Thesis

James Engleback

June 30, 2022

Contents

Chapter 1

Introduction

1.1 Abstract

This thesis develops two enzyme engineering methods. Both centre around developing a mutant of the bacterial Cytochrome P450 BM3 to hydroxylate the herbicide mesotrione at carbon C_5 for use in herbicide-resistant crop engineering projects.

Work was done in the Manchester Institute of Biotechnology under supervision of Prof. Andrew Munro and later Dr. Sam Hay. The project was funded by the BBSRC iCASE program in partnership with agricultural technology company Syngenta.

Virtual Directed Evolution (VDE) is a program developed to engineer enzymes with target performance in simulation using black box optimization algorithms. Simulations model an enzyme sequence *fitness* in a user-defined *task* and returns a score. A black box function optimizer is used to find progressively higher-scoring protein sequences.

In this case, the *task* involves:

1. Protein structure prediction from input sequence (side-chain repacking)
2. Prediction of likely protein-mesotrione binding conformations (molecular docking)
3. Score based on favourability of mesotrione binding and the sequence distance from BM3 template.

The sequence:score information is passed to the optimizer and a new batch of protein sequences is generated and evaluated. Genetic algorithms and Bayesian optimization are compared as optimization functions.

Once running, the process is fully automated. Simulations are run in parallel and at scale on cloud infrastructure. Here, throughput is 10^4 mutants per day on a 60 CPU virtual machine. Several methods of generating protein libraries for lab testing are presented.

Screening Fist is an enzyme-substrate screening operation directed by an learning algorithm. The screen detects binding activity between a purified enzyme and a compound screening

library.

Screening data is passed to an agent to fine tune its internal, pre-trained model of protein function from sequence, which is used to design an array of screening experiments that maximizes expected information gain (EIG). The end use of the model is to generate useful predictions of protein function from sequence in downstream enzyme engineering tasks.

A high throughput binding assay is developed, including software to automate test design, liquid handling and data analysis. Five purified BM3 mutants are screened against a chemical library of 900 FDA-approved drugs, some of which share common features with herbicides.

An agent containing a multi-task deep neural network is constructed and trained to functionally annotate an input protein sequence and estimate its likelihood of interacting with a particular compound.

Pre-trained from a custom database of 10^6 annotated protein sequences and their known ligands, the model is retrained on screening data, evaluated for accuracy and used to generate new BM3 mutants with predicted mesotrione binding capabilities.

1.2 Introduction

[1]

Chapter 2

Screening Fist

2.1 Abstract

Screening Fist is a screening operation used to train a machine learning model to predict the likelihood of a given enzyme binding to a given small molecule from sequence. The model is pre-trained on a large and general dataset of protein and small molecule binding pairs scraped from several online sources before retraining on the screening data. The screening data is generated in-house with a high-throughput, custom developed Cytochrome P450-small molecule binding assay, with which five P450 BM3 mutants were each screened for binding against 980 drug-like molecules.

The retrained model can be used for virtual screening of new enzyme sequences against a specific target molecule, or to query a single sequence against many prospective small molecule binding partners. It can also be used to design subsequent rounds of screening based on the expected information gain of an experimental design.

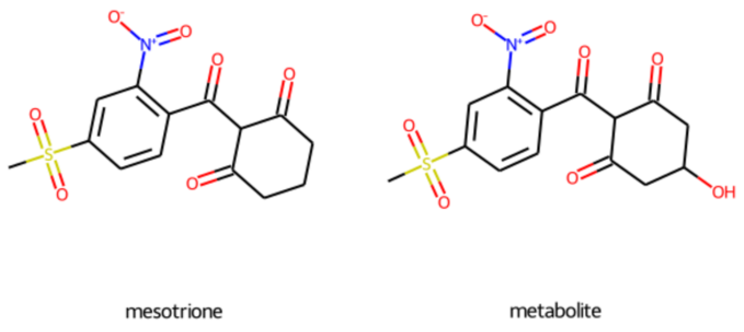
2.2 Introduction

2.2.1 Engineering Problem and Context

Herbicide-resistant crops are important for global agriculture because they mitigate yield losses due to weeds and give farmers extra flexibility in their herbicide application programs, which is important to suppress emergence of herbicide-resistant weeds.

Herbicides kill plants by inhibiting key metabolic processes and their species-specificity is determined by susceptibility of herbicide target and their ability to metabolize the herbicide. HPPD inhibitors are a key herbicide class that cause leaf bleaching and death in susceptible plants. HPPD inhibition disrupts tyrosine catabolism which disrupts UV-protection via carotenoid production and photosynthetic electron shuttling via plastoquinone, leading to death by UV damage and radical toxicity.

Figure 2.1: The HPPD-inhibiting herbicide mesotrione and its primary metabolite 5-hydroxy mesotrione - hydroxylated at the C_5 position naturally in some plants by indigenous Cytochrome P450s, priming it for subsequent stages of detoxification.



Engineering HPPD-inhibitor resistance into plants have used the HPPD and metabolic enzymes from naturally resistant species like *Avena fatua*. **note: ref this.** CYP72A1 is a cytochrome P450 that initiates metabolism of HPPD herbicides by ring hydroxylation at C_5 and has been explored as a means of engineering HPPD herbicide metabolism into soybean. **Figure 3.1** shows the chemical structure of mesotrione and its primary metabolite 5-hydroxy mesotrione.

Cytochrome P450s are a ubiquitous class of heme-dependent oxidoreductases that are frequently involved in xenobiotic metabolism. Bacterial P450s have been engineered to catalyse a range of xenobiotic biotransformations. The bacterial P450 BM3 from *Bacillus megaterium* is one such bacterial P450 whose structure has been studied extensively [2].

Notably, BM3 is a popular target for enzyme engineering studies, owing in part to its extensive documentation, ease of expression and *evolvability* [3]. Some examples of BM3 mutants with altered substrate scope are shown in **Table 2.4** [4].

The BM3 A82F/F87V mutant has a broad substrate specificity [5], and represents an attractive starting point for engineering altered substrate scope, however it has no binding activity towards the HPPD herbicide mesotrione.

