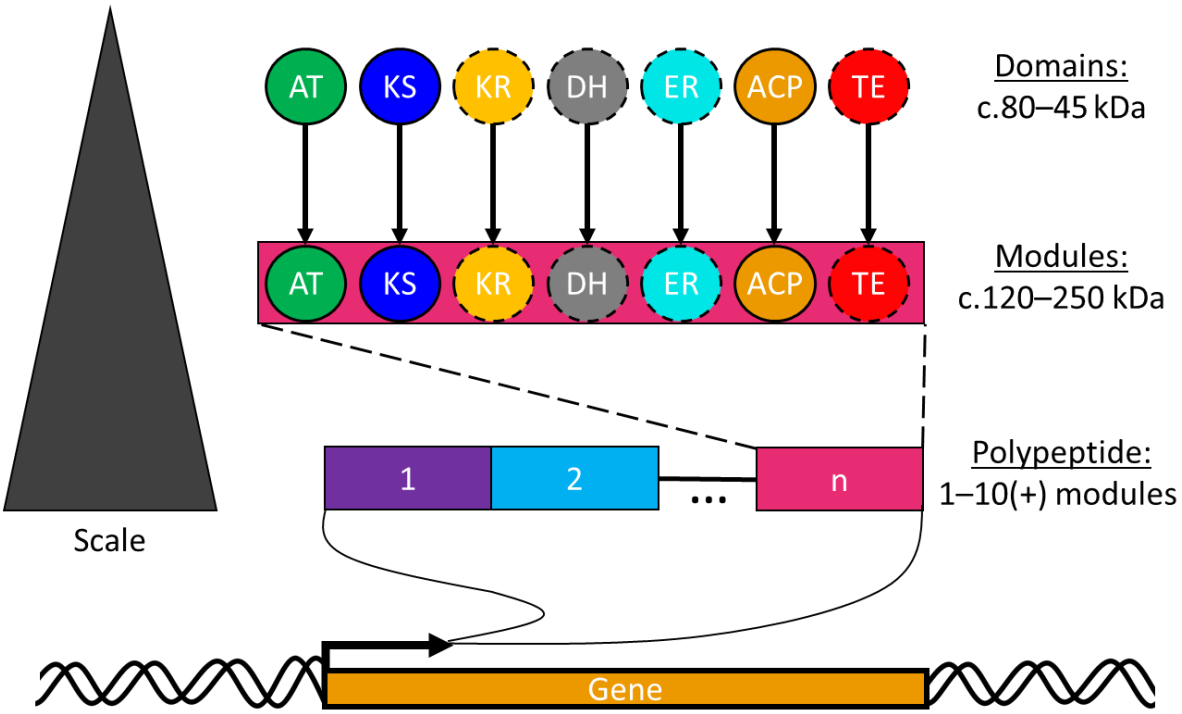


Figure G1: Visual representation of how domains, modules and genes for mPKSs relate to each other. A domain is a functional enzymatic unit, of which the AT, KS and ACP are essential to a module. The KR, DH, ER and TE are optionally included in modules and are marked in dashed lines. An mPKS gene/polypeptide will be a composition of one or more modules.

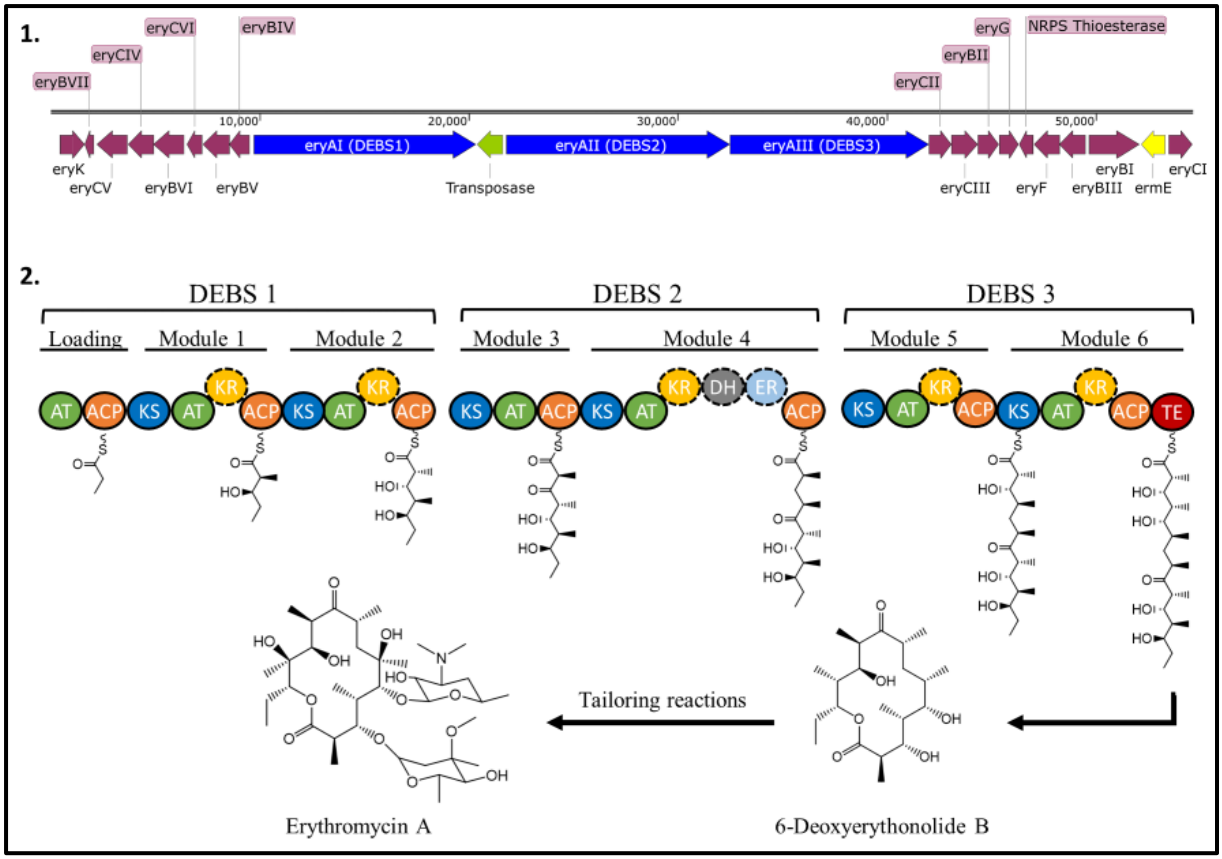


Figure G2: **1.** *Saccharopolyspora erythraea* Erythromycin A biosynthetic gene cluster. Scale bar displays length in nucleotides. Genes encoding polyketide synthases have been coloured blue, tailoring enzymes are burgundy, resistance genes yellow and transposases green. **2.** Biosynthesis of core polyketide 6-Deoxyerythonolide B (6-DEB) by mPKS polypeptides DEBS1–3. Input starter unit is propionyl-CoA (used in loading module) and extension units for modules 1–6 are 2*S*-methylmalonyl-CoA.

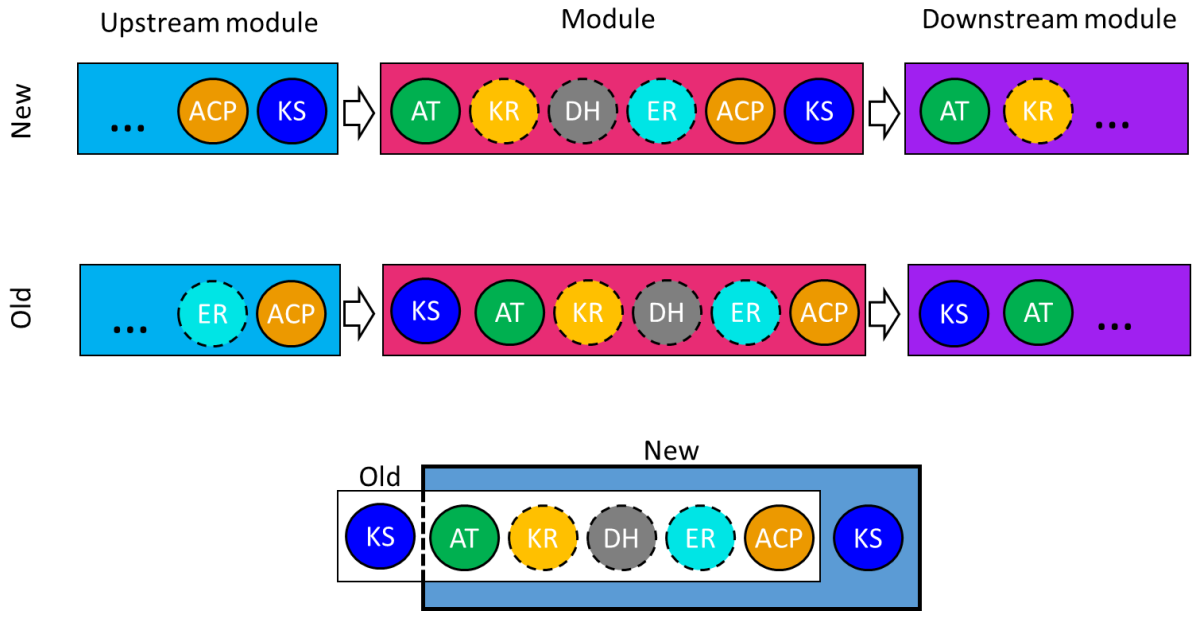


Figure G3: Contemporary (new) border definitions[1, 2] versus traditional (old) definition of mPKS modules. Essentially, the border between modules is either after the KS or after the ACP.

Chapter in abstract

Modular polyketide synthases (mPKS) are a class of biosynthetic pathway responsible for the production of many essential polyketide medicines, most notably antibiotics[3] (see figure G2 for an example). There is growing need for new antibiotics[4, 5] and it would be advantageous to be able modify these pathways to produce new polyketides that address this need. However, when we implement modifications to mPKSs that change the output polyketide, we typically break the pathway and it is not clear why. Several reasons for the modified pathways failing have been suggested in the literature, one of which is that one of the constituent enzymatic domains essential to mPKSs, the ketosynthase, may reject the non-natural products in a phenomenon referred to as proofreading[6-8].

In this chapter, we use neural networks to investigate ketosynthase substrate specificity (proofreading), by generating machine learning models that can discern between a subset of polyketide substrate properties, when presented with computationally predicted ketosynthase structures as input data.

To establish the validity and utility of our machine learning approaches, we first ran a pilot study on mPKS acyltransferase (AT) substrate specificity, which is a structural problem with known solutions – we know which polypeptide motifs determine AT substrate selection. Through this pilot study, we found that Graph Convolutional Networks (GCNs) can detect with 98.4% accuracy whether an AT domain has substrate specificity for methyl-malonyl-CoA or malonyl-CoA. We also developed a method for creating human interpretable explanations of the GCN models that identifies the known motifs, as well as providing information that, with experimental validation, could expand our understanding of AT domain substrate interaction.

Transferring the pilot GCN architectures to ketosynthases, we find that only ketosynthases receiving a dehydrated β-carbon are distinguishable against other reduction states:

Accuracy				
β-carbon reduction state	NR	KR	DH	ER
NR		X	73%	X
KR	X		86%	X
DH	73%	86%		82%
ER	X	X	82%	

AUC				
β-carbon reduction state	NR	KR	DH	ER
NR		X	86%	X
KR	X		92%	X
DH	86%	92%		82%
ER	X	X	82%	

Table 1: GCN binary classification model accuracy and AUC scores for β-carbon reduction state. States on the β-carbon are Non-reduced (ketone), KR (alcohol group), DH (dehydrated, carbon-carbon double bond) or ER (fully reduced). An "X" result indicates that the model was unable to convincingly distinguish between reduction states.

From the NR/KR/ER vs. DH models, we identify structural regions and associated residues that are frequently reported as significant by the model. Some of the residues detected are novel, some have already been have already been verified for substrate specificity in previous studies[6, 7]. The verified residues have been shown to have the capacity to increase a ketosynthases specificity for non-native

substrates when mutated. Our hope for the novel residues is, if experimentally verified, that they could form a basis for predicting and mitigating ketosynthase proofreading in modified mPKS systems, in order to facilitate better engineering outcomes when designing novel mPKSs.

Applied to the methylation state on the α-carbon, we find some evidence that α-methylated and α-non-methylated polyketide substrates from the upstream ACP are distinguishable, with a model accuracy score of 71.2% when tested on a 10% test partition. However, the confidence in this model is low, with a loss score of 0.61 and weak explanations of the model's predictions. We were unable to train a model that could discern between methylation state of the extension unit going into the condensation reaction.

2.1 Introduction

Modular polyketide synthases (mPKSs) are enzymatic assembly lines that produce diverse and complex carbon scaffolds[9]. These synthases are enormous by protein standards and are comprised of multiple modular units, where each module functionally comprises 3–7 conventional enzymes on a single polypeptide, and a single module typically ranges from 4–7 kb in length, translating to polypeptides in the mega-Dalton range. In nature, we observe as many as 9 modules[10] in a single gene.

Nature is not in the habit of wasting resources to make such large proteins without an equally large incentive. Often produced by soil bacteria, the mPKS family is responsible for the production of a range of potent antibiotics that allow their host to occupy highly competitive environmental niches. From a human standpoint, some of these mPKS products can be used as medicinal antibiotics, such as erythromycin[11] and amphotericin[12, 13], as well as other medicines, such as immunosuppressants[14]. Further uses for these compounds have been found as pesticides[15] and potentially biofuels/plastic feedstock[16], as well as in pseudo-medicinal tribal practices[17].

Polyketide structures produced from an mPKS are intrinsically coupled to the composition of enzymatic domains within each module, and the number and arrangement of modules in the assembly line This makes mPKS carbon scaffold products predictable from their gene sequence[18]. From an engineering perspective, this is attractive because it implies underlying rules and logic abstractable to mPKSs in general. In principle, we should be able to treat modules as decoupled components of mPKS systems, which is to say it should be possible to rearrange them such that a permissible desired target carbon scaffold can reliably be produced. Indeed, we can see these rearrangements happening over evolutionary periods, and this inherent modularity has likely facilitated the emergence of the wide range of unique mPKS assembly lines and products observed in nature[19].

However, when we treat modules as decoupled components in the lab, we find that mPKSs are not always amenable to modification; module rearrangements and mutagenesis produce a mixed bag of results. The general rule is that modifications can work, but frequently we will observe dramatically reduced productivity to no product at all[20, 21]. Furthermore, we find that the greater the chemical deviation from the natural product, the more pronounced the risk of failure[22, 23]. This raises the question: why do some modifications work whilst others do not?

Contemporary explanations for this are as follows:

1. Ketosynthases can exhibit a capacity for substrate specificity (proofreading) on the incoming polyketide from the previous module[6, 7] and can reject/stall when deviating from their native substrates.
2. Cryo-electron microscopy (cryo-EM) studies of the mono-modular PikAIII gene found that mPKS modules are highly coordinated structures with complex interactions between modules[24, 25]. This coordination and interdependency of components within/across modules provides some explanation as to why mPKSs are not robust to modification.

3. ACP handover across modules has been demonstrated as a potential point of failure[20, 26]. Mechanistically, this occurs when the ACPs are unable to interact with partnering KSs and can be a result of incompatible surface charges[6, 20] or physical distance
4. Module border definitions have changed recently in the literature to be more in line with how modules present as discrete units over evolutionary periods[1, 2] (see figure G3). The traditional definition of a module is KS-AT-(KR-DH-ER)-ACP. The revised definition moves the KS to the end of the module: AT-(KR-DH-ER)-ACP-KS. The difference that this creates is highly significant when reflecting on previous module swap experiments for 2 reasons:
    1. The ketosynthase evolves with the upstream reductive domains and AT domain. This gives a means for proofreading of the β-carbon reduction state and extension unit selection to be maintained across module duplication and translocation events, because these will still be relevant to the KS after the evolutionary event. This would not be the case in the traditional definition.
    2. The ACP most relevant to the KS over evolutionary periods is the upstream in the new definition, not the downstream, as indicated in the traditional definition. As mentioned above, ACPs interact with the KS via charged interactions on the domain surfaces that are complementary[6, 20]. The new border definition preserves this interaction.

The new definition comes with a caveat, which is that the borders of mPKS genes with docking domains conform to the traditional definition, and not the revised definition. We also find genes encoding mPKS modules do no always arrange into collinear operon on the genome (e.g. amphotericin, which forms a 4, 5, 6 – 1, 2, 3 two operon system), which would indicate that to some degree, the traditional definition is still correct.

Of the reasons outlined above, ketosynthase proofreading is the primary focus of this chapter, though we draw from the border definitions explanation in how we address this. In essence, our objective is to expand the set of design rules for modifying mPKSs that can explain past failures and provide solutions towards *de novo* mPKS design.

2.1.1 Exploring ketosynthase proofreading

Modular polyketide synthases represent a massive investment of cellular resources and so there is a strong incentive to ensure that the correct compound is produced. This gives a justification for molecular proofreading occurring in mPKSs.

On paper, there are many potential points of failure within an mPKS assembly line. For example, the 6-DEBS mPKS that produces the carbon scaffold for erythromycin has 6 KSs, 7 ATs, 6 KRs, 1 DH, 1 ER and 1 TE, catalysing a total of 22 chemical reactions that must occur sequentially, facilitated by 22 ACP-mediated translocations (see figure G2). Looking at the chemical reactions that occur in this pathway, there is no chemistry-prohibitive reason that the chemical reactions that occur in a module cannot be skipped or repeated. Iterative type I polyketide synthases are highly related protein structures and do repeat condensation and reduction reactions. Likewise, the reductive domains and methylases in an mPKS are not required for a polyketide to be extended by the next module, so could theoretically be skipped or partially utilised. We occasionally see examples of these alternative chemistries occurring in nature (e.g. amphotericin A/B, Ref.[13]), but overall, carbon scaffolds produced by mPKSs are precise. This presents mPKSs as highly coordinated assembly lines that protect against off-target chemistries and gives a basis for why proofreading may have evolved in mPKS modules.

Alongside AT and ACP domains, ketosynthases are an essential component of a polyketide module[27]. The AT domain does not interact directly with the growing polyketide, and thus does not represent a sensible place for proofreading to occur. An ACP interacts with two to six different substrates and can

interact with up to seven domains as part of a module's reaction cycle. Contrasted with the two substrates a ketosynthase receives, and two interacting domains (ACPs), the ketosynthase presents the simpler of the two to evolve proofreading capacity. On balance, ketosynthases are the more logical place for proofreading to have evolved in mPKS assembly lines.

This aligns with the contemporary, evolution-based module border definition[1], which places the ketosynthase at the end of the module, where proofreading would be relevant to the whole module, rather than a preceding module that may not be coevolving. This module definition better supports the observed translocation of modules over evolutionary periods, if proofreading is limited to the immediate condensation and reduction steps within a module.

Experimental evidence of ketosynthase proofreading

1. Chimeric, *cis*-AT bimodular mPKSs are bottlenecked by both substrate specificity and ACP interaction[6]. Point mutations to multiple residues proximal to the ketosynthase active site can increase turnover rate[6, 7]. Sequence analysis of *cis*-AT mPKSs also finds some weak/moderate consensus for fingerprint residues within ketosynthases[8].
2. *Trans*-AT mPKS ketosynthases sequences can be used to predict inter-module polyketide substrates, where the correlation between substrate and ketosynthase has been experimentally demonstrated for β-carbon branching[28]. *Cis*-AT mPKSs do not form substrate-specific clades however, and instead cluster by BGC[7, 29].

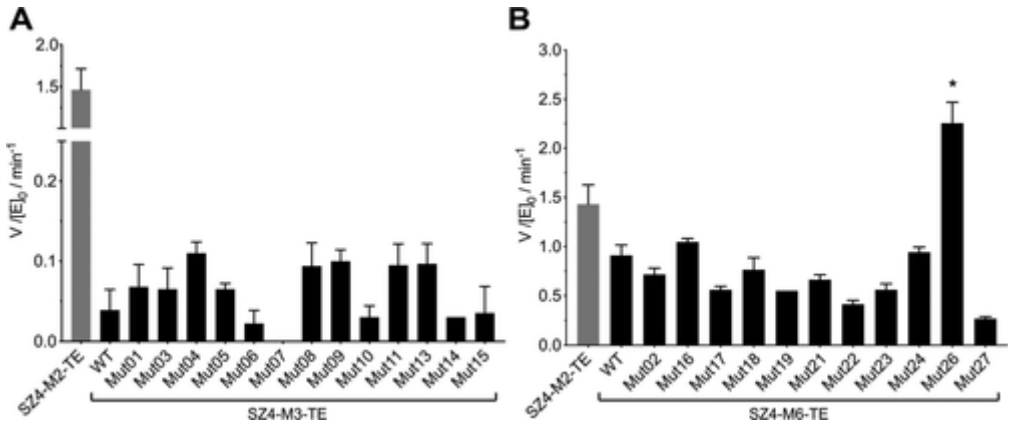


Figure 1: Figure 3, copied from Klaus et al 2020 Ref.[6]. In this study, a series of bi-modular chimeric mPKSs were created, where the second module is heterologous to the non-chimeric pathway, depicted in the graphs as SZ4-M2-TE. WT indicates the wild-type chimeric module and Mut## indicates a series of mutations to WT. Panels A and B show different modules swapped into the second position (EryAII_Mod.1 and EryAIII_Mod.2 respectively). Panels A and B show that mutagenesis of the ketosynthase can improve turnover (y-axis) in the chimeric-bimodular system and even surpass the native pathway (panel B, SZ4-M2-TE vs Mut26).

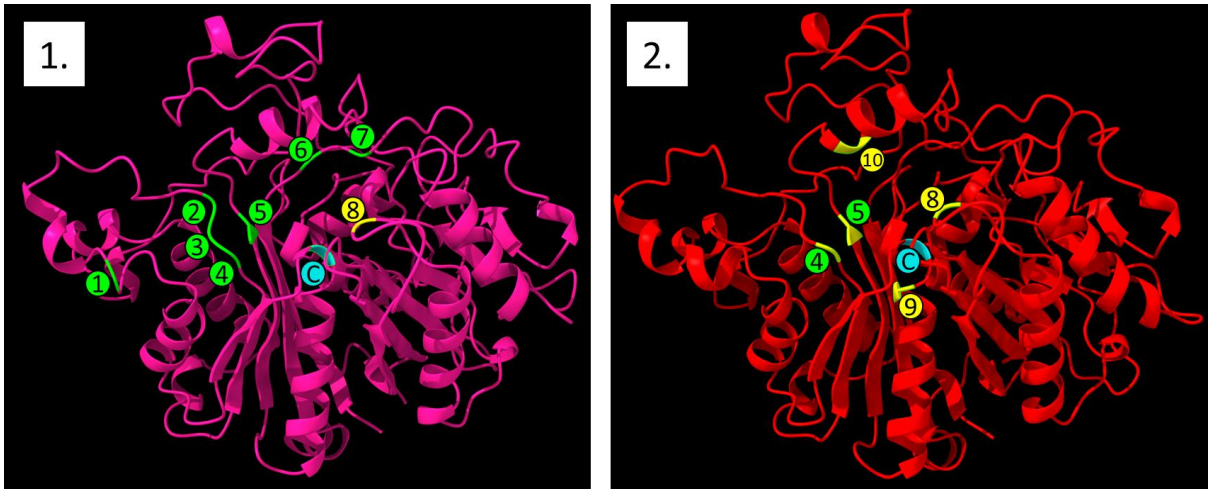


Figure 2: Experimentally validated substrate specificity associated residues for ketosynthases. Residues in green were tested in Murphy et.al, 2016[7] and residues in yellow were tested in Klaus et al, 2020[6]. Active-site Cysteine has been coloured cyan. Panels **1 and 2** are AlphaFold predicted structure of SeEryAII_Mod.1 and EryAIII_Mod.2, and correspond to the proteins use in the two studies.

On the basis of the above logic and experimental evidence, our hypothesis is that there are properties of the KSs that are intrinsically linked to the natural incoming substrate, and if we can link properties of the substrate to properties of the ketosynthase, then we have expanded the design rules for mPKS and improved our ability to engineer mPKSs. For example, a rule might present as "a positively charged residue in position X predicts a fully reduced β-substituent in the polyketide substrate". What we are aiming to contribute beyond Klaus and Murphy is a computationally guided explanation of how substrate specificity works in ketosynthases so that we can predict and mitigate it.

Over the last 30 years or so, there has been a cascade of research systematically unpicking how mPKSs work. Structurally, we have access to crystal structures for each of the sub-domains of a module, and from multiple different pathways[9, 11, 30-36]. Catalytic mechanisms and active site motifs are well-understood[9]. There is also literature showing a range of module swaps in a variety of pathways, which has led to a generalisation of expected poor productivity outcomes/stalling when modifying mPKS pathways[21-23]. There has also been significant cryo-EM work on conformational states in pikAIII[24, 25], which has been expanded to DEBS[37], and from this, we have an insight into how substrate moves through modules and how domains interact. SAXS has also yielded whole module structural information[33]. Whole-module, monomeric *trans*-AT structures have been verified as achievable with AlphaFold[38] (corroborated with cryo-EM)*. What is missing from the wider body of research on mPKSs is a complete narrative of structural confirmations that follows a growing polyketide across multiple modules and describes how the modules cooperate.

**We found that monomeric* cis*-AT modules resolve to plausible structures at individual domain levels, adopting a "beads on a string" confirmation. However, in our hands, we found that the more biologically relevant dimer structures were not resolvable in AlphaFold. For the current generation of protein structure prediction tools, our opinion is that the AlphaFold is a powerful tool for examining individual domains and inter-domain pairings (particularly with the ACP).*

2.1.2 Ketosynthase catalysis, structure and interactions

Ketosynthases catalysis and interactions

Functionally, ketosynthases combine a growing polyketide chain with a short, activated carbon chain referred to as an extension unit, in order to create a longer polyketide carbon chain. This occurs via a

decarboxylative Claisen condensation reaction[9, 35], releasing CO_2 in an exergonic reaction that leaves a ketone group at the β position of the carbon chain. A ketone group is therefore produced in each module's extension step, which is where the name "polyketide" originates. These ketones can then variably be reduced, depending on whether a ketoreductase (KR), a KR and a dehydratase (DH), or a KR, DH and enoyl-reductase (ER) are present in a module.

Within the reaction centre of a ketosynthase are 3 consensus motifs: KSNIG**H**T, TA**C**SSS and EA**H**GTG[9]. The catalytic residues are underlined. These motifs present with some variation to the consensus. In loading ketosynthases (KS_Q), a glutamine usually replaces the TA**C**SSS cysteine and enables decarboxylation to occur independent of an extension unit[9].

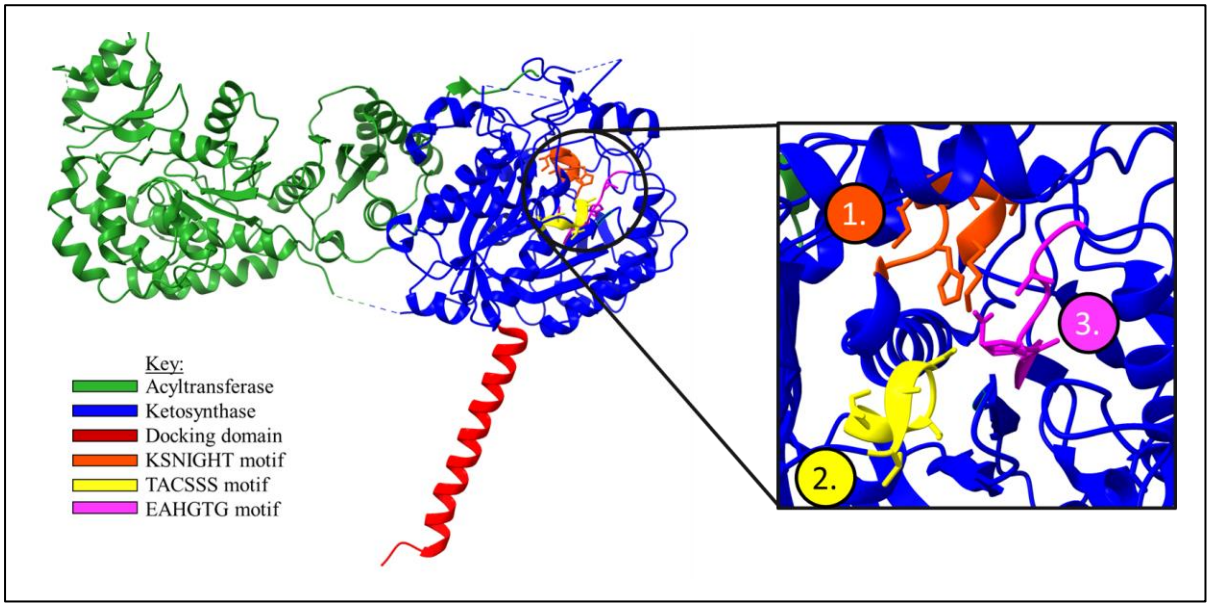


Figure 1: Monomer view of 8EE0, highlighting reaction centre (circled) and docking helix (red). Within the reaction centre are 3 motifs: 1. KSNIG**H**(T/A), orange, 2. TA**C**SSS, yellow and 3. EA**H**GTG, magenta.

Modular polyketide synthases assemble as homodimers*, with the ketosynthase forming one of the dimerization surfaces for PKS modules[24, 36, 39-41]. Contacting loops, sheets and helices have been highlighted in magenta in figure 2 below. AT domains also form a structural interact with the opposite chain KS[24].

Ketosynthases also form a significant structural interface with their C-terminal acyltransferase domain, adjoining via a ferredoxin-like linker[32] (see figure 8). This interaction is dynamic during the reaction cycle[24] and may trigger allosteric effects between KS and AT, depending on what substrate is bound where.

Ketosynthases interact with ACP domains immediately upstream and downstream in the assembly line. As part of a reaction cycle, the first of these interactions is with the upstream ACP, which provides the growing polyketide from the previous extension step to the active-site cysteine of the ketosynthase, in a process referred to as trans-acylation. The interaction between a ketosynthase and its upstream ACP is spatially facilitated either by a connecting loop, if the ketosynthase and ACP are on the same polypeptide, or via a set of charged docking helices on the ketosynthase N-terminal, which binds to a complementary docking domain at the C-terminal of the upstream ACP[42] (figure 1, red).

The second domain interaction is with the downstream ACP, which provides the activated short carbon-chain extension unit. Following handover of the extension unit the condensation reaction occurs. A

downstream ACP then removes the polyketide for modification by downstream reductive domains, if present, and then passes the polyketide to the next ketosynthase, or a thioesterase if at the end of the assembly line.

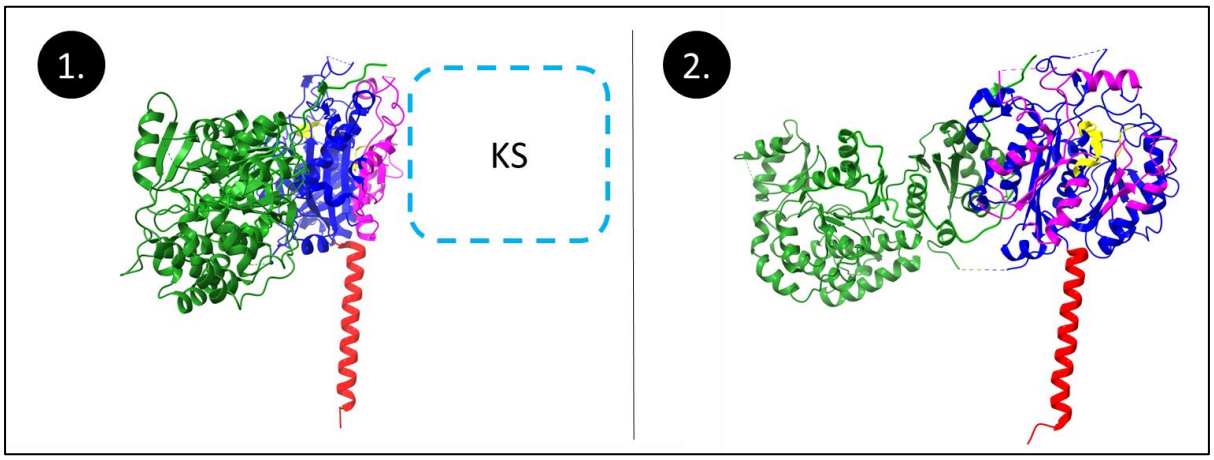


Figure 2: Monomer view of 8EE0[36] figure 1, highlighting all reaction centre motifs yellow and dimerization contacts magenta. Panel 2 show a 90° clockwise rotation of panel 1 around the vertical plane.

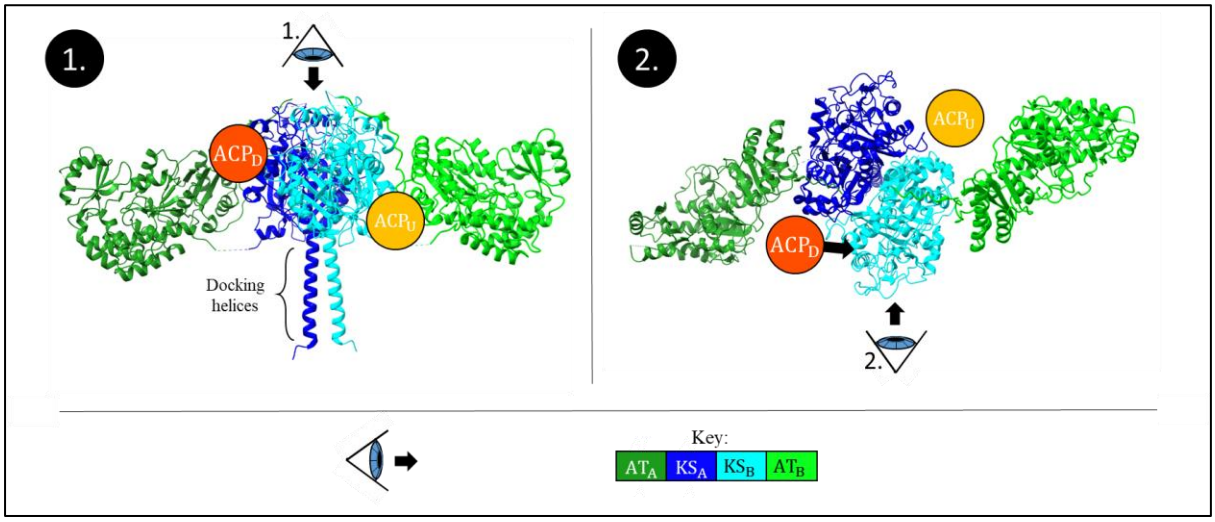


Figure 3: KS-AT didomain from module 2 of the 6-deoxyerythronolide B synthase (SeEryAI_Mod.2), PDB: 8EE0[36], viewed at 90° angles. Structures have been simplified by removing antibody fragments used in crystallisation. Arrows depict relative viewing angle in opposite photo. Ketosynthases have been depicted in cyan/blue, Acyltransferases in lime/green. The docking location or the upstream and downstream ACPs (ACP_u and ACP_D, respectively) have been added on the basis of erythromycin module 5 docking models in Kapur et al, 2010[43] and 2012[26]. We note that the relative angle that the AT domain is joining here may be an artefact of KS-AT didomains crystals, and may not representative of whole modules**. However we lean on the side of this representation of 8EE0 being correct, on the basis that this KS-AT angle is corroborated by a small-angle X-ray scattering (SAXS) study[33].

**Possibly higher order structures[44]. Not sure I fully agree with this – I suspect the EM work would have quickly picked up on higher order structures. Could be a crystal artefact? We already have evidence that the AT domains are 120º from the KS relative to what the KS-AT didomain structures show. Hard to say for sure. A counter argument is that a dehydrating or fully reducing module would have a dehydratase between the ketoreductase and AT domains, so it may be the case that the Dutta work is only applicable to the subset of ketoreductive modules.*

***Cryo-EM models of PikAIII shows the AT domains angled 120° relative to what is shown in figure 3[24], though this may be an idiosyncrasy of ketoreductive modules or single module PKS genes. The relative positions of AT and KS were also found to be dynamic during the reaction cycle[25], with AT domains moving to occlude active site entrances to the KS.*
This was an important development in the structural biology narrative for mPKSs because it highlights that any single crystal structure is going to miss the full story for how these multi-step, multi-domain synthases operate. Furthermore, this dynamism explains why crystal formation of entire PKS modules has been so elusive: crystals by definition are comprised of regular lattice structures, which becomes less feasible as the number of reaction cycle states increases for the target protein. This is not to say that X-ray crystallography is a futile pursuit in this arena, far from it. In fact, the cryo-EM models are dependent on high-quality X-ray structures to thread the EM models and so are very much dependent on them.

ACP docking to Ketosynthases

The body of literature on ACP-ketosynthase interactions proposes two models for how ACPs dock with ketosynthases. The first model is based on the canonical substrate channel for Fatty Acid Synthases (FAS) and PKSs (types I-III), which proposes that the upstream and downstream ACPs both dock at this channel, with the ACPs docking with different relative orientations to the channel[35]. We refer to this as the Khosla model, viewed in figure 3 above.

The second model is presented in figure 4, below, where the upstream and downstream ACPs dock at opposite sides of the ketosynthase, when viewing from the axis of dimerization. This model is supported by cryo-EM structures, which shows a second substrate channel open and close during the reaction cycle on the reaction centre side of the ketosynthase[24] (opposite side to docking domain). This gating of the channel is hypothesised to preclude a polyketide running in reverse and ensuring directionality in mPKSs[24, 25]. This model of separate sites for upstream and downstream ACP docking is further backed up by docking simulations for ACPs in the DEBS pathway[26, 43]. We refer to this model as the Dutta-Whicher model.

Downstream ACPs domains are semi-promiscuous in their interactions with the ketosynthase dimer. ACPs will take extension units from either AT domain in the dimer[45], but will only pass the extension unit to the KS on the opposite peptide. When picking up the extended polyketide, the ACP will only take from the ketosynthase on the same polypeptide[35, 46].

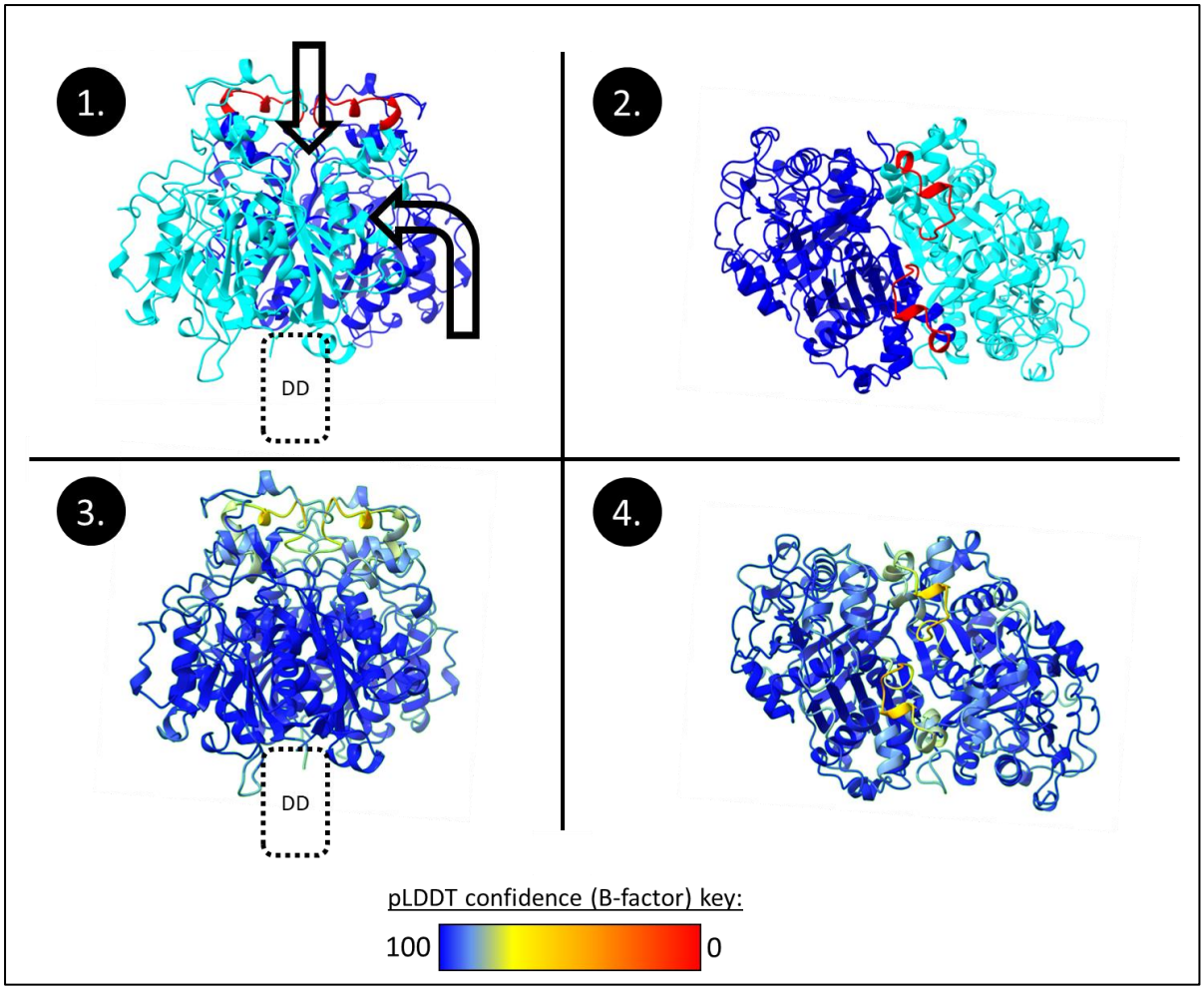


Figure 4: PikAIII_Mod.1 ketosynthase dimer generated in Colabfold multimer[47-49] for this chapter. Panels **1** and **2** show the show the monomer chains in cyan and blue. Residues 163-174 in cryo-EM PikAIII model[24, 25] (129-141 in this substructure) have been highlighted red in both monomers. These residues form a high-flexibility, low-conservation loop that surrounds the secondary access site to the KS[24]. This low conservation and high flexibility are echoed in the same residues in panels **3** and **4**, where they appear yellow, which display the beta factor scores of the AlphaFold model. Beta factor scores describe AlphaFold's confidence in the location of a residue and its R-group orientation. Mid to low confidence scores present as yellow through red[48]. Interestingly, low confidence frequently occurs for dynamic regions of proteins and can be an indicator that they are a moving component. This would agree with these residues have a gating function. **Panel 1** has 2 arrows which indicate the direction of approach of the downstream ACP (top arrow) and upstream ACP (curved arrow) in the Dutta-Whicher models[24, 25].

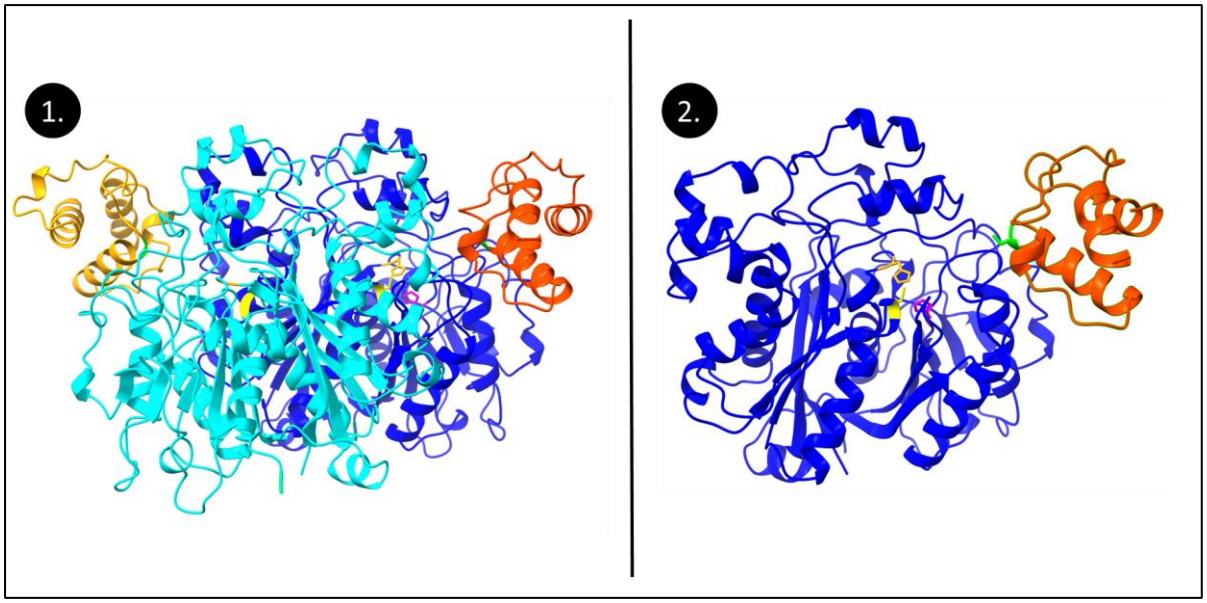


Figure 5: ColabFold (multimer)[47-49] prediction of EryAI_Mod.2 ketosynthase (blue and cyan) with upstream (orange) and downstream (red-orange) ACPs. Active site cysteine labelled yellow and histidines orange and magenta. **1.** Dimeric view. **2.** Monomeric view as seen from the perspective of the dimerization surface. ACPs both present facing into a substrate channel to the active site. Both structures are orientated downstream to upstream, top to bottom.

2.1.3 Ketosynthase data sources and interrogating data

For primary data sources of ketosynthases, at the time of writing, we have access to thousands of genomes for mPKS producing organisms through databases such as JGI[50] and NCBI[51], sampled from a range of environmental niches and biomes across the globe. Tools such as antiSMASH[18, 52-59] allow us to automate annotation of these genomes for natural product BGCs, which forms a symbiotic data eco-system with the MIBiG[52] database, a curated repository for mined BGCs. ClusterCAD[60] further builds on MiBIG by providing a verified-product curation of mPKS and NRPS BGCs, broken down into domain categories and associated compounds.