2.2.2 Technologies Used

Machine Learning

Transfer Learning

Transfer learning is a popular approach to enhancing model accuracy and involves pretraining the model on a large, general dataset before transferring to a new task. The effect is a reduction in the number of samples required to train the model. Pre-training datasets do no necessarily need to be closely related to the target task, for example: models initially trained

Table 2.1: Examples of engineered BM3 Mutants with altered substrate scope [4]

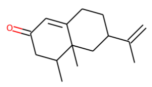
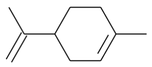
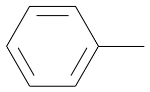
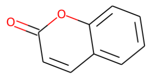
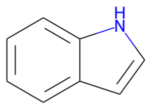
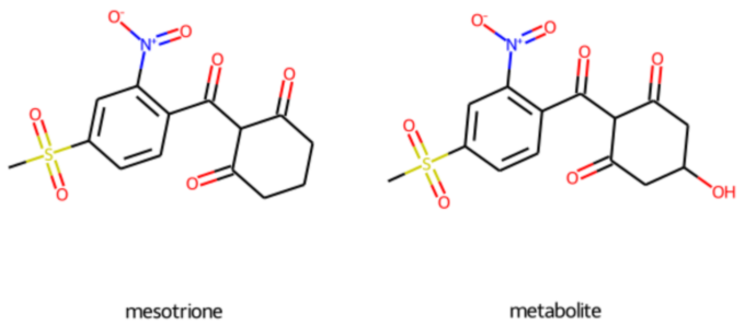
Mutant	**Substrate scope**	**Structure**
F87A/I263A/(A328I)	Nootkatone	
A328F	Limonene	
R47L/Y51F	Alkylbenzenes (no heteroatoms)	
R47L/F87V/L188Q (LVQ)	Coumarins	
A74G/F87V/L188Q (GVQ)	Indole, Lovastatinn, beta-ionine, organophosphates, polyaromatic hydrocarbons, chlorinated dioxins	

Figure 2.2: The HPPD-inhibiting herbicide mesotrione and its primary metabolite 5-hydroxy mesotrione - hydroxylated at the C_5 position naturally in some plants by indigenous Cytochrome P450s, priming it for subsequent stages of detoxification.



on the CIFAR 100 dataset (containing cars, animals etc) have been successfully re-tasked on phenotype identification from microscopy images [6].

In the domain of protein sequence-based machine learning, thoroughly pre-trained models are available for generating a neural embedding of a given protein that can improve sample efficiency in downstream learning tasks. Generally, these models are large attention-based models trained *unsupervised* on a large corpus of protein sequences, like the *TrEMBL* collection of *Uniprot*. In this case, unsupervised training often entails reconstruction of a distorted or masked protein sequence and is run on hardware far beyond the budget of this project.

The TAPE Benchmark

note: explain relevance, flesh out

Tasks Assessing Protein Embeddings [7] is a benchmark for comparing numerical representations of protein sequence (learned or otherwise) on a set of biological learning tasks from different domains of protein science. It currently contains five tasks:

1. **Secondary Structure Prediction Task:**
2. **Structural Contact Prediction Task:**
3. **Remote Homology Detection:**
4. **Fluorescent Protein Landscape Prediction:**
5. **Protein Stability Landscape Prediction:**

Tasks 4 and 5 are most applicable to protein engineering, since they involve metric prediction from a set of largely similar protein sequences. The leader boards for performance on these two tasks as of 5 Jun 2022 are:

Fluorescence:

Table 2.2: Fluorescence TAPE benchmark leader boards

Ranking	**Model**	**Spearman's rho**
1.	Transformer	0.68
2.	LSTM	0.67
2.	Unirep	0.67
4.	Bepler	0.33
5.	ResNet	0.21
6.	One Hot	0.14

Table 2.3: Stability TAPE benchmark leader boards

Ranking	**Model**	**Spearman's rho**
1.	Transformer	0.73
1.	Unirep	0.73
1.	ResNet	0.73
4.	LSTM	0.69
5.	Bepler	0.64
6.	One Hot	0.19

Stability:

Facebook AI Research - Evolutionary-Scale Modelling note: explain model architecture and purpose, set of pretrained models and availability

2.2.3 Related Work

note - do:

- **biswas**

2.2.4 Overview of This Work

This project is a proof of concept for a new approach towards enzyme engineering using machine learning. Broadly speaking, it involves screening a number of enzyme mutants for activity towards a large number of compounds of interest. This data is then used to train a machine learning model to predict the likelihood of activity between a given protein sequence and small molecule structure. The trained model has two applications:

1. Virtually screen a large number of potential protein and small molecule candidates for potential activity. These screens can be used to explore potential substrate scope of a given protein, or as a virtual fitness function for use in virtual directed evolution - where a sequence optimization algorithm evaluates sequences against a fitness function.

2. Optimal design of subsequent screening rounds based on the expected information gain (EIG) of a candidate experimental design.

The specific approach in this project is to screen a local area of sequence space surrounding the wild-type P450 BM3 against a library of 980 FDA-approved drug compounds. Drugs share chemical properties with herbicides so are suitable for a proof of concept study.

2.3 Methods and Development

2.3.1 Assay Development

In order to generate a P450 BM3-specific dataset on which a model could be re-trained to make binding likelihood predictions on drug-like molecules, a high throughput P450 binding detection assay was developed. The assay is based on traditional UV-Visible light spectroscopy-based techniques for detection of P450-ligand binding, miniaturized into a 384 well format. It relies heavily on automation and a throughput of 980 compounds per day is demonstrated.

Aim

The initial aims of this development work were:

- Develop a high throughput P450-ligand binding assay based on established biophysical characterisation techniques.
- Develop necessary software for design and analysis of each assay.
- Compare the precision and accuracy of the assay to existing techniques.

Basis: UV-Visible Spectroscopy for Monitoring Cytochrome P450-Ligand Binding

The assay is based on a technique for quantifying P450-ligand interactions based on UV-visible photospectroscopy. The technique consists of the purified Cytochrome P450 heme domain in question in a neutral buffer at around 5-10 μM in a optically clear cuvette. Since only the heme-containing domain of the P450 is used, no chemical reactions are expected to take place which removes time-sensitivity from the assay.

The UV-visible light absorbance of the sample is typically measured for all wavelengths between 200 and 800 nm, which for a P450 without a ligand bound in the active site should show a large and defined absorbance peak at around 420 nm.

After an initial absorbance measurement of the ligind-free P450, the compound of interest can be titrated into the sample. On binding to the ligand, the absorbance profile of the P450 changes such that the absorbance at 420 nm (A_{420}) decreases and absorbance at 390 nm (A_{390}) increases.

The change in A_{420} and A_{390} in response to change in ligand concentration can be quantified and used to derive metrics that indicate affinity between the ligand and P450 using Michaelis-Menten kinetics models.

The original Michaelis-Menten model of enzyme kinetics states:

$$v = V_{max} \frac{[S]}{[S] + K_M} \tag{2.1}$$

where v is the reaction velocity - the rate of an enzymatic reaction. V_{max} is the maximum possible v for this particular enzyme-substrate pair, $[S]$ is the concentration of the substrate and K_M is the $[S]$ at which $v = V_{max}$.

V_{max} and K_M are useful metrics for quantifying the binding interaction between enzyme and substrate, where low K_M indicates a tight binding interaction and a high V_{max} indicates a large magnitude of response.