A recent paradigm shift in the field of structural biology has been the development of accurate predictions of protein structures from primary sequences using artificial intelligence[61], namely AlphaFold[48]. With AlphaFold's general release on GitHub and subsequent community modifications such as ColabFold[47] and AlphaFold-multimer[49, 62], it is feasible to generate accurate structure predictions for most types of proteins in minutes with relatively affordable and accessible hardware.

AlphaFold opens a bridge between primary sequences and 3D structures at scale. This newly accessible data offers a novel route for interrogating ketosynthase proofreading in a meaningful context that was not previously available. We can now ask questions about the relationship between the substrate and enzyme structure, at scale, to abstract systemic rules.

Ketosynthase structures are complex and differences across thousands of highly similar structures are going to be difficult to identify using meat-based hardware[63], especially if they are subtle, and so a computational approach is required here. By using datasets at scale, we should have a greater sensitivity for detecting subtleties against noise. This sensitivity may provide the difference between previously used techniques in the field and allow us to make a novel contribution.

Interrogating structural data at scale with Machine Learning

Machine learning is defined in the Oxford English Dictionary as *"The capacity of computers to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and infer from patterns in data"*[64]. In the simplest terms, machine learning uses defined mathematical processes (transformations) to learn patterns in a provided dataset, in order to extrapolate predictions on unseen data of a similar nature. For a gentle subject primer, we direct the reader to the review in ref. 61[65], though we have included a "need-to-know" introduction for biologists below.

Over the past decade, there have been substantial advances in the field of machine learning and crucially, API (application programming interface) packages such as TensorFlow[66] and SciKit[67] have made this field more accessible in terms of time investment and entry skill. Comprehensive, developer-level knowledge of the underpinning mathematics and backend code for running machine learning, though helpful, is no longer *essential* to operate in this space.

Machine learning can broadly be broken down into three non-exclusive categories: Supervised learning, Unsupervised learning and Reinforcement learning[65]. In these fields, we typically refer to input data and output predictions of the machine learning as features and labels, or X and Y, where the objective is to predict a label from a set of features (X predicts Y). Labels can exist as categorical data or as a continuum, depending on the nature of the task. In supervised learning, training data labels are known at the point of training. In unsupervised learning, the model will designate its own categories for labels. Reinforcement learning applies to models that learn from their own behaviours, where training data is the model's previous behaviour and, instead of labels, a scoring system is used to promote/penalise behaviours. In this chapter, we use supervised learning and semi-supervised learning.

What happens between features and labels is the crux of machine learning, and can range from simple mathematical transformations to complex multi-layered neural networks, so-called deep learning. In this chapter, we make use of neural networks. In essence, a neural network is a series of interconnected mathematical transformers that modify and amplify/dampen numerical information passed through them. Neural networks are *inspired* by neurons in biological nervous systems and imitate information processing in the brain[68] *. Just like the brain, neurons in a computational neural network are interconnected and can have activation thresholds**, defined by a mathematical function instead of the depolarisation voltage that characterises their biological analogue[69]. In this manner, a neuron can be thought of as having decision-making capacity on when/what to output. Neurons are implemented within an artificial neural network as layer(s) of user-defined density, which collectively allows a neural layer to make sophisticated decisions on complex input data.

Neural networks are not innately able to make accurate predictions on data and need to learn from experience, much like their biological analogues. We refer to this learning period as training, where we refine a machine learning model. Neural networks are trained via an iterative process that serves to refine weightings between neurons, which determines the emphasis applied on a neuron's response to a signal in the data, effectively adding biases on when a neuron activates. Weightings can be thought of as numerical representations of how important a signal is to a connection between neurons to an accurate prediction. Each iteration of this training process is referred to as an epoch.

Between epochs, the neural network's performance is evaluated and the direction of improvement to the neural weights is mathematically determined. This evaluation of performance is calculated using a loss function, which is used to adjust weights in a process known as backpropagation, where weights move against the loss function's gradient, thereby moving towards minimal loss and better predictive models. Movement against the loss function gradient is carried out by a second function, the optimiser, which governs how to move against this gradient. This is coupled with the learning rate (a hyperparameter), which describes how far to move along the function in a single step.

To summarise, the loss function describes the model's performance and direction of travel to improve the model, and the optimiser takes it there; the two combine to create a process that builds a model that learns over epochs.

To understand the significance of results produced in this chapter, the reader needs to understand three concepts that we use to judge the performance of a machine learning model: Loss, Accuracy and Data Partitioning.

Loss

The output of the loss function describes the difference between a model's predictions and the true values of the supplied labels. Loss inversely correlates with confidence in a model's predictions: higher loss scores indicate a greater difference between the model's predictions and corresponding labels true values.

To give an example of how loss works, let us imagine that we have binary classification model (a neural network that tells the difference between 2 categories) that is being trained to determine the difference between photos of cats or dogs. At the tail end of the neural network (woof), the data passing through the network will be funnelled down to a pair of neurons that output values between 0 and 1, where each neuron represents a category, and a 1 would indicate a 100% confidence in the category. The output will look like this: `[cat, dog]`.

Let us say the model makes a prediction on a supplied photo of a cat and produces a prediction of `[0.7, 0.3]`. We can see that the model thinks there is a 70% chance the supplied photo is of a cat, which means that for a rounding model, we would return the prediction of "cat", or `[1, 0]`. This would be considered an accurate prediction, because a photo of a cat should have a label of `[1, 0]`. The loss function serves to describe the confidence before the final prediction and describes the difference between the `[0.7, 0.3]` prediction and the `[1, 0]` true value.

Averaged out across a dataset, we can think of loss as a measure of how confident we can be in a model's predictions. A model that consistently made correct predictions, i.e. a highly accurate model, may still have a high loss value if the predictions are made with low confidence. For example, a prediction of [0.51, 0.49] for a photo of a cat would make for an accurate prediction after rounding, but we can see that the model is not confident in the prediction.

Methods for calculating loss are described by the loss function, and the choice of loss function is dependent on the type of task at hand. There are different ways that we could have skinned the cat in the cat/dog classifier. We could have specified the task as "cat" and "not-cat" with a single neuron. Likewise, we could have allowed a model to predict a photo as both cat and dog, `[1, 1]`, or neither cat nor dog, `[0, 0]`, by defining that the categories are not mutually exclusive. This could have also been constructed as a more complex task, where a third category that is not mutually exclusive, such as whether the animal is black, and so an output that correctly assigns the animal but not the colour can be half right.

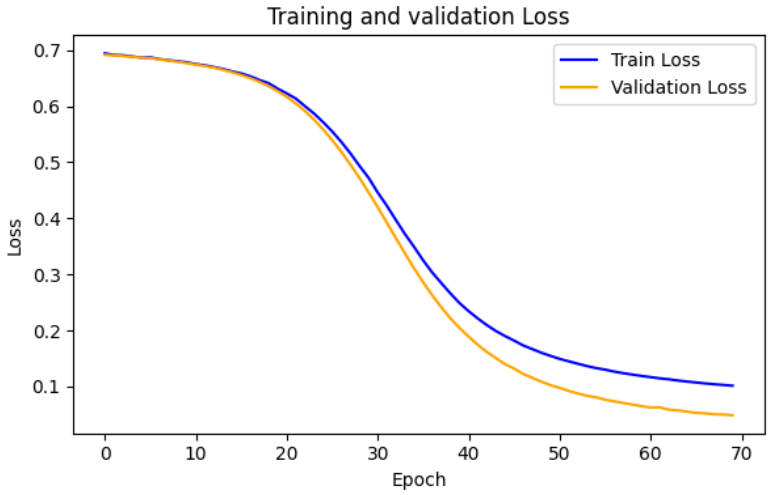


Figure 6: Example loss over epochs graph, derived from figure 38 in section 2.2.3 below, which is the product of a binary classification model. We can see that in early epochs, loss is high. As the model learns information about the relationship between the features and labels, loss decreases, eventually reaching a plateau. From a training optimisation standpoint, the loss graph can be used to inform hyperparameter tuning, which determines when and where the final model is reached.

Accuracy

In our cat/dog example, the photo was either a cat or a dog. We can specify this in the model design as mutually exclusive categories. Knowing this, we can enforce that the model must output `[1, 0]` or `[0, 1]`, cat or dog, so a hypothetical score of `[0.7, 0.3]` is rounded to `[1, 0]` because our task is categorical. Accuracy, when averaged across a dataset, can be thought of as a measure for how good a model is at making a correct prediction.

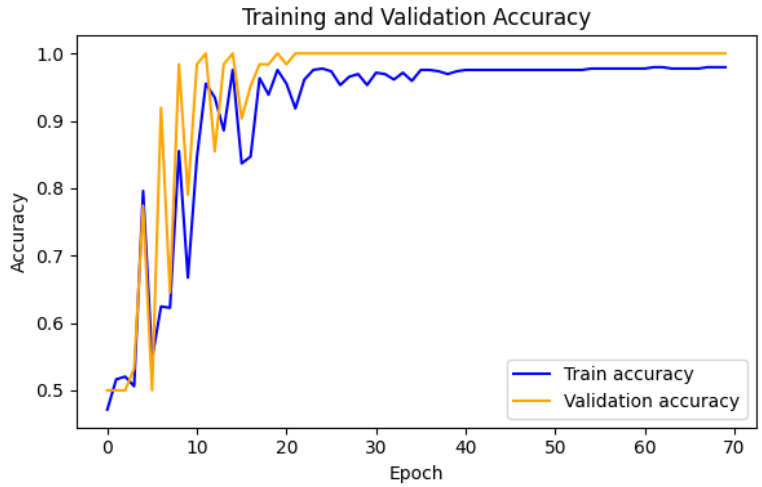


Figure 7: Example graph of accuracy over epochs. This was produced in conjunction with figure 6 above and is derived from figure 38 in section 2.2.3 below. These two graphs provide a good example of why loss and accuracy are both important metrics. We can see that between epochs 15 and 20, accuracy approaches its maximal value in the test and validation sets. However, we can see in the loss graph that during epochs 15-20, the model is not as confident in its predictions as it could be, which it reaches a plateau for around epoch 50.

Receiver Operating Characteristic (ROC) and Area Under Curve (AUC): an alternative to loss and accuracy

Accuracy and loss metrics offer simple measures of a model's performance. Loss is a useful metric from the perspective of the data scientist training the model because it tracks the rate and direction of learning over epochs***, which helps us to determine where and when to stop training, as well as spotting traits that can prompt hyperparameter tuning. Loss is not ideal for scientific communication because it does not have a defined upper limit (loss can go to infinity), and so is a relativistic measure. Likewise, as explained above, accuracy does not directly report on the confidence of predictions. We also lose information about how confident a prediction was due to rounding when measuring accuracy in classification tasks.

As an alternative to loss and accuracy, we can use Receiver Operating Characteristic (ROC) curves to measure a model's performance. A ROC curve is plotted as False Positive Rate (FPR) versus True Positive Rate (TPR) for classification thresholds within the classifier (i.e. both cat **and** dog) and is derived from the probabilities outputted by the model without rounding. This preserves information on a prediction's confidence. FPR and TPR form 0-1 probability scales on the x and y-axis respectively, and so scales here are comparable between models. What the ROC curve describes is the probability the model will make a correct classification prediction.

From the ROC curve, we can extract the Area Under the Curve (AUC) via integration, which measures the relationship between FPR and TPR and infers the model's performance. The AUC metric ranges from 0 to 1, where a value of 0 indicates 100% of predictions are wrong and a value of 1 indicates that 100% of predictions are correct. For a binary classifier, a random category assignment returns an AUC value of 0.5 (half correct, half incorrect).

As a measure, AUC can be thought of as a blend of the confidence provided by the loss score and the precision capacity of accuracy. Something to note about AUC is that it can report with higher scores than accuracy. This can be an indication that incorrect predictions are being made with low confidence, which AUC penalises less than high confidence incorrect predictions.

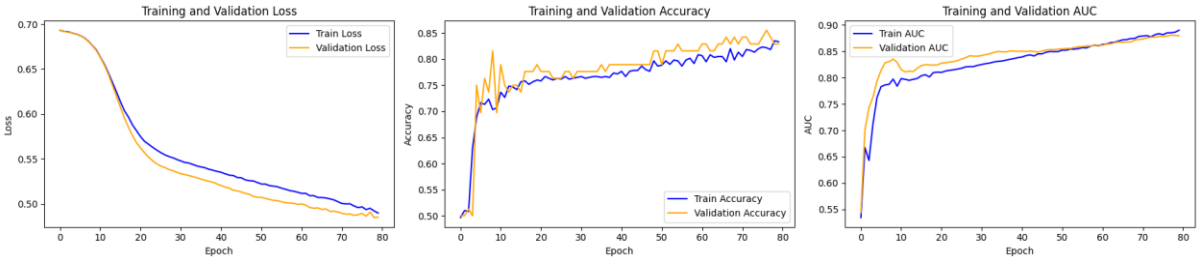


Figure 7.2: Loss, accuracy and AUC over an 80 epoch training period. We can see that accuracy and AUC broadly follow the same trend over epochs, however, AUC is less erratic over epochs and for this model, AUC concludes at a higher score than accuracy.

Data partitioning:

One of the problems that we can encounter with machine learning is that the neural network can learn the idiosyncrasies of the training data, rather than meaningful abstractions of categories at hand. This is referred to as overfitting. To evaluate overfitting, we can use a validation data set, which is a partition of the total data available. This partition is not used to train the model directly****, so the model *shouldn't* be able to learn idiosyncrasies of the validation set. Typically, we use 10–20% of the total data for the validation set.

As the model is trained, we want the training and validation data to trend with each other in the loss and accuracy metrics, which is indicative that the network is learning representations of the data that are

applicable to unseen data of a similar nature. Validation data sets need to be non-redundant with respect to the training data to be effective. Validation set non-redundancy is hard to achieve. In effect, we want to create a partition where individual examples retain the class-determining characteristics that the model needs to learn, whilst stripping out the idiosyncratic characteristics. Given that, for the most part, we will not be able to discern what is idiosyncratic and what is meaningful (that is why we are using machine learning in the first place), this is hard to implement.

A third partition, test data, is sometimes made. Typically, we use this if the validation data is being used to adjust hyperparameters as the model trains, such as modifying learning rate.

**Comparisons between artificial neural networks (ANNs) and the biological neurons are not perfect. ANNs can perform operations that neurons cannot. Likewise, neurons are capable of performing complex computations beyond on/off responses, such as taking in temporal and spatial information from connections, forming local charge environments etc. As ANNs have progressed, they have become more divorced from their brain-inspired inception.*

***Neurons can output either continuous activation values (i.e. any number on a continuum between 0 and 1) or a binary 0 OR 1 response. Traditional neural networks take the prior approach and spiking neural networks (SNNs) take the latter. SNNs would be considered more true to their biological inspiration[70].*

**** Reminder: the loss function that determines the direction of weight adjustments that provides the mechanism for learning over epochs.*

*****Validation data is often used to indirectly influence training, where is can be used to tune hyperparameters such as the learning rate between epochs. This can also be used to determine where the model stops in a process called early stopping, which is a protection against overfitting (we manually approximate this process later, see figures 53 and 54 for an example). Early stopping and dynamic learning rates are subject to the idiosyncracies of the validation set, and so may not always be optimal, which is where a third, test partition comes in.*

2.1.4 Pilot on AT domains

One of the limitations of neural networks is that the process from input to output, and by extension an explanation of why a network makes its decisions, is not readily available in human interpretable format[71] *. Compounded with this, machine learning can produce over-fitted models that tests as accurate, but are actually just learning idiosyncrasies of their training data, and so can fail to make meaningful predictions on non-redundant new data.

This prompts the question: if we build a model that presents as accurate, how can we trust it? Likewise, if a model is unable to produce accurate results, how do we discern between a failure of the model's design from a failure due to the data having no learnable associations between features and labels? To answer these questions, we tested our data processing and machine learning models on a pilot data set with known solutions, to act as a benchmark.

AT domains:

Like ketosynthases, Acyltransferase (AT) domains are an essential component of mPKS modules. Functionally, they import the carbon extension units used to elongate the growing polyketide[9, 27]. AT domains have been observed to have a range of substrates that they are specific for, however the most frequently occurring substrates in canonical mPKS AT domains are 2S-methyl-malonyl-CoA (Mmal) and malonyl-CoA (Mal)[72], which we find holds true in this body of research (see figure 13).

AT domains have been well studied and the structural and catalytic motifs that distinguish a Mmal-AT from a Mal-AT are well defined. These motifs have been demonstrated at an experimental level with catalytic-motif directed mutations[73-76], albeit with reduced activity of the mPKS:

- Mmal: YAS**H**, GH**S**QG
- Mal: HAF**H**, GH**S**[I/V]G

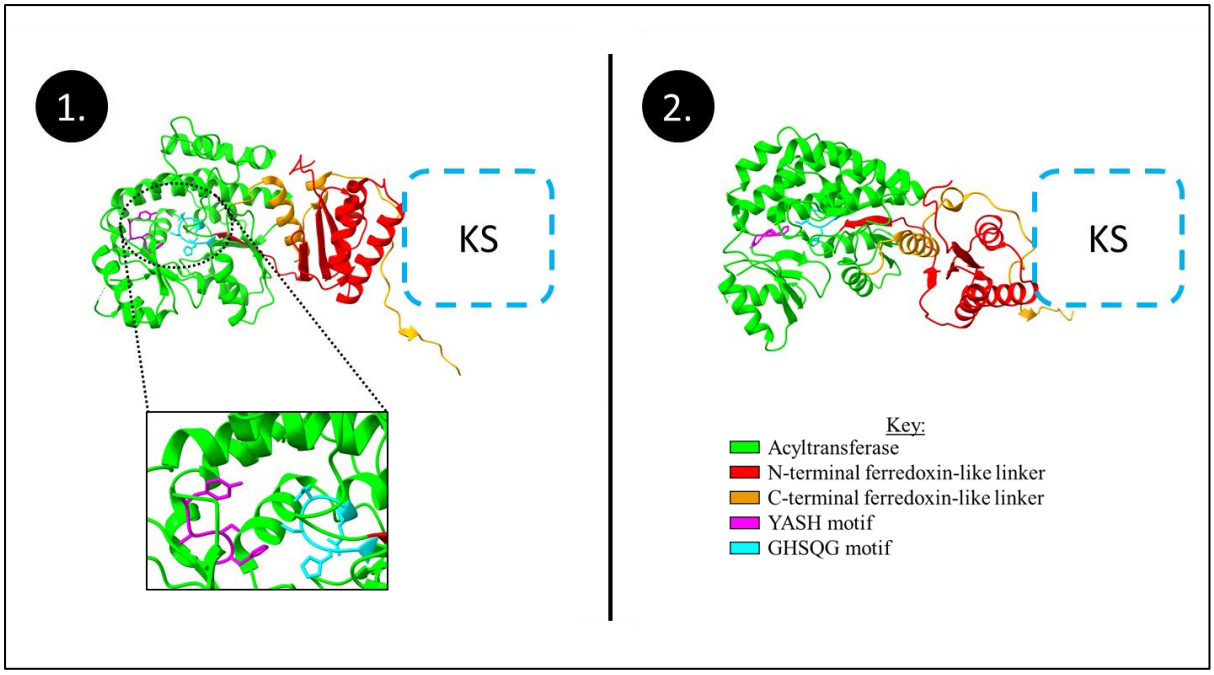


Figure 8: Acyltransferase and ferredoxin-like linker from module 2 of the 6-deoxyerythronolide B synthase (SeEryAI_Mod.2), PDB: 8EE0[36]. This AT domain has substrate specificity for Mmal and contains the associated consensus YASH (magenta) and GHSQG (cyan) motifs. Our definition for the borders of an acyltransferase are highlighted in green, with the N-terminal and C-terminal components of the ferredoxin-like interface coloured red and orange respectively. For simplicity of annotation, we use the antiSMASH annotation, which ultimately derives from the InterPro definition of Acyltransferases[77, 78]. Panels 2 is a 90º rotation of **1** in the vertical axis (2 views 1 from above).

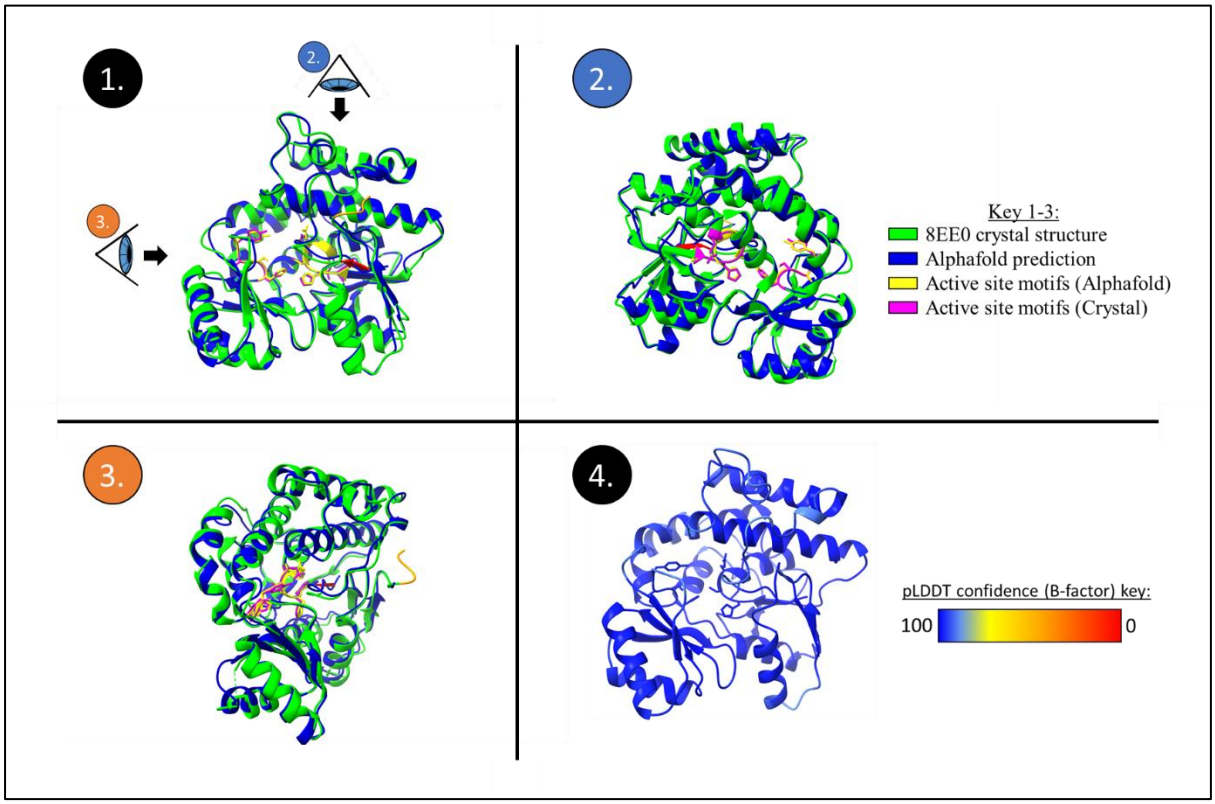


Figure 9: **Panels 1-3:** Alignment of AT domain structures of SeEryAI module 2 as a crystal structure (PDB: 8EE0, green) and an AlphaFold predicted structure for AeEryAI_Mod.2, which PDB does not have a crystal for (Blue. See section 2.2.1 for method of generation). AlphaFoldprediction uses the InterPro definition for AT domain borders[77, 78]. Active sites YASH and GHSQG motifs have been coloured magenta and yellow respectively. **Panel 4:** AlphaFold crystal structure coloured by pLDDT confidence score. Note: all areas exhibit high confidence (above 90%). Active site residues show near perfect alignment of R-groups.

From the information presented above, I am of the opinion that distinguishing between Mmal and Mal AT domains is a case-study with a known motif-based solution, and one that can involves comparable datasets as for KS domains. Mmal and Mal AT domains can therefore serve as a benchmark for method development in predicting substrate from 3D structures.

**Interpretability of neural networks is stymied by the number and complexity of neurons in the network, which makes it difficult to disentangle what the model is using to make decisions. Furthermore, activation functions can create one-directional mathematical transformations, such as converting a continuum into a binary, which means that a neural network cannot be run backwards from label to features.*

2.2 Results

N.B. All scripts mentioned in this chapter were written from scratch by the author, unless stated otherwise.

2.2.1 Data acquisition and processing, from online database to aligned 3D protein structures

The process for generating the protein structure datasets used in this chapter is as follows:

1. Reviewed MiBIG BGC accession numbers were acquired from ClusterCAD[60], via webscraping script `01.01.01_ClusterCAD_Webscrape.R` and stored as a dataframe, `DF3.csv`.
2. DF3.csv is then used to specify antiSMASH[18, 59] BGC files to download from MiBIG[52, 79] in script `01.01.02_MiBIG_Webscrape.R`. These files are parsed and annotated mPKS domains are sorted to `.csv` files, with columns physically arranged to capture the domain and module arrangement of domains. This script is imperfect and outputs a `missing_BGC.csv` file that require manual curation. Largely these problems come from naming domains in NRPS-PKS hybrid pathways. This script then outputs a file *`yymmdd`*`_mastersheet.csv`, which is a collation of all `BGC.csv` files mined without error, where *`yymmdd`* indicates the date the script was run.

	N.term.do	C.term.do	KS.0	AT.0	KR.0	DH.0	ER.0	MT.0	ACP.0	TE.0	KS.1	AT.1	KR.1	DH.1	ER.1	MT.1	ACP.1	TE.1	KS.2	AT.2	KR.2	DH.2	ER.2	MT.2	ACP.2	TE.2	KS.3
eryCIV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
eryBIV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
eryAI	NA	NA	NA	FVFPGQG	NA	NA	NA	NA	APPHVLRE	NA	PVAVVAN	FVFPGQG	TGTVLVTG	NA	NA	NA	ALASLPAP	NA	PIAIVGMA	FVFPGQG	GGTILVTG	NA	NA	NA	RLAGLPRA	NA	NA
eryAII	SEKVAEYL	GDRLDELE	NA	NA	NA	NA	NA	NA	NA	NA	PIAIVSMA	FLFPGQG	SHGTVLVTG	NA	NA	NA	LLELVANA	NA	PIAIVGIG	CLVFPGQG	DGTVLITG	EHPLLLAA	DGAIDSVA	NA	AELVRSHA	NA	NA
eryAIII	TEEKLRRY	NA	NA	NA	NA	NA	NA	NA	NA	NA	PIAIVGMA	MVFPGQG	RGTVLVTG	NA	NA	NA	RLAALSTA	NA	PIAIVGMA	FVFPGQG	AGTALVTG	NA	NA	NA	AAPAREM	VICCAGTA	NA
eryG	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
thioesterz	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	ARLVCFPH	NA	NA	NA	NA	NA	NA	NA	NA	NA

Figure 10: BGC0000055 (erythromycin, *Saccharopolyspora erythraea*) sample parsing script output. KS, AT and ACP domains have been highlighted blue green and orange respectively. NA columns indicate that there is no relevant domain in this position. Note that all genes in BGC are captured in this process and any genes that contain a relevant key word will for an entry (example, *trans*-acting thioesterase gene, bottom row, has an entry for TE.1).

3. For AT domains, the mastersheet.csv file is used in script `01.01.03_ClusterCAD_AT_scrape.R`, which scrapes buttons on ClusterCAD for SMILES, AT substrate type and reductive domain information, and combines this into file *`yyyymmdd`*`_AT_sheet_and_reductive_domains.csv`.

    a. Dataset `AT#1` was manually sampled and erroneously passed off as correct (see section 2.2.3 below). This produced an interesting result.
    b. A full manual review was completed to create dataset `AT#2`, which corrected the errors in `AT#1`.
4. For KS domains, the `mastersheet.csv` file is used in script `01.01.04_ClusterCAD_KS_scrape.R`, which scrapes buttons on ClusterCAD for SMILES and KS substrate type, and combines this into file yymmdd_KS_SMILES_unified.csv. This corresponds to dataset `KS#1`

N.B: Module data in 3 and 4 was stored by module number in the gene in format gene_Mod.X, where X is the module position in the gene. A "Mod.0" suffix indicates that the domain precedes the first ketosynthase.

5. Sequences for the AT and KS domains are then converted to .fasta files using script `01.01.05_fasta_write.R`. This script has the option to write as monomers (for ATs) or dimers (for KSs).
6. The .fasta files were run through a local (server) instance of ColabFold[47-49] version 1.5.1, using default settings to generate 3D structures monomer structures (ATs). The multimer setting was used to generating KS dimers.
7. ColabFold outputs 5 models per structure, and the top ranked structures (outputted as "rank 1") from the output of step 6 were extracted and renamed using `01.01.06_Rank_1_extraction.R`.
8. Rank 1 structures were then aligned against a template structure, using Python script `01.01.07_alignment_python.py`.

Packages used in steps 1 to 8 include Rvest 1.0.3[80], Polite 0.1.3[81], Tidyverse 2.0.0[82], Genbankr 1.26.0[83], Stringr 1.5.0[84], Biostrings 2.66.0[85], Dplyr 1.1.3[86], Seqinr 4.2-30[87] and Prody 2.4.0[88].

Summary of datasets

Datasets can be found on the GitHub page under section `02 Data`.

67-121C	Abyssomicin	Actinoallolide	Aculeximycin	Aldgamycin
Angolamycin	Ansamitocin	Argimycin	Azalomycin	Bafilomycin
BE-14106	Borrelidin	Brasilinolide	Butyrolactol	Candicidin
Chalcomycin	Chaxamycin	Chlorizidine	Chlorothricin	Coelimycin
Concanamycin	Coronafacic	Cremimycin	Cylindrospermopsin	Cystothiazole
Dihydrochalcomycin	Divergolide	E-837	Ebelactone	ECO-02301
Epothilone	Erythromycin	FD-891	Filipin	FK520
Fluvirucin	Fostriecin	Geldanamycin	Gephyronic	Glidobactin
Griseochelin	Halstoctacosanolide	Herbimycin	Herboxidiene	Heronamide
Hitachimycin	Hygrocin	Incednine	Indanomycin	Jerangolid
Kendomycin	Kijanimicin	Laidlomycin	Lankamycin	Lasalocid
Lipomycin	Lobophorin	Lydicamycin	Macbecin	Mediomycin
Megalomicin	Meilingmycin	Meridamycin	Methymycin	ML-449
Monensin	Mycinamicin	Mycolactone	Nanchangmycin	Naphthomycin
Natamycin	Nemadectin	Niddamycin	Nigericin	Niphimycin
Nocamycin	Nocardiopsin	Nostopeptolide	Nystatin	Nystatin-like
Oligomycin	Phenalamide	Phenylnannolone	Phoslactomycin	Piericidin
Pladienolide	Quartromicin	Rapamycin	Reveromycin	Rifamycin
Rosamicin	Salinilactam	Salinomycin	Sanglifehrin	Selvamicin
Simocyclinone	Spectinabilin	Spinosad	Stambomycin	Streptazone
Streptolydigin	Tautomycetin	Tautomycin	Tetronasin	Tetronomycin
Tiacumicin	Tirandamycin	Tylactone	Versipelostatin	

Figure 11: Biosynthetic gene clusters used in dataset `KS#1`.

Domain type	Total
Acyltransferases (AT) – total	971
• Mmal-AT	359
• Mal-AT	529
• Other-AT	83
Ketosynthases (KS) - total	1208
• Non-reducing (NR)	122
• Ketoreducing (KR)	410
• Dehydrating (DH)	404
• Enoyl-reducing (ER)	164
• Edge cases	108

Figure 12: Summary of total protein structures used in this chapter in datasets `KS#1` and `AT#2`. Note that where a reductive state is being given for a KS, this is referring to the β-carbon coming off the ACP upstream to the KS. The difference between the total numbers of AT and KS is a result of using an expanded range of BGC for the KS structures.

Number of modules in gene	Frequency
1	234
2	196
3	92
4	48
5	12
6	16
7	1
8	1
9	1
10	0

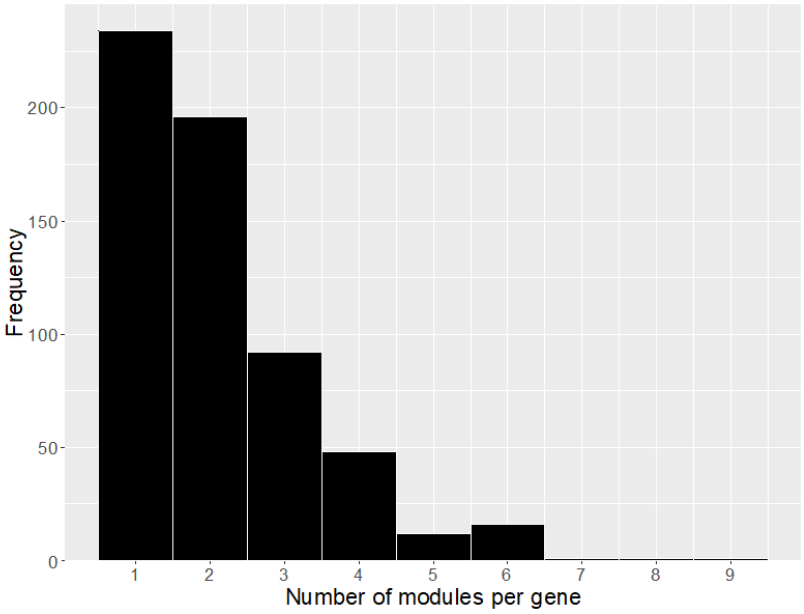


Figure 13: Frequency of modules per gene/polypeptide within dataset, table and histogram.

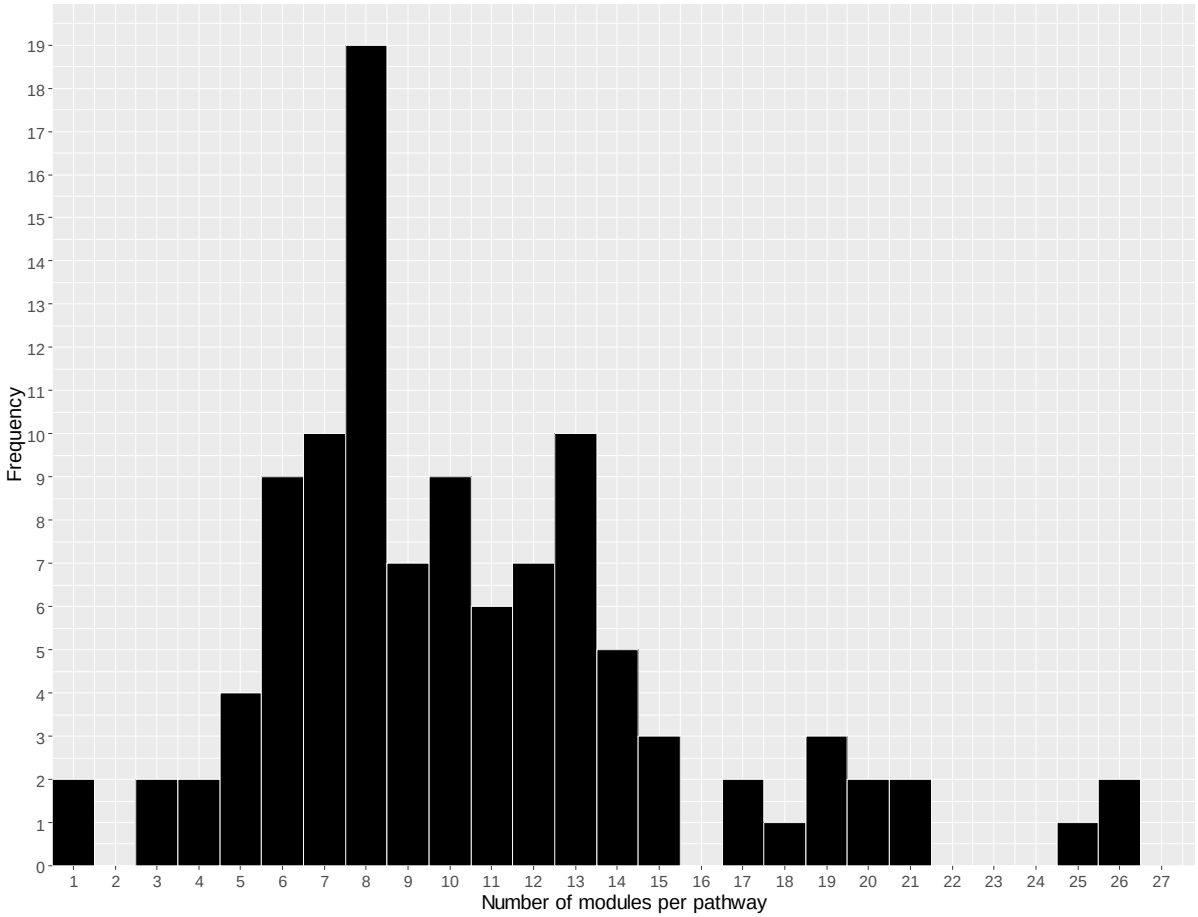


Figure 14: Frequency of modules per mPKS pathway within dataset. Note: Only modules that contain a ketosynthase are counted. AT-ACP loading modules are not counted. Two pathways are NRPS hybrids that only contain a single PKS module (Nostopeptolide and Glidobactin).

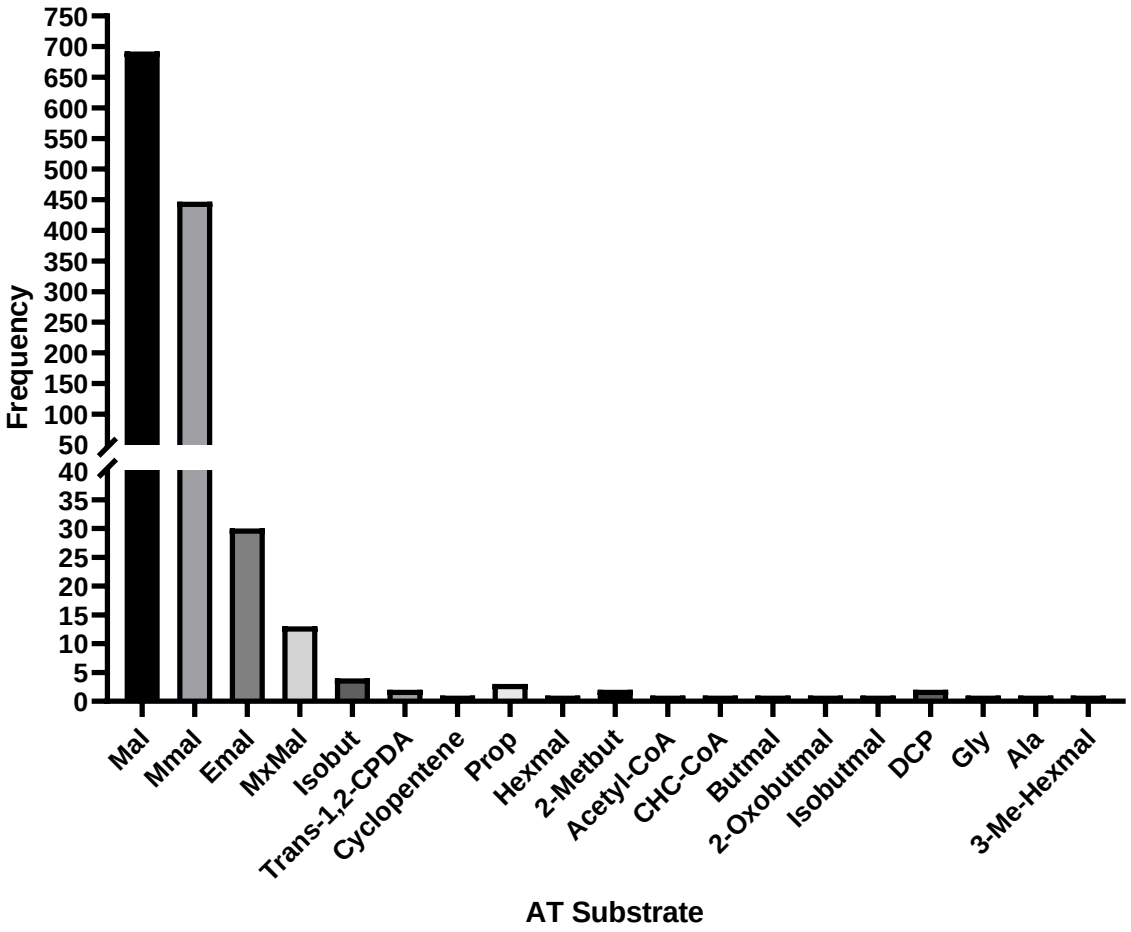


Figure 15: Distribution of AT substrate preferences in output KS#1. Note split y-axis at 40.

Data sanitation

Structure data was manually screened for extreme coordinates in X, Y and Z axes using an interactive plot (exampled in figure 16 below). The objective of this was to assess the uniformity of the data and identify structures that deviated substantially from the mode. This is important because we normalise data to 0-1 scales for machine learning, and extreme cases can skew results. In figure 16 below, we can see a general trend of uniformity in the minimal/maximal coordinates.

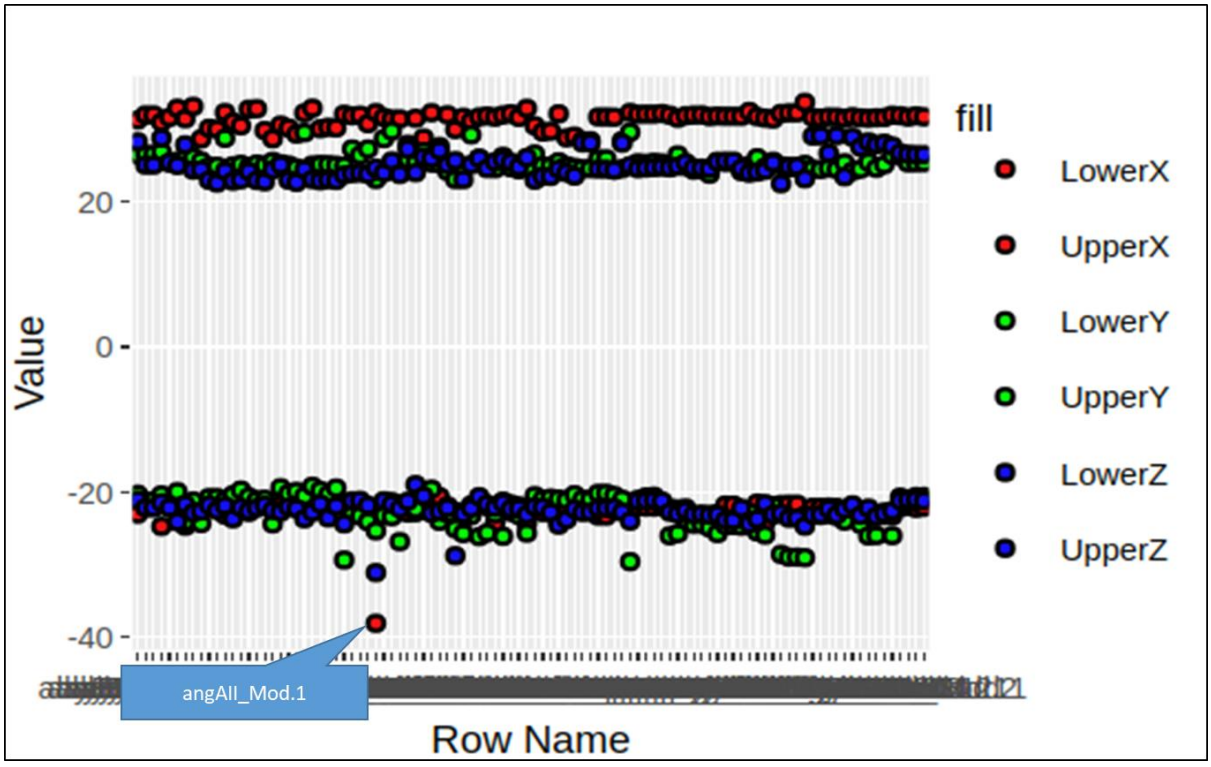


Figure 16: Dataset AT#2 AT domain maximal and minimal X, Y and Z coordinates. Highlighted in the dataset is angAII_mod.1, which has unusual lower bound X and Z coordinates.

Figure 17: Structure alignment of angAII-mod.1 AT domain (red) to abyB1-mod.1 AT domain (white). Note the red loop on the left of the structure. This is explains the outlying coordinate values in figure 16. The structure for angAII-mod.1 was removed from the structure dataset.

2.2.2 Principal component analysis of datasets

AT Domains in dataset AT#2

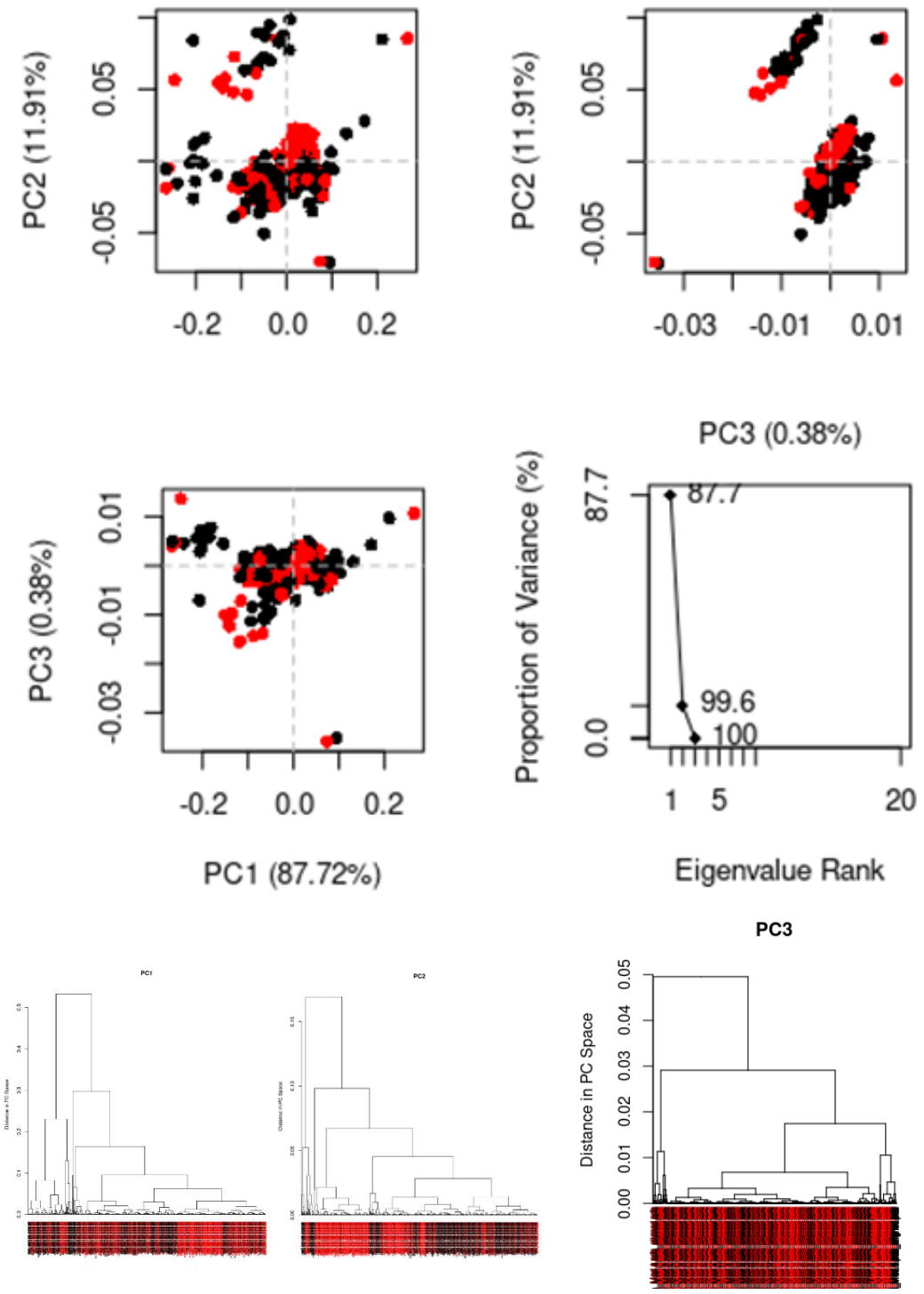


Figure 18: Principal component analysis of AT structures in dataset AT#2 using Bio3D's `PCA()` function[89] (top) and dendrographic representations of principle component graphs using `hclustplot()` function. Structures with a malonyl-CoA substrate preference are labelled red and those with a methyl-malonyl-CoA preference are labelled black. Dendrograms describe the relational clustering within components.

Principal component analysis in dimension PC1 and PC3 do not appear to form distinct clusters based on substrate preference. PC2 accounts for 11.91% of variation and does appear to show a degree of clustering in the PC2 dendrogram.

Phylogenetically, AT domains form clades based on Mal/Mmal, and not by species [27] [90].This means that we might expect to see clustering of structures based on substrate specificity in the figure 18 graphs. We do not see clear clusters forming in the PCA graphs (top) though the dendrogram for PC2 does appear too show some separation of mal from mmal (red from black).

KS domains in dataset KS#1

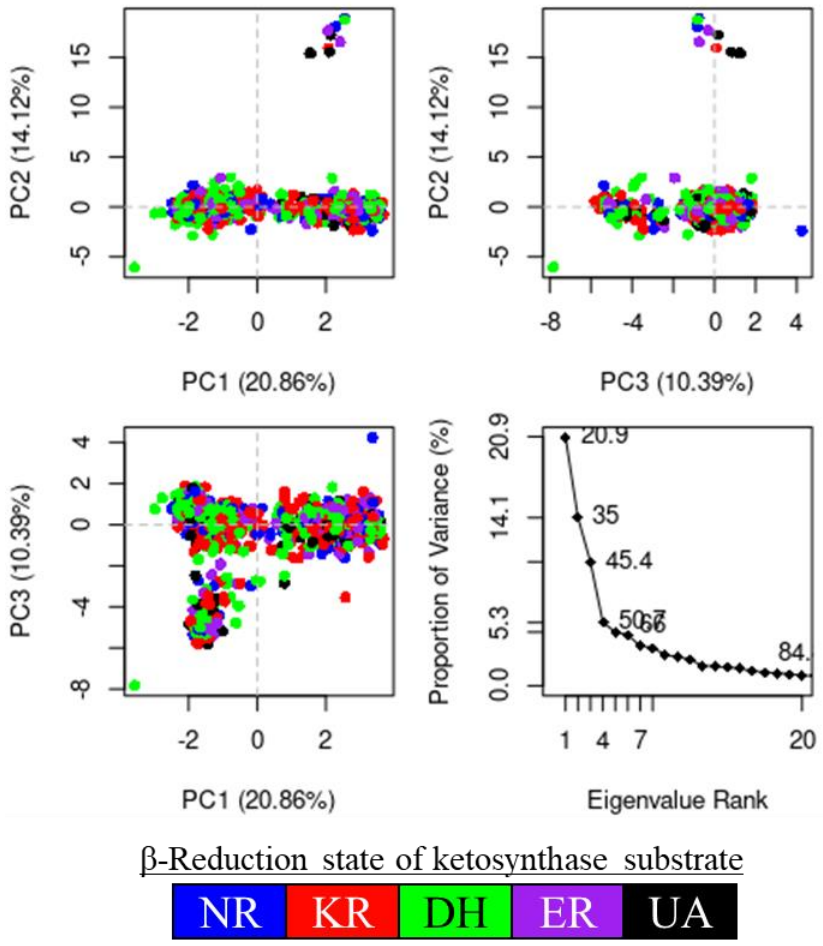


Figure 19: Principal component analysis of ketosynthase structures in data set KS#1. Reduction states are non-reducing (NR), ketoreduced (KR), dehydrated (DH), enoyl-reduced (ER) and unassigned (UA). Categorisation as "unassigned" is applied to ketosynthases that have irregular reductive domain compositions, such as containing a dehydratase but no ketoreductase.

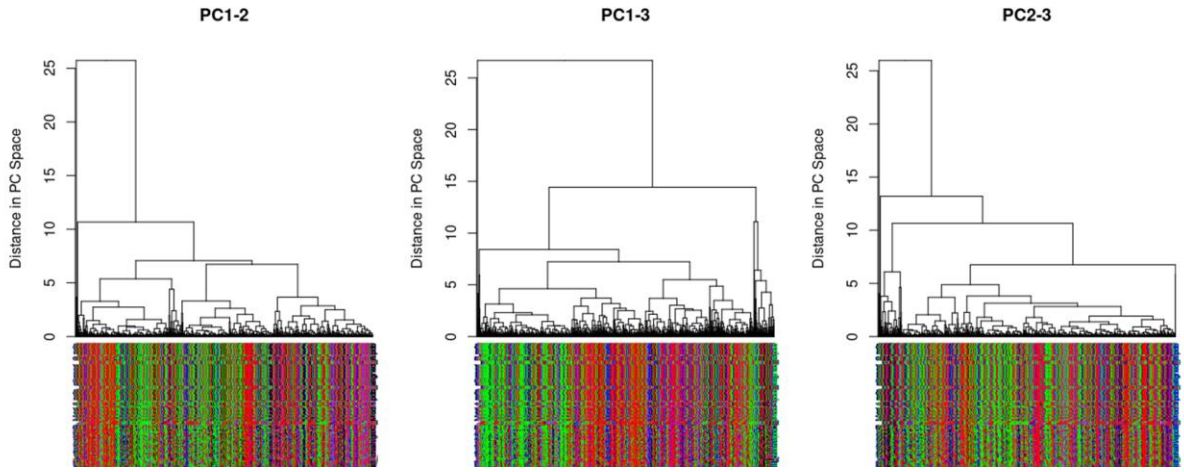


Figure 20: Dendrogram representation of principal components graphs in figure 19. Colour coding follows figure 19.

PCA analysis for ketosynthase structures does not appear to show substantial clustering by reduction state on the polyketide substrate's β-carbon.

2.2.3 Analysis of AlphaFold structures

AlphaFold `.pdb` file β-factor scores (pLDDT) were used to create aligned average β-factor structures, using scripts *01.01.08_AT_pairwise_alignment_B_and_NetX.R*, for AT domains, and *01.01.09_KS_pairwise_alignment_B_and_NetX*.R for KS dimers. The process in this script is as follows:

1. Import AlphaFold `.pdb` files from folder and define a reference sequence to perform the alignment against (arbitrarily chosen).
2. Create a dataframe of structures and β-factor scores per residue.
3. Create an alignment index per substrate category.
4. Perform pairwise alignment against reference, removing unaligned residues from query.
5. Average β-factor scores across alignments.
6. Write new `.pdb` file for reference sequence with average β-factor scores.
7. Script *01.01.10_Crossword_graphs.R* can then be run on the `.pdb` files to generate word search diagrams seen in figure 21 below.

N.B: these scripts are also used to generate the heat map structures and logo diagrams used in the graph learning section later.

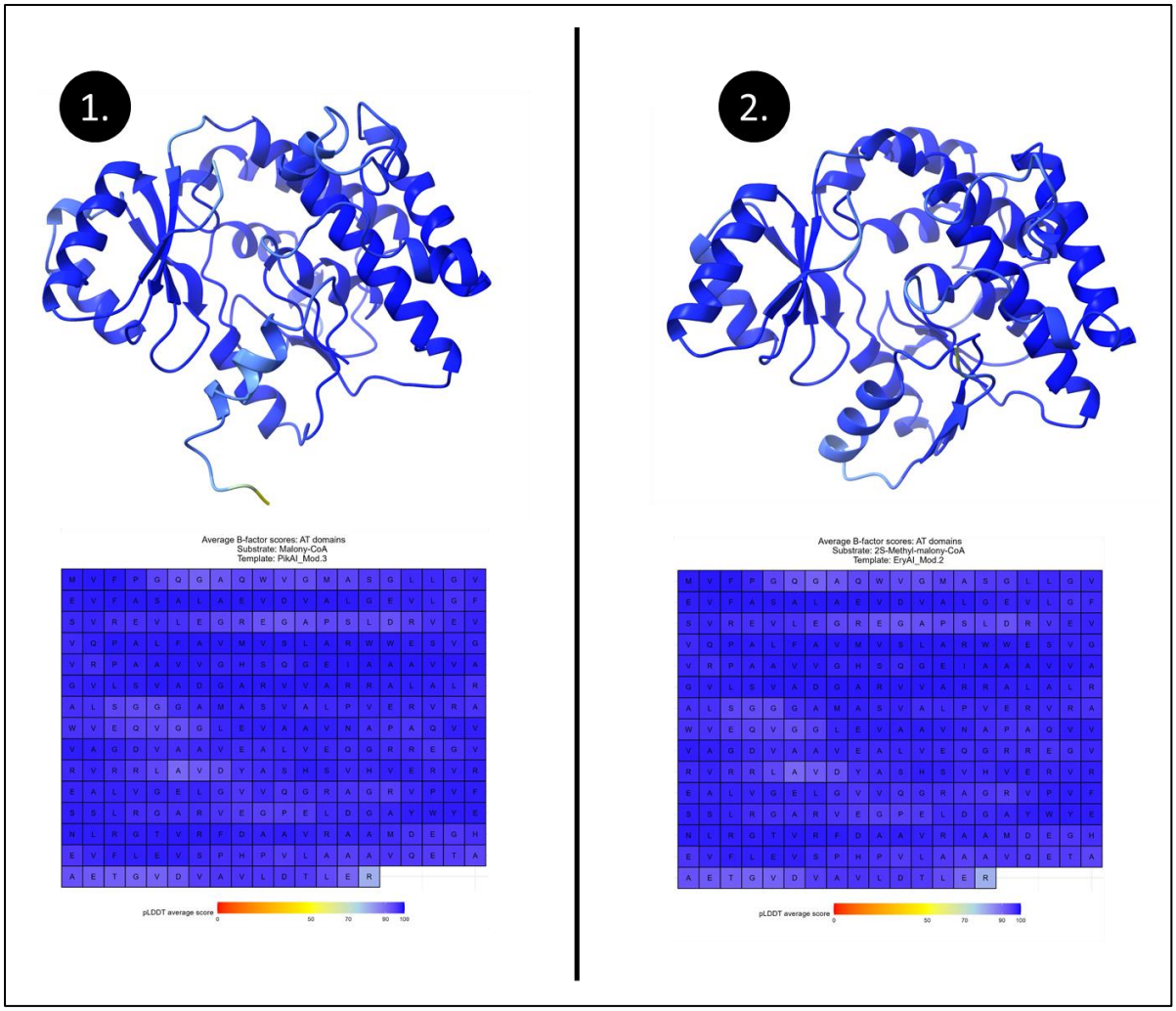


Figure 21: Total dataset average β-factor scores for **1.** Malonyl-CoA selective (n=529) and **2.** 2S-methyl-malonyl-CoA selective (n=359) AT domains. Alignments have been performed against PikAI_Mod.3 and AeEryAI_Mod.2 respectively. Top panels show the 3D structures, bottom panels display word search diagrams. Colouring in both diagrams follows AlphaFold's pLDDT colour scheme[48].

AT domains are predicted with very high accuracy, with a pLDDT score >90% for almost all residues. This indicates a high degree of confidence in both the position of the alpha-carbons, as well as the rotational confirmations of the amino acid R-groups.

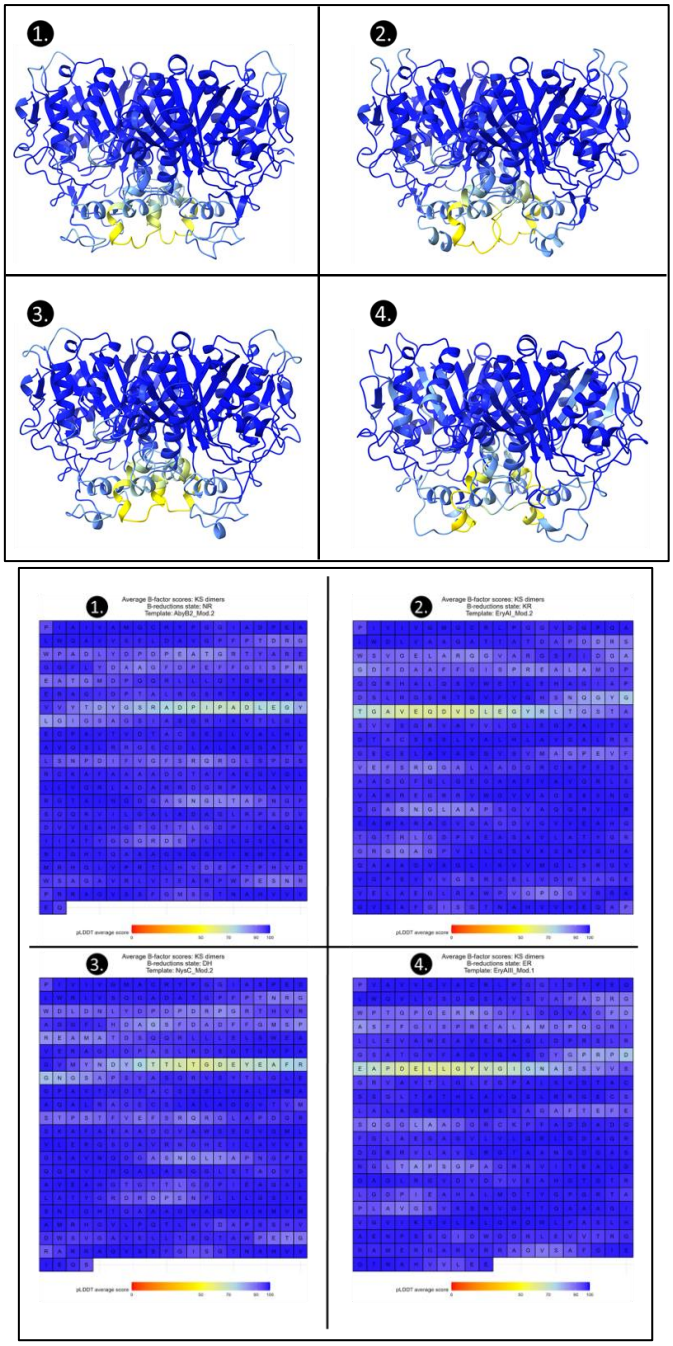


Figure 22A: Total dataset average β-factor scores for ketosynthase dimers, divisible by incoming polyketide substrate β-carbon reduction state. **1.** Non-reduced (ketone), aligned to AbyB2_Mod.2. **2.** Keto-reduced (alcohol), aligned to AeEryAI_Mod.2. **3.** Dehydrated (carbon-carbon double bond), aligned to NysC_Mod.2. **4.** Enoyl-reduced (fully reduced), aligned to AeEryAIII_Mod.1. Top panels show the 3D structures, bottom panels display word search diagrams. Colouring in both diagrams follows AlphaFold's pLDDT colour scheme.

Ketosynthase dimer structures are not predicted with uniform high accuracy by AlphaFold. The core structure around the active site is confidently predicted (>90% pLDDT) for all reduction types; however, a pair of α-helices and neighbouring loops at the bottom of the structure (opposite end to the docking domain position, so on the downstream side of the structures) appear yellow (50-70% pLDDT).

This indicates a lower confidence in the positions of the α-carbons, and low reliability on the positioning of R-groups.

I am of the opinion that the lower scores for these α-helices are explicable. Cryo-EM modelling found these helices to be dynamic part of the ketosynthase reaction cycle, where they serve to open a secondary access route for ACP access to the active site, post condensation reaction[24, 25]. Given that we expect these helices to be dynamic, we would not necessarily encounter expect high positional confidence in these regions.

It is also worth noting that the AT domain runs back to the KS before forming the reductive loop, where it forms an interface with the yellow helices. It may well be the case that the AT domain is needed to fold these confidently.

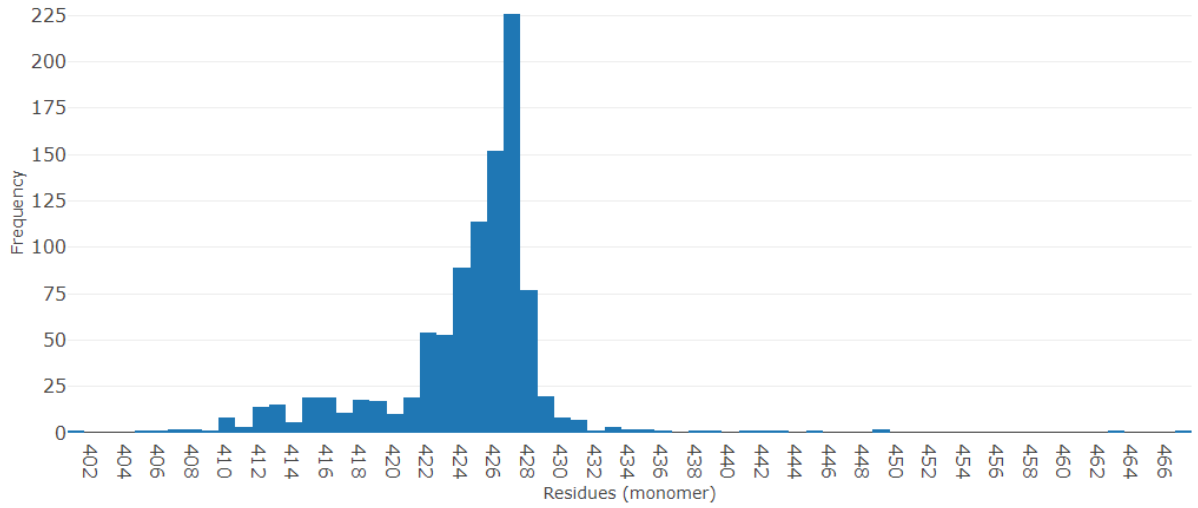


Figure 22B: Frequency of residues per ketosynthase monomer in dataset KS#1. Structures mostly fall within a range of 420 to 430 amino acids.

2.2.4 Machine learning approach 1: 3D convolutional neural networks for predicting AT domain substrate and KS substrate reduction state

Convolutional neural networks

The first approach we used on the AT-domain pilot data was to convert atomic coordinate data into voxels (pixels with a Z-dimension), then deploy 3-Dimensional (3D) convolutions on the voxel data and run through a neural network (referred to as a convolutional neural network, CNN).

Convolutions, in a mathematical context, refer to mathematical operations that combine two functions to create an amalgamated third function. Through convolutions, we combine input features to create a description of how the features relate to each other. This is particularly useful for describing spatial relationships.

In this section, convolutions provide a means for mathematical description and pattern recognition of atomic neighbourhoods. The strength of CNNs is that a convolutional layer creates representations of the local area around each pixel/voxel. In image recognition, this is effective at picking out edges and contrasts in an image. Applied here, a CNN's strength is that it enables pattern recognition in the atomic structure arrangement. The idea to use these for protein structure recognition was prompted by the success and wide adoption of convolutional neural networks (CNNs) in image recognition, where it is an industry standard tool[91].

Convolutions for image recognition and the approached used in this section (2.2.4) operate via a sliding kernel method, which takes a small n-D matrix (a kernel) and multiplies it across a n-D data structure to create a second, convoluted matrix.

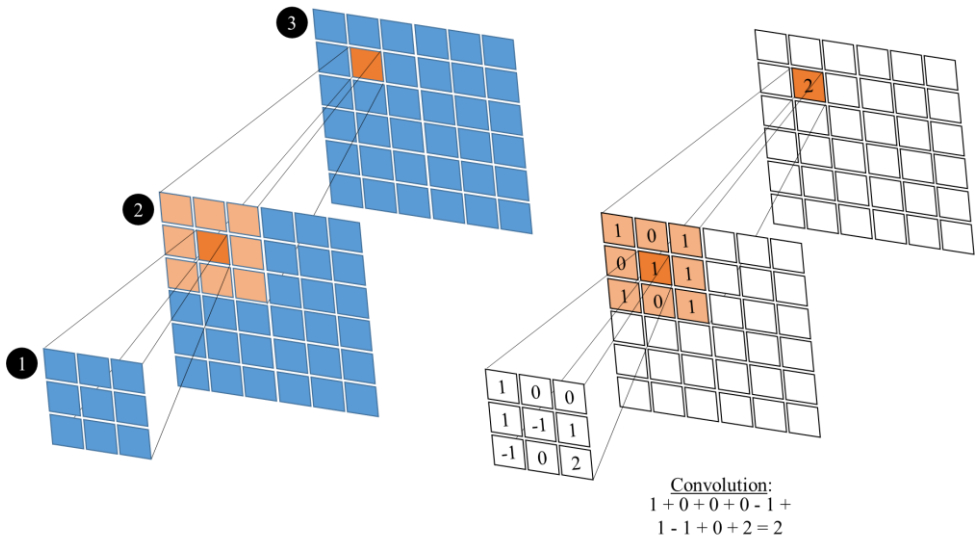


Figure 23: Visual representation of how a 2D kernel (1) passes over a 2D feature matrix and performs a matrix multiplication (2) to produce a convoluted feature matrix (3). For a 3D convolution, the kernel and matrices would be cube shaped. What we see in the left hand side is that the kernel creates a numerical receptive field around a data point in a matrix. On the right, we see that the kernel allows for differential amplification of data around the target data point. In this example, the data point in the bottom right is being given particular attention, whilst the bottom left and centre are being penalised. Through this particular instance of the convolution, we can see that a value of 1 is amplifying to a value of 2.

Generation 1: 100x100x100 voxel cube
Script: 01.02.01_100xcube.R
Packages: Bio3D[89], Tensorflow[92], Keras[93]
R version: 4.3.1
OS: Ubuntu 20.04.6 LTS

In this script, we convert 3D atomic coordinates contained within the rank 1 AlphaFold `.pdb` files into a cubic representation of the location of atoms. A 100x100x100 voxel cube was chosen for two reasons: first, the larger the cube gets, the further apart the atoms are, which increases the void space and reduces the ability of the convolutional layer to describe an atom's neighbourhood effectively. This can be compensated to a degree by increasing the kernel size, to take in a wider neighbourhood, but the amount of empty space remains the same and the kernel's output becomes more complex.

The second reason is more practical: for a dataset of 668 AT structures, the RAM (Random Access Memory) requirement during training is approximately 40GB, which is feasible on a local computer with a swap file. Increasing the cube size has a linear relationship with the RAM requirement, so a 1000x cube would require 400GB, an unfeasible amount. When presented with datasets comprised of mostly zeros values, referred to as sparse data, one typically compresses this into an encoding format known as a sparse tensor[94]; however, this is not readily available for convolutional networks in our chosen API[93] (the convolution requires decompression*). At 100x, 668 structures take approximately 12 minutes to complete an epoch for the hardware available. As with the RAM, this will scale with the size of cube, increasing training time.

The drawback of compressing to a 100x cube is that we decrease the accuracy of distances between atoms, due to rounding. PDB files use a 1:1 coordinate to angstrom ratio, which are specified to 3 decimal places. All coordinates in the AT files fall within a 100 Å range (figure 16), so when we confine the atomic model to a 100x cube, our axis will be a 1:1 scale. The rounding will mean that the atoms are up to 0.5 Å out from their pdb file coordinates, in any axis. Reducing accuracy is not ideal, but when we consider that crystal structures in a 1.5 Å to 2.8 Å range are considered medium to high-resolution structures[95] this presents as a minor issue**.

**Solutions exist for performing convolutions on sparse tensors[94], but these have not been implemented in general use APIs, such as Keras and PyTorch (yet). Using these would involve rewriting the convolutional network functions in Keras, which is beyond the scope of my ability for the time constraints of the PhD.*

***We could pull at this thread even more. To what degree do we trust the accuracy of AlphaFold's predictions? For that matter, if structures we use were genuine X-ray structures, reporting at sub-Ångstrom accuracy, how much would we trust the biological relevance of the structure? Most crystals are grown under conditions far removed from their biological native state, with respect to buffering, temperature and concentration.*

Process:

1. Aligned rank 1 `.pdb` files are collated into a unified data frame.
2. Mmal/mal bias is reported and excess mal domains are removed at random from the data frame. This produces a dataset with balanced labels.
3. Atomic coordinates are moved into a positive only space via a linear transformation. Here we take the lowest negative values for X, Y and Z across all structures, then add them to all atoms in all structures, respectively.
4. Coordinates are then converted into a 0–1 range by dividing all atomic coordinates by the highest value coordinate in X, Y and Z across all structures.
5. Data is then partitioned into training and validation data, with a 90:10 split respectively.
6. A data frame consisting of a stack of 100, 100x100 tables is created for each structure.
7. Atomic coordinates are used to populate the stack in binary. Z atomic coordinates are used to specify which table in the stack, X and Y specify which cell in the table the atom is recorded.
    a. Due to the condensed nature of the cube, atoms that pass within 0.5 Å may occupy the same voxel. Where this happens, the value remains at 1.
8. The stacks are then collated into a 5D tensor (tensor dimensions: `[number of genes, 100, 100, 100, 1]`) for training and validation data.
9. The tensor is run through a CNN in Keras. Architecture map below.

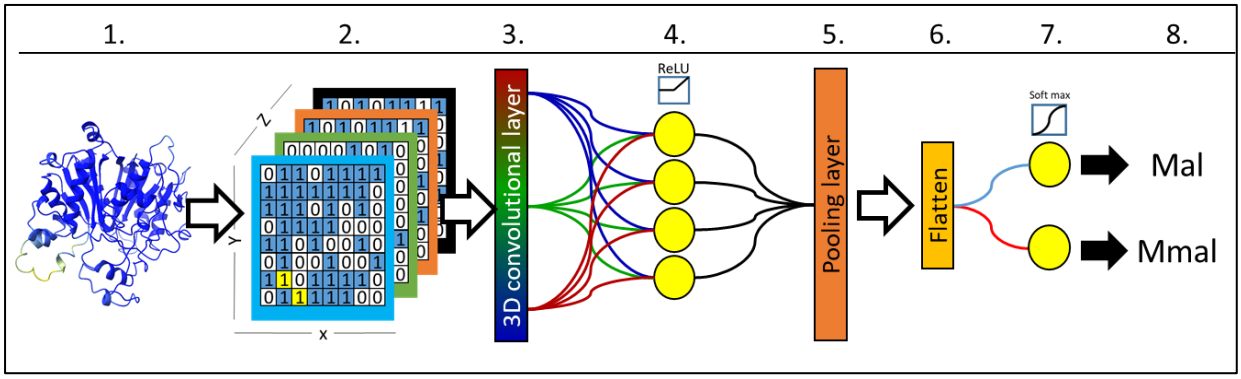


Figure 24: Data processing workflow and CNN architecture for 20230614_100xcube.R **1-2:** Structures are converted to a voxel dataframe. **3.** A 3D convolutional layer is run on the voxel dataframe and, **4.,**

passed through a 64x ReLU layer. **5-6.** Data is pooled and flattened. **7.** Data outputs to a binary neural layer with a sigmoid activation function. **8.** Output is rounded to `[0, 1]` or `[1, 0]` corresponding to a Mal/Mmal prediction.

Generation 1 results

Algorithms and Hyperparameters

Dataset: AT#3
Loss function: Categorical cross entropy
Optimiser: Adam (Adaptive Momentum estimation)[96]
Epochs: 10
Batch_size: 35
Validation_split: 10%
Seed number: Random

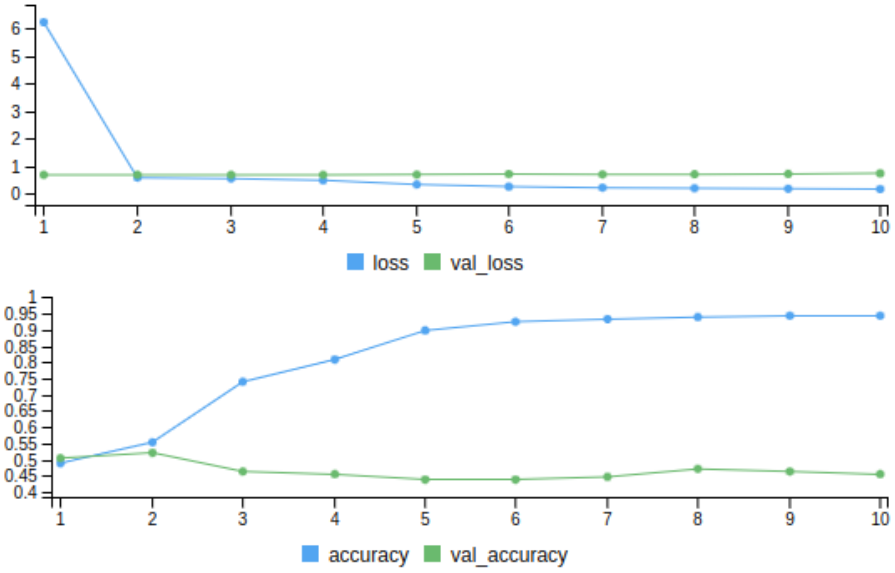


Figure 25: 10 epochs of training.

The validation data results show no substantial improvement in accuracy over training epochs, whilst the training data accuracy reaches 94% in the final epoch. Where meaningful representations of the training data are being learnt we expect the validation data to trend with the training data (some variation within epochs is normal). The accuracy graph in figure 25 does not show this trend and so we can conclude that the model is not learning meaningful representations of the training data.

This generation of model architecture was considered to have failed on the pilot data and was not used to make predictions on the KS dataset as a result.

Generation 2: photographs of a cube method

Script: 01.02.02.01_AT_cube_face_CNN.R
Packages: Bio3D[89], Tensorflow[92], Keras[93]
R version: 4.3.1

OS: Ubuntu 20.04.6 LTS

In the second generation, the convolutional dimensions were reduced from 3D to 2D. In this method, we compress the voxel cubes made in the previous generation into 3x (100x100) pixel surfaces in the XY, XZ and YZ axial planes of the cube. Atoms are mapped to the surfaces based on their planar coordinates. The compressed dimension's coordinates in the plane are converted to a 0-1 value, based on the depth from the surface to the atom and applied as a pixel value. This can be thought of as taking photographs of the voxel cubes generated in the previous method, where the intensity of the pixel corresponds to how near/far it is to the photographer. In this manner, we encode depth in the missing flattened axis.

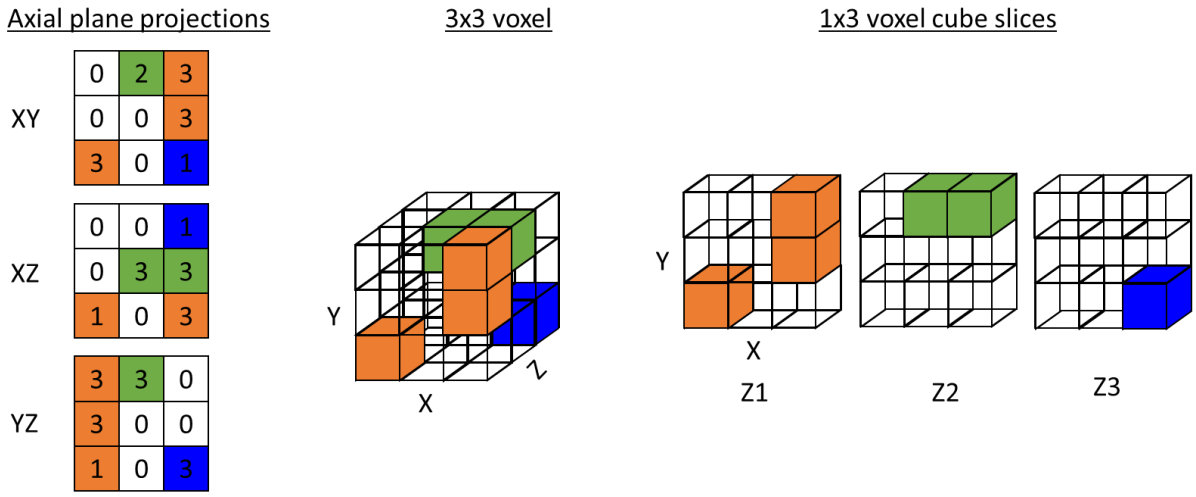


Figure 26: Axial plane projections of a hypothetical 3x3 voxel cube. For these projections, we are encoding the depth in missing direction numerically (these numbers are converted into a 0-1 range for the machine learning step, what is shown here would be the relative numerator for this conversion).

Process:

1. Aligned rank 1 `.pdb` files are collated into a unified data frame.
2. Mmal/mal bias is reported and excess mal domains are dumped at random from the data frame.
3. The pdb atomic coordinates are moved into a positive-only space via a linear transformation. Here, we take the lowest negative values for x, y and z across all structures, then add them to all atoms in all structures, respectively.
4. Coordinates are then converted into a 0-1 range by dividing all atomic coordinates by the highest value coordinate in x, y and z across all structures.
5. Data is then partitioned into training and validation data, with a 90:10 split respectively.
6. A data frame consisting of a stack of 3x (100x100) tables is created for each structure.
7. Atomic coordinates for each plane are used to populate the relevant stack, with the missing plane being used to encode a depth as a 0-1 value (see figure 27).
8. The stacks are then collated into a 4D tensor [`number of genes, 3, 100, 100`].
9. This is run through a CNN in Keras. Architecture map below.

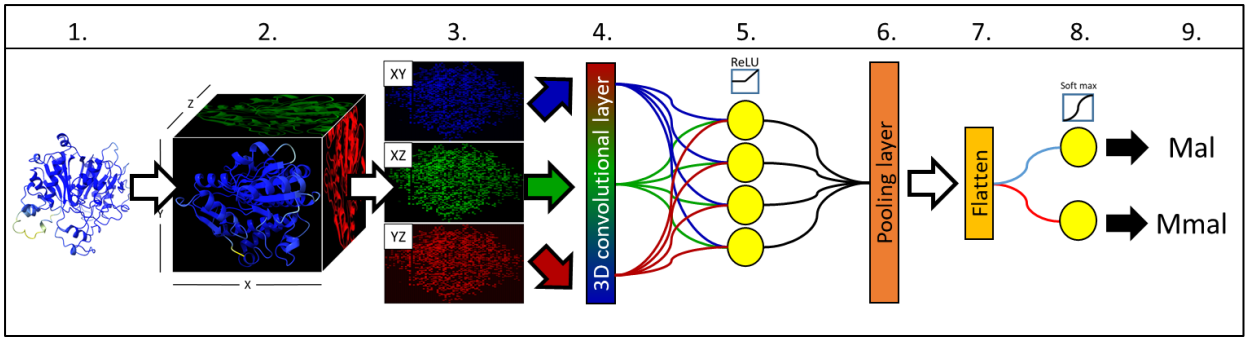


Figure 27A: Data processing workflow and CNN architecture for script *01.02.02.01_AT_cube_face_CNN.R*. **1-3:** AlphaFold structures are converted to 3x 2D representative pixel maps in XY, XZ and YZ planes, with depth in missing plane encoded as pixel intensity. **4-5.** A 3D convolutional layer is run over the 3 planar representations and passed through a 64x ReLU layer. **6-7.** Data is pooled and flattened. **8.** Data outputs to a binary neural layer with a sigmoid activation function. **9.** Output is rounded to [0,1] or [1,0] corresponding to a Mal/Mmal prediction.

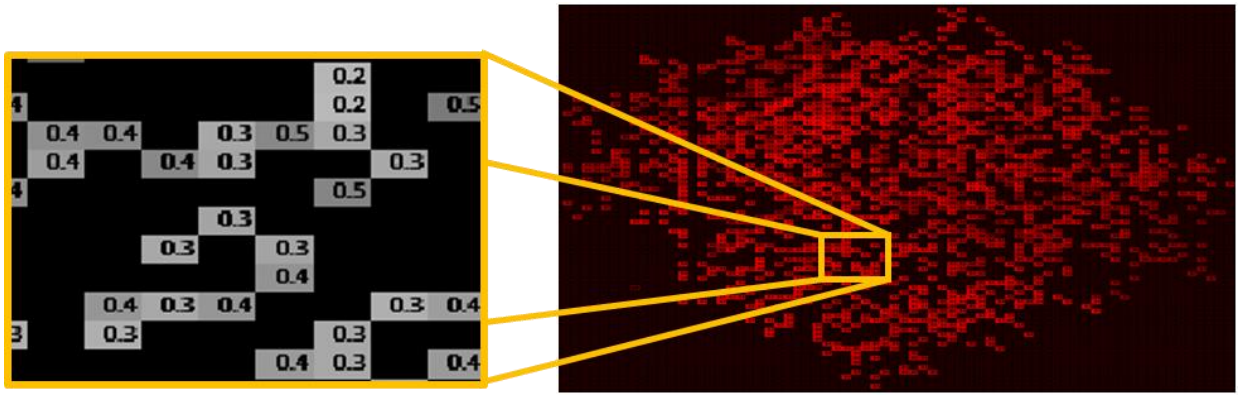


Figure 27B: Pixel intensities correspond to depth in missing axis.

Generation 2 results

Algorithms and Hyperparameters:

Dataset: AT#3
Loss function: Categorical cross entropy
Optimiser: Adam (Adaptive Momentum estimation)[96]
Epochs: 15
Batch_size: 35
Validation_split: 0.3
Seed number: 2

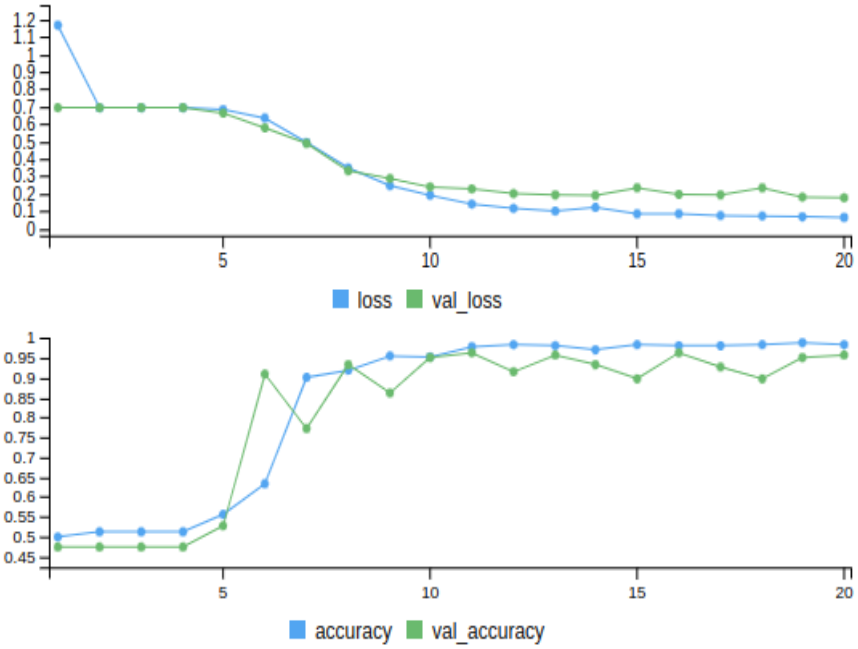


Figure 28: loss (top) and accuracy (bottom) over a 20 epoch training cycle for training data set (blue) and validation data set (green).

The accuracy graph in figure 28 shows a model that is learning to perform accurate predictions in a meaningful way. Validation accuracy increases proportionately with training accuracy, showing that what is being learnt for the training data is also applicable to the validation data.

Mislabelled AT domain predictions

Running the script *01.02.02.01_AT_cube_face_CNN.R* several times on dataset AT#1 using random data partitioning and initial seeds, we observed that the training approached but never reached 100% accuracy. This prompted us to question if the outliers were consistent across runs. To investigate this, we recorded which genes are incorrectly predicted in the training and validation data after each independent run. We found after 10 runs that there was a subset of modules that were frequently being predicted as the opposite label type (figure 29, below).

The AT domain labels data (mal/mmal) used in dataset AT#1 was produced from the web-scraping process (section 2.1.1), and had been manually sampled to check that the data was consistent with ClusterCAD and the antiSMASH outputs. Random sampling of BGCs had not indicated any immediate errors in the labelling. Using genes identified in red in figure 29 below, we found that an omission had been made in the module number binning if the PKS was an NRPS hybrid for the first module. This prompted a full manual review of all BGC labels to ensure they were correct. This is interesting because the model was able to detect that the labels we gave were incorrect, further supporting the notion that the trained model was learning meaningful representations.

However, manual review did not account for all the frequent-offender AT domains. We found that 12 modules were predicted incorrectly with a frequency higher than 20%. Protein sequences for these modules were searched for the catalytic motifs known to be determining mal/mmal selectivity. Nine were found to have catalytic motifs that are associated with the prediction the model made, and where the ClusterCAD annotation shows the opposite, indicating that the model thinks that ClusterCAD's annotation is incorrect. Three of the AT domains were found to have catalytic motifs that deviate from

the consensus. A final AT domain, gfsD_Mod.2, was both correctly annotated on ClusterCAD and in agreement with its catalytic sites.

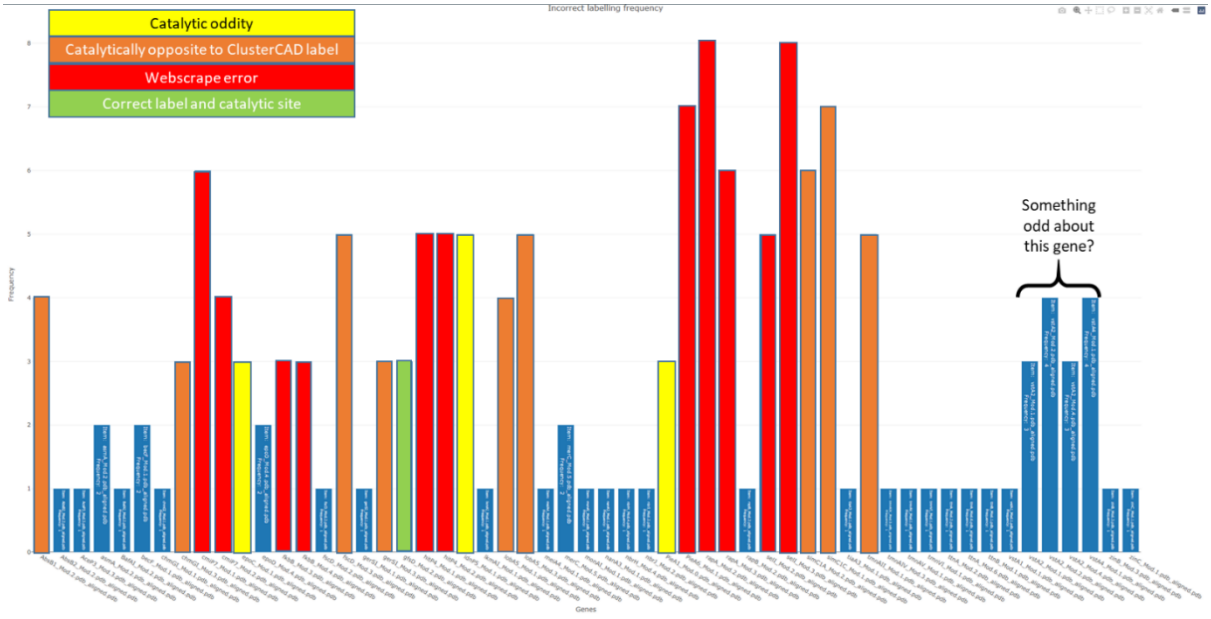


Figure 29: Histogram recording frequency of incorrect predictions of AT domains over 10 independent training runs. Modules with a frequency above two were manually reviewed and colour coded. **Red:** Modules mislabelled by the webscraping script. **Orange:** Modules that contain active site motif of the opposite substrate label but are correct per ClusterCAD's annotation. **Yellow:** Catalytic oddities, where the active site motifs do not conform or approximate the YASH/HAFH and GHSQG/GHS[I/V]G motifs consensus. **Green:** Correctly labelled and with canonical active sites motifs.