Important assumptions in the Michaelis-Menten model of kinetics are:

1. The concentration of enzyme is $< K_d$
2. The rate of reaction is directly proportional to the concentration of the substituents
3. The reaction is at chemical equilibrium at the time of measurement
4. The interaction is reversible

A variant of this model is applied to Cytochrome P450 photospectroscopy assays, where the response to a ligand is detectable in the 390-420 nm region. Ligand binding to a P450 alters the electron environment in the heme, which changes the UV-Visible light absorbance profile. Specifically, absorbance at 420 nm is reduced and absorbance at 390 nm increases.

In this case, v is substituted for $\Delta A_{390} - \Delta A_{420}$ - the magnitude of the P450 response and K_M is substituted for K_d - the dissociation constant between the enzyme and ligand. This yields the formula:

$$Response = \Delta A_{390} - \Delta A_{420} = V_{max} \frac{[S]}{[S] + K_d} \tag{2.2}$$

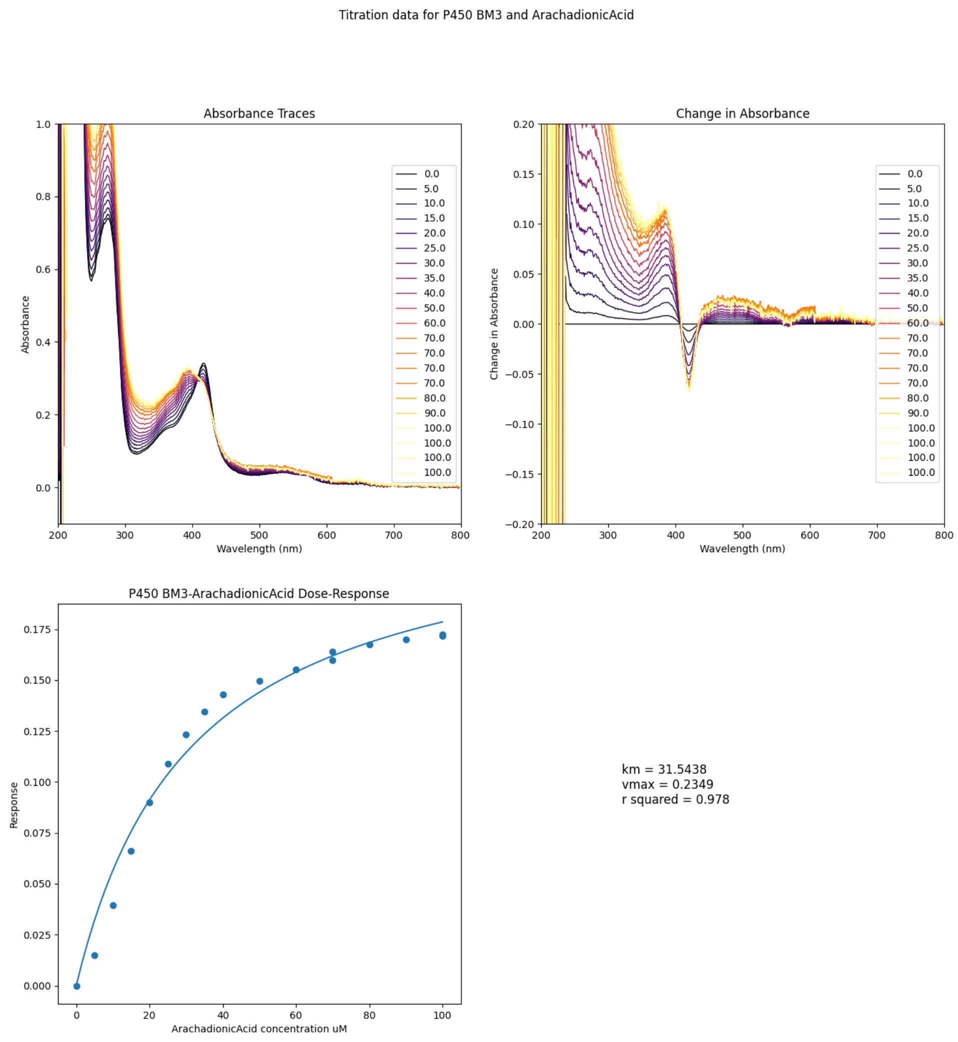
An example of this is shown in **figure 2.3**, where the fatty acid arachadionic acid was titrated into a sample of P450 BM3 wild-type.

Development

This style of assay was miniaturized into a 384-well format for the purpose of this project. The 384-well format permits high throughput screening of compounds for binding with a given P450 provided it is sufficiently stable to last the duration of the experiment without degrading and interfering with measurement.

At micro-scales, precision dispensing of both compound and protein is critical for overall assay precision. To achieve sufficient precision, compounds were dispensed using an *Echo 550* acoustic liquid handling device, which uses ultrasound to move liquid from a source plate to a destination plate. In order to achieve the desired dispensing pattern, a custom python package, `echo`, was developed. `echo` facilitates dispensing of compounds in the gradients required by output of a picklist `csv` file that can be used directly in the *Echo*. A critical function of this package is that it tracks the volume of compounds in the source plate, which is important

Figure 2.3: Response of P450 BM3 to Arachadionic Acid from a titration experiment, from which steady-state Michaelis-Menten kinetics can be derived. The legend shows the concentration of arachadionic acid in μM.



in the common case that one compound must be spread over multiple wells. The package was indispensable in this work.

During early stages of development, an automated bulk liquid dispensing device - a *ThermodropMulti* was available. The device uses precise peristaltic pumps to dispense a single liquid into a plate at a single volume very quickly. Precision in the bulk fluid volume in the plate is helpful in downstream analysis due to the consistent path lengths and concentrations in each well, something that was found to be less consistent when using a multichannel pipette to achieve the same task. **Figure 2.4** shows an example of both the *ThermodropMulti* and *Echo* being used to capture K_d and V_{max} for the binding interaction between P450 BM3 wild-type and arachadionic acid.

2.3.2 Assay Protocol

This is a assay protocol for detecting binding interactions between a Cytochrome P450 and multiple small molecule compounds. The assay has demonstrated scale to a library of 980 compounds and five P450 mutants and with some small adjustments could be improved in scale and precision.

It works in 384 well microplates and uses a microplate reader to capture absorbance profiles from 220-800 nm wavelengths, from which a pattern associated with a P450-small molecule binding interactions can be detected and quantified.

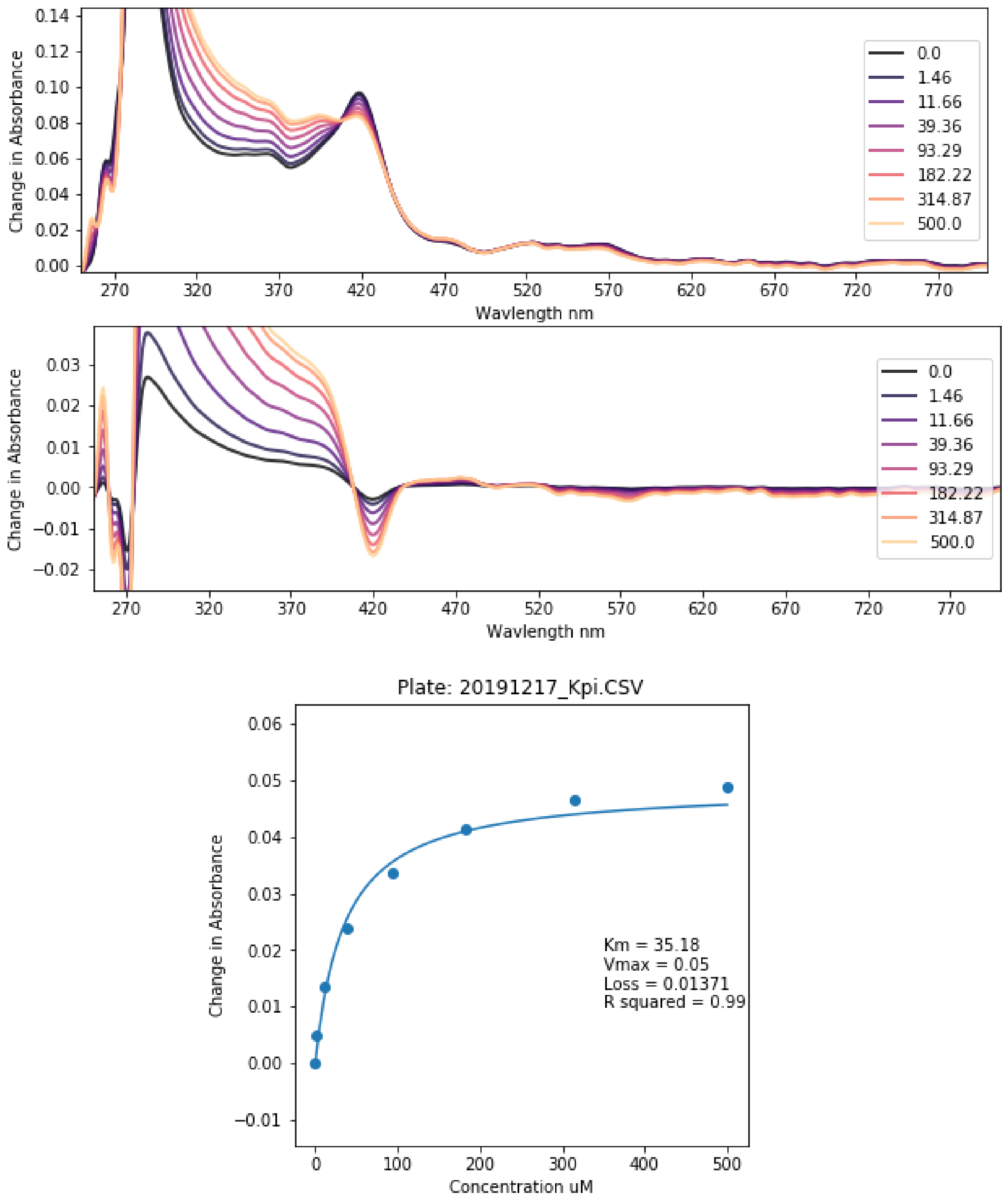
It was designed for profiling and modelling the effect of mutations on a P450's substrate binding preferences. This was tested with five P450 mutants against 980 drug-like compounds. It requires purified P450 protein which limits the rate of data generation, though can scale to more compounds.

Requirements

Essential:

- **Hardware:**
  - **Microplate reader:** Able to read absorbance for all wavelengths between 220 and 800 nm. Used here: *BMG ClarioStar and FluoStar* microplate readers.
  - **Labcyte Echo [500 550]:** Acoustic liquid handlers for precise compound dispensing.
  - **Consumables:** In absence of a high precision liquid handling machine, serial dilution of compounds would probably be fine.
  - **Enzyme:** A purified Cytochrome P450 - used here were mutants of P450 BM3 at 800 μM. Note that BM3 is fairly stable at room temperature which facilitates queuing large batches of plates to the plate reader. You could run the assay at a low temperature if you use a solvent other than DMSO, which freezes at 19°C, which interferes with measurement.

Figure 2.4: Michaelis-Menten steady-state kinetics of arachadionic acid and wild-type P450 BM3 captured during assay development. Notably, in this experiment both an *Echo* and *ThermodropMulti* were used in dispensing compound and protein respectively, which accounts for the neatness. The traces were smoothed using a Gaussian kernel to account for noise.



  - **Compound Library:** A large number of compounds in solvent (e.g. DMSO) in a microplate format. Used here was a 980 compound library, dissolved at 10 mM in DMSO in 96 well format.
  - **Buffer:** must be optically clear the protein must be stable in it. Must not contain potential ligands. Used here was 100 mM Potassium Phosphate at pH 7.0 - chosen based on traditional wisdom.
  - **384 well microplates - clear bottom:** Assay plates with at least 30 µl working volume. Some treated surfaces may be more suitable for unstable proteins. Ideally have minimal absorbance in the 390-420 nm region, but this can be corrected for with controls. Used here: *ThermoFisher Nunc 384 Well Plates*
  - **384 well *Labcyte Echo* source plates:** for dispensing compounds to assay plates. Used here were the *Low Dead Volume (LDV)* variety, which have a working volumne of 2.5-12.5 µl, which limits compound waste compared to the standard *Echo* plates (12-65 µl).

Optional:

- **Hardware:**
  - **Bulk liquid dispensing:** can be far more accurate than a multichannel pipette when dispensing protein or buffer into wells. During development, both a *ThermodropMulti* peristaltic dispenser and a *Hamilton Star* liquid handling robot. Both work well, though use of a bulk liquid dispensing machine is recommended given their speed, lower unit cost and lack of requirement for pipette tips.
  - **Microplate reader plate loader:** Autoloading plates into the reader increases throughput capacity significantly. I used a **BMG ClarioStar** plate reader with a stacker module.
- **Consumables:**
  - **BSA:** in assay buffer may have a stabilizing effect on the enzyme - which would improve time stability and reduce errors. Time stability is important for scalability.
  - **384 well Labcyte Echo DMSO Trays:** for control for DMSO concentration in assay wells by topping up each assay plate to a fixed concentration. Around 5% is ok with BM3.

Procedure

Summary:

1. **Design *Echo* picklists:**
   - An *Echo* can accept a `csv` file with column headers: `Transfer Volume`, `Volume`, `Destination Well`, `Source Well` and optionally: `Destination Plate Name` and `Source Plate Name`. The Volume must be in nano litres and a multiple of 2.5 and the Source and Destination wells must be in the format `[A-Z][0-9]+` and exist in the plate layout specified to the *Echo* client at runtime.