Gene	BGC	Site 1	Site 2	ClusterCAD label	Correct label
absB1_Mod.2	Abyssomicin BGC0001492	YASH	GHSQG	mal	Strong case for mmal
Notes on absB1_Mod.2: 2 clusters for abyssomicin are given on ClusterCAD: BGC0001492 and BGC0000001, sourced from genomes for *Verrucosispora maris* and *Streptomyces sp. LC-6-2* respectively. Both SMILES are accurate for abyssomicin C. BGC0000001 module 2 is has the HAFH, indicating a mal substrate. There exists several isoforms of abyssomicins, of which *S. sp. LC-6-2* is reported to produce 12 *""enantiomeric"-like abyssomicin metabolites (M-X)*", of which abyssomicin-W and abyssomicin-X are reported with a structures coherent with a mmal being incorporated by module 2[97]. Abyssomicin I produced in *Streptomyces* sp. CHI39 also has a structure consistent with a module 2 selecting for mmal[98].					
chmGI_Mod.3	Chalcomycin	YASH	GHSQG	mal	mal[99]
epoC_Mod.1	Epothilone	VASH	GHSMG	mmal	Mmal and mal[76]
EpoC is a promiscuous AT domain that can use either Mal or Mmal to make epothilone A and epothilone B respectively[76].					

fscD_Mod.3	Candicidin	YASH	GHSQG	mal	Mmal[100]
Notes on fscD_Mod.3: Chen et al.[100] report module 13 (fscD_Mod.3) as selecting for mmal. The 2S methyl group is not part of the final product however. They hypothesise that FscP and FscFE act as a tailoring P450 monooxygenase that oxidise this group, though they note that this is missing an oxidoreductase step.					
gerSI_Mod.3	Dihydrochalcomycin	YASH	GHSQG	Mal	Mal[101, 102]
idnP5_Mod.1	Incednine	HAAH	GHSTG	Mmal	Mmal[103]
lobA5_mod.1	Lobophorin	YASH	GHSQG	Mal	Mal[104, 105]
lobA5_Mod.3	Lobophorin	YASH	GHSQG	Mal	Mal[104, 105]
pieA1_Mod.0	Piericidin	LAAH	GQCLG	Mal	?
Notes on pieA1_Mod.0: Catalytic serine of site 2 has been replaced with a cysteine. Site 1 also has a less-canonical structure for mal, though retains the catalytic histidine. There is not a KS within the loading module to decarboxylate (KS_Q)[106] mal to make a carbon chain coherent with the final structure.					
simC1A_Mod.2	Simocyclinone	YASH	GHSQG	Mal	Mal[107]
simC1C_Mod.1	Simocyclinone	YASH	GHSQG	Mal	Mal[107]
Notes on simC1A_Mod.2 and simC1C_Mod.1: Evidence for mal being used in both modules is inferred from the final structure. The polyketide undergoes substantial tailoring (see figure 1, Trefzer et al, 2002[107]) so it is conceivable that Mmal could be used in either of these modules.					
tmnAII_Mod.1	Tetronomycin	YASH	GHSQG	Mal	Mal[108]

Figure 30: Analysis of catalytic motif oddities and inverse predictor identified in figure 29, above. Colour scheme as per figure 29.

Though this is a small subset of the total data, the results in figure 30 provide evidence that this method is generating meaningful predictions on the AT substrate type. The model can pick out when a small subset of false labels is being provided, which was the case for 11 of the domains mislabelled in the web-scraping (red) and likely 2 more (absB1_Mod.2 and fscD_Mod.3).

These results may be indicating a bias towards accurately identifying mislabelled Mmal domains – we do not see any false Mmal errors.

Validation with catalytic flips set AT-Inverted

To scrutinise the utility of the method 100x100 "photographs" method, and whether the trained models follow the literature for AT-substrate classification, a new dataset was generated by partitioning off 20 Mmal and 20 Mal AT domains from dataset AT#2. These were selected at random. The remaining data left in dataset AT#2 is now named dataset AT#3. The partition is named dataset AT#3_Inverted.

New .fasta files were created for the 40 Mal/Mmal partitioned, where the active site motifs YASH/ HAFH and GHSQG/GHS(I/V)Q were flipped in the files. The modified files were run through the structure preparation pipeline outlined in section 2.1.1*, and then stored with the original, unmodified structures in `AT#3_Inverted` with a suffix "`was_[substrate of wild type]_SNAKE`".

**On the topic of using AlphaFold for mutagenized protein structures, the use case for these modified AT domains is a grey area. The AlphaFold 2.0 paper[48] recommends that the strength of the predictions is primarily based on strength of the MSA, and so is reliant on input sequence being "natural proteins" and that unnatural single mutations may present with high accuracy but are not necessarily to be trusted. This is a grey area here, because the motif swaps we are demonstrably within the scope of a natural mutation for the active site[73-76], but we may well have missed other co-evolving areas of the structure that would strengthen AlphaFold's conviction. This in itself is interesting because it may throw out other areas of the AT-domain that interact with the active site beyond direct catalytic sites we modified, upon further exploration.*

Predictions on dataset AT-Inverted were made using a model trained in *01.02.02_cube_face_CNN.R* using dataset AT#3. The difference in prediction confidence scores for mal and mmal were calculated (figure 31).

Our hypothesis is that if the model has learned that the active site motifs are informative for accurate labelling, as the literature suggests, then the modified structures should predict with a higher percentage probability for the new active site motif type than the native structure.

Catalytic flip Results

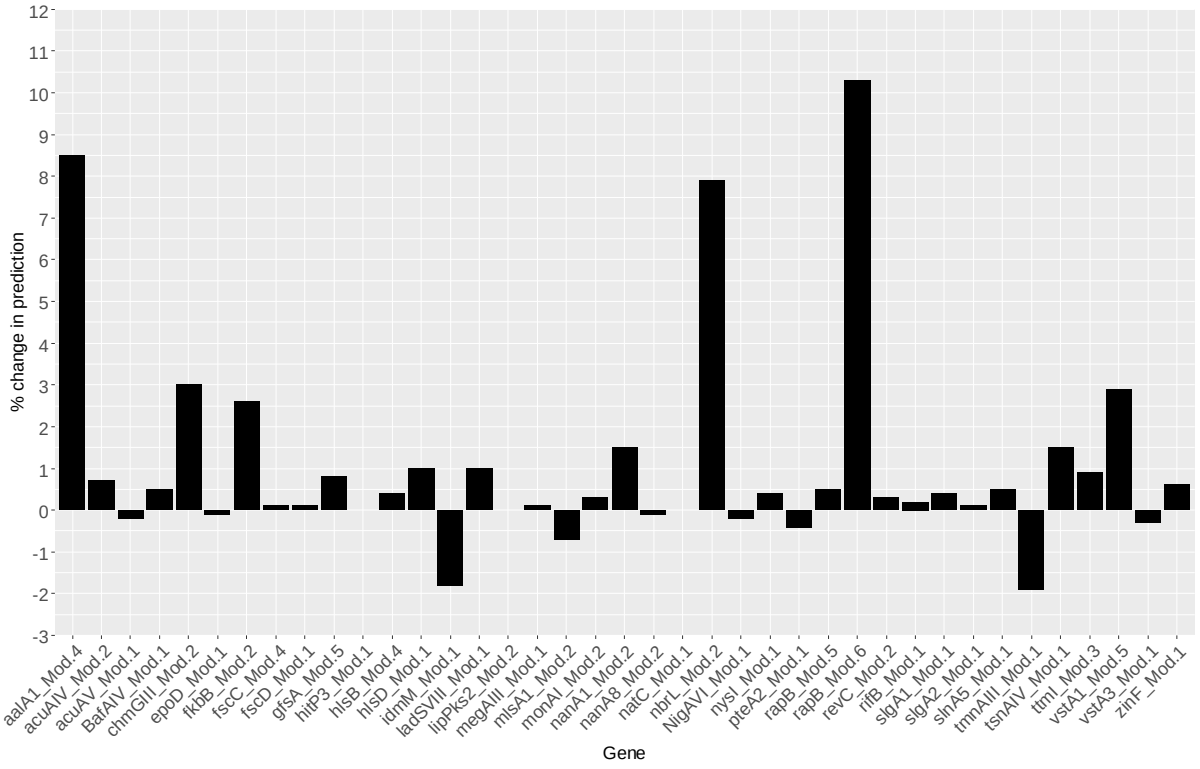


Figure 31: Percentage change in CNN predictions between wild type AT structures and catalytically inverted structures.

Averaged across genes, structures with an altered substrate specificity active site are 1.06% more likely to be assigned to the altered class type. For a two-tailed, paired T-test comparing the model's substrate prediction values before and after treatment (substrate preference mutation), we report a p-value of 0.012

Application to KSs: Can we predict the β-carbon reduction state of an incoming polyketide substrate from a KS structure?

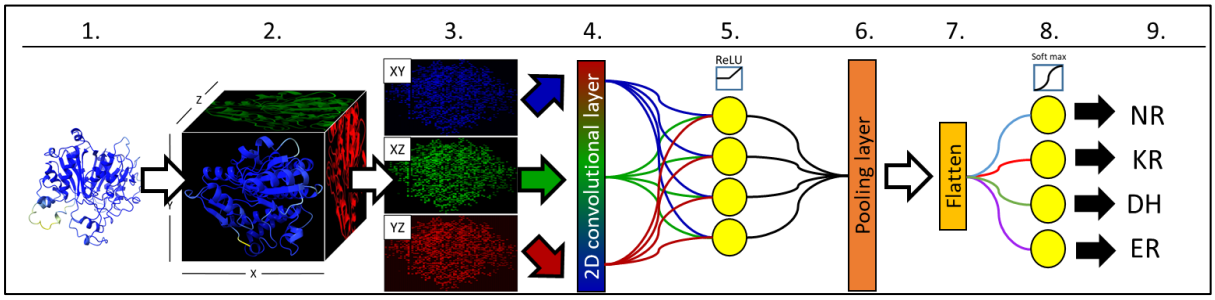


Figure 32: Data processing workflow and CNN architecture for script `01.02.02.01_KS_B-factor_cube_face_CNN.R`. **1-3:** AlphaFold structures are converted to 3x 2D representative pixel maps in XY, XZ and YZ planes, with depth in missing plane encoded as pixel intensity. **4-5.** A 3D convolutional layer is run over the 3 planar representations and passed through a 64x ReLU layer. **6-7.** Data is pooled and flattened. **8.** Data outputs to a binary neural layer with a sigmoid activation function. **9.** Output is rounded to a 4-way binary corresponding to a prediction for a β-carbon reduction state.

Results

Script: `01.02.02.01_KS_B-factor_cube_face_CNN.R`
Dataset: KS#1
Loss function: Categorical cross entropy
Optimiser: Adam (Adaptive Momentum estimation)[96]
Epochs: 60
Batch_size: 35
Validation_split: 0.2
Seed number: 2

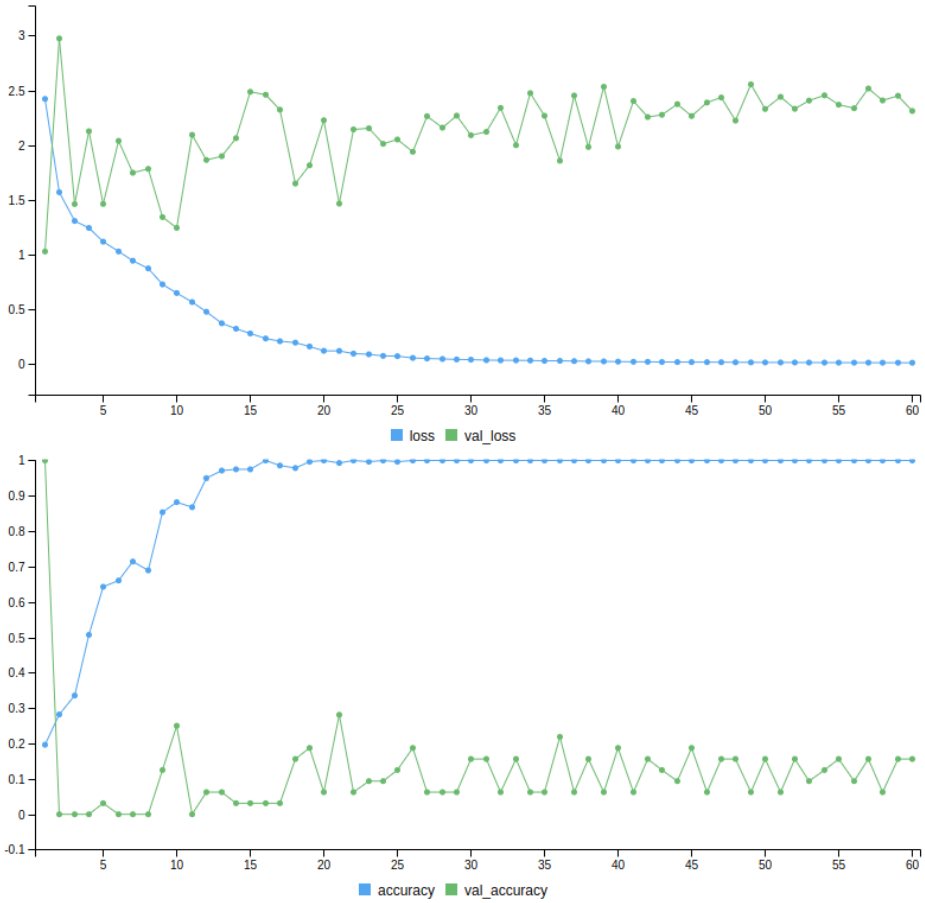


Figure 34: Loss (top) and accuracy (bottom) over a 60-epoch period for training data set (blue) and validation set (green).

Figure 34 displays a model that has learnt to recognise patterns in the training data that are not applicable to the validation data, and displays a rather extreme example of overfitting. The trained model has therefore not learnt meaningful representations of the data categories.

Conclusions and discussion on approach 1:
Voxel-based methods failed to learn meaningful patterns in the data.

The cube photographs method was able to accurately distinguish between substrate types in AT domains. Further, when a small number of mislabelled AT domains were accidentally included in the training data, they were correctly predicted by the model, further indicating that the model was learning meaningful representations of the structures.

However, the inability to identify that the active site had changed was a somewhat of a concern. Lastly, we find that the method does not train models that can make accurate predictions on the ketosynthase β-reduction state problem.

2.2.3 Machine learning approach 2: Graph neural networks

Introduction to network graphs

Converting protein structures to voxels is one way to present protein structural data for machine learning. Another approach is to model encode the structures using network graphs.

Network graphs represent and describe interactions between objects[109]. This field of mathematics is used in wide range of applications, from modelling social groups[110] to mapping optimal paths through a city[111], and in our case, the arrangement of amino acids in 3D space.

In a network graph, objects are represented as nodes. Using the above examples, these could be people in a social network, locations in a city, and atoms or residues in a polypeptide. Nodes can have numerically encoded features associated with them, such the atom's element or the amino acid type for a residue, which can act as features for machine learning.

Interactions between objects are represented by connections between nodes, referred to as edges. Like nodes, edges can associate with features that describe the connection, be this a physical distance or the nature of the interaction, such as whether a chemical bond is covalent, ionic etc. As with nodes, edges can also be encoded with features for use as input data for machine learning.

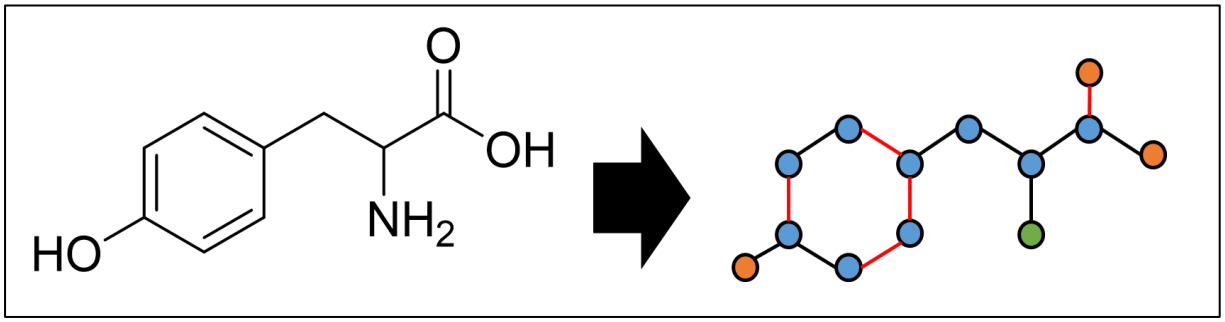


Figure 35: Representation of the amino acid tyrosine's atomic structure as a graph network (ignoring hydrogens). Node features have arbitrarily been represented by colour, as have the bond types.

Graph-based machine learning can be used to perform one of three operations: the prediction of nodes, the prediction of edges, or the classification of entire graphs (or sub-graphs within). In this approach, we are associating network graphs drawn from AlphaFold[48] structures, with graph-level labels.

An advantage of graph networks is that, through certain architectures, they can be orientation invariant and so the structure alignment is not an issue here. Compared to our first approach using a model in 3D space, graph networks do not require memory for void spaces between nodes – all that is encoded is the nodes and their partnering edges*. This leads to memory efficient representations.

For a (relatively) light, mathematics-orientated introduction to graph networks, we recommend these reviews: Pearce et al, 2021 Refs.[112, 113] and Daigavane et al, 2021 Ref.[113].

**There are several methods to encode graph connectivity. We use PyTorch-geometric for the machine learning, which store edge indexes in a sparse coordinate tensor format. This takes a `[2, num_edges]` format.*

Graphs and non-Euclidian space

Central to the utility of graph networks is their occupation of non-Euclidian space. This area of geometry was codified by Euclid (not to be confused with Euler, who is widely attributed as the founder of graph theory). Assuming that you have not ingested psychedelic mushrooms recently, Euclidian geometry is

almost certainly what you imagine geometry to be and is grounded in our three-dimensional existence*, forming a branch of the highly controversial field of Common Sense**. Euclid gives this field his name by virtue of laying down a set of postulates that codifies how Euclidian geometry operates***.

Non-Euclidian space does not operate by conventional rules of Euclidian geometry with respect to parallel lines, referred to as Euclid's 5th postulate****. In the 5th postulate, parallel lines cannot ever meet; however, in non-Euclidian geometry, they can meet an infinite number of times and from our inherently Euclidian perspective, straight lines are not *visibly* straight anymore but instead curved (this is particularly relevant to manifold learning, a technique not used here).

In application, removing the 5th postulate is useful because it allows us to ignore spatial dimensions when we draw non-Euclidian data architectures such as graph networks, and is suited for data that does not inherently possess a unifying set of global coordinates, vector space or parametrization[114]. Is this valuable for modelling protein structures? Not necessarily, given that atomic/residue nodes will be representative of objects confined to a 3D space, and so are inherently Euclidian. However, when we consider drawing edges that do not exist as chemical bonds, in a manner that ignores Cartesian coordinates, we start to see why this geometry is useful. A good example would be linking transient interactions within a protein, such as during a conformational change in ligand binding proteins, or amino acid R groups that can rotate around the alpha carbon. Removal of the 5th postulate is most useful for drawing edges because it means that they describe a relationship between two objects within a set of objects, and that an edge can be independent of other edges. Therefore, the significance of information encoded in edges is not intrinsically linked to the properties of other edges.

In contrast to the non-Euclidian graph, Euclidian data structures are typically represented as an n-dimensional grid where data exists at the intersections of perpendiculars in the grid (i.e., a table or data frame). Returning to the convolution method used in the previous section (see figure 23), if we imagine that two pixels on opposite sides of the photo are intrinsically linked (maybe the protein has a dynamic component), how would the spatially limited kernel capture this relationship? The options here are to warp or distort intervening data*****, which would exaggerate and distort data elsewhere in the grid, or use a much larger kernel, which would remove the locality of the convolution. This is a reason why using machine learning on Euclidian data structures with longitudinal dependencies within the data can prove challenging. Removing the 5th postulate can help to get around this.

**Over short, human observable distances. Einstein's theory of relativity throws a spanner at Euclidian geometry over large distances. The truth is, Euclidian space is the best approximation our limited primate brains can deal with.*

The introduction of non-Euclidian geometry to the mathematical canon is a proverbial "rabbit hole" and may, in part, be some of the inspiration for this phrase. Lewis Carroll, professor of mathematics and author of the famously psychedelic "Alice in Wonderland" and sequel "Through the Looking Glass" uses these novels to poke fun at non-Euclidian geometry through various elements of the plot[115]. Nobody got the jokes (shocker) so he went on to write the satirical play "Euclid and His Modern Rivals", 1879 Ref [116], where the ghost of Euclid defends his postulates in a courtroom in hell. The play was meant to serve as a defence of perceived logic and by extension, the existence of god[115]. Apparently, questioning what it means to be straight can upset the religious community.

***First codified by Roman philosopher Commonus Sensicus the Unpopular, shortly before being torn apart by an angry mob.*

**** Must have been nice publishing as an ancient Greek.*

*****There are subsets of this: hyperbolic and spherical. These differ in whether space is positively or negatively curved and have different applications. Spherical geometries are used in navigation and astronomy, for example. Did you know it is possible to sail in a straight line from the UK to New Zealand? That's some non-Euclidian, spherical geometry right there.*

******Cue Event Horizon paper and pencil wormhole.*

Machine learning on graph structures

Graph neural network (GNN) is an umbrella term for neural networks that operate on graph network data structures. Some networks that fall under this umbrella include Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), Graph Sample and Aggregated Embeddings (GraphSAGE)[117] and Graph Isomorphism networks (GINs). Figure 36 below gives an overview of how these network types relate to each other.

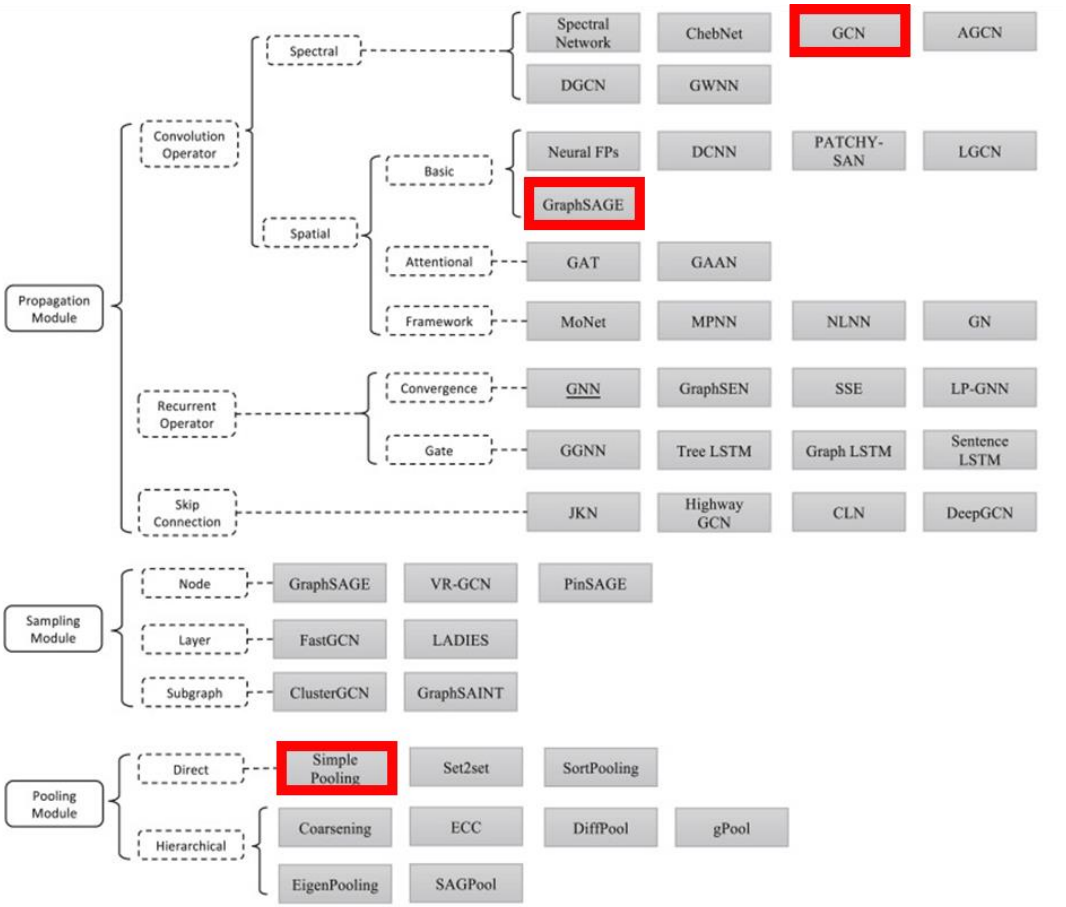


Figure 36: Overview of network graph machine learning techniques, reproduced from Ref.[109]. Methods used in this chapter are highlighted red. Propagation modules are used as a means of communicating feature data across nodes with respect to network topology. Sampling modules create a means to accelerate/facilitate learning of large graphs over multiple layers by breaking the task down into samples. Pooling modules condense data, which is particularly needed when moving from dense data to categorical outputs used in classification tasks[109].

In this chapter, we use GCNs and GraphSAGE, which use convolutional operators in their propagation modules. We also make use of simple pooling layers. We explore these methods in detail below.

In propagation modules, convolutional operators act via a message passing system, where each layer of a GNN updates a representation of nodes by an additional k-hop of separation. So, for any arbitrarily chosen node in a graph:

- In layer 0, the representation is for the node only, without connecting node. Information contained here will only include the target node and its assigned features.
- Layer 1 will incorporate first-degree neighbours that connect to the arbitrary node. Information at the target node now also describes its immediate neighbourhood

- Layer 2 will incorporate second-degree neighbours, those connected to the first-degree neighbours. Information on the target node now describes a neighbour(s) and the neighbour's neighbours.

The objective of message passing is to aggregate information from neighbouring nodes to update nodal data representations. This information can be presented to a convolutional graph network as spectral or spatial, as described next.

Spectral representations focus on global graph structures and operate by leveraging eigenvectors and eigenvalues* associated with the graph, such as the adjacency or Laplacian** matrices. When derived from the Laplacian, Eigenvalues represent the frequencies at which patterns oscillate in the graph, and eigenvectors correspond to the spatial distribution of these patterns. Convolutions of the Laplacian matrix therefore capture both local and global information about the graph, thereby enabling longitudinal dependencies in data to be brought together in a form detectable by a neural architecture. Somewhat unhelpfully, GCN typically refers to a spectral convolution, even though convolutional operators can be applied to spatial data, as in a GraphSAGE.

We use the Kipf and Welling GCN algorithm[118] via the PyTorch geometric API[119], which leverages spectral data in a GCN (when referring to GCNs in this text , we are referring to the Kipf and Welling type). We discussed section 2.2.4 how convolutions operate over matrices in approach 1, and the principle for operation on the Laplacian is the same.

Spatial representations operate directly on the graph structure, by examining the topological relationships of nodes in the graph. Spatial methods are therefore well equipped for capturing local dependencies in the data. Graph Attention Networks (GATs) use spatial representations[120].

** Eigenvectors are vectors used to retain relative directionality when performing linear transformations on data, where the magnitude is scaled by eigenvalues. Eigenvalues are scalars that are used to transform Eigenvectors. In effect, this allows for rotation and compression of data without losing information about the spatial relationships between data. If you are familiar with principal component analysis, this is how dimensionality is managed to produce a 2D/3D graphical representation of higher dimensional data.*

*** The Laplacian is a square matrix associated with a graph, which provides information on the structure and connections in the graph.*

Graph Attention Neural Networks (GATs)

GATs assign attention weights to each k-degree node, relative to a target node, in the k^{th} layer (so in the second neural layer, weights are being assigned to second degree neighbours). These weights are applied via an attention mechanism, which effectively builds a description of how important edges are, allowing the neural network to focus on what is important and ignore what is not[121]. These will be associated with an activation function (ReLU, Sigmoid etc.), which introduces non-linearity to the neural elements (non-linearity allows for signal amplification/silencing). GATs are permutation equivariant[120], meaning that the same result is achieved regardless of the input order of nodes and edges to the algorithm. This is particularly useful for a protein graph because it means that the structures do not have to be aligned and further, nodes and edges are not required to be sequentially arranged or evenly balanced across graphs in a training set.

In a GAT, each node has a feature vector, which the attention mechanism uses to compute attention coefficients between the target node and its neighbours. Attention coefficients are used to create a weighted sum of the neighbour node features, which are combined with the target node's feature, updating its representation. Representations are then operated on by neural layers.

GATs can have multiple attention heads, where each attention head independently calculates attention weights and perform feature aggregations in a local area. Having multiple attention heads provide GATs

with the ability to expand into global information by allowing their outputs to be averaged or concatenated.

Based on these properties, one might expect a GAT to perform well on protein structure tasks where one or more discrete motif/amino acid network is specific to a classification.

Explaining graph neural networks with GNNExplainer

Once a model is trained and reports as making reliable predictions, we would like to be able to interpret which features of the protein structure the neural is using to make its predictions. An insight into what features (in this case, residues) the model uses for making its predictions may be translatable into a laboratory context. For example, a residue that is frequently detected in one label class and not another class would be a sensible target for mutagenesis.

GNNExplainer[122] is a model-agnostic approach to providing human-interpretable explanations of graph neural network models, through the PyG API[119]. This falls within the category of explainable artificial intelligence (xAI)*.

GNNExplainer presents as an optimisation task that maximises the mutual information between a GNN's prediction and the distribution of possible subgraphs within an input graph. The inputs are the GNN model and an input graph with corresponding labels. The output is a subgraph of the input nodes that are most influential to the model's assignment of the prediction. This is carried out as a separate machine learning task that operates on the model and input graph, effectively learning which nodes are relevant and not relevant to the label.

Applied to the substrate prediction task at hand, we input a single input AlphaFold structure and are returned a map of residues that are influential the substrate.

Our method for explaining our trained models' predictions is to run GNNExplainer on all structures within the test partition following training. The output of the explainer is a .pdf file (figure 37, below) displaying the subgraph structure, per structure.

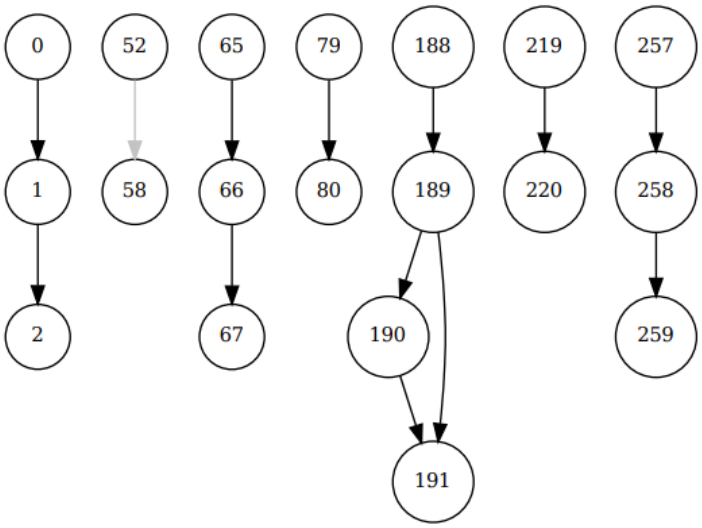


Figure 37: ChmGI_Mod.1 (AT domain) explainer output subgraph. Note nodes 52 and 58 have high transparency arrows. Note that node numbers presented here are 1 less than the residue position, due to Python indexing 0 as the first number in any list.

These .pdf subgraphs are manually viewed and subgraph nodes are tallied into a .csv**. Node values are recorded +1 to the .pdf value to account for Python indexing. The csv is then run through script `01.01.08_AT_pairwise_alignment_B_and_NetX.R` or `01.01.09_KS_pairwise_alignment_B_and_NetX.R`. This converts the coordinates from a dimeric input into a monomeric format (if required) and maps the node numbers to the corresponding `.pdb` structure.

- Nodes are recorded in the B-value column of the `.pdb` file, replacing the AlphaFold pLDDT[48] scores for the file.
- Scores are assigned based on the subgraph connection opacity. Dark arrows are given a score of 100, high transparency arrows a score of 50.

The script proceeds to run an alignment of the protein sequence in each `.pdb` against a reference sequence, using a pairwise alignment algorithm from the R package MSA[123]. Explainer scores across the aligned test set are averaged per residue in the reference sequence, effectively creating a heat map scoring in the B-score for frequency of residue occurrence in the explainer output. Finally, the heat-mapped structure is saved as a `.pdb` file. This structure can be used as an input file to make a "word search" heat map figure, such as figure 39 below, using script `01.01.10_Crossword_graphs.R`.

This approach creates a problem, however, which is that nodes that do appear in the explainer, but do not align to the reference, are lost. For instance, if we have a variable, 5-6 residue motif that is frequently occurring in the explainer results and we use a 5-residue variant as the template, we are going to lose information about the missing 6th residue. We propose two solutions to this problem:

1. Aligned heat maps to a reference structure for exemplified mapping. This gives a sequence-specific alignment that can be represented in 3D, however it comes with the caveat that unaligned residues of significance may be lost.
2. Alignment agnostic, binned histograms to identify approximate areas of high node predictive frequency.

**Explainable AI (xAI) is currently a hot topic in the field of artificial intelligence. Neural networks and deep learning are capable of identifying patterns in large, complex datasets. Understanding which underlying features a network uses to make predictions is valuable to data scientists for a range of task-specific reasons; from safety concerns (e.g. a driverless car) to gaining insight into the fundamental patterns within a dataset. The problem with neural networks is that the flow of data through them is directional as a result of the degree of neural connections and the presence of non-linear activation functions and thresholds. In effect, we can input features to return labels, but we cannot input labels to return features. This has led to neural networks being described as a "black box" where the neurones form convoluted interactions that are not readily interpretable. That said, we do not necessarily need to understand the minutia of these interactions to understand which input features are significant to an output. After all, we have control of the input data and a trained model is a static mathematical transformer, so we can perform incremental changes to an input to learn what changes the outcome. Through this experimentation, we can extract explanations of isolated examples.*

***There may well have been a way to do this digitally. I looked but gave up.*

Graph convolutional networks for AT-domain substrate specificity

Script: `01.03.01_AT_GCN_binary.ipynb`
Packages: OS, torch, Graphein, pandas, re, functools, sklearn, torch_geometric, matplotlib, numpy,
Python version: 3.9, run as Jupyter notebook.
OS: Ubuntu 20.04.6 LTS
Optimiser: Adam[96]
Loss function: Cross entropy (PyTorch)

AT structures are pulled from a file directory and randomly sampled to produce a balanced input of Mal and Mmal domains. This dataset is then run through a graph generation tool, Graphein[124], to convert the `.pdb` data into graph format.

For graph construction, we opted to run a simplified model that only includes the alpha carbons in the structure as nodes. Edges were created from both the primary protein sequence and using a K-nearest neighbour method, which draws edges between nodes in physical proximity in the structure. Node features are defined as the amino acid type. Edges are not assigned features.

Script overview:

1. Import AT domains from rank 1 files and combine with substrate labels data. Labels were produced from script `01.02.02.01_AT_cube_face_CNN.R` and saved as a .csv file, which was imported to this script.
2. Generate graph networks using alpha carbons, drawing edges using a k-nearest neighbour method in Graphein version 1.4 [124].
    - K = 3 (joins to 3 nearest), long interaction threshold = 0.
        - This draws the primary structure plus the next nearest residue.
    - Node features are implemented as a one-hot encoding (OHE) of amino acid type.
    - Labels and structural data are unified in the graph constructor.
    - Graph data is converted from networkX to PyG format.
3. Graph data is imported to PyTorch Geometric using a data loader. Data is balanced for even Mmal:Mal, batched and partitioned to training, test and validation data, using an 80:10:10 split.
4. Training and validation data is run through a loop that trains the model. Network architecture is in figure 37. The trained model is then used to generate a final accuracy and loss score against the test data partition. Optimiser algorithm and loss function used were ADAM and cross entropy.
5. A final code block in the script loops the test data partition through GNN explainer, using a 10,000 epoch setting. Output `.pdf` files are manually curated per method above.

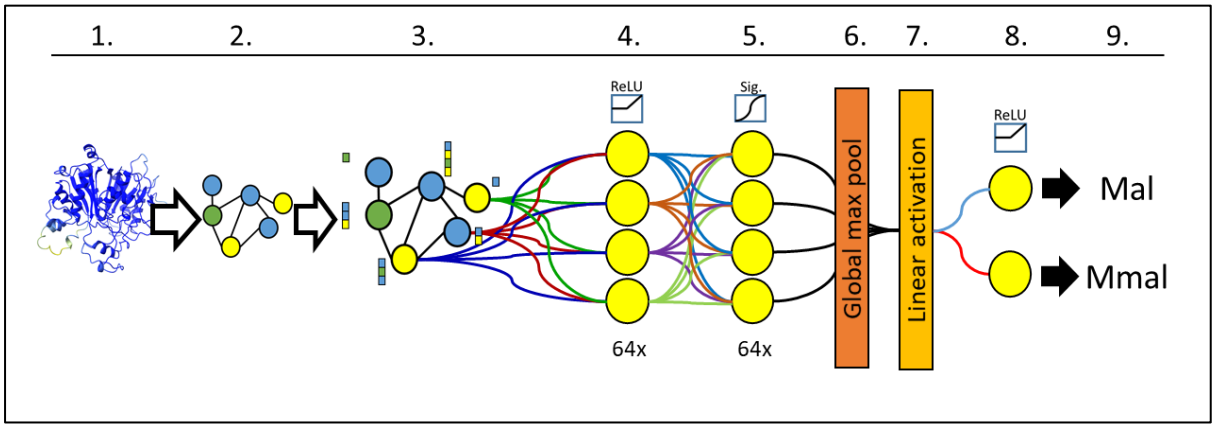


Figure 37: Process map and network architecture for script `01.03.01_AT_GCN_binary.ipynb`. AlphaFold structures are converted into network graphs (1-2) and run through of two layer GCN (3-5). Outputs are then pooled (6) and run through a linear activation (7-8) to give a prediction of Mmal or Mal.

Results

Input data:

Dataset: AT#3

Post-balancing:

Training Set Composition:
Mal: 245 samples, Mmal: 245 samples, Total: 490 samples
Mal Ratio: 50%, Mmal Ratio: 50%

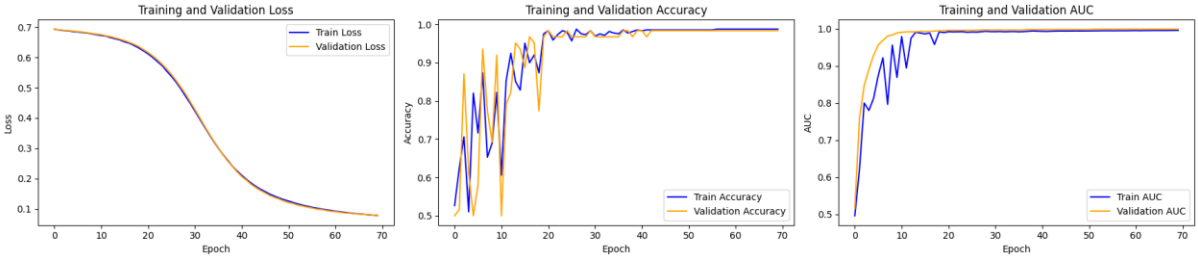
Validation Set Composition:
Mal: 31 samples, Mmal: 31 samples, Total: 62 samples
Mal Ratio: 50%, Mmal Ratio: 50%

Test Set Composition:
Mal: 31 samples, Mmal: 31 samples, Total: 62 samples
Mal Ratio: 50%, Mmal Ratio: 50%

Hyperparameters:

Batch size: 64
Seed: 15706112877486103401
Epochs: 70
Learning rate: 0.001
Neural layer density: 64

Training:



Test Accuracy: 0.9839 (98.39%)
Test AUC: 0.9844(98.44%)
Figure 38: Loss and accuracy over 70 epochs.

Training finds relatively consistent loss and accuracy between training and validation data. This suggests that the model is being learning meaningful representations. Loss plateaus around 0.1, which is a good score. This corresponds with a very high accuracy score.

xAI results

Malonyl-CoA xAI average scores

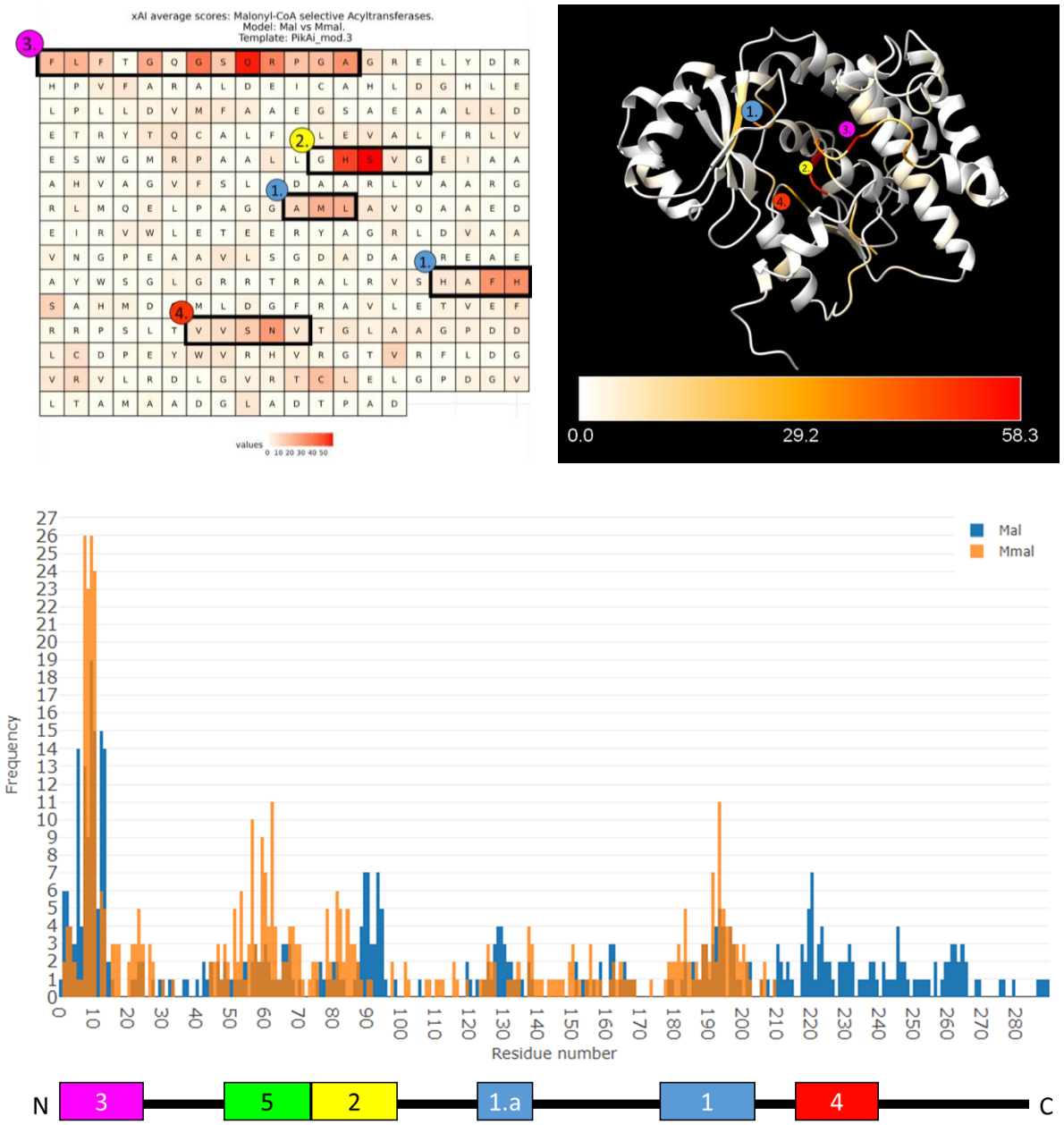


Figure 39: **Left:** Heat map of average xAI results (n =31) for Malonyl-CoA specific AT domains in the test partition, averaged to alignment for PikAI_mod.3 (a native Malonyl-CoA AT domain). Raster pattern follows left to right, top to bottom, N terminal to C terminal. Heat map is red hot relative to maximal scores in averages specific to this heat map – colours not comparable between graphs. Regions 1-4 have been highlighted to correspond with figure 40 below (reproduced in right panel). Note that regions 1 and 2 correspond to the YASH and GHSQG active site motifs. **Right:** Explainer averages for Malonyl-CoA, aligned to PikAI_Mod.3. Scale bar indicates average score (max achievable of 100). **Bottom:** Alignment agnostic histogram display node frequencies detected by xAI. Below is an approximate map of nodes to regional diagrams. Note this is an approximation because the nodes are not aligned – any insertion/deletions in the sequences will move residues out of relative frame.