- The picklist(s) can be generated in a spreadsheet exported to `.csv` or programmatically. Documentation for the `python` tools used are [documented here.](picklists.md)

2. **Dispense compounds into *Echo* source plates** This can be done with a multichannel pipette, and requires one tip per compound. If the total volume of each compound required is greater than 60 µl then a standard polypropylene *Echo* plate should be used, otherwise a low dead volume plate may be economical If not, or for valuable compounds, Low Dead Volume *Echo* may plates should be used. These have a working volume of 2.5-12.5 µl, outside of which the *Echo* will refuse to dispense. You may need to dispense the same compounds into multiple source wells and the picklists must be designed accordingly.

3. **Dispense compounds from *Echo* source plates to empty assay plates**
   (a) Transfer the picklist `.csv` to the *Echo* host computer.
   (b) Launch the *Echo Plate Reformat* client there:
   (c) Create New Protocol
   (d) Select Custom Region Definitions
   (e) `File` > `Import Regions` and select your picklist `.csv`
   (f) **Optional:** Specify the log output in `Options`, simulate with `Run` > `Simulate`
   (g) Save and run, optionally simulating the run first. Multiple copies of a set of destination plates can be specified if the source plates contain sufficient compound volume.

4. **Stopping point:** Length of pause depends on rate of DMSO evaporation from destination/assay plates and the stability of the compounds at the plate storage conditions. Plates stored in a stack should limit evaporation rate to an extent, though specialised lids for *Echo* plates that limit DMSO evaporation are available. Up to 24 hours seems ok.

5. **Thaw purified P450, make stock of 10 µM in a neutral buffer, enough for 15.36+ ml per plate (40 µl per well)** I heard that thawing fast limits ice crystal formation, which could destroy some protein. Optionally, in a microcetrifuge, pre-cooled to 4C, spin the protein at 14,000 rpm and carefully transferr to fresh tubes to remove unfolded protein.
   (a) Measure the stock concentration of the protein in a UV-Vis spectrometer by taking an absorbance trace from 200-800 nm, diluted in the destination buffer. There should be a peak at 420 nm, the height of which can be used to calculate the protein concentration with the following equation:

$$[P450] = ael \tag{2.3}$$

   where a is absorbance and e is the extiction coefficient - 95 for P450 BM3 heme domain. l is the path length in cm and for the cuvettes used was 1. Use the measured stock concentration of P450 to create a working stock of around 10 µM. 10µM was chosen because it yeilds a reasonably strong signal when in plates. Varying the protein concentration doesn't have a big effect on measurements, so err towards using more.

(b) Dilute in neutral buffer to the target working concentration. Filtration through a 22 µm syringe filter can remove some precipitates. Vacuum filtration can work too but in practice, the protein can pick up binding partners from the filtration equipment contaminants, which can ruin downstream measurements.

6. **Dispense the diluted protein into the assay plates, centrifuge**

   An electric multichannel pipette works but accuracy is more limited than with automated dispensing. 38 µl of protein working stock needs to be dispensed into each well, which brings the total well volume to 40 µl in cases where the volume of compounds in DMSO in each well is 2 ul. If the volume of DMSO in destination wells is not a constant 2 ul, then default to 38 µl of the protein stock. The variation in total volume can be corrected for in compound concentration calculations, though the path lenght will vary which affects precision.

   Better than that is a precise bulk liquid handling device. I used a *ThermodropMulti* for a while which was fast and accurate. Occasionally a nozzle would become blocked either with DMSO ice or precipitates, though the protein still dispensed into the correct well. Blockages can be cleared by disassembling the pump head, coupling a syringe of water to the nozzle and flushing.

   It may be necessary to dispense some control plates, with everything but protein. This is useful to correct for the intrinsic absorbance of the plate and buffer, as well as the compounds themselves which sometimes have absorbance at the measurement wavelengths. A control set of plates for every protein screen may be unnecessary and expensive. One good one should be ok.

   Centrifuge the plates for 2-5 mins at around 2000 rpm to push the well contents to the bottom. This step can also ensure that meniscuses are flat and uniform and remove bubbles. If possible, centrifuge at room temperature to avoid DMSO ice formation.

7. **Capture UV-Visible light absorbance data between 220 and 800 nm from plates in a microplate reader at room temperature within 3 hours:**

   The protein is fairly stable over the course of 3 hours. On a BMG platereader, measurements take about 15 minutes per plate including the data transfer from device to host machine. Using an automated plate loader is recommended, for example a BMG Stacker module. In that case, put an empty or waste plate on the top of the stack to limit evaporation from the top assay plate. The BMG ClarioStar can be instructed not to read the last plate.

   The stacker occasionally jams due to a solenoid error, which can be due to a misaligned stack of plates. It is advisable to un-stack and re-stack the plates using the stacker to check for this kind of issue prior to measurement.

8. **Data analysis overview**

   (a) Export the plate measurement data to a workable format, like `.csv`. In the BMG Mars software, the operation is simple but on all host machines I've tried it on have been unreasonably slow to open the data files prior to export.

(b) Index the files to their experiments. I used a `config.yml` file to track this.

(c) **Analysis**

i. Match compounds to plate well data.

ii. Match the *Echo* exceptions report to wells to find the actual compound volume in each well.

iii. From each trace, subtract its own absorbance A_{800} at 800 nm. This accounts for baseline drift which can be caused by light scattering from precipitates.

iv. If correcting for compound absorbance with control plates, then subtract the absorbance of each test well from each control well. If the actual compound volumes of the test and control don't match up, it can be an issue if the compound interferes with the absorbance in the 390-420 nm region. If the compound absorbance changes predictably then it can be interpolated.

v. Curves can be smoothed with Gaussian smoothing using `scipy.ndimage.gaussian_filter1d` if necessary. Sources of a jagged curve can be precipitates, which can interfere with downstream analysis.

vi. At this point, changes in the P450 absorbance trace can be identified. Common categories of trace are:

- Clean absorbance trace, no shift.
- Clean absorbance trace, peak shift from 420 to 390 nm.
- Clean absorbance trace, peak shift from 420 to 424 nm.
- Compound interference in absorbance trace.
- Light scattering in absorbance trace.

vii. For clean traces with a peak shift from 420 to 390 or 424 nm, biding response can be calculated using the $|\Delta A390| - |\Delta A_{420}|$ or $|\Delta A420| - |\Delta A_{420}|$ for each compound concentration. With a set of concentration-response data points, the binding dissociation constant K_d can be calculated using the Michaelis-Menten equation for enzyme kinetics:

$$Response = \frac{V_{max} \times [S]}{K_d + [S]} \tag{2.4}$$

$$Response = |\Delta A_{390}| - |\Delta A_{420}| \tag{2.5}$$

Where $[S]$ is a given substrate concentration and V_{max} is the maximum response magnitude possible from the P450 being tested from this compound. The metrics K_d and V_{max} can be derived by fitting $|\Delta A390| - |\Delta A_{420}| = \frac{V_{max} \times [S]}{K_d + [S]}$ can be fit to the P450 substrate concentration-response data points using a curve fit algorithm like `scipy.optimize.curve_fit`.

Table 2.4: BM3 mutants used in screening

ID	Mutations	PDB
WT		1BU7
A82F	A82F	4KEW
DM	A82F/F87V	4KEY
1YQO	T268A	1YQO
1YQP	T268N	1YQP

Useful additional metrics for each compound are R^2 score of the curve fit, a data quality metric.

An ideal end output of this analysis as a table of compounds, P450s and a qualification or quantification of their binding interactions.

2.3.3 Enzyme Production

Summary

This page contains the methods for producing the enzymes used in this screening program. The enzymes are variants of the Cytochrome P450 BM3:

The page shows the method used to create the mutant BM3 expression plasmid DNA, expression of the mutants in *E. coli* and their purification.

Aims

- Create expression plasmids containing the target mutants from an in-house starting point - `bm3-wt.gb`.
- Sequence the plasmids to confirm they carry the mutations
- Express the mutants in *E. coli* using those plasmids.
- Purify the mutant protein from the *E. coli* harvest.

DNA

Starting Material

An heirloom BM3 Wild-type (heme domain) expression plasmid, `bm3-wt.gb`, was inherited and used as the basis for DNA work in this project. The plasmid is a pET14 expression vector where the BM3 gene has a 6xHis purification tag at the N-terminus, flanked by a T7 promoter and terminator which leads to high yields in strains of *E. coli* containing the T7 RNA polymerase. The plasmid also encodes ampicillin resistance and a *ColE1* replication origin which leads to a low copy number.

Primer Design and Acquisition

Mutations were introduced to the wild-type sequence via Polymerase Chain Reaction (PCR)-based site-directed mutagenesis. Two methods were considered for this task based on commercially available kits, where each imposes different constraints on primer design. Efforts were made to automate primer design as far as possible with scalability in mind.

The PCR kits used were:

1. *New England Biolabs (NEB) Q5 mutagenesis kit* - which requires that primers facilitate cloning of a linear DNA strand from the circular template plasmid and mutation payloads are carried in the tail of one primer. The kit includes a cocktail of the DNAse *DPN1*, which disassembles template plasmid methylated in *E. coli* and a kinase and ligase that work to join the ends of the linear DNA into a circular plasmid. The reaction is restricted to one payload.

2. *Agilent Quickchange mutagenesis kit* - which requires a pair of overlapping primers that carry the mutation payload in the mid-section. This cloning method produces circular DNA carrying the targeted changes. It has the advantage of allowing multiple payloads carried by multiple primer sets.

Two important considerations based on the template sequence are:

1. Adenine-Thymine (AT) richness of the template sequence. Compared to cytosine and guanine (C and G), A and T bind to their complimentary bases weakly. This results in weak primer binding to the template sequence, measurable by a low primer *melting temperature* T_m. To compensate, primers must be longer than they otherwise would be for a sequence richer in CG, which increases their cost and their chance of self-binding. The template sequence used here is AT-rich - at 42%.

2. Repetitions and palindromic regions of the template sequence. If the sequence surrounding a mutation target area contains these features, then the likelihood of *mis-priming* by binding to an off-target sequence area is high, so too is the likelihood of a non-functional, self-binding primer.

note: talk about `mxn`

PCR and Work Up

Sequencing

Purified plasmid DNA ostensibly conataining the target mutations, having been harvested and purified from DH5a *E. coli* cells, was shipped to *Eurofins Genomics* for sequencing using their *TubeSeq* service, which uses a variant of Sanger Sequencing. Sequencing primers for this matched the T7 promoter and terminator and provided coverage of the targetted region.

note: include sequencing, flesh out

2.3.4 Protein Expression

Having been sequenced and confirmed to carry the target mutations, the mutant plasmids were used to produce the mutant protein *en masse* via a *BL21 DE3 E. coli* strain, which contains a T7 RNA polymerase under the control of a *lac* promoter.

BM3 mutants were expressed in *NEB* BL21 (DE3) *E. coli* cells grown in *Formedium* auto-induction TB media at 25°C for 24 hours with shaking at 180 rpm. Cells were harvested by centrifugation in a *Beckman* JA16 rotor at 6000 rpm for 10 minutes and stored at -20 °C. Variants were purified from the cells by nickel-affinity chromatography, dialysis into buffer 1, concentration to 500-1000 μM and storage at -80 °C.

Materials

- Expression plasmid encoding mutant P450 BM3
- *BL21 DE3 E. coli* - NEB. This domesticated *E.coli* strain is shipped in a transformation buffer.
- Auto-induction *Terrific Broth* (TB) media, which contains glucose and a lactose analog. The lactose analog triggers expression of T7 RNA-polymerase in *BL21 DE3 E. coli* and the subsequent expression of the target protein between the T7 promoter and terminator regions. The glucose inhibits this until it is consumed by the cells, which allows them to multiply to sufficient numbers before diverting energy to production of the target protein.
- Ampicillin - the antibiotic for which resistance is encoded in the target plasmid, ensuring that all cells in the growth media contain this resistance. Assuming no ampicillin-resistant contaminants, all cells should be *BL21 DE3 E. coli* containing the target plasmid.
- Δ Amino-Levulnic acid (Δ-ALA) - a precursor to heme, ensuring heme availability for the large amount of BM3.

Method

Purification

2.3.5 Screening

Throughput the screening process, the protocol described in section 2.3.2 was employed. Six concentrations of each target compound were dispensed with the aim of determination of the V_{max} and K_d: 0.0, 5.26, 55.61, 156.72, 306.74 and 500 μM. These values were generated from six linearly spaced values between 0 and 1 squared, then scaled to the starting concentration and final volume. This method saved some compound compared to a straightforward linear spacing of values between 0 and 500 μM - the maximum concentration possible given a 10 mM starting stock in DMSO without exceeding a total of 5% v/v solvent.

During the first run, control experiments where compounds were present in buffer only were included for downstream correction of the compounds inherent UV-Visible light absorbance, though it was thought to be unnecessary to repeat this for each subsequent screen, since it is wasteful of resources, particularly the compounds themselves.

Each of the five enzymes was screened against all 980 compounds over the course of a day each.

2.3.6 Model Design and Construction

The model aims to predict the binding likelihood between a given protein sequence and chemical structure, illustrated in equation 2.3.6.

$$P_{binding} = fn(sequence, smiles) \tag{2.6}$$

This page describes the deep learning model constructed for this project. The model is designed to estimate the likelihood of a binding interaction between a given Cytochrome P450 sequence and a ligand SMILES code. The intended end uses of the model are:

1. Virtually screening sequences for potential activity with a given compound. 2. Optimally design an enzyme-ligand screening experiment.