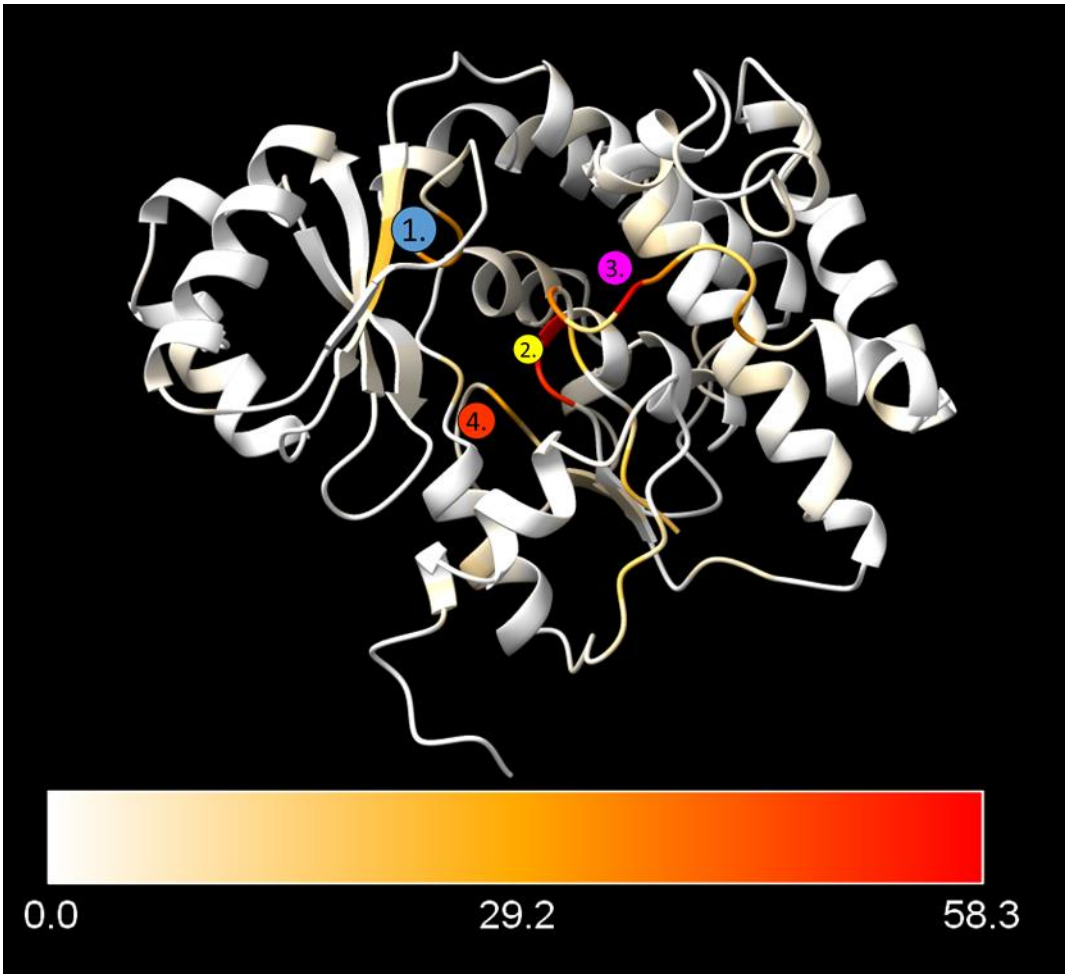


Figure 40: Explainer averages for Malonyl-CoA, aligned to PikAI_Mod.3. Scale bar indicates average score (max achievable of 100).

The averaged xAI scores identify regions within and proximal to the active sites as significant in the trained model's predictions for Malonyl-CoA. Notably, the substrate predictive GHSVG and HAFH domains are being detected (this is expected), as is a loop proximal to the active site.

Methyl-malonyl-CoA xAI average scores

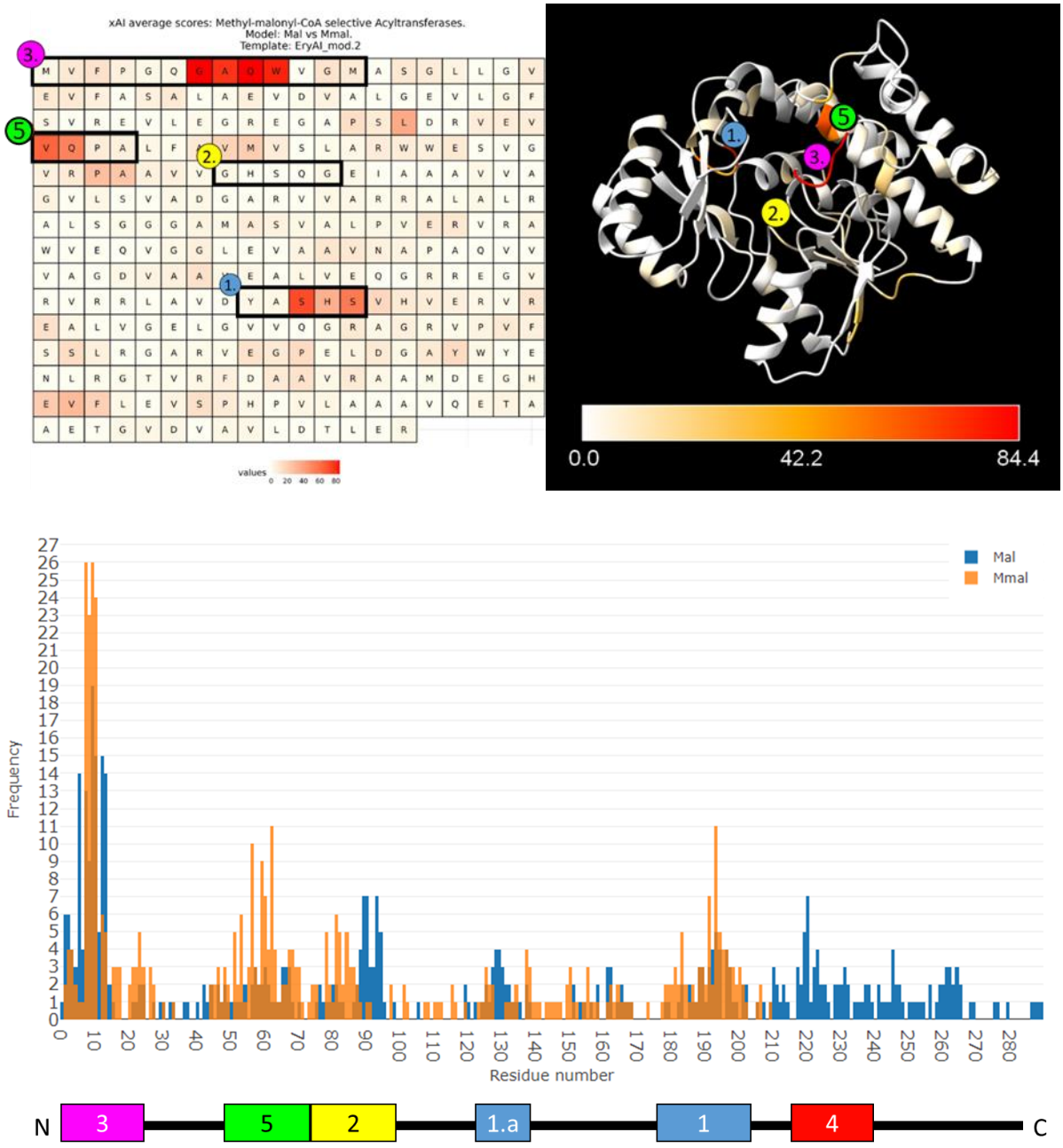


Figure 41 (left) and figure 42 (right): **Left:** Heat map of average xAI results (n =31) for Methyl-malonyl-CoA specific AT domains in the test partition, averaged to alignment for AeEryAI_Mod.2 (a native Methyl-malonyl-CoA AT domain). Raster pattern follows left to right, top to bottom, N terminal to C terminal. Heat map is red hot relative to maximal scores in averages – colours not comparable between graphs. Regions 1-4 have been highlighted to correspond with figure 42 below (reproduced in right panel). Note that regions 1 and 2 correspond to the YASH and GHSQG active site motifs. **Right:** Explainer averages for Methyl-malonyl-CoA, alignment to AeEryAI_Mod.2. **Bottom:** Alignment agnostic histogram display node frequencies detected by xAI. Below is an approximate map of nodes to regional diagrams. Note this is an approximation because the nodes are not aligned – any insertion/deletions in the sequences will move residues out of relative frame.

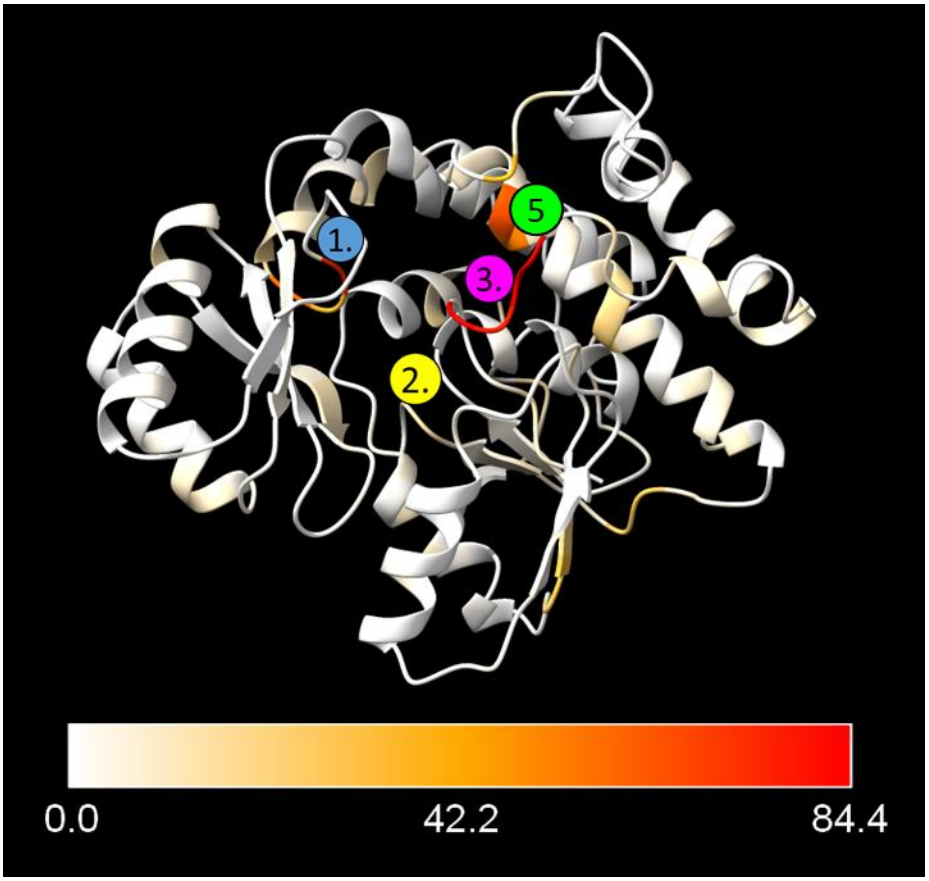


Figure 42: Explainer averages for Methyl-malonyl-CoA, alignment to AeEryAI_Mod.2.

The averaged xAI scores identifies the YASH motif but does not present the GHSQG motif as being significant for the prediction. As with the malonyl-CoA graphs, a loop proximal to the active site is detected in region 3.

Results on AT-inverted dataset

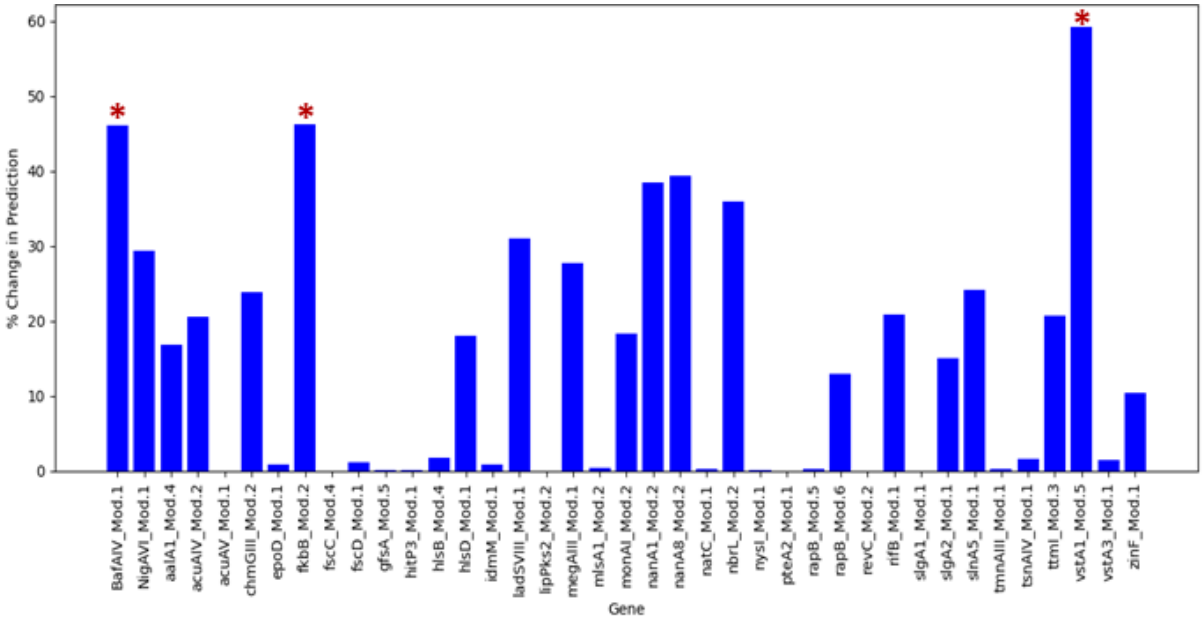


Figure 43: Difference in prediction scores for AT domain with modified catalytic motifs dataset AT-inverted and wild-type counterparts. AT domains with a great enough difference to change the model's categorical designation for the structures are marked with an asterisk.

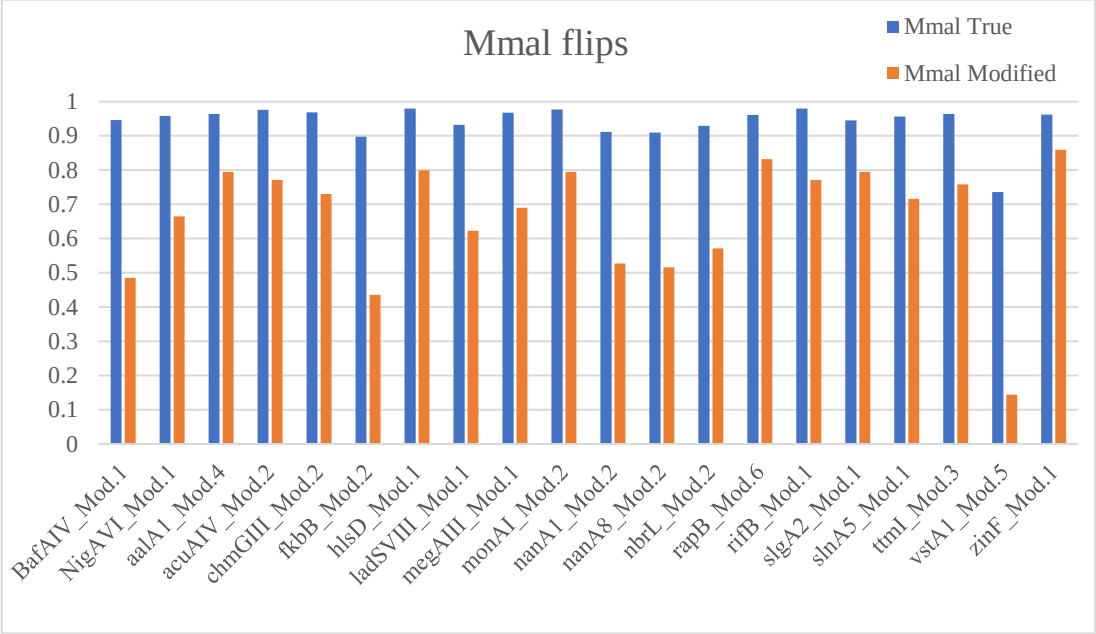


Figure 44: Comparison of model's confidence in a prediction of Mmal for catalytically native Mmal AT structures (blue, wild-type) and catalytically modified variants (orange, HAFH→YASH and GHS(I/V)G→GHSQG). Average shift in prediction across the set is 33.6%.

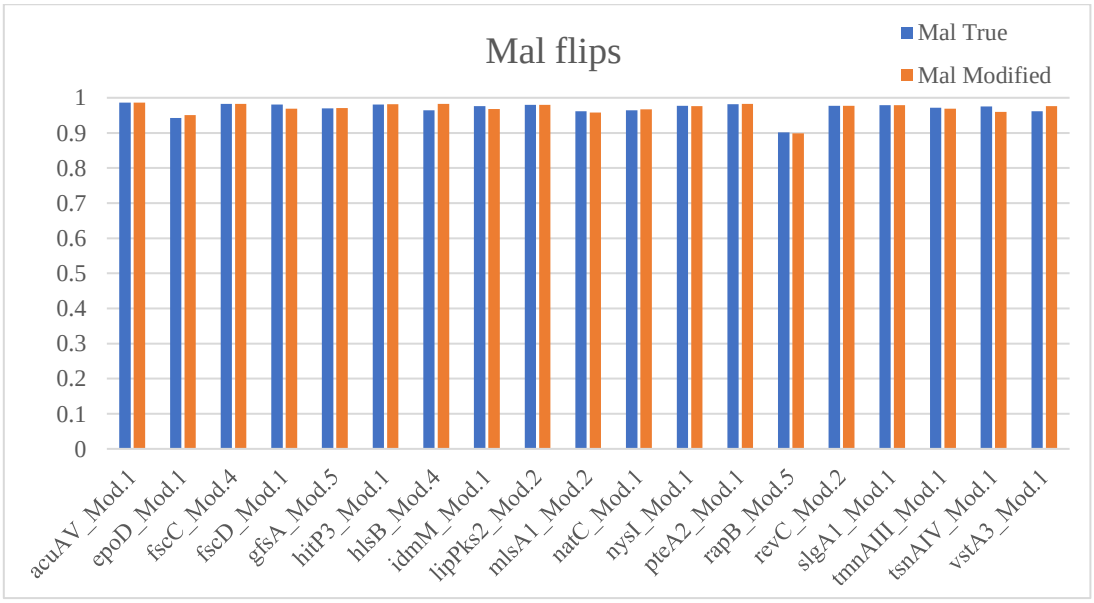


Figure 45: Comparison of model's confidence in a prediction of Mal for catalytically native Mal AT structures (blue, wild-type) and catalytically modified variants (orange, YASH→HAFH and GHSQG→GHSVG).

Figures 43-45 show that the catalytic motif swaps were detected in Mmal→Mal, but not in Mal→Mmal. From the word search figures 39 and 41, we can see the following:

- GHSVG is indicative for Mal, whilst GHSQG is not indicative for Mmal. As a binary classifier, the model only needs to detect one to be accurate (e.g. if GHSVG is present, make a prediction of Mal), so this does line up with our expectations for the active site.
- Mal predictions based on the HAFH motif may be being learnt in conjunction with a proximal AML equivalent motif on a neighbouring a-helix, though this is not clear from the NetworkX subgraphs.
- HAFH-AML is less significant an indicator than YASH, as denoted by a lower average score in the heat maps.
- Region 3 on both diagrams shows a loop at the N-terminal that joins to the KS-AT ferredoxin-like linker. This is a stronger predictor in both word searches for Mal and Mmal.
- Two other, separate loci, marked as region 4 are also detected proximal to that active site pocket that are indicated by the explainer averages as predictive for substrate preference. .

Region 3

Region 3 is a loop that connects the N-terminus of the AT domain to the KS-AT ferredoxin-like linker. Following the sequence into this linker, a β-strand threads through the AT structure, with the strand quite clearly forming part of the AT structure. In hindsight, we maybe should have included this as part of the definition for the AT domain structures borders, rather than the InterPro derived definition.

The region 3 loop exists on the opposite side of the active site pocket to the KS-AT dimer cleft (Figure 47, below), and away from the ACP docking helices (Figure 46, below). Given the proximity to the active site, we speculate that perhaps this loop might have a role in filtering which activated carbon-chain substrate that the AT selects, or in the catalysis itself. Previous mutagenesis studies were found to be incomplete for converting Mmal selective domains to Mal selective domains[125]. We would suggest that region 3 is an interesting candidate for a follow up mutagenesis study.

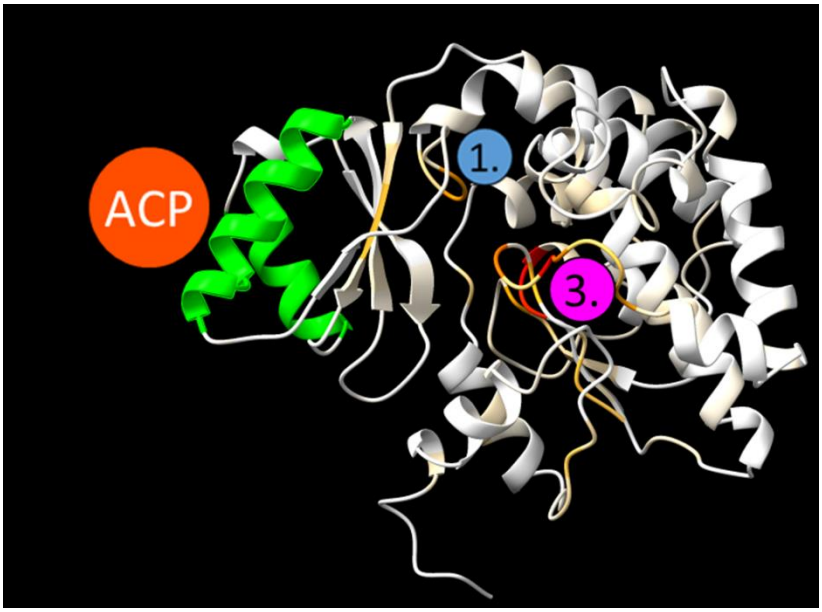


Figure 46: PikAI_Mod.3 with ACP interacting helices coloured green. The activated carbon substrate binds at site 1 (HAFH, in this case). ACP binding helices are based on cryo EM studies of PikAIII

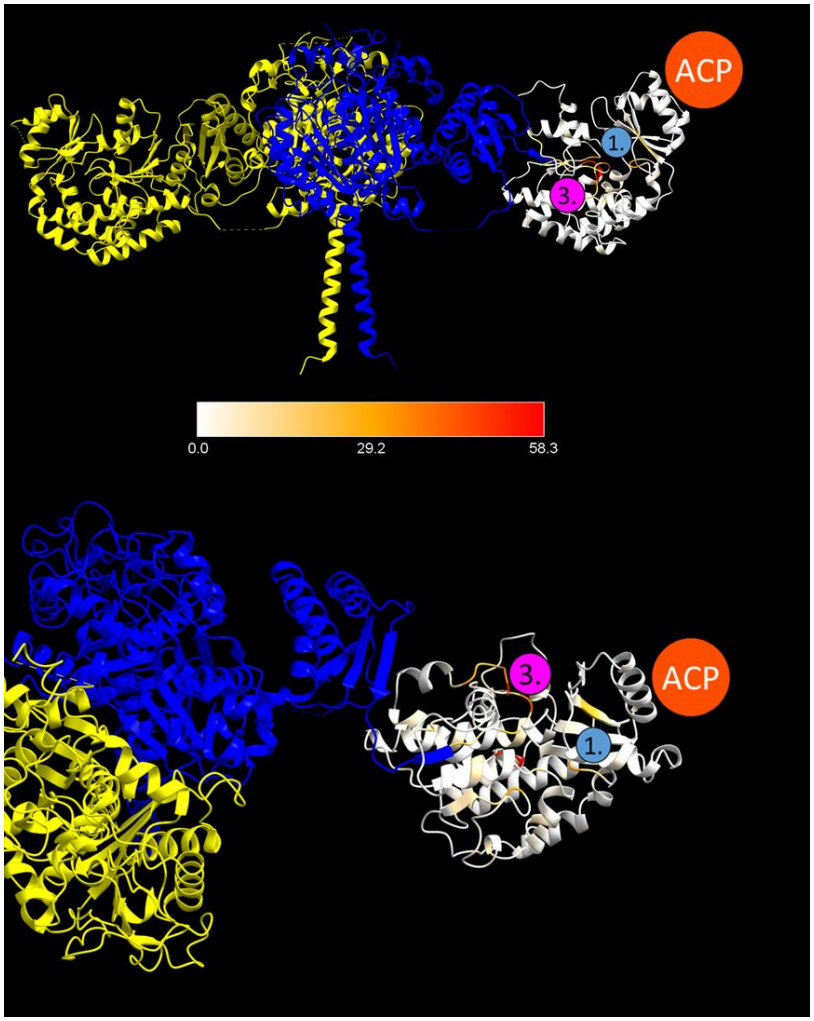


Figure 47: PikAI_Mod.3 aligned to Bioassembly for AeEryAI_Mod.2 KS-AT dimer (PDB: 8EE0[36]). Chains A and B coloured blue and yellow. Corresponding AT domain has been hidden from blue chain to highlight PikAI_Mod.3. Bottom structure view is top-down relative to the top view. Added are sites 1 and 2 from figure 46 above, corresponding to the active site (1) and the active site guarding loop (3).

Discussion of pilot study on AT domains

GCNs are able to produce comparable results to the voxel method, in terms of accuracy and loss (figures 28 and 38). When tested on the set of catalytic site-inverted AlphaFold structures in dataset `AT#3_Inverted`, the graph model was more sensitive to these mutations in the Mmal set. Experimental data for these active sites mutations indicates that we should expect a change in substrate preference[125], illustrating that the graph method is a comparatively better model for the Mmal/Mal problem.

Using GNNExplainer we produce averaged explanations that can be used to create actionable heat maps of residues significant to the models predictions. These in turn may provide actionable sites for mutagenesis to alter substrate specificity.

As noted in the heat maps and associated structures, we find a novel N-terminal site proximal to the AT domain's active site cavity, which may have a role in substrate selectivity. From the somewhat limited literature on AT domain mutagenesis[27, 73, 74], we find that modification to YASH/HAFH and GHSQG/GHS[I/V]G motifs do not fully alter substrate selectivity, which gives scope for other

residues/motifs in AT domains adding to substrate selection. We propose that the tertiary site identified in this chapter would make for an interesting follow-up to these earlier mutagenesis studies, both to validate the *in vivo* utility of using graph networks in protein mutagenesis space, and to gain greater insight into AT domain catalysis.

Graph-level learning for β-carbon reduction state prediction using downstream KS-domains
Script: `01.03.02_KS_4-way_GCN.ipynb`
Packages: OS, torch, graphein, pandas, re, functools, sklearn, torch_geometric, matplotlib, numpy,
Python version: 3.9, run as Jupyter notebook.
OS: Ubuntu 20.04.6 LTS
Optimiser: Adam[96]
Loss function: Cross entropy loss

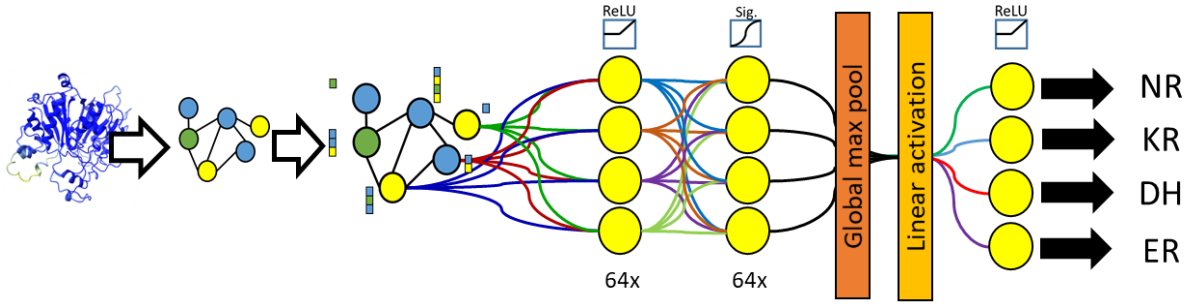


Figure 48: Process map and network architecture for script `01.03.02_KS_4-way_GCN.ipynb`. AlphaFold structures are converted into network graphs (1-2) and run through of two layer GCN (3-5). Outputs are then pooled (6) and run through a linear activation (7-8) to give a prediction of NR, KR, DH or ER.

This script trains a neural network to predict the β-carbon reduction state of a polyketide chain being passed to a KS domain, and builds from the AT domain prediction pilot script `01.03.01_AT_GCN_binary.ipynb`. The process and network architecture are predominantly the same, by design, with the following differences:

- We have increased the number of label categories from 2 to 4. Categories are based on the domain composition of the traditionally defined module immediately upstream of the ketosynthase (derived from ClusterCAD annotations). The criteria for categorisation are as follows:
    - NR: no reductive enzymes detected;
    - KR: active KR domain only;
    - DH: active KR and DH domains;
    - ER: active KR, DH and ER domains.
    - Anything that falls outside of this (e.g. module with a DH but no KR) is removed for simplicity. We do not take account of KR type or epimerase activity for this script.
- We use a different loss function to account for the change in the number of categories.

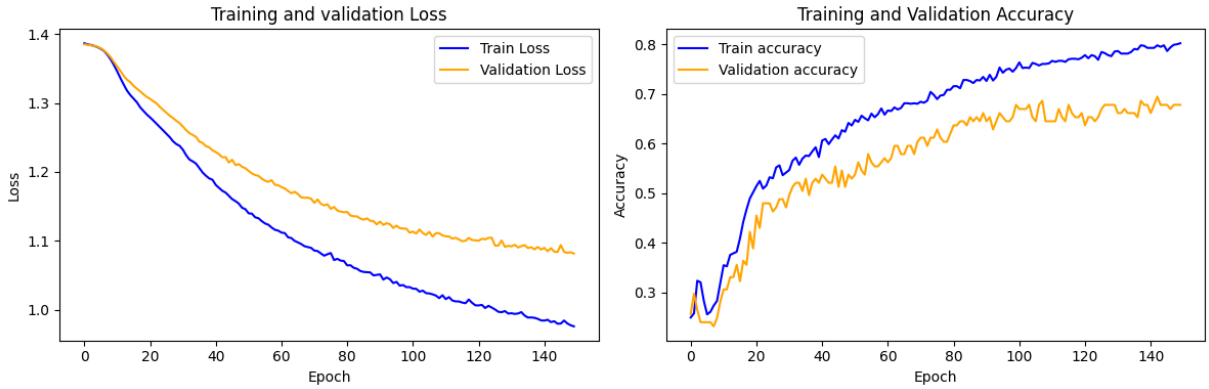
As outlined in figure 12 in section 2.1.1, the composition of reductive domains is not evenly distributed and has ~3x difference between the highest and lowest reduction state categories. This means that to produce a balanced, non-replicative training set we need to drop a substantial amount of data, which is not ideal, or pad the data with replicates, also not ideal*.

This script contains two separate code cells to address this imbalance at the point of selecting data pre-partitioning. The first is to distribute the data without repeats, and so provides a relatively small dataset. This second uses data padding to repeat structures within the minority categories to a user-defined size, thereby artificially increasing the size of the data. This second method comes with the caveat that data will contain redundancies, and so will be more prone to over-fitting and less generalizable in the padded categories.

**Padding is a process used in machine learning to artificially inflate a dataset to balance the input categories. This typically involves duplicating data within the under-sized (minority) category. This allows the larger categories to retain the benefit of their size.*

Results

4-way classification: Padding up to 300 structures per domain



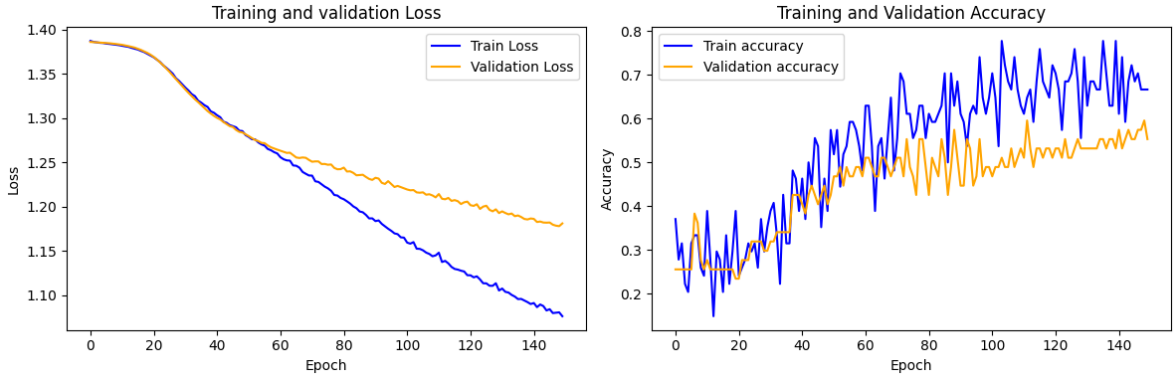
Test Accuracy: 0.6417 (64.17%)
Figure 49: β-carbon reduction state model's accuracy and loss over 150 epochs training, padding to 300.

Training loss and validation loss bifurcate around epoch 10 in figure 49. This would suggest that, in part, the model is over-learning the training data idiosyncrasies and that what is being learnt is not meaningfully applicable to unseen data.

Reduction state	Frequency of incorrect prediction on test set (total 60)
NR	6
KR	18
DH	12
ER	7

Figure 50: Frequency of incorrect predictions of reductions state, within test data partition.

4-way classification: No padding



Test Accuracy: 0.4043 (40.43%)

Figure 51: β-carbon reduction state model's accuracy and loss over 150 training epochs, no padding (total 117 per reduction state label).

Reduction state	Frequency of incorrect prediction on test set (total 34)
NR	10
KR	8
DH	4
ER	6

Figure 52: Frequency of incorrect predictions of reductions state, within test data partition. No padding.

Comparison of padded and unpadded 4-way reduction state classification

Both the padded and unpadded data result in models that seem to learn a degree of representative patterns relating to the categorisation of reduction state, as indicated by the model exceeding the 25% random success rate we would expect from random assignment to 1 of 4 categories. However, the validation and training data loss and accuracy diverging is cause for concern.

The padded data produces a higher accuracy for test and validation data compared to the unpadded data over the course of 150 epochs; however, the model trained on the padded data is inherently less reliable due to the replication of data and redundancy across data partitions. We can see that the loss values for training and validation data bifurcates earlier in the padded data, indicating that the model is increasingly more representative of the training data's idiosyncrasies over epochs. However, loss does continue to decrease in the unpadded data, albeit at a slower rate, potentially indicating the some of what the model learns is representative of the true categories.

To disambiguate what is learnable in the 4-way classifier, we divided the classification problem into a series of binary classifiers that explore each of the 4 categories in a pairwise manner.

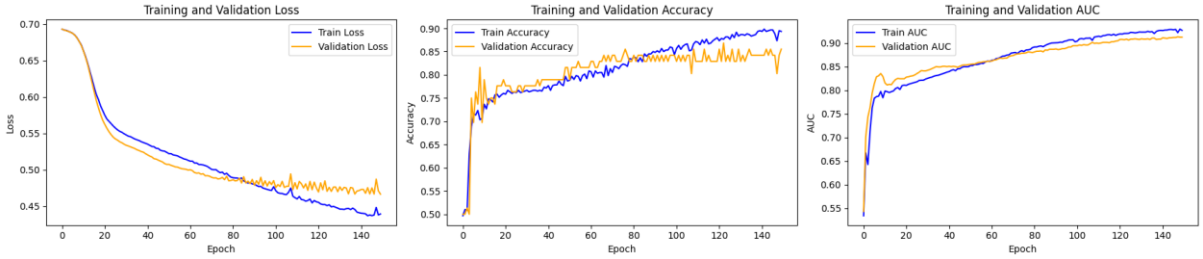
Ketosynthase GCN binary classification models

Packages: OS, torch, Graphein, pandas, re, functools, sklearn, torch_geometric, matplotlib, numpy,
Python version: 3.9, run as Jupyter notebook.
OS: Ubuntu 20.04.6 LTS
Optimiser: Adam[96]
Loss function: Cross entropy (PyTorch)

Binary classification scripts for β-carbon reduction state used in this section use identical architectures and initial seeds. Scripts vary slightly in which reduction state types are being inputted to the model and whether the validation partition is included in the explainer loop. The architecture is identical to that of the AT models in figure 37, as is the process for generating xAI scores.

1. KR vs DH, no padding

Script: `01.03.03.01_KRvsDH_Binary.ipynb`
Dataset size: 376 per label



Test Accuracy: 0.8684 (86.84%)
Test AUC: 0.9266 (92.66%)
Figure 53: Loss and accuracy over a 150-epoch training period for validation and training data sets.

Figure 53 shows a model that initially learns meaningful representations of the underlying category assignments. Between epochs 40 and 60, the training data's loss values separate from the validation set, suggesting that what is being learnt beyond epoch 60 is not applicable to the validation set. Beyond epoch 60, the validation data does not have decreasing loss and accuracy beyond this point decreases. On the basis, we conclude that this model was best at epoch 60, and so the training was repeated (using the same seed).

Test Accuracy: 0.8553 (85.53%)
Test AUC: 0.9245 (92.45%)
Figure 54: Loss and accuracy over a 60-epoch training period for validation and training data sets.

The model generated in figure 54 was used to run the xAI on structures within the test data partition (n=38 per class, 76 structures total).

xAI results

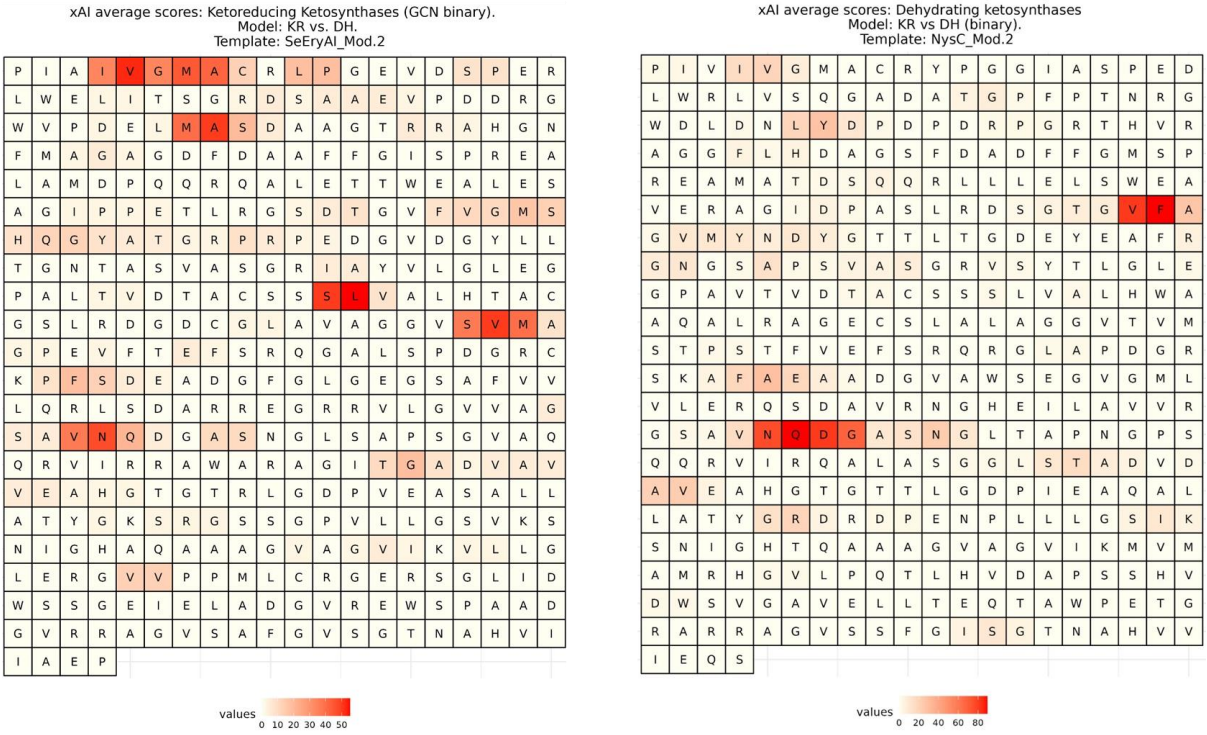
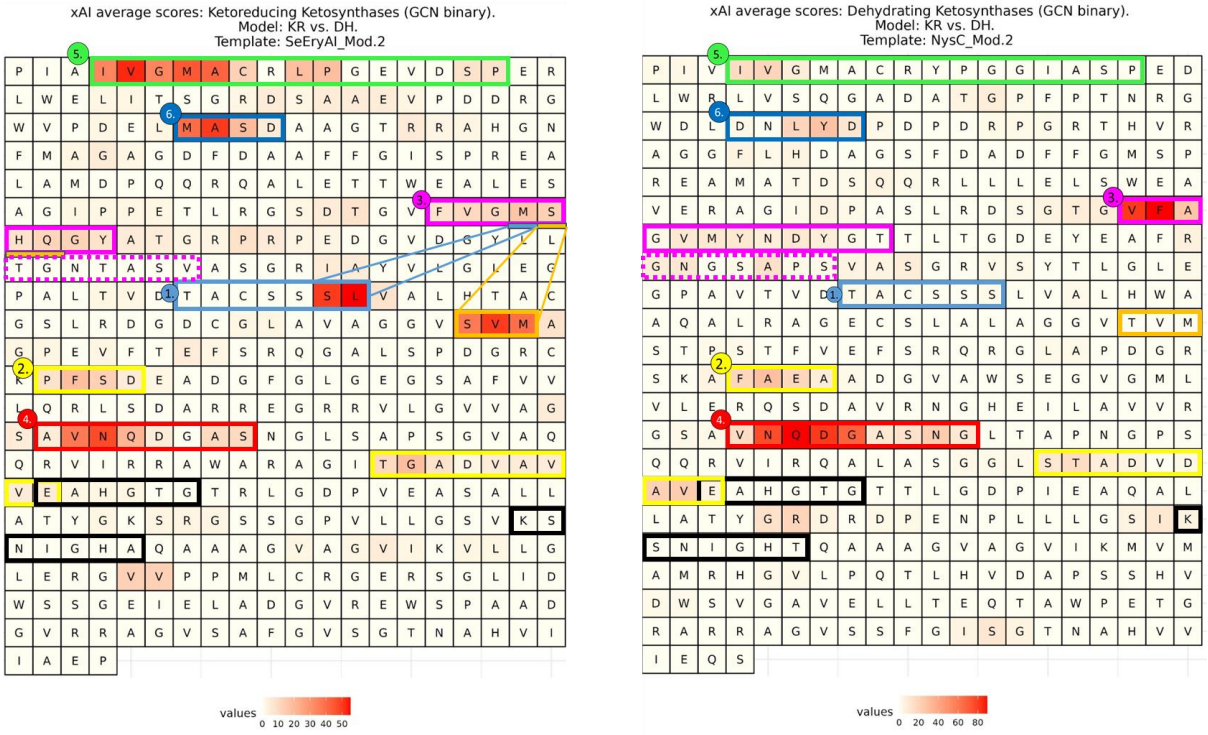


Figure 55: Word search heat maps of xAI averages (n=38) generated from KR vs DH, for β-ketoreduced accepting ketosynthases (KR-type) (left) and β-dehydrated accepting ketosynthases (DH-type). Note that the heat map score colour scale is per model and not comparable between word searches.



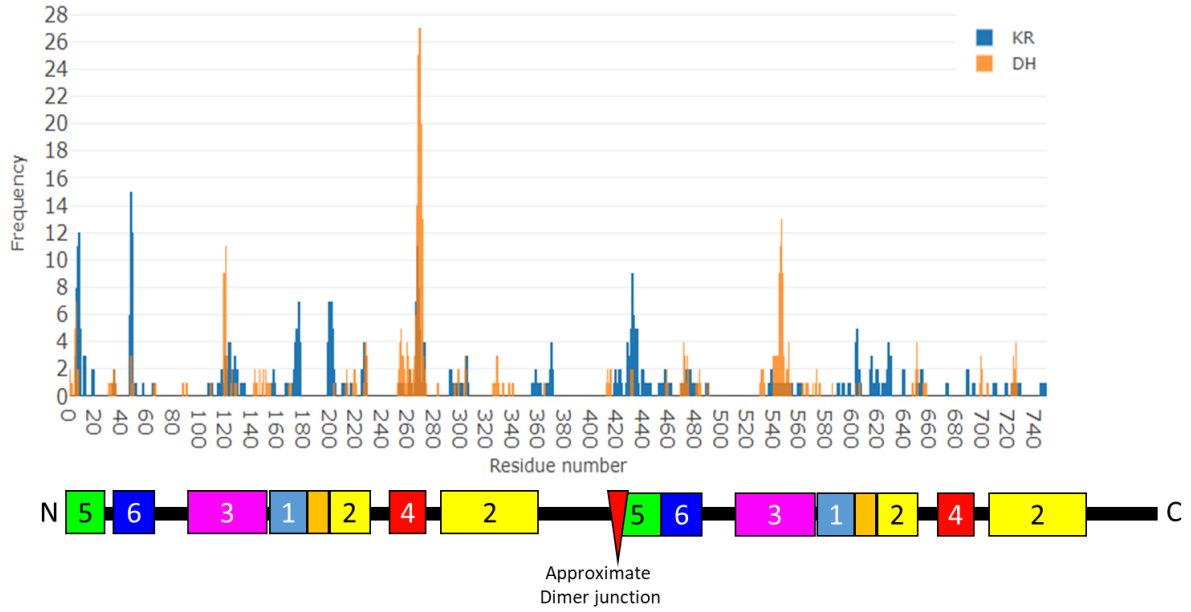


Figure 56: **Top:** Word searches in above figures 55 with regional grouping applied to model, to be comparable with figures 57-59 below. **Bottom:** Alignment-agnostic histogram displaying node (residue) frequencies detected by xAI. This graph has been constructed with raw values and has not been condensed to a monomeric view, as shown in the word searches. Below the histogram is an approximate map of nodes to regional diagrams. Note this is an approximation because the nodes are not aligned – any insertion/deletions in the sequences will move residues out of relative frame. This approximation will be more inaccurate across the dimerization junction due to variance in protein length (see figure 22B – ketosynthase monomers are mostly between 420 and 430 amino acids in length, so this will create an approximate range of +/- 20 at the end of the second dimer).

- Region 1, light blue, highlights the active site TACSSS domain. Histidine supplying active site motifs EAHGTG and KSNIGHT have been highlighted in black.
- Region 2, yellow, covers a grouping of multiple secondary structures proximal to the active site histidines. These are predominantly behind the histidines relative to the active site.
- Region 3, magenta, highlights a beta-sheet through to a loop proximal to the active site cysteine, proposed as a polyketide pocket (see below). This region loops back round to form a pocket on the opposite dimer, where we have used dashed lines. A closely associated region detectable in the β-ketoreduced accepting ketosynthases has been highlighted gold, and proximal resides are linked to region 3.
- Region 4, red, is a beta-sheet though to a loop that presents on the KS-AT cleft side of the KS.
- Region 5, green, is a beta-sheet through to a loop that contacts the ferredoxin-like linker that adjoins the KS to the on-peptide AT domain.
- Region 6, dark blue, is found in helical regions, described in the Dutta-Whicher model[24, 25] as reaction-cycle motile to create the post-condensation ACP access route.