Approach: Recommender Systems

Abstractly, the problem of predicting the binding likelihood between a one of m proteins and one of n small molecules can be likened to filling empty the values of an $n \times m$ matrix, where rows and columns refer to proteins and small molecules and values are the probability of a binding interaction between the two:

$$\begin{array}{ccccc} & compound_i & compound_{i+1} & ... & compound_{i+n} \\ sequence_i & P_{binding_{i,j}} & & & \\ sequence_{i+1} & & & & \\ \vdots & & & \ddots & \\ sequence_{i+m} & & & & \end{array}$$

Some $P_{binding}$ values are known, which in the perspective of $n \times m$ possible values where n and m approach infinity, coverage is sparse.

This type of problem has been addressed in recommender systems, which in the context of streaming services translates to a matrix of n users and m paces of content. Known values are likes and engagement metrics and are similarly sparse, and blanks can be filled with the probability of a successful recommendation.

Machine learning models can be trained to predict the unknown values based on a numerical representation of the user and content. The prediction can be cast as a classification problem. To overcome the lack of negative data points, presumed negative data can be generated by sampling a random user: content pair, which should be treated with caution.

In this work, a machine learning model classifies pairs of protein sequence and small molecules as binding or not. Negative samples are generated by randomly sampling a sequence and small molecule, which given the vastness of sequence and chemical space may be reasonable in a large number of cases, though this assumption is treated with caution. A Binary Cross Entropy loss function is employed here where true positives and synthesized negatives are weighted evenly.

Data

Small molecules are represented as SMILES codes in the dataset, which are parsed using `rdkit` and then into 2048 bit fingerprint vectors using the `RdkitFingerprint`. Molecular fingerprints are generated by hashing functions based on an input molecule such that similar molecules are assigned similar fingerprints, which makes them useful in featurizing small molecules for machine learning tasks. The fingerprints are represented as a $b \times 2048$ tensor where b is batch size.

Sequences are represented in the dataset as strings where each character c_i is a single letter amino acid code:

$$c_i \subset ACDEFGHIKLMNPQRSTVWY$$

Roughly, characters are encoded as tensors of the integers that index their position in the list $ACDEFGHIKLMNPQRSTVWY$, with extra positions to represent null values, start of frame and end of frame characters.

Pre-Training Data

A dataset of around 10^6 protein sequences and SMILES codes for their corresponding binding partners are gathered here from several sources.

There are several sources of data containing protein sequences and information on their binding partners.

- Uniprot is a large, publicly available database containing protein sequences and a variety of functional annotations. *Uniprot* comprises three databases:
- *UniProt Knowledgebase (UniprotKB)* contains protein sequences and their annotations. *UniprotKB* mainly comprises two collections:
- *SProt:* - The SwissProt collection. Manually annotated and non-redundant protein sequences. Annotations can include *Enzyme Comission (EC)* classification numbers, which can be linked to substrates and products. At the time of collection (2022) the *SProt* collection contains 1,706,449 data points.
- *Trembl:* - The TrEMBL collection. Automatic annotations, N data points.
- *UniProt Archive (UniParc)* - an archive of sequences pooled from other public protein sequence databases. This collection does not contain annotations.
- *UniProt Reference Clusters (UniRef)* - Three collections of clustered sequences from *UniprotKB* and *UniParc*. Each collection is clustered with the CD-HIT algorithm and a different

similarity cutoff: *UniRef100*, *UniRef90* and *UniRef50* use 100, 90 and 50In the case of UniRef100, similarity measurements are made against sequence fragments. The clusters do not contain annotations. Importantly: *Uniprot* files can be freely downloaded via an `ftp` server at https://ftp.uniprot.org/pub/databases/uniprot/current_release/uniref/

- KEGG is another large source of proteins and biological function annotations, though download of the full dataset is restricted to paying customers. *KEGG* does however have a *REST API* which can be used to query the database, though requests are handled somewhat slowly. Despite this limitation, it was useful to cross-reference *EC* numbers against substrate and product identities.

- BindingDB is a large collection of proteins and known small molecule binding partners mostly derived from high throughput drug screening datasets, which is reflected in the chemical diversity of the set.

- PDBBind contains protein sequences and structures and their binding ligands derived from the *Protein DataBank (PDB)*.

- BRENDA contains manual curations from primary literature for enzymes.

For this work, the Uniprot SProt and BindingDB collections were selected because they met the necessary requirements of containing protein sequences that can be matched to a chemical binding partner. For data mining a cloud machine was hired (Linode `g6-dedicated-8`, 1TB Disk space, 8 `x86_64` cpus, 16 GB RAM, Debian 10, London). Scripts used here are in `screening-fist/data/pretraining/scripts/`.

Large files and files in large numbers can prove slow to search, a problem that databases address. For this work, a database was set up on the hired machine for storage and retrieval of the mined data. MongoDB, an unstructured database, was chosen.

MongoDB is unstructured in as far as it does not store tabular data, instead opting for a `json`-like schema which uses attribute:value arrays that themselves can contain more attribute:value arrays. This offers considerable flexibility in data input from heterogeneous sources with varying presence, absence and categories of annotation.

SProt was gathered as a compressed `.dat` file using `screening-fist/data/pretraining/scripts/uniprot.sh`, which decompresses, parses entries and uploads them to the local MongoDB in a single UNIX pipe. Using a single pipe saves the need to store the decompressed file on disk, which becomes significant in the case of the `TrEMBL` collection which has an uncompressed size of around 700GB as opposed to the 3.5GB of `sprot`.

BRENDA was used for its comprehensive list of EC numbers which would later be used for reference. BindingDB was downloaded in full as a `.tsv` file. Both the BRENDA and BindingDB sets were downloaded using `curl` commands and BindingDB was uploaded to the MongoDB instance.

The EC numbers from BRENDA were used with the KEGG REST API to collection compound identification numbers, which in turn were used to retrieve SMILES codes from PubChem using the script `ec.py` `mkdataset1.py` was used to assemble a `csv` containing protein sequences and SMILES codes of their binding partners (1,587,674), which was further filtered to exclude

SMILES codes and sequences that could not be parsed using RDKit and the tokenizer used in the sequence model.

Finally, an attempt to improve the chemical diversity of the pre-training dataset was made. A diversity filter that aims to return a subset of a compound set by maximizing the pairwise Tanimoto similarity of the subset compound fingerprints was employed using the MaxMin algorithm.

Unfortunately, MaxMin is $O(n^2)$ complexity, so is only feasible with relatively small batches. 64 compounds are selected this way from each batch of 512 from the filtered set, yielding `o3.csv` (1359834 data points and 908M uncompressed size).

`o3.csv` was compressed with `gzip` and loaded to an area of *Linode* object storage in Frankfurt, making it accessible to the *Linode* instances that would be used for model training.