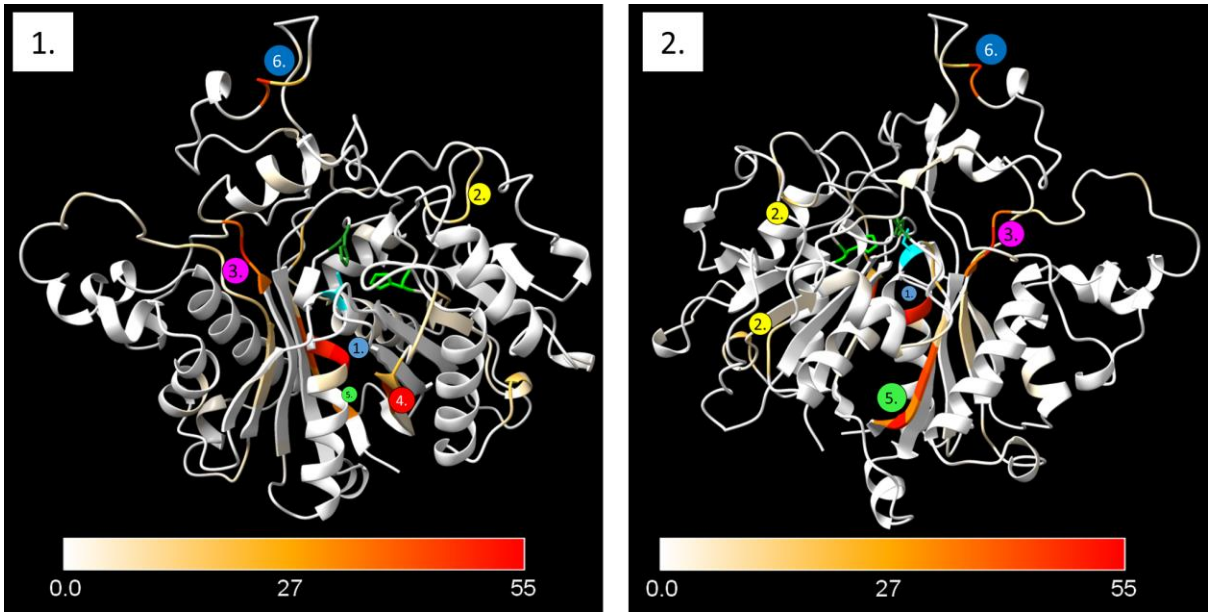


Figure 57: SeEryAI_Mod.2 xAI heat map coloured structures, with regional markers per figure 56 above. Active site cysteine and histidines are coloured cyan and green. Panel 1 views the ketosynthase from the dimerization surface. Panel 2 views from the opposite side.

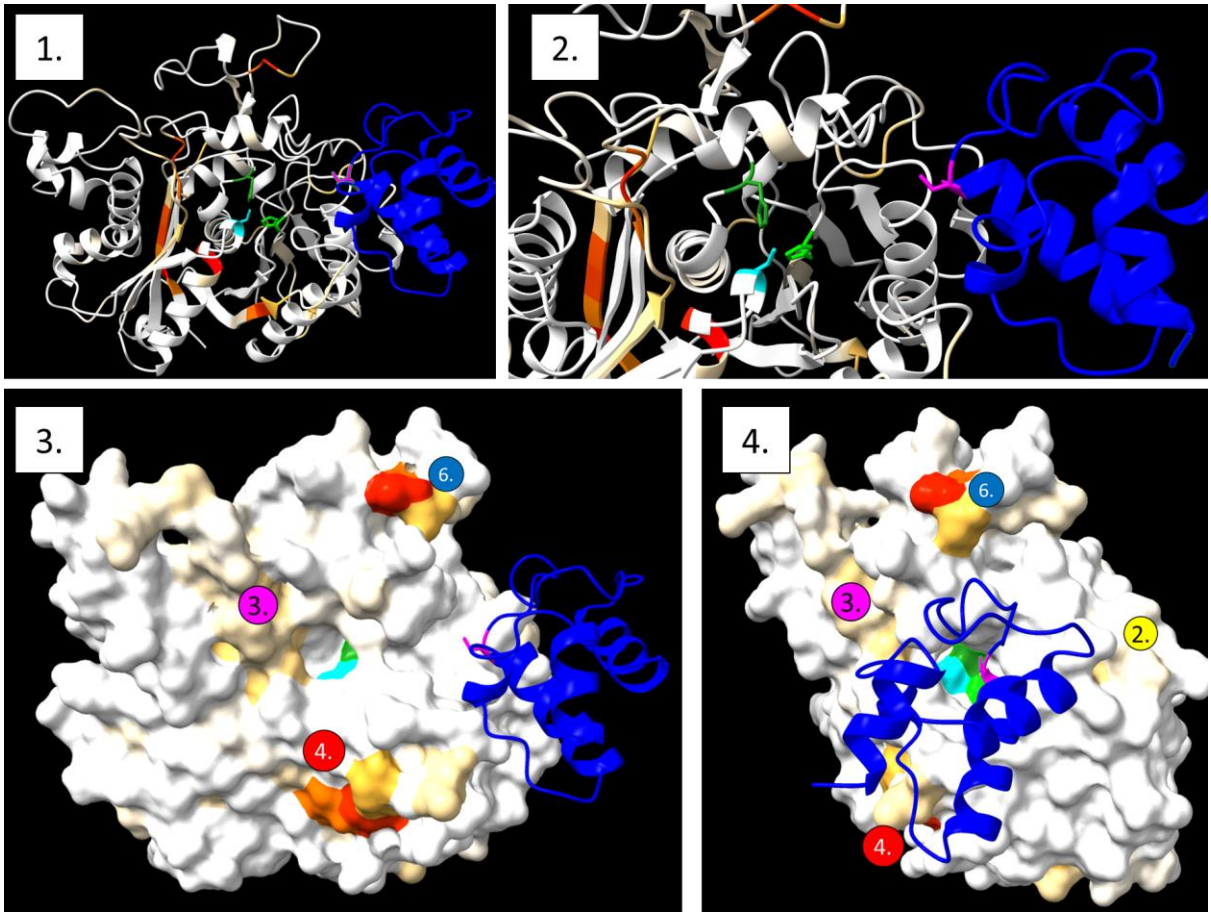


Figure 58: AlphaFold-multimer[49] structure of dimeric SeEryAI_Mod.2 ketosynthase, modelled with partner ACP (blue). Coloured numbers correspond to figure 56 above. **1-2:** Monomeric, as viewed from dimerization surface. Note that the polyketide/extension unit would be bound to a phosphopantetheine arm, which would extend the ACPs reach further into the active site. **3:** Monomeric surface view* SeEryAI_Mod.2 as viewed from the dimerization surface, looking into the active site. Heat map scoring applied to all structures is per xAI averages and corresponds to the colouring in figure 56. **4:** Surface view from ACP approach of the ketosynthase dimer.

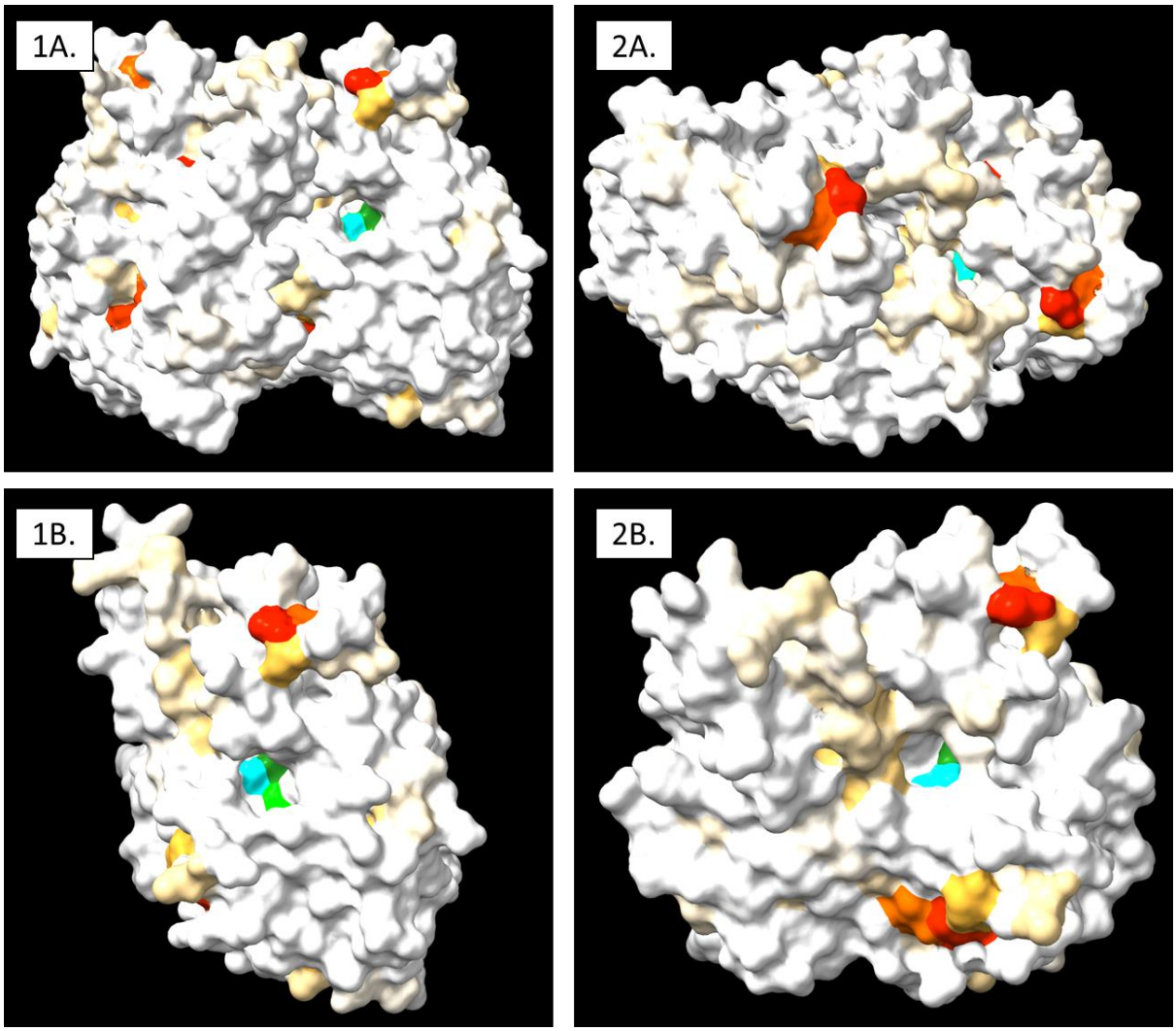


Figure 59: Surface view of SeEryAI_Mod.2, heat map coloured with xAI average scores (key in figure 56). Active site cysteine has been coloured cyan, histidines in greens. Panels **1A-2A** show dimeric views and panels **1B-2B** show monomeric views. **1A-1B** show a view from the KS-AT cleft, where the docking helices would point down (so net direction of substrate movement through this module would be bottom to top). What is noteworthy about these views is the presence of a tunnel from the direction of upstream ACP approach, through the surface map, to a pocket containing the active site cysteine. **2A** views the active site from the perspective of the downstream ACP (in the Dutta-Whicher model) and finds another tunnel through the surface map to the active site. Interestingly, along this second tunnel, the xAI averages report the residues to be significant to β-reduction state classification. **2B** views the monomer's dimerization surface.

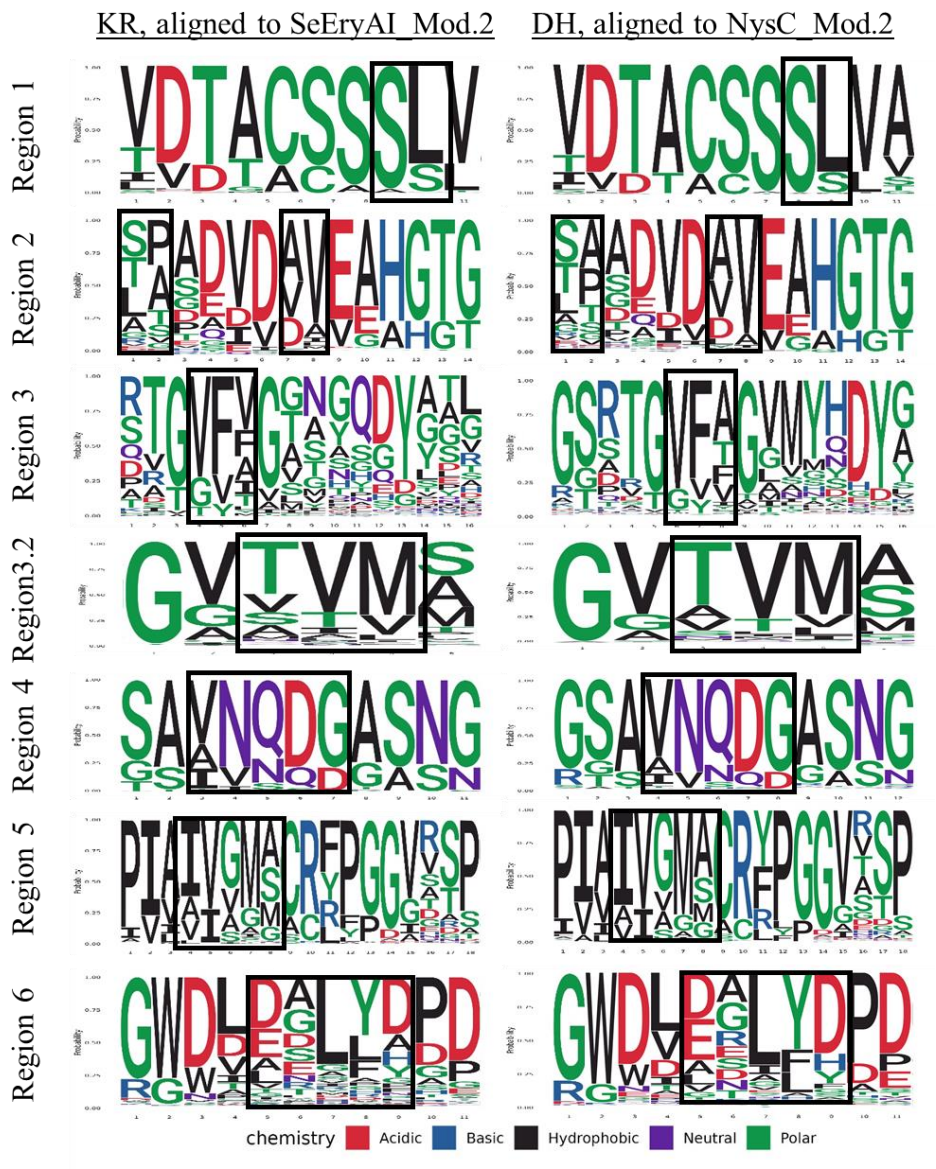


Figure 60: Logo diagrams** generated from sequences aligned to SeEryAI_Mod.2 (KR) and NysC_Mod.2 (DH), based on regional associations in figures 56. Residues of highest predictive value in the explainer averages have been boxed in black.

Analysis of logo diagrams:

- Region 1: No significant variation.
- Region 2: No significant variation.
- Region 3: Boxed region does not show a particularly informative consensus. There is a slightly higher frequency of V over A in the third residue of the DH alignment. Three residues down from this box (M in KR, N in DH) we do find more of a defined consensus in the DH logos, with a weak consensus for GVMYHDY, which presents as mostly hydrophobic for residues 2 and 3, with the equivalent in the DH logos presenting as more polar and less hydrophobic on average.
- Region 3.2: No significant variation.
- Region 4: No significant variation.
- Region 5: No significant variation.
- Region 6: No significant variation.

From the logo diagrams, we find that the residues with highest predictive scores from the xAI averages broadly have consensus across KR and DH type ketosynthases.

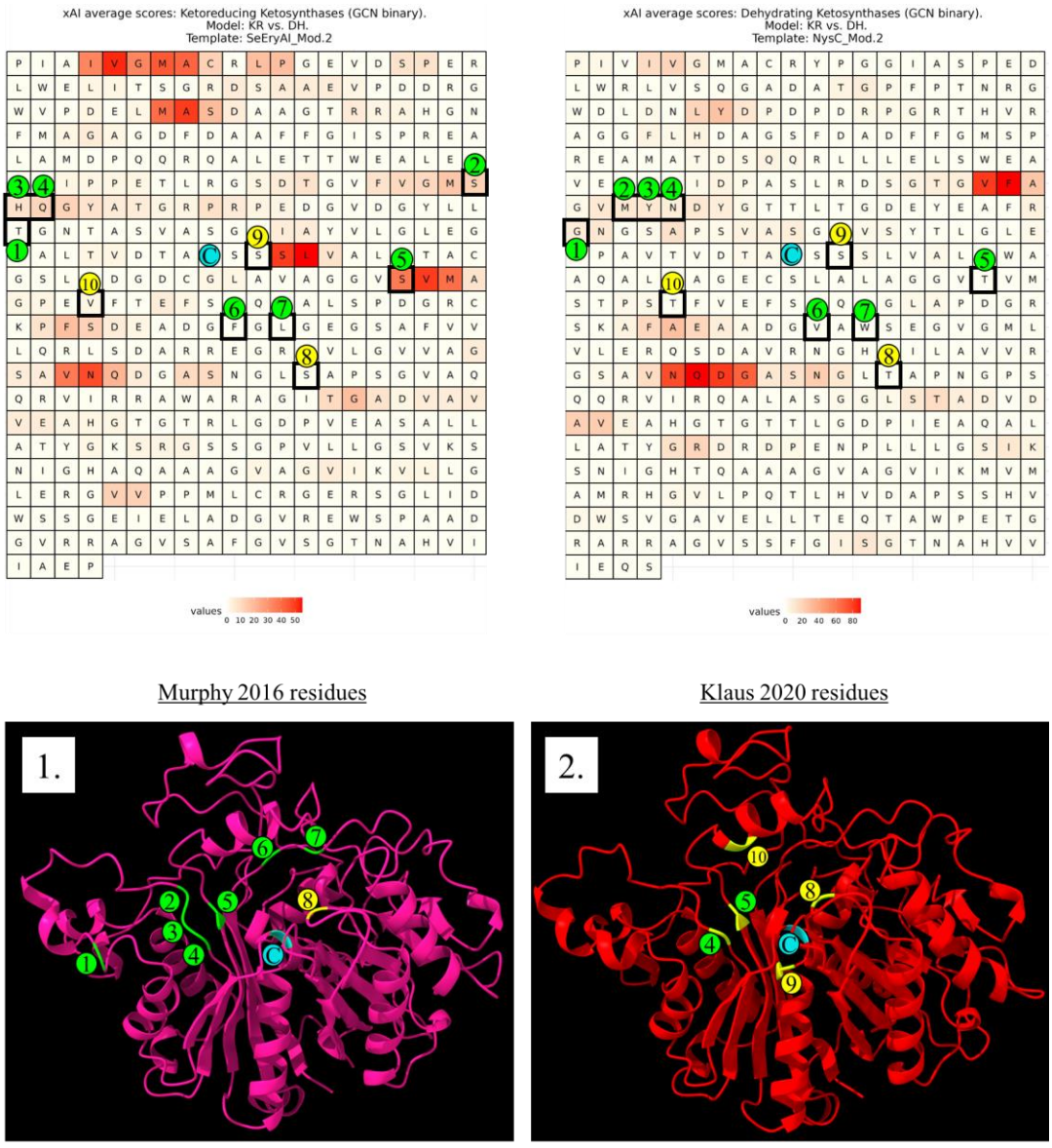
Information contained here is for those less familiar with structural biology:

**Surface view of a protein maps the solvent excluding radii of atoms in a the protein[126]. As part of their maturation following translation (the production of the protein), proteins fold to form a tertiary (3D) structure. Within this structure, there will be areas of the protein that are exposed to the solvent/cytosol (liquid containing the protein) and areas that are excluded from the solvent/cytosol. Viewing protein structures in this way is helpful because it allows us to see what is readily able to contact the environment outside the protein and can give indications of where/what external interactions may occur♠. This has been highlighted in figures 58 and 59, where we show the active site (substrate interacting) residues in cyan and green. What we are illustrating with the surface view figures is that there are residues accessible to the cytosol and proximal to the active site that the xAI is flagging (red/orange). Given that the polyketide substrate is coming from the cytosol, the residues represent a significant point of contact.*

- *♠Its worth noting that protein structures, ketosynthases included, are often dynamic structures and do not form static structures such as those presented.*

***Logo diagrams presented here display the probability of encountering a particular amino acid (residue) at a given position in the protein sequence, for a substrate type. The key of this graph displays the more prominent chemical properties of the amino acids (electrostatic properties). Logo diagrams are used to identify consensus over evolutionary periods and differences between closely related groups. For example, if a residue is kept the same across hundreds of related protein sequences, it is likely that the residues in question is important to the function of the enzyme. We can see a good example of this in the TACSSS motif in region 1 in figure 60 – TACSSS constitutes the primary catalytic residues for interacting with a polyketide and performing the enzymatic function of the ketosynthase and so appears with little variation. Similarly, when comparing between familial proteins, as we are doing here, variation in consensuses may indicate group-defining properties.*

Comparison with experimental evidence from mutagenesis studies



SeEryAII_Mod.1 target (1)

1. L142 = G141 in KR = G141 in DH
2. Y124 = Q122 in KR = N125
3. A124 = H121 in KR = Y124
4. A123 = S120 in KR =M123 in DH
5. S199 =S197 in KR = T198 in DH
6. F234 = F230 = V231
7. F236 = L232 = W233
8. A275 = S273 in KR = T274 DH
9. S173 = S171 = S172
10. V206 = V204 = T205

Figure 61: comparison of GCN xAI heat maps to experimentally validated residues identified in Murphy et al, 2016[7] (green) and Klaus et al, 2020[6] (yellow). Structures presented are AlphaFold predictions for SeEryAII_Mod.1 (Ery3) and SeEryAIII_Mod.2 (Ery6) in panels 1 and 2 respectively. Residues in panels 1 and 2 were aligned in ChimeraX to create the mapping used in the heat maps. Active site Cysteine has been labelled cyan.

Figure 61 shows that residues 1, 2, 3 and 4 are weakly predictive for substrate specificity (20–30%). Residues 1–4 are present in region 3 in figure 56. Residue 5 is strongly predictive (37%) and is associated with a region 3 proximal loop. Residues 1–5 provide some reassurance that the GCN model has learnt non-arbitrary information about the post-cysteine pocket. Residue 10 is also weakly detected in the KR-type.

Residues 6, 7 and 8 are not identified by the model, though residue 8 does present at the C-terminal of the strongly predictive region 4 beta-sheet.

Residue 9 is interesting. From the logo diagrams in figure 60, region 1, we see that the TACSSS motif is highly conserved in both KR and DH-type ketosynthases, and so makes for an unusual candidate for substrate specificity. However, returning to how the node messaging and convolutional operations are carried out in the GCN (see section introduction), information encoded on the nodes following the second convolution are descriptive for 2 nodal k-hops. This could suggest that although the TACSSS motif is constant, the network around it is distinct for DH and KR-types.

Residues of moderate significance within 5 Å and 10 Å distance of highly significant residues

The most significant residues identified in the heat maps seem to be consensus regions (figure 60). Given the nature of convolutions, it is conceivable that the high-scoring consensus residues act as consistently detectable nodes that differentially change in response to the convolutions.

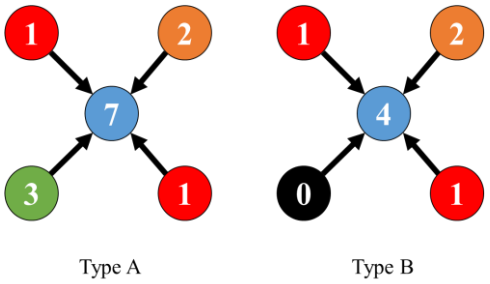


Figure 62: Nexus analogy for why consensus regions may be being identified in the xAI. Here, we designate the blue central node as being a consensus residue in the ketosynthases. Surrounding the node are 4 edge linked nodes that pass information to the blue node as part of a convolution, which we are adding together to give a value on the blue node. If we imagine that the green and black nodes that add to this total is variable between label groups A and B, we can see that this would create a difference in the network, and that this difference can be detected on both the green and black nodes, and on the blue node. If we suppose that the green and black nodes correspond to a set of amino acids (or missing amino acids), rather than one in particular, we may find that detecting a value on the blue node is more parsimonious to detect. Alternatively, it may be the case that the network surrounding the green and black nodes is too noisy, or too generic, and so does not represent an effective discriminator in classification. This could explain why consensus regions are being identified in the xAI scores.

Amino acid R-groups range up to ~5 Å (Ref[127]) , so for 2 convolutions, a range of 10 Å is a reasonable search radius around the consensus regions.

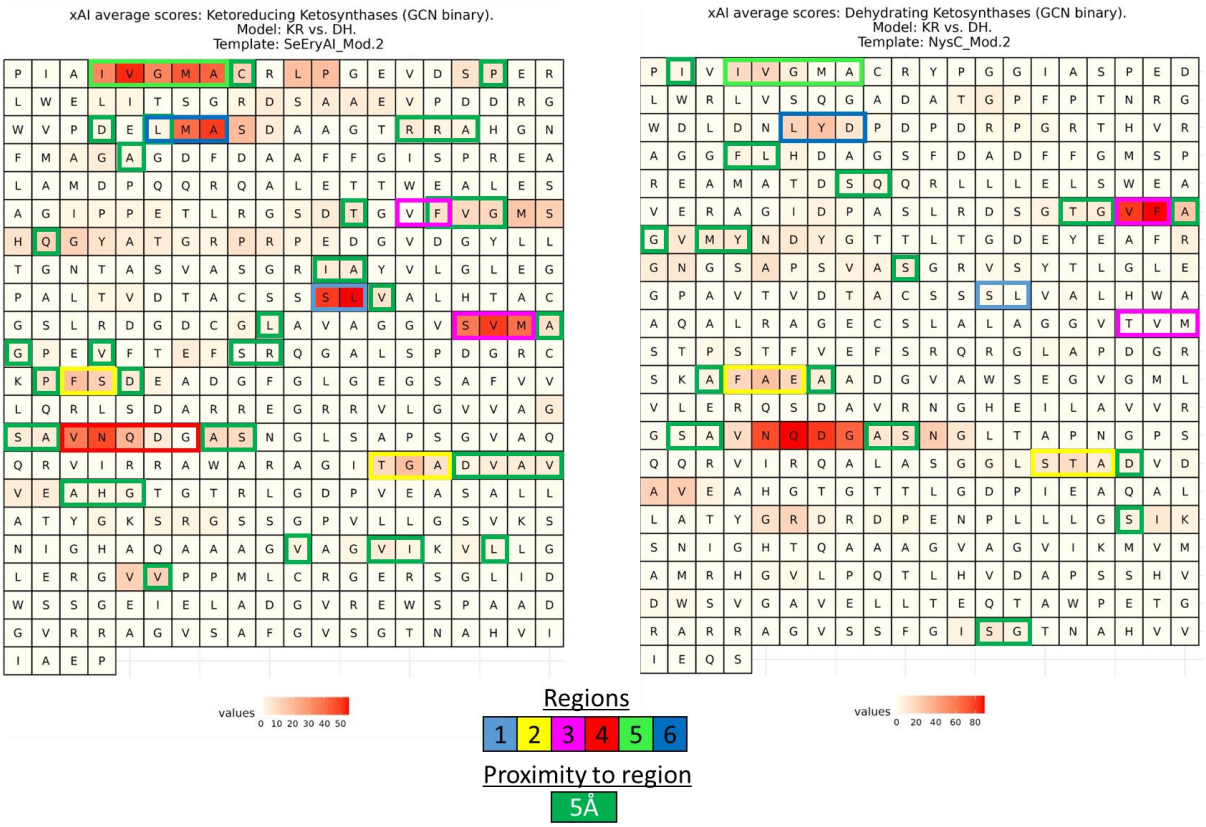


Figure 63: Residues within a 5 Å radius of highest scoring residues in regions 1-6 (marked by colouring in regions key). All 5 Å residues have been combined for a compact figure. Radius determined in ChimeraX using select radius function.

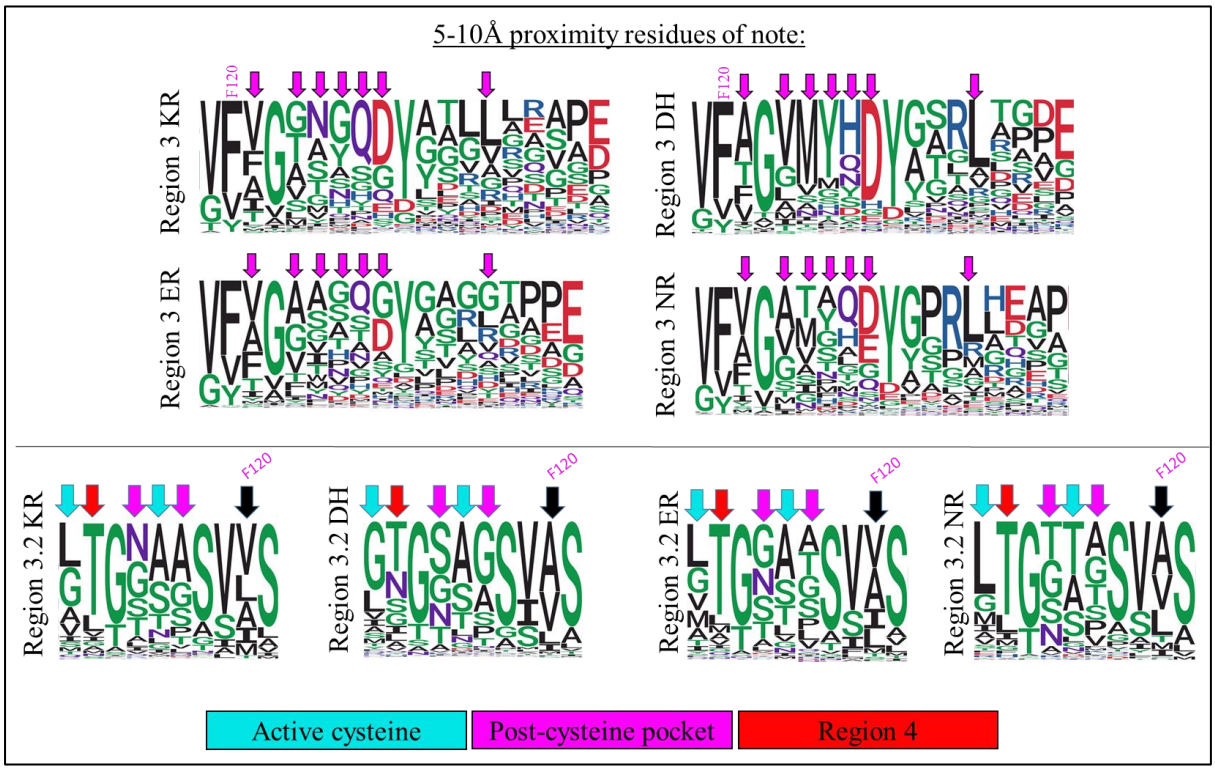


Figure 64: Logo diagrams covering region 3 and dashed region 3 (3.2). These residues have largely been identified in Hirsch et. al 2021[8], Murphy et. al 2016[7] and Klaus 2020[6] already.

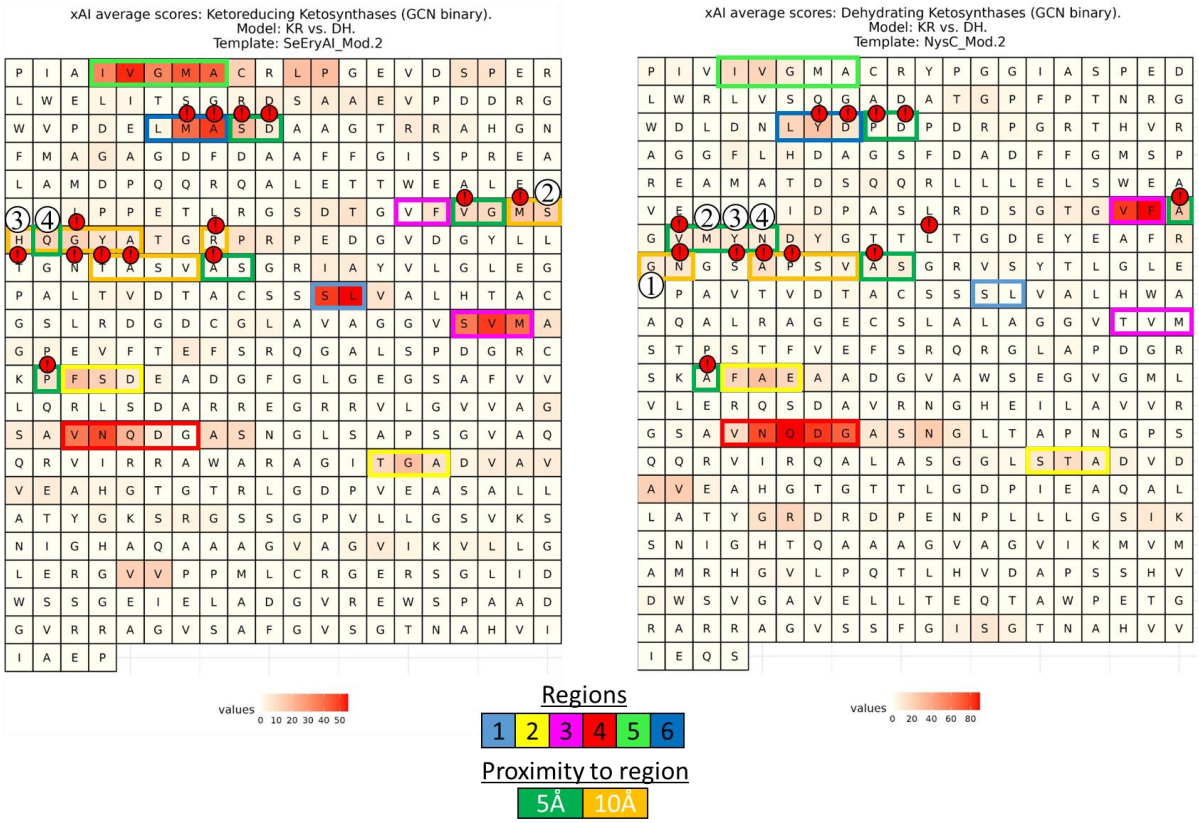
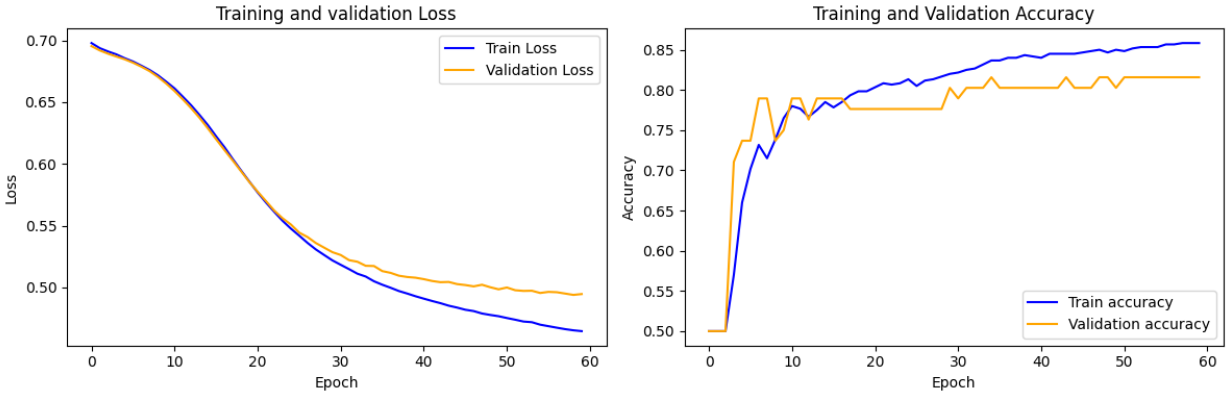


Figure 65: 5 Å and 10 Å proximity mapping of moderate significance residues to highly significant residues. White circled numbers correspond to residues in figure 61 above. Red exclamation marks correspond to residues with reduction type-specific variations not in figure 61, and have not been experimentally verified to our knowledge. These would make for interesting targets for mutagenesis.

This explains region 3 and associated SVM/TVM motif, as well as region 1, which is within 10 Å. Explanation for region 2 is very weak, if explicable at all. Regions 4 and 5 are unexplained.

GraphSAGE 2 layer network
Layer 1: 32
Layer 2: 8

We arrived at this above reduced layer density through iterative optimisation of validation/training loss. Higher neural densities were found to diverge at higher loss values and so were deemed less fit.



Test Accuracy: 0.8158 (81.58%)
Figure 66: GraphSAGE model training

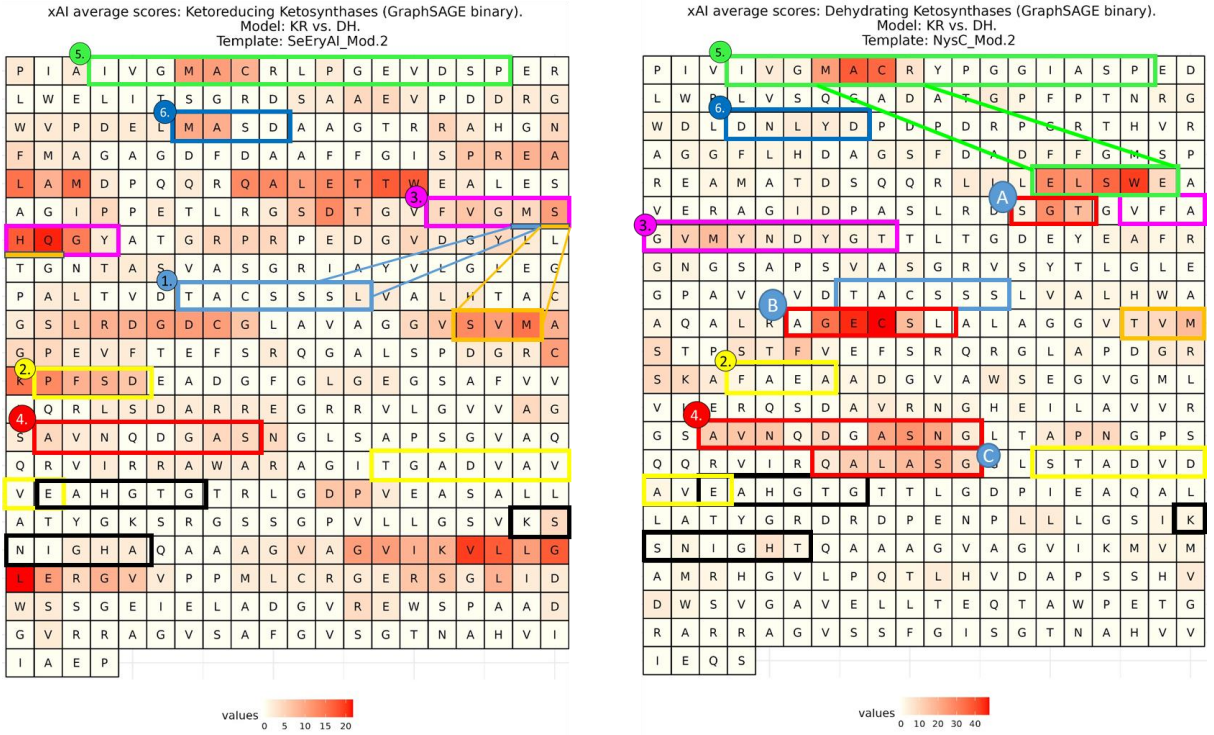


Figure 67: xAI averages for GraphSAGE model predictions on test data. Note that the scale bar indicates that the heat map scoring is lower than figure 56.

GraphSAGE is a spatial convolutional algorithm and does not operate on the Laplacian matrix. As such, GraphSAGE will generate convolutions in a node's local (k-hop) environment, and so should result in more motif-analogous models for the xAI to detect, when compared to the spectral GCN method that detects global patterns.

The word search diagrams in Figure 67 have lower average xAI scores when compared to the GCN model's equivalent (this is denoted in the difference in maximal score on the red-hot encoding). We see in figure 67 poor consistency for the predictions of KR type ketosynthases, as denoted by the scale bar in the left word search and weak regional focus on residues. Regional boxing has been kept from figure 56 GCN for the KR-type diagram.

The DH type ketosynthase word-search form highlights regions that are more distinct, with a higher maximal xAI average score. Region 4 in this model expands across the dimerization interface to capture

two proximal loops SGT and AGECS (red). Region 6 also finds predictive residues in a helix to the beta-sheet in region 6. These results

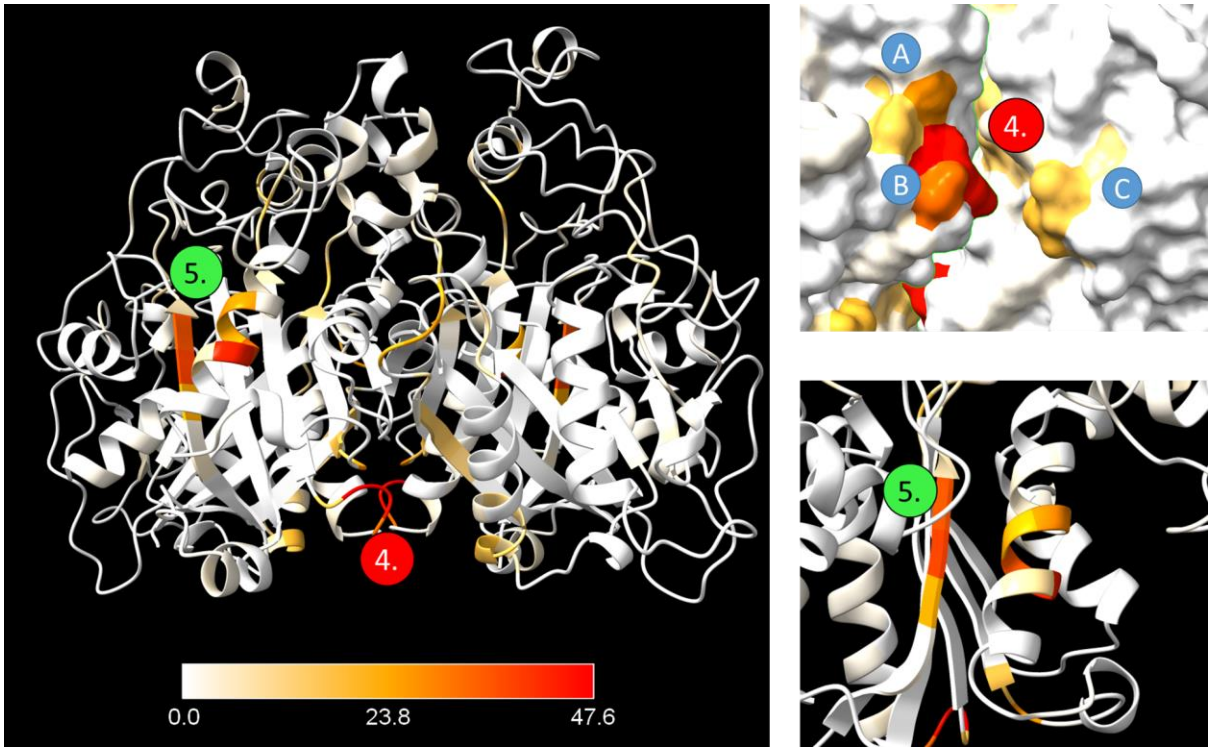


Figure 68: NysC_Mod.2 xAI score heat map AlphaFold structure. Regions 4 and 5 have been highlighted to illustrate expanded regions in figure 67.

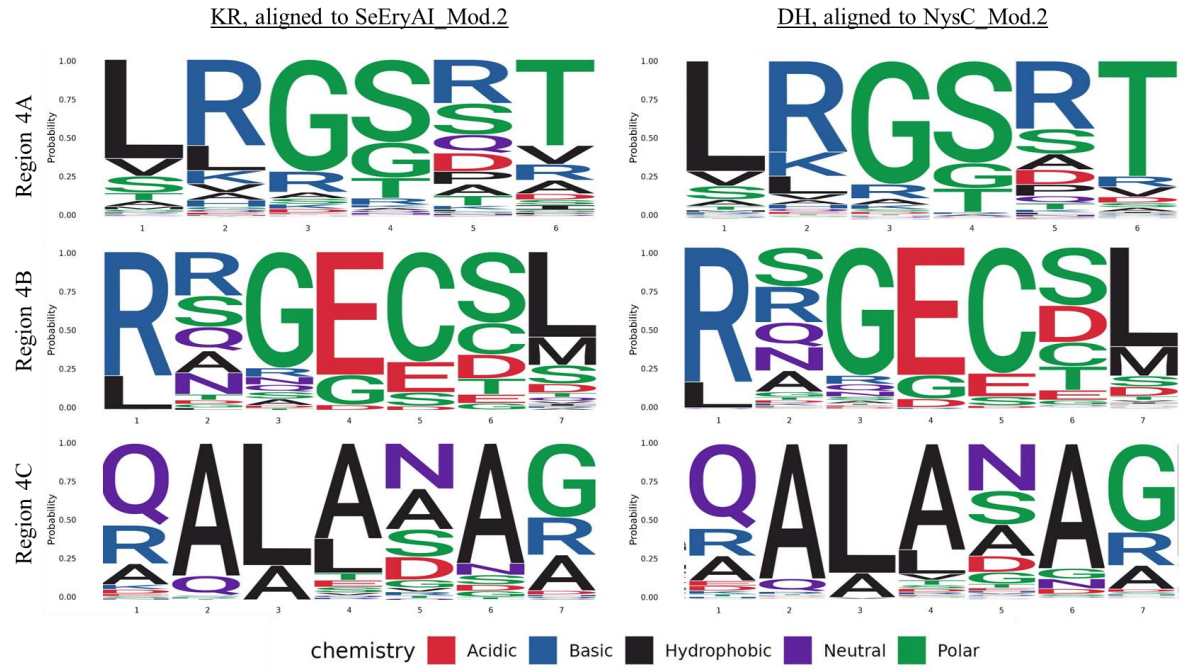
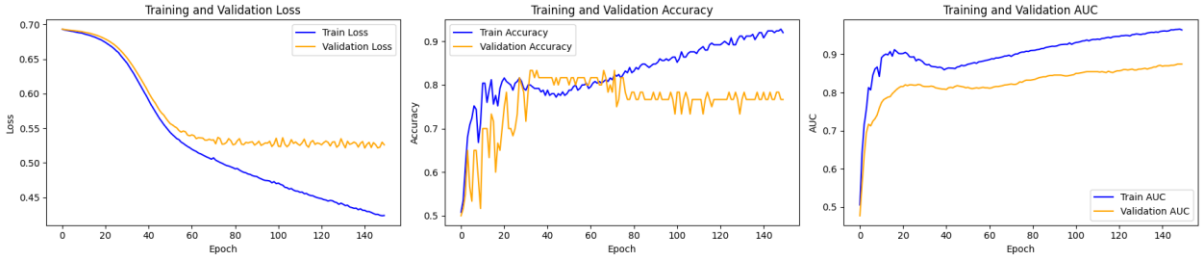


Figure 69: Logo diagram consensuses in region 4 adjacent high scoring motifs for KR-type and DH-type. This figure does not show compelling evidence for a consensus describing a reduction state specific motif.

Interestingly, GATs do not work on this data/seed but GraphSAGE does. GATs and GraphSAGE both convolute on the spatial domain. One of the differences between GATs and GraphSAGE is that GraphSAGE incorporates a node's degrees of connectivity as a feature set, which GCNs does. This might suggest that actually one of the determinants of these networks operating on vs. DH models is detecting nodal connectivity, in addition to residue type.

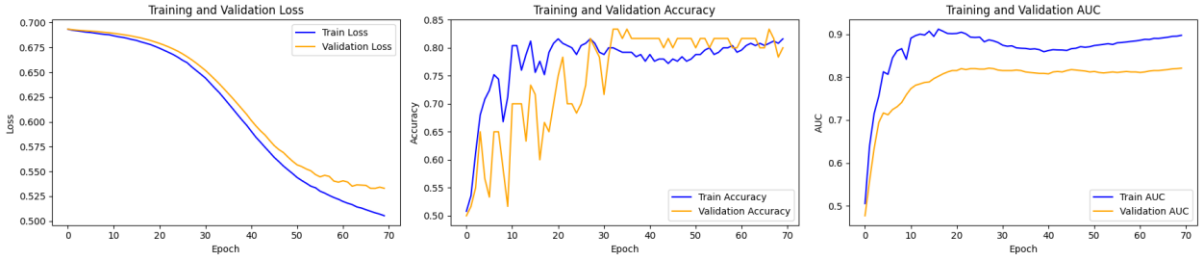
2. DH vs ER, padding to 200

Script: `01.03.03.02_DHvsER_binary.ipynb`
Dataset size: 200 per label (ER padded from 164)



Test Accuracy: 0.8750 (87.50%)
Test AUC: 0.8211 (82.11%)
Figure 70: DH vs ER binary classification model loss and accuracy over a 150-epoch training period for validation and training data sets.

Training graphs in Figure 70 shows a model that initially learns meaningful representations of the underlying category assignments. Between epochs 50 and 70, the training data's loss values separate from the validation set, suggesting that what is being learnt beyond epoch 70 is not applicable to the validation set. Beyond epoch 70, the validation data loss starts increasing and accuracy beyond this point decreases. On this basis, we conclude that this model was best around epoch 70, and so the training was repeated using the same seed.



Test Accuracy: 0.8160 (81.60%)
Test AUC: 0.8211 (82.11%)
Figure 71: DH vs ER binary classification model loss and accuracy over a 70-epoch training period for validation and training data sets.

The model generated in figure 71 was used to run the xAI on structures within the test data partition (n=20 per class, 40 structures total).

xAI results

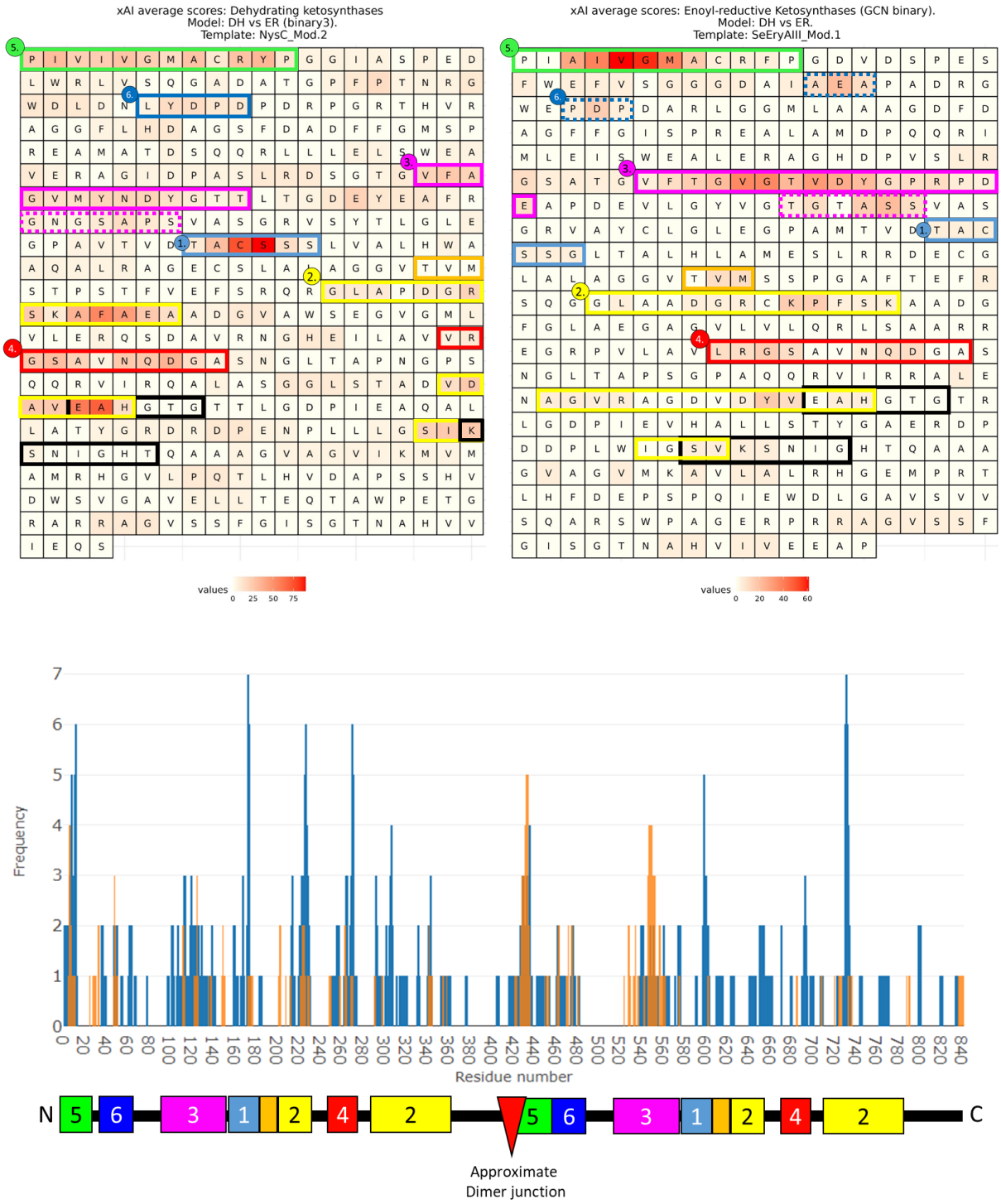


Figure 72: Word searches with regional grouping applied to model to be comparable with figures 56 above. Region 2 has been expanded from the KR vs. DH model, encompassing a wider area behind the active site histidines in KSNIGHT and EAHGTG. Region 3 also appears to be more expansive and significant to this model. Regions 6, dark blue, has a pair of proximal regions detected in the ER-type word search that have been highlighted in dashed lines. **Bottom:** Alignment-agnostic histogram displaying node (residue) frequencies detected by xAI. This graph has been constructed with raw values and has not been condensed to a monomeric view, as shown in the word searches. Below the histogram is an approximate map of nodes to regional diagrams. Note this is an approximation because the nodes are not aligned – any insertion/deletions in the sequences will move residues out of relative frame. This approximation will be more inaccurate across the dimerization junction due to variance in protein length

(see figure 22B – ketosynthase monomers are mostly between 420 and 430 amino acids in length, so this will create an approximate range of +/- 20 at the end of the second dimer).

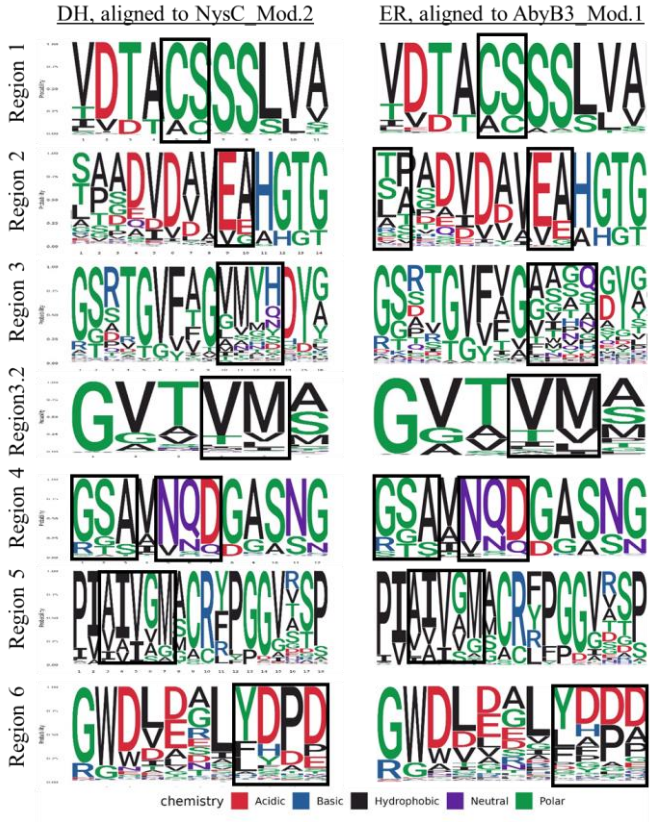


Figure 73: Logo diagrams generated from sequences aligned to NysC_Mod.2 (DH) and AbyB3_Mod.1 (ER), based on regional associations in figures 72. Residues of highest predictive value in the explainer averages have been boxed in black.

Again, we find that the highest scoring residues are appearing in consensus regions, with the exception of region 3. We find somewhat similar residue fingerprint patterns in region 3 the ER-type as seen in the KR-type (see logo figure 64 diagram above). ER-types present as more like KR-types, but not as distinct as from DH-types.

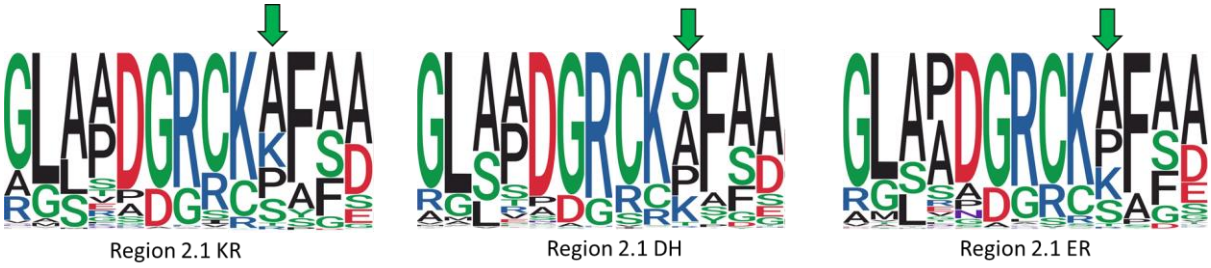
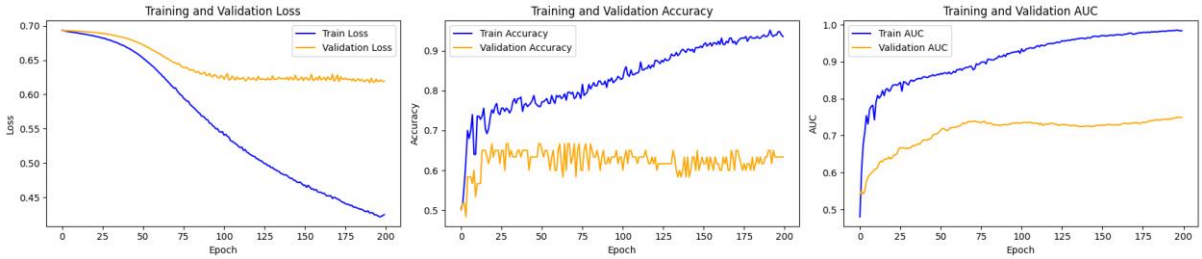


Figure 74: some VERY weak sequence variation behind histidine in EAHGTG

3. KR vs ER: no padding binary

Script: `01.03.03.03_KRvsER_Binary.ipynb`

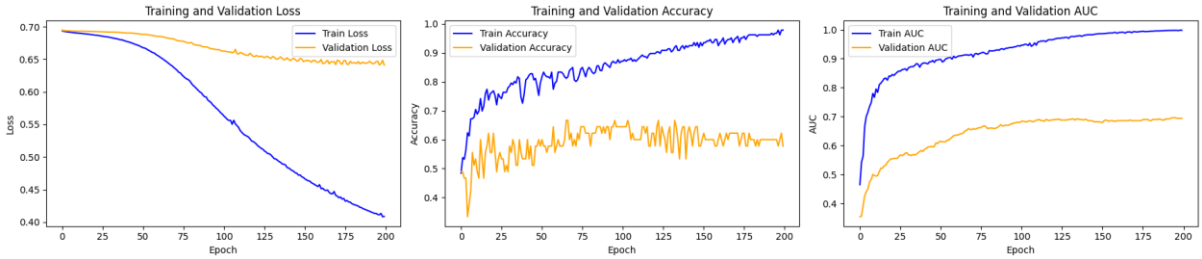
Dataset size: 167 per label



Test Accuracy: 0.7500 (75%)
Test AUC: 0.7500 (75%)
Figure 75: KR vs ER binary classification model loss and accuracy over a 300-epoch training period for validation and training data sets.

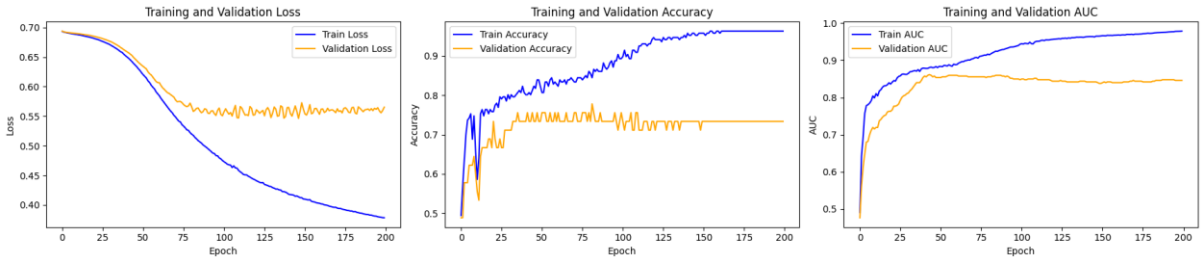
The graphs in figure 75 do not indicate that the model has learnt meaningful representation of the underlying categories of KR and ER, as denoted by the rapid divergence between training and validation data's loss and accuracy scores over epochs. This model was not taken further.

4. KR vs NR: no padding, binary
Script: `01.03.03.04_KRvsNR_Binary.ipynb`
Dataset size: 167 per label



Test Accuracy: 0.5778 (57.78%)
Test AUC: 0.6937 (69.37%)
Figure 76: KR vs NR binary classification model loss and accuracy over a 300 epoch training period for validation and training data sets.
The graphs in figure 76 do not indicate that the model has learnt meaningful representation of the underlying categories of KR and ER, as denoted by the rapid divergence between training and validation data's loss and accuracy scores over epochs. This model was not taken further.

5. DH vs NR: no padding, binary
Script: `01.03.03.05_DHvsNR_Binary.ipynb`
Dataset size: 122 per label



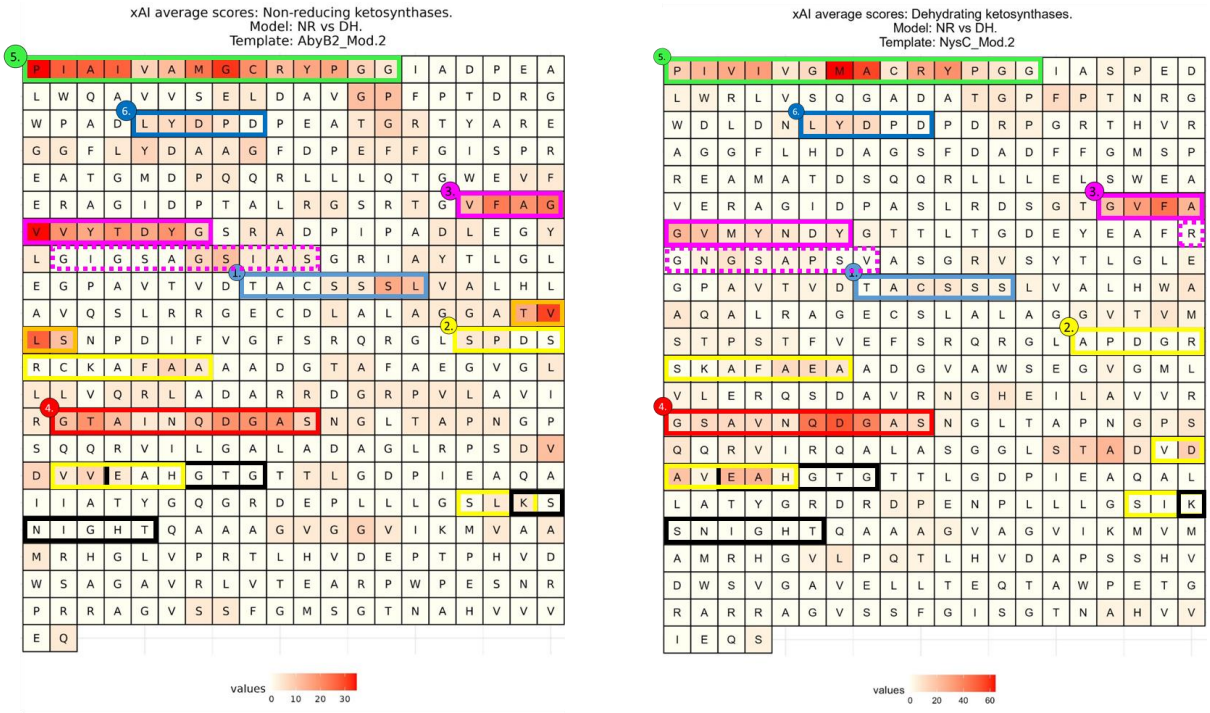
Test Accuracy: 0.7333 (73.33%)
Test AUC: 0.8458 (84.58%)

Figure 77: DH vs NR binary classification model loss and accuracy over a 300 epoch training period for validation and training data sets.

Training and validation data trend together for the first 80-100 epochs in the loss graph and somewhat in the accuracy graph. This model was repeated for a 100 epoch period.

Test Accuracy: 0.7333 (73.33%)
Test AUC: 0.8557 (85.77%)
Figure 78: DH vs NR binary classification model loss and accuracy over a 100-epoch training period for validation and training data sets.
The 100-epoch model was run on the explainer using both validation and test data sets (n=48).

xAI results



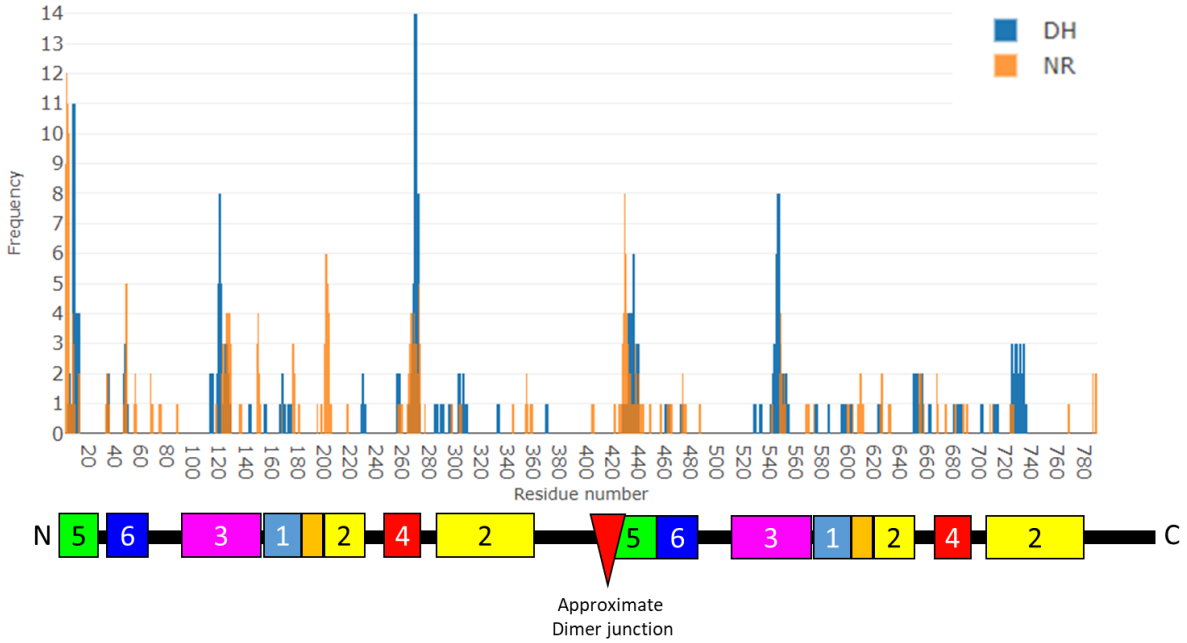
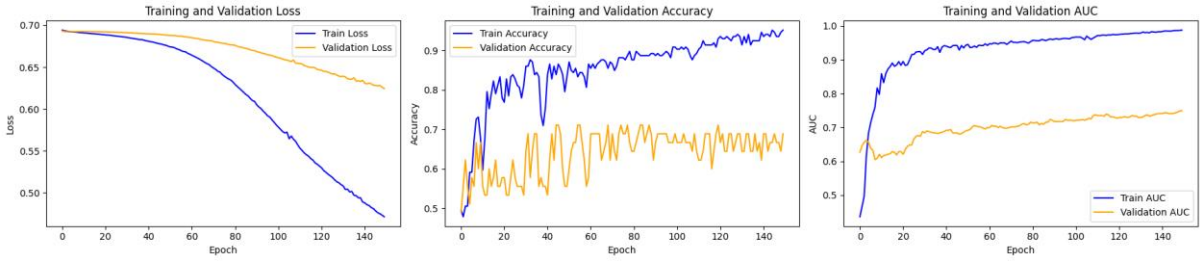


Figure 79: **Top:** Word searches with regional grouping applied to model, to be comparable with figures 56 and 72. **Bottom:** Alignment-agnostic histogram displaying node (residue) frequencies detected by xAI. This graph has been constructed with raw values and has not been condensed to a monomeric view, as shown in the word searches. Below the histogram is an approximate map of nodes to regional diagrams. Note this is an approximation because the nodes are not aligned – any insertion/deletions in the sequences will move residues out of relative frame. This approximation will be more inaccurate across the dimerization junction due to variance in protein length (see figure 22B – ketosynthase monomers are mostly between 420 and 430 amino acids in length, so this will create an approximate range of +/- 20 at the end of the second dimer).

6. ER vs NR: no padding, binary

Script: `01.03.03.06_ERvsNR_Binary.ipynb`

Dataset size: 122 per label



Test Accuracy: 0.6889 (68.89%)

Test AUC: 0.7490 (74.9%)

Figure 80: ER vs NR binary classification model loss and accuracy over a 150-epoch training period for validation and training data sets.

The graphs in figure 80 do not indicate that the model has learnt any meaningful representations of the underlying categories of NR and ER. This model was not taken further.

Discussion on β-reduction state binary classifiers results

From the GCN loss and accuracy graphs, we found that a 4-way classifier for β-carbon reduction states was able to learn partially representative patterns of the reduction state categories (64% accuracy on padded data). To investigate this further, we ran a series of binary classification models comparing reduction states on a pairwise basis.

Accuracy				
β-carbon reduction state	NR	KR	DH	ER
NR		58%	73%	69%
KR	58%		86%	75%
DH	73%	86%		82%
ER	69%	75%	82%	

AUC				
β-carbon reduction state	NR	KR	DH	ER
NR		69%	86%	75%
KR	69%		92%	75%
DH	86%	92%		82%
ER	75%	75%	82%	

Figure 81: Binary classification model accuracy and AUC scores at final epoch. Models that did not have convincing validation/training data loss scores have been highlighted red.

We found that only models trained against the DH-type presented loss and accuracy graphs indicative of meaningful representations of the underlying categories. This suggests that when it comes to the β-carbon reduction state, a carbon-carbon double bond requires a different ketosynthase architecture to those without a double bond.

Double bonds differ to the other β-carbon reduction states (ketone, alcohol group or hydrogen), in that a double bond enforces a rigid bond angle and prevents conformational isomerism around the carbon-carbon bond. In effect, a polyketide has less rotational freedom when entering the ketosynthase. Polyketide dehydratases can produce *cis* or/and *trans* double bonds in the polyketide, which enforce a kink or a linearity in the polyketide respectively. Whether this is why the DH-type ketosynthases form a detectable category of ketosynthase is unclear.

Compared to the pilot on acyltransferase domains, the ketosynthase models give higher loss scores and return lower accuracy predictions. However, this difference in the results was anticipated. Proofreading does not always occur when a module is modified, and as shown by ketosynthase mutagenesis studies, substrate specificity can exist as a continuum[6, 7] and instances where module swaps work[22, 23]. Therefore, it is likely that not all ketosynthases will possess properties that can be learnt or detected by a model.

Another observation on the binary models is that we have ~60% less structures for the ER and NR categories than we do for the KR and DH categories. This may have limited the ability of the model to detect learnable patterns in the structural data, as well as introducing noise into the DH vs ER/NR models.

Discussion by region

Regions 3 and 6 (and maybe 1 by proximity)

The order of operations for a ketosynthase are as follows:

1. Upstream ACP hands the polyketide to the active site cysteine, from the direction of the KS-AT cleft, passing the polyketide through a substrate channel seen in figure 58.3 and 59.1A/1B.
    - The polyketide bound to the ACP is ordered such that the first condensation step of the molecule should be entering the ketosynthase first, “omega” carbon to alpha carbon, and will need to pass through the active site, such that the active site cysteine can

interact with the α-keto group of the polyketide and displace the ACP, completing the *trans*-acylation step.
    - This model requires a cavity for the omega end of the polyketide to reside[9], which region 3 spans.
2. Next, the AT domain downstream (structurally adjoining, relative C-terminal AT) passes an extension unit to the downstream ACP, which will be inserted into the ketosynthase active site via the substrate channel and the condensation reaction occurs.
3. The polyketide is now extended and needs to leave the KS. The Khosla model suggest that the PKS exits through the entry used for the input substrates to the condensation. The Whicher/Dutta model find that a second entry opens with the topside helices seen in region 6 parting to expose a new channel to the active site.

Given this order of events, we postulate that we may not see substrate specificity the area between the active-site cysteine and the ACP, and instead find it occurring behind the active site. At first glance, this may sound counter intuitive; it would be logical to reject the substrate before it goes somewhere it might become stuck, where a *trans*-acting thioesterase cannot reach.

Post-cysteine proofreading is the more logical proposition when we consider that the entire polyketide must pass through the active site entrance region, and so this region inherently needs the most degrees of freedom for the chemical properties of the polyketide. This region also has to be charge-compatible with the ACPs[20]. Based on this logic, we suggest that the point of translocation, when the polyketide is inserted into the post-cysteine pocket, is the point where we could see proofreading of the β-keto reduction state occurring, as well as properties of the polyketide as a whole. This logic is supported by the mutagenesis studies illustrated in figure 61, which show that modifying this pocket alters the substrate specificity.

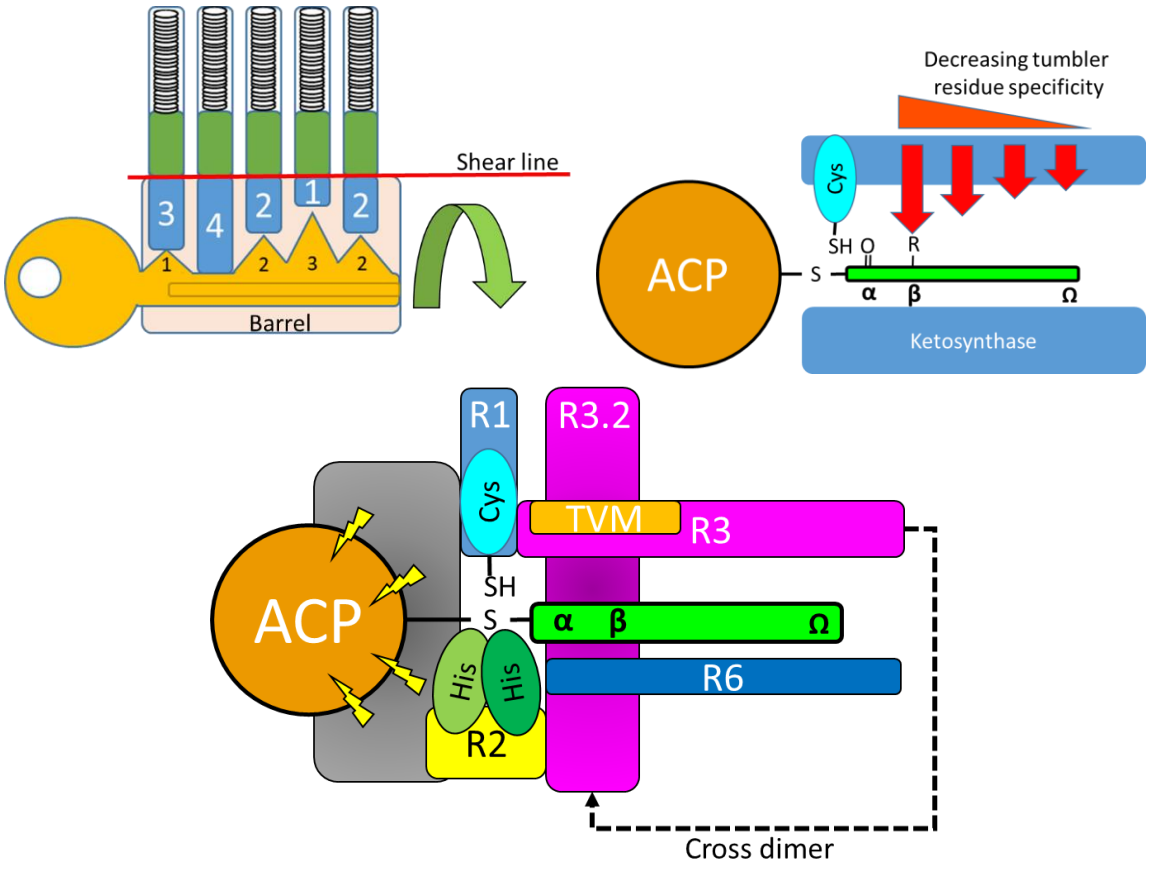


Figure 82: Proposed lock and key model for ketosynthase-substrate interaction. **Left:** A model of a lock and key. In this model, a key has teeth of length inversely proportional to the size of the tumblers (blue, AKA pins) in the lock. In order to turn the key in the lock, all the tumblers need to line up along the sheer line for the key to rotate and open the lock. **Right:** Schematic explanation for how residues within the post-cysteine pocket could create a tumbler-key type interaction, based on the chemical properties of the polyketide. *Trans*-acylation from the ACP to the KS active site cysteine would be analogous to the lock opening. Bottom: Lock and key analogy applied to regions 1, 2 3 and 6. Active site histidines and cysteine have been added. Grey box denotes the ACP interacting charged surface of the ketosynthase, which we did not find to be predictive for substrate β-carbon chemistry but has been shown to guide ACP docking[6, 20].

Region 1 and 2

Region 1 is immediately below/includes the active site Cysteine. Region 2 is behind the active site histidines. We did not find support for substrate-specific motifs within these regions that could be used to direct mutagenesis. However, the functionality of these regions to the ketosynthase suggests that perhaps this is not nonsense – the catalytic cysteine and histidine residues are critical to enzymatic function. It may be the case that there are detectable spatial differences in how the residues organise in space, and what they partner with, and this may be critical to how the ketosynthases implement substrate specificity.

Region 4

Region 4 seems to be part of a dimerization surface in the KS-AT cleft, seen in figure 47. No substrate specific motifs were found. Some speculation on this region can be inferred from the KR vs. DH GraphSAGE model, where region 4 seems to join the two monomers. Maybe the network at region 4 alters how the dimers join. If it does, this might change the dimensions of the region 3 pocket. Another speculation is that region is on the upstream side of the ketosynthase and might have something to do with how the upstream ACP interacts with the ketosynthase. Again, this is all speculative.

Region 5

Similar to region 4, analysis of region 5 did not find any motifs within a 10 Å radius that would indicate residues for mutagenesis. Region 5 immediately follows the N-terminal docking domain or post-ACP linker, depending on the gene's modular composition. Region 5 also forms part of the interface to the ferredoxin-like linker that adjoins to the downstream AT domain. None of this information offers a credible explanation to why the models would detect these.

Binary classifiers for α-carbon methylation state in KS-domains

Polyketide substrate α-carbon methylation state prediction from KS-domains

Script: `01.03.04.01_KS_GCN_upstream_methylation.ipynb`
Packages: OS, torch, graphein, pandas, re, functools, sklearn, torch_geometric, matplotlib, numpy,
Python version: 3.9, run as Jupyter notebook.
OS: Ubuntu 20.04.6 LTS

In this script, we examine if KS structures have an association with the α-carbon methylation state of the polyketide substrate. Our method of labelling is defined by the upstream AT domain type: whether Mal or Mmal. This script is identical to the AT domain graph prediction script, with the difference that we are using KS structures and a different label set, which corresponds to the upstream AT domain.

Script overview:

1. Import KS domains from files and combines with substrate labels data. Labels were produced from script `01.03.02_KS_4-way_GCN.ipynb` and saved as a .csv file 20231129_upstream_methylation.csv, which was imported to this script.
2. Generate graph networks from carbon alphas using a k-nearest neighbour method in Graphein.
    a. K = 3, long interaction threshold = 0.
    b. Node features are implemented as a one-hot encoding (OHE) of amino acid type.
    c. Labels are unified in the graph constructor.
    d. Graph data is converted from networkX to PyG format.
3. Graph data is imported to Torch Geometric using a data loader. Data is balanced for even mmal:mal, batched and partitioned to training, test and validation data, using an 80:10:10 split. Optimiser algorithm and loss function used are ADAM and cross entropy.
4. Training and validation data is run through a loop that trains the model. Network architecture is in below figure 83. The trained model is then used to generate a final accuracy and loss score against the test data partition.
5. A final code block in the script allows a user-defined file to be imported, converted to a graph and run through an explainer script.

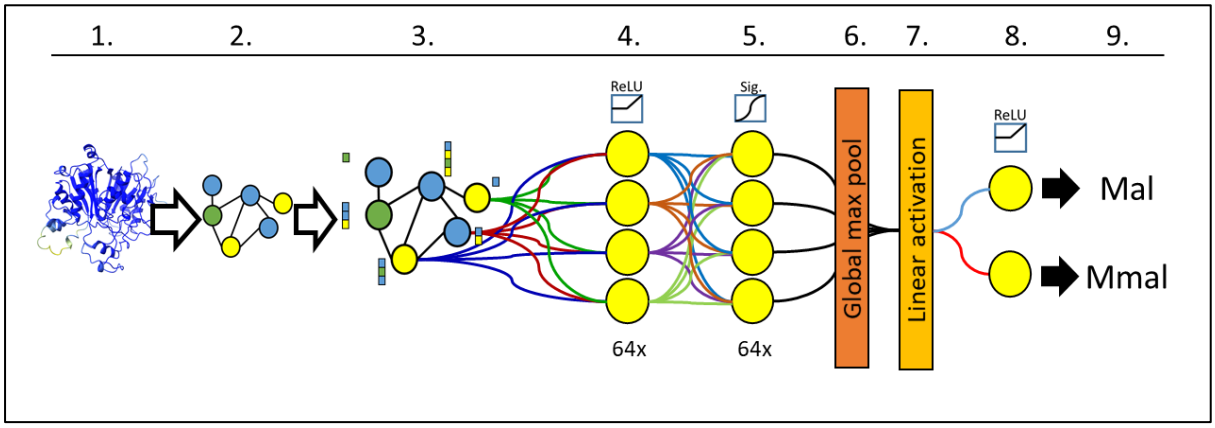
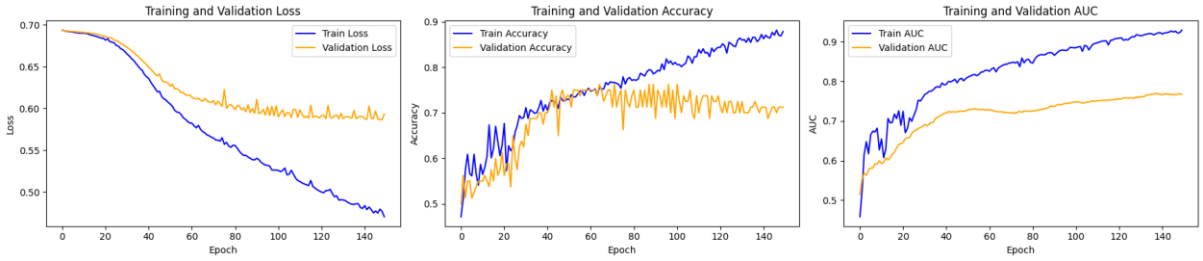


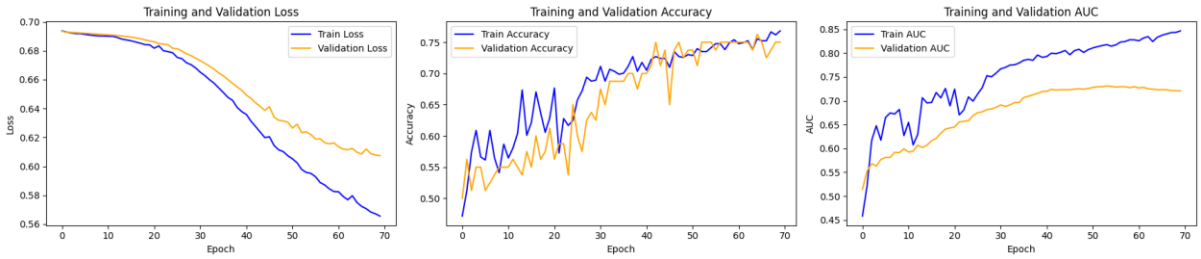
Figure 83: Process map for script `01.03.04.01_KS_GCN_upstream_methylation.ipynb`.

Results



Test Accuracy: 0.7250 (72.5%)
Test AUC: 0.8287 (82.87%)
Figure 84: binary classification model for α-carbon methylation state loss and accuracy over a 200-epoch training period for validation and training data sets.

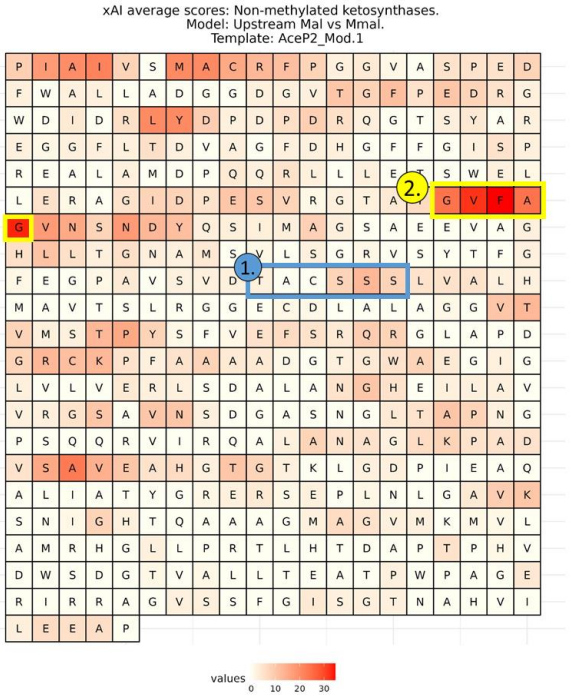
The training and validation data in figure 84 presents a model that is learning something applicable to the underlying categories for the first 100 epoch. The degree of confidence improvement over these epochs is very low, as seen by the shallow decrease of loss from ~0.7 down to ~0.6 in epoch 100.

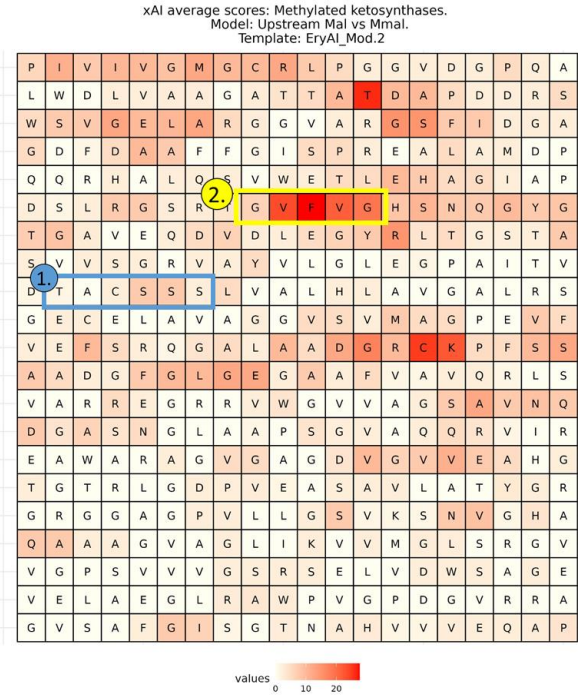


Test Accuracy: 0.7125 (71.25%)
Test AUC: 0.8225 (82.25%)
Figure 85: Binary classification model for α-carbon methylation state loss and accuracy over a 200-epoch training period for validation and training data sets.

The 100-epoch model was run through the explainer loop. Results are displayed below.

xAI results





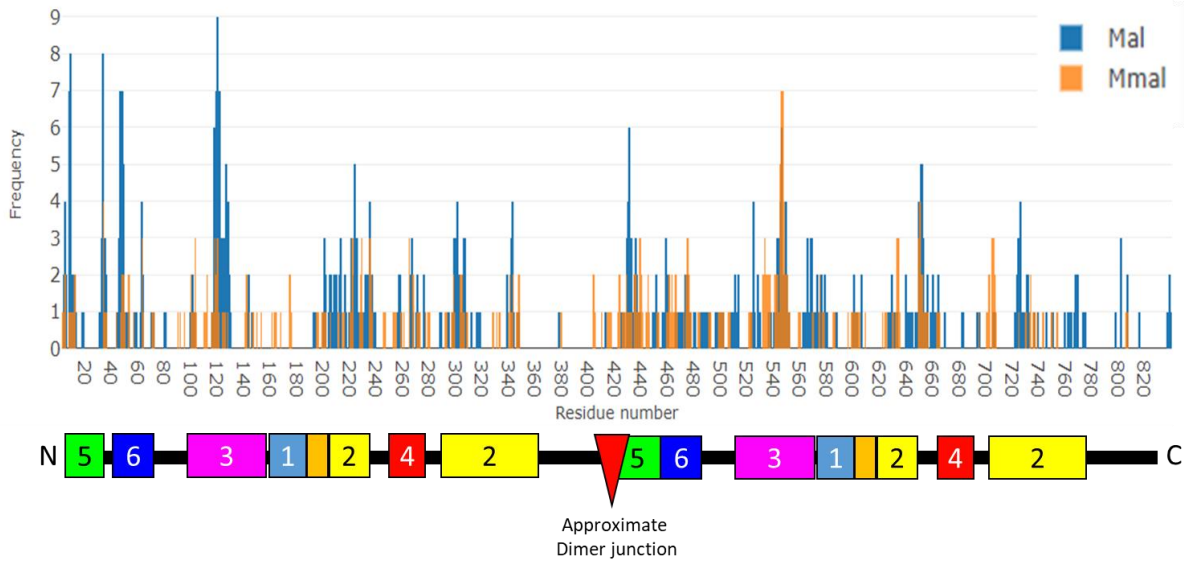


Figure 86: Word search diagrams depicting averaged explainer results (n=80) for a-methylated polyketide accepting ketosynthases, aligned to AceP2_Mod.1 (non-methylated) and AeEryAI_Mod.2 (methylated). Raster pattern follows left to right, top to bottom, N terminal to C terminal. Heat map is red hot relative to maximal scores in averages – colours are not comparable between figures. Region 1 (blue) highlights the TACSSS motif, which contains the active site cysteine. Note that the heat map scale bars differ between the two diagrams. **Bottom:** Alignment-agnostic histogram displaying node (residue) frequencies detected by xAI. This graph has been constructed with raw values and has not been condensed to a monomeric view, as shown in the word searches. Below the histogram is an approximate map of nodes to regional diagrams. Note this is an approximation because the nodes are not aligned – any insertion/deletions in the sequences will move residues out of relative frame. This approximation will be more inaccurate across the dimerization junction due to variance in protein length (see figure 22B – ketosynthase monomers are mostly between 420 and 430 amino acids in length, so this will create an approximate range of +/- 20 at the end of the second dimer).

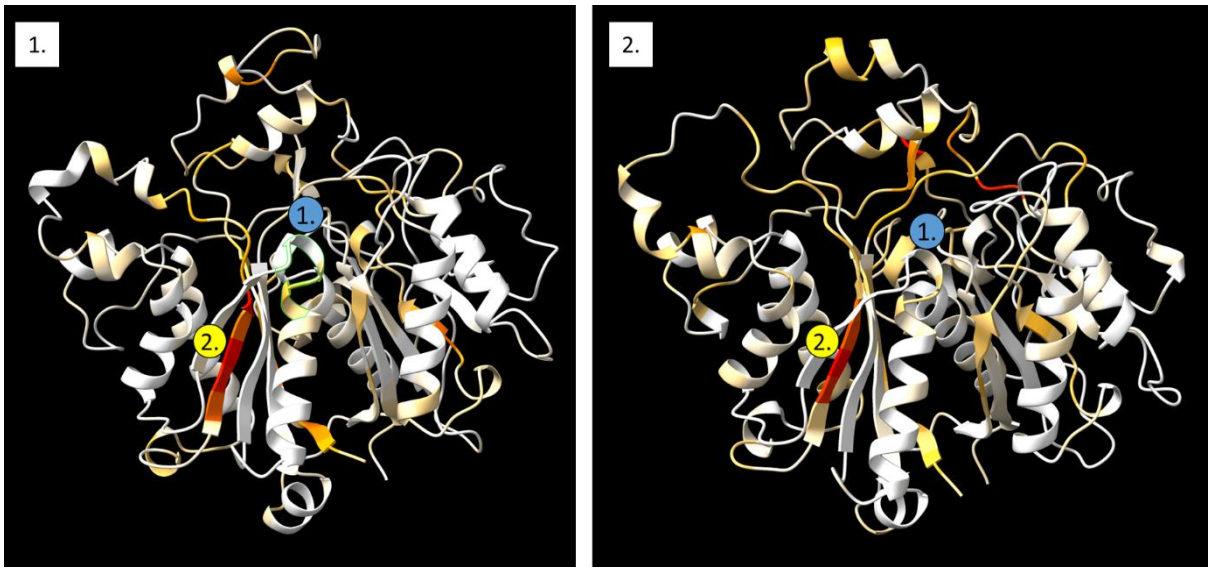


Figure 87: Explainer averages, heat-mapped structures of AceP2_Mod.1 (non-methylated, 1) and EryAI_Mod.2 (non-methylated, 2) ketosynthase structures, depicted as monomers view from the dimerization surface. Region 1 marker is place over the active site cysteine.

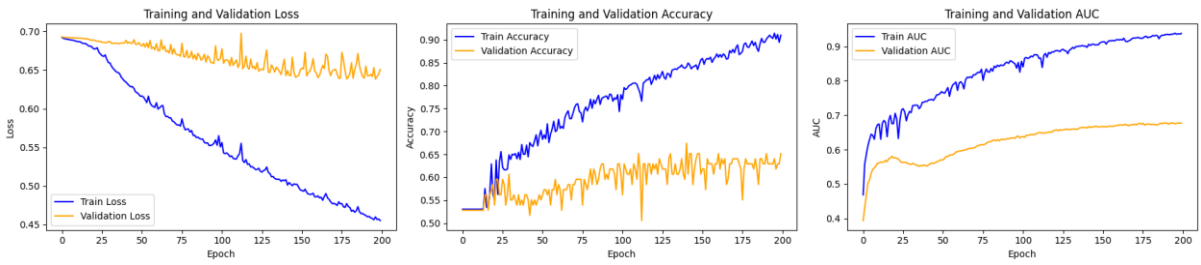
Binary classifier for the prediction of extension unit specificity (mal or mmal)

Script: `01.03.04.02_KS_methylation_adjoining.ipynb`
Packages: OS, torch, graphein, pandas, re, functools, sklearn, torch_geometric, matplotlib, numpy,
Python version: 3.9, run as Jupyter notebook.
OS: Ubuntu 20.04.6 LTS

This script investigates whether KSs can predict whether the extension unit it uses in the condensation reaction is mal or mmal.

This script is identical to the AT substrate script `01.03.01_AT_GCN_binary.ipynb`. The only difference is that ketosynthase dimers are being used as the input structures for graph generation, and that the k-nearest neighbour graph generation method draws 4 neighbours instead of 3.

Results



Test Accuracy: 0.6517 (65.17%)
Test AUC: 0.7285 (72.85%)
Figure 88: binary classification model for incoming extension unit type (mmal or mal). Graphs display loss and accuracy over a 200-epoch training period for validation and training data sets.

Loss and accuracy in figure 88 do not trend together, indicating that the model is not learning meaningful representations of the underlying categories. This indicates that the model is not able to detect differences in the ketosynthase structures that pertain to the type of extension unit.

Conclusions on methylation state models

Methylation state introduced by the relative upstream AT domain is detectable, to a degree, in ketosynthase structures. This would suggest that a ketosynthase has properties that can detect whether the correct extension unit, in a Mal or Mmal scenario, was used in the incoming polyketides previous condensation reaction.

The explainer averages are noisy, as seen in figure 86, suggesting that there is not a clearly defined motif/residue that determines

We find no evidence that the relative downstream ketosynthase has learnable features specific to the substrate being passed to the ketosynthase for condensation. This suggests that the ketosynthase only proofreads the incoming polyketide prior to extension.

These findings support the revised borders definition[1] of polyketide synthase modules, placing the ketosynthase at the end of the module.

Discussion of future work ideas:

1. Can length of polyketide be predicted?
    - As we describe in this body of work, ketosynthases have structural properties that associate with chemical properties imparted by enzymatic domains upstream. This body of work is not exhaustive for substrate properties however. In natural systems, ketosynthases receive polyketide substrates that range from 0 to 30 (Ref[2]) condensation reactions occurring upstream. It is conceivable that ketosynthases could have modified architectures to accommodate a potentially 30-fold difference in substrate size. Returning to the analogy in figure 82, we are asking if the length of the key is detectable in the lock. If they do discriminate for length, this is important to understand for purposes of designing new to nature mPKS pathways.
    - We propose that GCNs could be used in a similar approach to that used for the binary classification tasks, where length is used as a label. There are two routes of different sensitivity that we could employ here:
        1. A binary classifier examining if the incoming polyketide is towards the start or end of the pathway. Effectively, asking the model to predict if the polyketide short or long.
        2. A single neuron, continuous output, predicting the condensation step count on the polyketide substrate. This may be best capped to the modal number of modules per path (which is eight for our datasets, see figure 14) for purposes of balancing training data.
    - Comparatively, the binary classifier option would have best chance of detecting learnable features associating with length, because of the larger available data from rounding into categories of short and long. This would give an easier answer to the question of "is length a factor to ketosynthase substrate specificity?" The continuous output option, I would hazard a guess, would be more difficult to get reliable results from, given the smaller pool of training data per condensation step.
2. Are any features from the SMILES predictable?
    - SMILES (simplified molecular-input line-entry) are a standardised descriptor of chemical structures. As part of the webscraping in section 2.2.1, we extracted SMILES for each condensation step each mPKS path (this was what we used to arrange genes sequentially into paths – the SMILES get longer at each condensation).
    - SMILES inherently contain features of the polyketide product, which we demonstrated in this body of work to be associable with features of the ketosynthase structure. We suggest a feature association strategy could be employed on bipartite graph networks of protein structures and SMILES. The aim is to establish if the protein structure can predict features of the SMILES graph.
3. Can Natural Language Processing (NLP) operate in this space?
    - Protein sequences can be considered as almost analogous to a natural language (e.g. English, French, etc.). Naturally evolved proteins can be represented by an alphabet of 20 letters and can contain syntactic modular elements, akin to words and phrases. We can identify varying levels of linguistic structure in protein, such as enzymatic motifs, secondary/tertiary common fold patterns such as beta-barrels and hot-dog folds. These

examples may be likened to words, sentences and paragraphs, which make up The Complete Works of Shakespeare that is a proteome. The field of Natural Language Processing seeks to use statistical models to enable computers to interpret such natural languages[128].

We found in this present chapter that changing the data organisation from voxel cubes to graph networks led to better predictive models. Perhaps a NLP approach might find new insights within the datasets generated here.

References

(1) Keatinge-Clay, A. T. Polyketide Synthase Modules Redefined. *Angewandte Chemie-International Edition* **2017**, *56* (17), 4658-4660, Editorial Material. DOI: 10.1002/anie.201701281.
(2) Zhang, L. H.; Hashimoto, T.; Qin, B.; Hashimoto, J.; Kozone, I.; Kawahara, T.; Okada, M.; Awakawa, T.; Ito, T.; Asakawa, Y.; et al. Characterization of Giant Modular PKSs Provides Insight into Genetic Mechanism for Structural Diversification of Aminopolyol Polyketides. *Angewandte Chemie-International Edition* **2017**, *56* (7), 1740-1745, Article. DOI: 10.1002/anie.201611371.
(3) Walsh, C. T. Polyketide and nonribosomal peptide antibiotics: modularity and versatility. *Science* **2004**, *303* (5665), 1805-1810. DOI: 10.1126/science.1094318.
(4) Larsson, D. G. J.; Flach, C.-F. Antibiotic resistance in the environment. *Nature Reviews Microbiology* **2022**, *20* (5), 257-269. DOI: 10.1038/s41579-021-00649-x (acccessed 2024-04-29T16:16:06).
(5) Miethke, M.; Pieroni, M.; Weber, T.; Brönstrup, M.; Hammann, P.; Halby, L.; Arimondo, P. B.; Glaser, P.; Aigle, B.; Bode, H. B.; et al. Towards the sustainable discovery and development of new antibiotics. *Nature Reviews Chemistry* **2021**, *5* (10), 726-749. DOI: 10.1038/s41570-021-00313-1 (acccessed 2024-04-29T16:58:22).
(6) Klaus, M.; Buyachuihan, L.; Grininger, M. Ketosynthase Domain Constrains the Design of Polyketide Synthases. *Acs Chemical Biology* **2020**, *15* (9), 2422-2432, Article. DOI: 10.1021/acschembio.0c00405.
(7) Murphy, A. C.; Hong, H.; Vance, S.; Broadhurst, R. W.; Leadlay, P. F. Broadening substrate specificity of a chain-extending ketosynthase through a single active-site mutation. *Chemical Communications* **2016**, *52* (54), 8373-8376, Article. DOI: 10.1039/c6cc03501a.
(8) Hirsch, M.; Fitzgerald, B. J.; Keatinge-Clay, A. T. How cis-Acyltransferase Assembly-Line Ketosynthases Gatekeep for Processed Polyketide Intermediates. *ACS Chemical Biology* **2021**, *16* (11), 2515-2526. DOI: 10.1021/acschembio.1c00598 (acccessed 2023-01-20T11:21:35).
(9) Keatinge-Clay, A. T. The structures of type I polyketide synthases. *Natural Product Reports* **2012**, *29* (10), 1050-1073, Review. DOI: 10.1039/c2np20019h.
(10) Stinear, T. P.; Mve-Obiang, A.; Small, P. L. C.; Frigui, W.; Pryor, M. J.; Brosch, R.; Jenkin, G. A.; Johnson, P. D. R.; Davies, J. K.; Lee, R. E.; et al. Giant plasmid-encoded polyketide synthases produce the macrolide toxin of

Mycobacterium ulcerans. *Proceedings of the National Academy of Sciences* **2004**, *101* (5), 1345-1349. DOI: 10.1073/pnas.0305877101 (acccessed 2024-03-18T13:11:47).
(11) Khosla, C.; Tang, Y.; Chen, A. Y.; Schnarr, N. A.; Cane, D. E. Structure and Mechanism of the 6-Deoxyerythronolide B Synthase. *Annual review of biochemistry.* **2007**, *76* (1), 195-221. DOI: 10.1146/annurev.biochem.76.053105.093515.
(12) Weissman, K. J. Chapter 1 Introduction to Polyketide Biosynthesis. *Methods in Enzymology* **2009**, *459*, 3-16. DOI: 10.1016/S0076-6879(09)04601-1.
(13) Caffrey, P.; Lynch, S.; Flood, E.; Finnan, S.; Oliynyk, M. Amphotericin biosynthesis in Streptomyces nodosus: deductions from analysis of polyketide synthase and late genes. *Chemistry & Biology* **2001**, *8* (7), 713-723, Article. DOI: 10.1016/s1074-5521(01)00046-1.
(14) Schwecke, T.; Aparicio, J. F.; Molnár, I.; König, A.; Khaw, L. E.; Haydock, S. F.; Oliynyk, M.; Caffrey, P.; Cortés, J.; Lester, J. B. The biosynthetic gene cluster for the polyketide immunosuppressant rapamycin. *Proceedings of the National Academy of Sciences* **1995**, *92* (17), 7839-7843. DOI: 10.1073/pnas.92.17.7839 (acccessed 2024-03-18T13:38:33).

(15) Li, S.; Yang, B.; Tan, G.-Y.; Ouyang, L.-M.; Qiu, S.; Wang, W.; Xiang, W.; Zhang, L. Polyketide pesticides from actinomycetes. *Current opinion in biotechnology*. **2021**, *69*, 299-307. DOI: 10.1016/j.copbio.2021.05.006.
(16) Zargar, A.; Bailey, C. B.; Haushalter, R. W.; Eiben, C. B.; Katz, L.; Keasling, J. D. Leveraging microbial biosynthetic pathways for the generation of 'drop-in' biofuels. *Current opinion in biotechnology*. **2017**, *45*, 156-163. DOI: 10.1016/j.copbio.2017.03.004.
(17) Perlis, R. H.; Lunz Trujillo, K.; Green, J.; Safarpour, A.; Druckman, J. N.; Santillana, M.; Ognyanova, K.; Lazer, D. Misinformation, Trust, and Use of Ivermectin and Hydroxychloroquine for COVID-19. *JAMA Health Forum* **2023**, *4* (9), e233257. DOI: 10.1001/jamahealthforum.2023.3257 (acccessed 2024-03-18T14:23:49).
(18) Blin, K.; Shaw, S.; Augustijn, H. E.; Reitz, Z. L.; Biermann, F.; Alanjary, M.; Fetter, A.; Terlouw, B. R.; Metcalf, W. W.; Helfrich, E. J. N.; et al. antiSMASH 7.0: new and improved predictions for detection, regulation, chemical structures and visualisation. *Nucleic Acids Research* **2023**, *51* (W1), W46-W50. DOI: 10.1093/nar/gkad344 (acccessed 2024-01-24T14:27:38).
(19) Ridley, C. P.; Lee, H. Y.; Khosla, C. Evolution of polyketide synthases in bacteria. *Proceedings of the National Academy of Sciences* **2008**, *105* (12), 4595-4600. DOI: 10.1073/pnas.0710107105 (acccessed 2024-03-18T14:47:39).
(20) Klaus, M.; Ostrowski, M. P.; Austerjost, J.; Robbins, T.; Lowry, B.; Cane, D. E.; Khosla, C. Protein-Protein Interactions, Not Substrate Recognition, Dominate the Turnover of Chimeric Assembly Line Polyketide Synthases. *Journal of Biological Chemistry* **2016**, *291* (31), 16404-16415, Article. DOI: 10.1074/jbc.M116.730531.
(21) Staunton, J.; Weissman, K. J. Polyketide biosynthesis: a millennium review. *Natural product reports /* **2001**, *18* (4), 380-416. DOI: 10.1039/A909079G.
(22) Menzella, H. G.; Carney, J. R.; Santi, D. V. Rational design and assembly of synthetic trimodular polyketide synthases. *Chemistry & Biology* **2007**, *14* (2), 143-151, Article. DOI: 10.1016/j.chembiol.2006.12.002.
(23) Menzella, H. G.; Reid, R.; Carney, J. R.; Chandran, S. S.; Reisinger, S. J.; Patel, K. G.; Hopwood, D. A.; Santi, D. V. Combinatorial polyketide biosynthesis by de novo design and rearrangement of modular polyketide synthase genes. *Nature Biotechnology* **2005**, *23* (9), 1171-1176, Article. DOI: 10.1038/nbt1128.
(24) Dutta, S.; Whicher, J. R.; Hansen, D. A.; Hale, W. A.; Chemler, J. A.; Congdon, G. R.; Narayan, A. R. H.; Hakansson, K.; Sherman, D. H.; Smith, J. L.; et al. Structure of a modular polyketide synthase. *Nature* **2014**, *510* (7506), 512-+, Article. DOI: 10.1038/nature13423.
(25) Whicher, J. R.; Dutta, S.; Hansen, D. A.; Hale, W. A.; Chemler, J. A.; Dosey, A. M.; Narayan, A. R. H.; Hakansson, K.; Sherman, D. H.; Smith, J. L.; et al. Structural rearrangements of a polyketide synthase module during its catalytic cycle. *Nature* **2014**, *510* (7506), 560-+, Article. DOI: 10.1038/nature13409.
(26) Kapur, S.; Lowry, B.; Yuzawa, S.; Kenthirapalan, S.; Chen, A. Y.; Cane, D. E.; Khosla, C. Reprogramming a module of the 6-deoxyerythronolide B synthase for iterative chain elongation. *Proc Natl Acad Sci U S A* **2012**, *109* (11), 4110-4115. DOI: 10.1073/pnas.1118734109.
(27) Dunn, B. J.; Khosla, C. Engineering the acyltransferase substrate specificity of assembly line polyketide synthases. *Journal of The Royal Society Interface* **2013**, *10* (85), 20130297. DOI: 10.1098/rsif.2013.0297 (acccessed 2024-01-24T10:40:30).
(28) Jenner, M.; Frank, S.; Kampa, A.; Kohlhaas, C.; Pöplau, P.; Briggs, G. S.; Piel, J.; Oldham, N. J. Substrate Specificity in Ketosynthase Domains from trans- AT Polyketide Synthases. *Angewandte Chemie International Edition in English* **2013**, *52* (4), 1143-1147. DOI: 10.1002/anie.201207690.
(29) Nguyen, T.; Ishida, K.; Jenke-Kodama, H.; Dittmann, E.; Gurgui, C.; Hochmuth, T.; Taudien, S.; Platzer, M.; Hertweck, C.; Piel, J. Exploiting the mosaic structure of trans-acyltransferase polyketide synthases for natural product discovery and pathway dissection. *Nature Biotechnology* **2008**, *26* (2), 225-233. DOI: 10.1038/nbt1379 (acccessed 2024-03-18T15:29:42).
(30) Tsai, S. C.; Miercke, L. J. W.; Krucinski, J.; Gokhale, R.; Chen, J. C. H.; Foster, P. G.; Cane, D. E.; Khosla, C.; Stroud, R. M. Crystal structure of the macrocycle-forming thioesterase domain of the erythromycin polyketide synthase: Versatility from a unique substrate channel. *Proceedings of the National Academy of Sciences of the United States of America* **2001**, *98* (26), 14808-14813, Article. DOI: 10.1073/pnas.011399198.

(31) Pan, H.; Tsai, S.-C.; Meadows, E. S.; Miercke, L. J. W.; Keatinge-Clay, A. T.; O'Connell, J.; Khosla, C.; Stroud, R. M. Crystal Structure of the Priming β-Ketosynthase from the R1128 Polyketide Biosynthetic Pathway. *Structure* **2002**, *10* (11), 1559-1568. DOI: 10.1016/s0969-2126(02)00889-4 (acccessed 2024-01-22T13:43:51).
(32) Tang, Y. Y.; Chen, A. Y.; Kim, C. Y.; Cane, D. E.; Khosla, C. Structural and mechanistic analysis of protein interactions in module 3 of the 6-deoxyerythronolide B synthase. *Chemistry & Biology* **2007**, *14* (8), 931-943, Article. DOI: 10.1016/j.chembiol.2007.07.012.
(33) Edwards, A. L.; Matsui, T.; Weiss, T. M.; Khosla, C. Architectures of Whole-Module and Bimodular Proteins from the 6-Deoxyerythronolide B Synthase. *Journal of Molecular Biology* **2014**, *426* (11), 2229-2245. DOI: 10.1016/j.jmb.2014.03.015 (acccessed 2024-01-22T16:42:46).
(34) Robbins, T.; Kapilivsky, J.; Cane, D. E.; Khosla, C. Roles of Conserved Active Site Residues in the Ketosynthase Domain of an Assembly Line Polyketide Synthase. *Biochemistry* **2016**, *55* (32), 4476-4484, Article. DOI: 10.1021/acs.biochem.6b00639.
(35) Robbins, T.; Liu, Y. C.; Cane, D. E.; Khosla, C. Structure and mechanism of assembly line polyketide synthases. *Current Opinion in Structural Biology* **2016**, *41*, 10-18, Article. DOI: 10.1016/j.sbi.2016.05.009.
(36) Guzman, K. M.; Cogan, D. P.; Brodsky, K. L.; Soohoo, A. M.; Li, X.; Sevillano, N.; Mathews, I. I.; Nguyen, K. P.; Craik, C. S.; Khosla, C. Discovery and Characterization of Antibody Probes of Module 2 of the 6-Deoxyerythronolide B Synthase. *Biochemistry* **2023**, *62* (11), 1589-1593. DOI: 10.1021/acs.biochem.3c00156.
(37) Cogan, D. P.; Zhang, K.; Li, X.; Li, S.; Pintilie, G. D.; Roh, S.-H.; Craik, C. S.; Chiu, W.; Khosla, C. Mapping the catalytic conformations of an assembly-line polyketide synthase module. *Science.* **2021**, *374* (6568), 729-734. DOI: 10.1126/science.abi8358.
(38) Dell, M.; Tran, M. A.; Capper, M. J.; Sundaram, S.; Fiedler, J.; Koehnke, J.; Hellmich, U. A.; Hertweck, C. Trapping of a Polyketide Synthase Module After C-C Bond Formation Reveals Transient Acyl Carrier Domain Interactions. *Angewandte Chemie International Edition* **2023**. DOI: 10.1002/anie.202315850 (acccessed 2024-01-19T15:35:17).
(39) Tang, Y. Y.; Kim, C. Y.; Mathews, II; Cane, D. E.; Khosla, C. The 2.7-angstrom crystal structure of a 194-kDa homodimeric fragment of the 6-deoxyerythronolide B synthase. *Proceedings of the National Academy of Sciences of the United States of America* **2006**, *103* (30), 11124-11129, Article. DOI: 10.1073/pnas.0601924103.
(40) Lohman, J. R.; Ma, M.; Osipiuk, J.; Nocek, B.; Kim, Y. C.; Chang, C. S.; Cuff, M.; Mack, J.; Bigelow, L.; Li, H.; et al. Structural and evolutionary relationships of "AT-less" type I polyketide synthase ketosynthases. *Proceedings of the National Academy of Sciences of the United States of America* **2015**, *112* (41), 12693-12698, Article. DOI: 10.1073/pnas.1515460112.
(41) Bagde, S. R.; Mathews, I. I.; Fromme, J. C.; Kim, C.-Y. Modular polyketide synthase contains two reaction chambers that operate asynchronously. *Science.* **2021**, *374* (6568), 723-729. DOI: 10.1126/science.abi8532.
(42) Broadhurst, R. W.; Nietlispach, D.; Wheatcroft, M. P.; Leadlay, P. F.; Weissman, K. J. The structure of docking domains in modular polyketide synthases. *Chemistry & Biology* **2003**, *10* (8), 723-731, Article. DOI: 10.1016/s1074-5521(03)00156-x.
(43) Kapur, S.; Chen, A. Y.; Cane, D. E.; Khosla, C. Molecular recognition between ketosynthase and acyl carrier protein domains of the 6-deoxyerythronolide B synthase. *Proceedings of the National Academy of Sciences of the United States of America* **2010**, *107* (51), 22066-22071, Article. DOI: 10.1073/pnas.1014081107.
(44) Keatinge-Clay, A. T.; Miyazawa, T.; Zhang, J.; Ray, K. A.; Lutgens, J. D.; Bista, R.; Lin, S. N. Crystal structures reveal the framework ofcis-acyltransferase modular polyketide synthases. Cold Spring Harbor Laboratory: 2023.
(45) Gokhale, R. S.; Lau, J.; Cane, D. E.; Khosla, C. Functional Orientation of the Acyltransferase Domain in a Module of the Erythromycin Polyketide Synthase. *Biochemistry* **1998**, *37* (8), 2524-2528. DOI: 10.1021/bi971887n.
(46) Kao, C. M.; Pieper, R.; Cane, D. E.; Khosla, C. Evidence for Two Catalytically Independent Clusters of Active Sites in a Functional Modular Polyketide Synthase. *Biochemistry* **1996**, *35* (38), 12363-12368. DOI: 10.1021/bi9616312.

(47) Mirdita, M.; Schütze, K.; Moriwaki, Y.; Heo, L.; Ovchinnikov, S.; Steinegger, M. ColabFold: making protein folding accessible to all. *Nature Methods* **2022**, *19* (6), 679-682. DOI: 10.1038/s41592-022-01488-1 (acccessed 2023-04-17T14:57:41).
(48) Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596* (7873), 583-589. DOI: 10.1038/s41586-021-03819-2 (acccessed 2023-02-14T13:17:38).
(49) Evans, R.; O'Neill, M.; Pritzel, A.; Antropova, N.; Senior, A.; Green, T.; Žídek, A.; Bates, R.; Blackwell, S.; Yim, J.; et al. Protein complex prediction with AlphaFold-Multimer. Cold Spring Harbor Laboratory: 2021.
(50) Nordberg, H.; Cantor, M.; Dusheyko, S.; Hua, S.; Poliakov, A.; Shabalov, I.; Smirnova, T.; Grigoriev, I. V.; Dubchak, I. The genome portal of the Department of Energy Joint Genome Institute: 2014 updates. *Nucleic Acids Research* **2014**, *42* (D1), D26-D31. DOI: 10.1093/nar/gkt1069 (acccessed 2024-01-23T16:49:56).
(51) Sayers, E. W.; Bolton, E. E.; Brister, J. R.; Canese, K.; Chan, J.; Comeau, C., Donald; Connor, R.; Funk, K.; Kelly, C.; Kim, S.; et al. Database resources of the national center for biotechnology information. *Nucleic Acids Research* **2022**, *50* (D1), D20-D26. DOI: 10.1093/nar/gkab1112 (acccessed 2024-01-23T16:48:25).
(52) Kautsar, S. A.; Blin, K.; Shaw, S.; Navarro-Munoz, J. C.; Terlouw, B. R.; van der Hooft, J. J. J.; van Santen, J. A.; Tracanna, V.; Duran, H. G. S.; Andreu, V. P.; et al. MIBiG 2.0: a repository for biosynthetic gene clusters of known function. *Nucleic Acids Research* **2020**, *48* (D1), D454-D458, Article. DOI: 10.1093/nar/gkz882.
(53) Blin, K.; Shaw, S.; Steinke, K.; Villebro, R.; Ziemert, N.; Lee, S. Y.; Medema, M. H.; Weber, T. antiSMASH 5.0: updates to the secondary metabolite genome mining pipeline. *Nucleic Acids Research* **2019**, *47* (W1), W81-W87, Article. DOI: 10.1093/nar/gkz310.
(54) Blin, K.; Medema, M. H.; Kottmann, R.; Lee, S. Y.; Weber, T. The antiSMASH database, a comprehensive database of microbial secondary metabolite biosynthetic gene clusters. *Nucleic Acids Research* **2017**, *45* (D1), D555-D559, Article. DOI: 10.1093/nar/gkw960.
(55) Blin, K.; Wolf, T.; Chevrette, M. G.; Lu, X. W.; Schwalen, C. J.; Kautsar, S. A.; Duran, H. G. S.; Santos, E.; Kim, H. U.; Nave, M.; et al. antiSMASH 4.0-improvements in chemistry prediction and gene cluster boundary identification. *Nucleic Acids Research* **2017**, *45* (W1), W36-W41, Article. DOI: 10.1093/nar/gkx319.
(56) Weber, T.; Blin, K.; Duddela, S.; Krug, D.; Kim, H. U.; Bruccoleri, R.; Lee, S. Y.; Fischbach, M. A.; Muller, R.; Wohlleben, W.; et al. antiSMASH 3.0-a comprehensive resource for the genome mining of biosynthetic gene clusters. *Nucleic Acids Research* **2015**, *43* (W1), W237-W243, Article. DOI: 10.1093/nar/gkv437.
(57) Blin, K.; Medema, M. H.; Kazempour, D.; Fischbach, M. A.; Breitling, R.; Takano, E.; Weber, T. antiSMASH 2.0-a versatile platform for genome mining of secondary metabolite producers. *Nucleic Acids Research* **2013**, *41* (W1), W204-W212, Article. DOI: 10.1093/nar/gkt449.
(58) Medema, M. H.; Blin, K.; Cimermancic, P.; de Jager, V.; Zakrzewski, P.; Fischbach, M. A.; Weber, T.; Takano, E.; Breitling, R. antiSMASH: rapid identification, annotation and analysis of secondary metabolite biosynthesis gene clusters in bacterial and fungal genome sequences. *Nucleic Acids Research* **2011**, *39*, W339-W346, Article. DOI: 10.1093/nar/gkr466.
(59) Blin, K.; Shaw, S.; Kloosterman, A. M.; Charlop-Powers, Z.; Wezel, V., Gilles P; Medema, H., Marnix; Weber, T. antiSMASH 6.0: improving cluster detection and comparison capabilities. *Nucleic Acids Research* **2021**, *49* (W1), W29-W35. DOI: 10.1093/nar/gkab335 (acccessed 2024-01-24T14:28:24).
(60) Eng, C. H.; Backman, T. W. H.; Bailey, C. B.; Magnan, C.; Martin, H. G.; Katz, L.; Baldi, P.; Keasling, J. D. ClusterCAD: a computational platform for type I modular polyketide synthase design. *Nucleic Acids Research* **2018**, *46* (D1), D509-D515, Article. DOI: 10.1093/nar/gkx893.
(61) Lin, Z.; Akin, H.; Rao, R.; Hie, B.; Zhu, Z.; Lu, W.; Smetanin, N.; Verkuil, R.; Kabeli, O.; Shmueli, Y.; et al. Evolutionary-scale prediction of atomic level protein structure with a language model. Cold Spring Harbor Laboratory: 2022.

(62) Bryant, P.; Pozzati, G.; Zhu, W.; Shenoy, A.; Kundrotas, P.; Elofsson, A. Predicting the structure of large protein complexes using AlphaFold and Monte Carlo tree search. *Nature Communications* **2022**, *13* (1). DOI: 10.1038/s41467-022-33729-4 (acccessed 2024-01-23T16:55:35).
(63) Frackowiak, R. S. *Human brain function*; Elsevier, 2004.
(64) Dictionary, O. E. machine learning, n. In *Oxford English Dictionary*, Oxford University Press: 2024.
(65) El Naqa, I.; Murphy, M. J. What Is Machine Learning? In *Machine Learning in Radiation Oncology*, Springer International Publishing, 2015; pp 3-11.
(66) TensorFlow: A system for large-scale machine learning.
(67) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12* (null), 2825–2830.
(68) Rosenblatt, F. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*; Spartan books Washington, DC, 1962.
(69) Alzubaidi, L.; Zhang, J.; Humaidi, A. J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M. A.; Al-Amidie, M.; Farhan, L. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data* **2021**, *8* (1). DOI: 10.1186/s40537-021-00444-8 (acccessed 2024-01-23T18:18:59).
(70) Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S. R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Networks* **2019**, *111*, 47-63. DOI: 10.1016/j.neunet.2018.12.002 (acccessed 2024-01-26T16:18:06).
(71) Linardatos, P.; Papastefanopoulos, V.; Kotsiantis, S. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* **2020**, *23* (1), 18. DOI: 10.3390/e23010018 (acccessed 2024-01-25T13:20:30).
(72) Chan, Y. A.; Podevels, A. M.; Kevany, B. M.; Thomas, M. G. Biosynthesis of polyketide synthase extender units. *Nat. Prod. Rep.* **2009**, *26* (1), 90-114. DOI: 10.1039/b801658p (acccessed 2024-01-24T10:42:19).
(73) Reeves, C. D.; Murli, S.; Ashley, G. W.; Piagentini, M.; Hutchinson, C. R.; McDaniel, R. Alteration of the Substrate Specificity of a Modular Polyketide Synthase Acyltransferase Domain through Site-Specific Mutations. *Biochemistry* **2001**, *40* (51), 15464-15470. DOI: 10.1021/bi015864r.
(74) Sundermann, U.; Bravo-Rodriguez, K.; Klopries, S.; Kushnir, S.; Gomez, H.; Sanchez-Garcia, E.; Schulz, F. Enzyme-Directed Mutasynthesis: A Combined Experimental and Theoretical Approach to Substrate Recognition of a Polyketide Synthase. *Acs Chemical Biology* **2013**, *8* (2), 443-450, Article. DOI: 10.1021/cb300505w.
(75) Wang, H.; Liang, J.; Yue, Q.; Li, L.; Shi, Y.; Chen, G.; Li, Y.-Z.; Bian, X.; Zhang, Y.; Zhao, G.; et al. Engineering the acyltransferase domain of epothilone polyketide synthase to alter the substrate specificity. *Microbial Cell Factories* **2021**, *20* (1). DOI: 10.1186/s12934-021-01578-3 (acccessed 2024-01-26T17:41:28).
(76) Petković, H.; Sandmann, A.; Challis, I. R.; Hecht, H.-J.; Silakowski, B.; Low, L.; Beeston, N.; Kuščer, E.; Garcia-Bernardo, J.; Leadlay, P. F.; et al. Substrate specificity of the acyl transferase domains of EpoC from the epothilone polyketide synthase. *Organic & biomolecular chemistry.* **2008**, *6* (3), 500-506. DOI: 10.1039/B714804F.
(77) Bunkoczi, G.; Misquitta, S.; Wu, X.; Lee, W. H.; Rojkova, A.; Kochan, G.; Kavanagh, K. L.; Oppermann, U.; Smith, S. Structural Basis for Different Specificities of Acyltransferases Associated with the Human Cytosolic and Mitochondrial Fatty Acid Synthases. *Chemistry & biology.* **2009**, *16* (6), 667-675. DOI: 10.1016/j.chembiol.2009.04.011.
(78) Paysan-Lafosse, T.; Blum, M.; Chuguransky, S.; Grego, T.; Pinto, B. L.; Salazar, A., Gustavo; Bileschi, L., Maxwell; Bork, P.; Bridge, A.; Colwell, L.; et al. InterPro in 2022. *Nucleic Acids Research* **2023**, *51* (D1), D418-D427. DOI: 10.1093/nar/gkac993 (acccessed 2024-01-25T16:45:53).
(79) Terlouw, B. R.; Blin, K.; Navarro-Muñoz, J. C.; Avalon, N. E.; Chevrette, M. G.; Egbert, S.; Lee, S.; Meijer, D.; Recchia, J. J., Michael; Reitz, L., Zachary; et al. MIBiG 3.0: a community-driven effort to annotate experimentally validated biosynthetic gene clusters. *Nucleic Acids Research* **2023**, *51* (D1), D603-D610. DOI: 10.1093/nar/gkac1049 (acccessed 2024-01-24T14:34:53).
(80) Wickham, H. rvest: Easily Harvest (Scrape) Web Pages. 2024.

(81) Perepolkin, D. polite: Be Nice on the Web. https://github.com/dmi3kno/polite, https://dmi3kno.github.io/polite/: 2023.
(82) Wickham, H.; Averick, M.; Bryan, J.; Chang, W.; McGowan, L. D. A.; François, R.; Grolemund, G.; Hayes, A.; Henry, L.; Hester, J.; et al. Welcome to the tidyverse. *Journal of Open Source Software* **2019**, *4* (43), 1686.
(83) Becker, G.; Lawrence, M. genbankr: Parsing GenBank files into semantically useful objects. 2022.
(84) Wickham, H. stringr: Simple, Consistent Wrappers for Common String Operations. https://github.com/tidyverse/stringr: 2023.
(85) Pagès, H.; Aboyoun, P.; Gentleman, R.; DebRoy, S. Biostrings: Efficient manipulation of biological strings. https://bioconductor.org/packages/Biostrings: 2024.
(86) Wickham, H.; François, R.; Henry, L.; Müller, K.; Vaughan, D. dplyr: A Grammar of Data Manipulation. https://github.com/tidyverse/dplyr: 2023.
(87) Charif, D.; Lobry, J. R. *SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis*; Springer Verlag, 2007. DOI: 207-232.
(88) Bakan, A.; Meireles, L. M.; Bahar, I. ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics* **2011**, *27* (11), 1575-1577.
(89) Grant, B. J.; Skjærven, L.; Yao, X. Q. The <scp>Bio3D</scp> packages for structural bioinformatics. *Protein Science* **2021**, *30* (1), 20-30. DOI: 10.1002/pro.3923 (acccessed 2024-03-20T15:55:15).
(90) Jenke-Kodama, H.; Sandmann, A.; Muller, R.; Dittmann, E. Evolutionary implications of bacterial polyketide synthases. *Molecular Biology and Evolution* **2005**, *22* (10), 2027-2039, Article. DOI: 10.1093/molbev/msi193.
(91) Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural computation.* **2017**, *29* (9), 2352-2449. DOI: 10.1162/neco_a_00990.
(92) Mart\'n~Abadi and Ashish~Agarwal and Paul~Barham and Eugene~Brevdo and Zhifeng~Chen and Craig~Citro and Greg, S. C. a. A. D. a. J. D. a. M. TensorFlow : Large-Scale Machine Learning on Heterogeneous Systems. 2015.
(93) Chollet, F. c. a. A. J. J.; et al. R Interface to Keras. 2017.
(94) Baoyuan, L.; Min, W.; Foroosh, H.; Tappen, M.; Penksy, M. Sparse Convolutional Neural Networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* **2015**, 2148-2814. DOI: 10.1109/CVPR.2015.7298681.
(95) Acharya, K. R.; Lloyd, M. D. The advantages and limitations of protein crystal structures. *Trends in Pharmacological Sciences* **2005**, *26* (1), 10-14. DOI: 10.1016/j.tips.2004.10.011 (acccessed 2024-01-30T13:40:41).
(96) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.
(97) Wang, X.; Elshahawi, S. I.; Cai, W.; Zhang, Y.; Ponomareva, L. V.; Chen, X.; Copley, G. C.; Hower, J. C.; Zhan, C. G.; Parkin, S.; et al. Bi- and Tetracyclic Spirotetronates from the Coal Mine Fire Isolate Streptomyces sp. LC-6-2. *J Nat Prod* **2017**, *80* (4), 1141-1149. DOI: 10.1021/acs.jnatprod.7b00108.
(98) Igarashi, Y.; Yu, L.; Miyanaga, S.; Fukuda, T.; Saitoh, N.; Sakurai, H.; Saiki, I.; Alonso-Vega, P.; Trujillo, M. E. Abyssomicin I, a modified polycyclic polyketide from Streptomyces sp. CHI39. *J Nat Prod* **2010**, *73* (11), 1943-1946. DOI: 10.1021/np100292h.
(99) Ward, S. L.; Hu, Z.; Schirmer, A.; Reid, R.; Revill, W. P.; Reeves, C. D.; Petrakovsky, O. V.; Dong, S. D.; Katz, L. Chalcomycin biosynthesis gene cluster from Streptomyces bikiniensis: novel features of an unusual ketolide produced through expression of the chm polyketide synthase in Streptomyces fradiae. *Antimicrob Agents Chemother* **2004**, *48* (12), 4703-4712. DOI: 10.1128/AAC.48.12.4703-4712.2004.
(100) Chen, S.; Huang, X.; Zhou, X.; Bai, L.; He, J.; Jeong, K. J.; Lee, S. Y.; Deng, Z. Organizational and mutational analysis of a complete FR-008/candicidin gene cluster encoding a structurally related polyene complex. *Chem Biol* **2003**, *10* (11), 1065-1076. DOI: 10.1016/j.chembiol.2003.10.007.

(101) Malla, S.; Thuy, T. T.; Oh, T. J.; Sohng, J. K. Identification and characterization of gerPI and gerPII involved in epoxidation and hydroxylation of dihydrochalcolactone in Streptomyces species KCTC 0041BP. *Arch Microbiol* **2011**, *193* (2), 95-103. DOI: 10.1007/s00203-010-0648-7.
(102) Thuy, T. T.; Liou, K.; Oh, T. J.; Kim, D. H.; Nam, D. H.; Yoo, J. C.; Sohng, J. K. Biosynthesis of dTDP-6-deoxy-beta-D-allose, biochemical characterization of dTDP-4-keto-6-deoxyglucose reductase (GerKI) from Streptomyces sp. KCTC 0041BP. *Glycobiology* **2007**, *17* (2), 119-126. DOI: 10.1093/glycob/cwl060.
(103) Takaishi, M.; Kudo, F.; Eguchi, T. Identification of the incednine biosynthetic gene cluster: characterization of novel β-glutamate-β-decarboxylase IdnL3. *J Antibiot (Tokyo)* **2013**, *66* (12), 691-699. DOI: 10.1038/ja.2013.76.
(104) Tan, B.; Chen, S.; Zhang, Q.; Chen, Y.; Zhu, Y.; Khan, I.; Zhang, W.; Zhang, C. Heterologous Expression Leads to Discovery of Diversified Lobophorin Analogues and a Flexible Glycosyltransferase. *Org Lett* **2020**, *22* (3), 1062-1066. DOI: 10.1021/acs.orglett.9b04597.
(105) Niu, S.; Li, S.; Chen, Y.; Tian, X.; Zhang, H.; Zhang, G.; Zhang, W.; Yang, X.; Zhang, S.; Ju, J.; et al. Lobophorins E and F, new spirotetronate antibiotics from a South China Sea-derived Streptomyces sp. SCSIO 01127. *J Antibiot (Tokyo)* **2011**, *64* (11), 711-716. DOI: 10.1038/ja.2011.78.
(106) Chisuga, T.; Nagai, A.; Miyanaga, A.; Goto, E.; Kishikawa, K.; Kudo, F.; Eguchi, T. Structural Insight into the Reaction Mechanism of Ketosynthase-Like Decarboxylase in a Loading Module of Modular Polyketide Synthases. *ACS Chem Biol* **2022**, *17* (1), 198-206. DOI: 10.1021/acschembio.1c00856.
(107) Trefzer, A.; Pelzer, S.; Schimana, J.; Stockert, S.; Bihlmaier, C.; Fiedler, H. P.; Welzel, K.; Vente, A.; Bechthold, A. Biosynthetic gene cluster of simocyclinone, a natural multihybrid antibiotic. *Antimicrob Agents Chemother* **2002**, *46* (5), 1174-1182. DOI: 10.1128/AAC.46.5.1174-1182.2002.
(108) Demydchuk, Y.; Sun, Y.; Hong, H.; Staunton, J.; Spencer, J. B.; Leadlay, P. F. Analysis of the tetronomycin gene cluster: insights into the biosynthesis of a polyether tetronate antibiotic. *Chembiochem* **2008**, *9* (7), 1136-1145. DOI: 10.1002/cbic.200700715.
(109) Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI open.* **2020**, *1*, 57-81. DOI: 10.1016/j.aiopen.2021.01.001.
(110) Zachary, W. W. An Information Flow Model for Conflict and Fission in Small Groups. *Journal of Anthropological Research* **1977**, *33* (4), 452-473. (acccessed 2024/01/30/).JSTOR.
(111) Euler, L. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae* **1741**, 128-140.
(112) Pearce, A.; Wiltschko, A.; Sanchez-Lengeling, B.; Reif, E. A Gentle Introduction to Graph Neural Networks. **2021**.
(113) Daigavane, A.; Ravindran, B.; Aggarwal, G. Understanding convolutions on graphs. *Distill* **2021**, *6* (9), e32.
(114) Bronstein, M. a. B. J. a. L. Y. a. S. A. a. V. P. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* **2016**, *34*. DOI: 10.1109/MSP.2017.2693418.
(115) Evers, N. S. Lewis Carroll's Defense of Euclid: Parallels or Contrariwise. In *Handbook of the Mathematics of the Arts and Sciences*, Springer International Publishing, 2021; pp 1093-1113.
(116) Carroll, L. *Euclid and His Modern Rivals*; 1879.
(117) Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems* **2017**, *30*.
(118) Kipf, T. N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* **2016**.
(119) Fey, M.; Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* **2019**.
(120) Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903* **2017**.
(121) Ashish Vaswani, N. S., Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin. Attention is all you need.
(122) Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; Leskovec, J. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* **2019**, *32*.

(123) Bodenhofer, U.; Bonatesta, E.; Horejš-Kainrath, C.; Hochreiter, S. msa: an R package for multiple sequence alignment. *Bioinformatics* **2015**, *31* (24), 3997-3999. DOI: 10.1093/bioinformatics/btv494 (acccessed 2024-03-12T13:04:33).
(124) Jamasb, A. R.; Viñas, R.; Ma, E. J.; Harris, C.; Huang, K.; Hall, D.; Lió, P.; Blundell, T. L. Graphein - a Python Library for Geometric Deep Learning and Network Analysis on Protein Structures and Interaction Networks. Cold Spring Harbor Laboratory: 2020.
(125) Lau, J.; Fu, H.; Cane, D. E.; Khosla, C. Dissecting the role of acyltransferase domains of modular polyketide synthases in the choice and stereochemical fate of extender units. *Biochemistry* **1999**, *38* (5), 1643-1651, Article. DOI: 10.1021/bi9820311.
(126) Sanner, M. F.; Olson, A. J.; Spehner, J.-C. Reduced surface: An efficient way to compute molecular surfaces. *Biopolymers.* **1996**, *38* (3), 305-320. DOI: 10.1002/(SICI)1097-0282(199603)38:3<305::AID-BIP4>3.0.CO;2-Y.
(127) Liu, S. a. X. X. a. L. H. NEPRE: a Scoring Function for Protein Structures based on Neighbourhood Preference. **2018**. DOI: 10.1101/463554.
(128) Ofer, D.; Brandes, N.; Linial, M. The language of proteins: NLP, machine learning & protein sequences. *Comput Struct Biotechnol J* **2021**, *19*, 1750-1758. DOI: 10.1016/j.csbj.2021.03.022.