`o3.csv` contained at least one invalid SMILES or sequences which had the costly effect of crashing the first model training run. A filter for troublesome inputs was built and yeilded `o3f.csv` (x xMB). and For the purposes of model training and evaluation, the `o3f.csv` dataset was split 3:1 into a training and test datasets `o3f.train.csv` and `o3f.test.csv` of sizes 907214 and 226804 respectively.

Training Data

Model Architecture

Given the model aims to predict:

$$P_{binding} = fn(sequence, smiles)$$

A neural network was devised to generate a prediction of $P_{binding}$ from an input of a protein sequence and a chemical SMILES code. The network can be split into three parts:

- **Sequence Embedding:** For a given protein $sequence$, output a tensor encoding a neural embedding $z_{sequence}$.
- **Chemical Embedding:** For a given chemical $smiles$ encoding, outputs an embedding z_{smiles}.
- **Prediction Head:** For the embeddings z_{smiles} and $z_{sequence}$, output a prediction $P_{binding}$.

The architecture is illustrated in 2.5.

Sequence Embedding

Although this model is the smallest of the ESM collection, on the single *NVIDIA Quadro RTX 6000* used it still occupied most of the 24 GB of available memory and most of the processing capability, which lead to long training times and difficulty in training more than one model in parallel on the same machine.

Figure 2.5: Architecture of the model constructed for this project

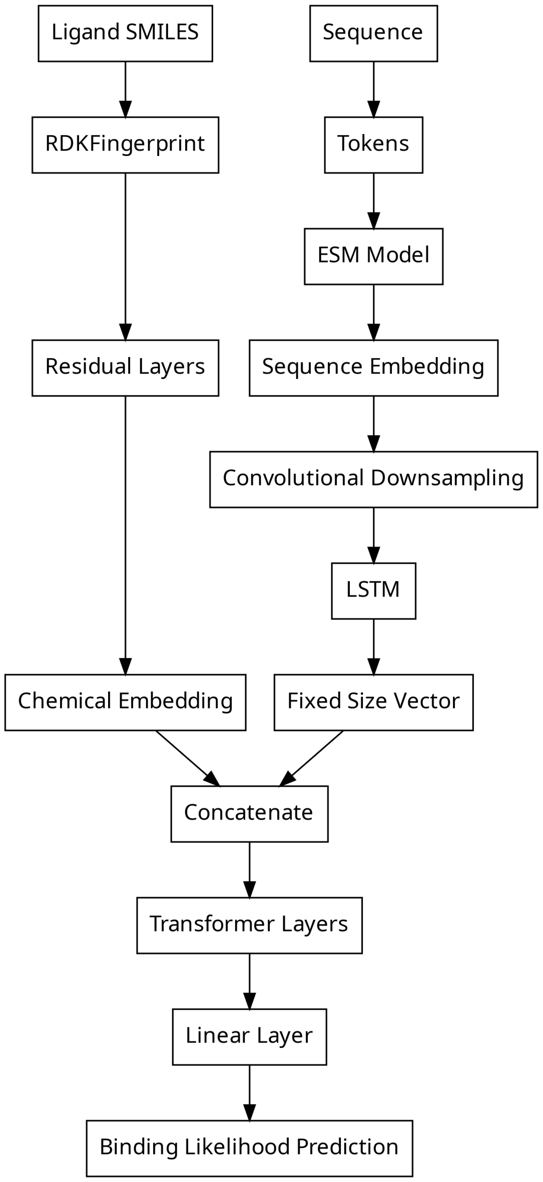


Table 2.5: Hardware specifications for machine used in model training.

Item	**Specifications**	**Number**	**Size**
CPU	Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz	8	
RAM	?	N/A	30 GB
Disk	?	1	630 GB
GPU	NVIDIA Quadro RTX 6000	1	20 GB VRAM

This could be remedied, however since in the complete `o3f.csv` dataset and the screening dataset there are 2947 unique sequences, so it was economical to pre-compute the embeddings and save them to disk. This resulted in a roughly 4$\times$ speedup in training time and massively reduced the memory requirements, allowing several models to be trained in parallel on a single GPU. This also saved costs significantly.

Chemical Embedding

As mentioned, chemical SMILES were hashed into chemical fingerprints using the `rdkit` `RDKFingerprint` method as a means of representation, yielding a 2048-bit vector for each compoundagainThe vectors were converted to tensors and served as an input to a residual neural network that output an embedding that would later be used to form a combined representation of both compound and sequence for binding likelihood prediction.

Prediction Head

The combined sequence and compound embeddings served as input to the prediction head, which output a single number that indicated a binding likelihood prediction for the two inputs.

Both residual neural networks and transformers were compared as prediction head architectures. Each consisted of 2-6 stacked layers of either residual or transformer layers with a fixed hidden layer size for convenience of automated assembly. The final layer in both cases was a single linear layer with a single output and a sigmoid function to output a number between 0 and 1 representing binding probability.

Pre-Training, Training and Evaluation

Training was done in two stages, each with a performance evaluation, during which several models with varying architectures and hyper-parameters were trained and compared. All training was done on a *Linode* `g1-gpu-rtx6000-1` which cost $1.50 per hour and was equipped with hardware specifications described in **table 2.5**.

1. **Pre-Training:** This was done with the larger, more general `o3f` dataset, which was randomly split into training and validation partitions, the latter of which was used sparingly

to avoid model bias. Pre-training lasted up to 64 epochs with a batch size up to 64. For each sample, a random sequence and SMILES pair were sampled as a presumed negative sample. The loss function used was binary cross entropy used with an Adam (Adaptive momentum) optimizer. Loss was tracked live using the *Weights and Biases* API which was useful to evaluate models as they trained and terminate them where necessary. Model weights were saved in each epoch and after training the model was evaluated for precision and accuracy on a subset of the training data. The metrics gathered were:

   (a) Mean binary cross entropy loss over evaluation.
   (b) Mean precision
   (c) A confusion matrix
   (d) A receiver operator curve (ROC)
   (e) A precision recall curve
   (f) A detection error trade-off (DET) curve

2. **Training:** This was done with the manually annotated screening dataset. An issue with the data was the class imbalance in that there were very few positive examples relative to negative. This was addressed by using *Synthetic Minority Oversampling* (SMOTE) whereby the rarer positive data were re-sampled until they number that of the negative data. The total size of the re-sampled data was 6666 points, which were then split 3:1 into training and validation sets of size 4999 and 1667 respectively. A model pre-trained on the larger `o3f` dataset was re-trained on this set and evaluated for performance in the same manner as with the `o3f` data, visualised in the following section.

Evaluation

2.4 Results

2.4.1 Screening Data Analysis

The data gathered for the screening experiments was saved in the `git` repository `https://github.com/jamese` in the `lab` directory. Within the `lab` directory are directories for each screening experiment. These include a `config.yml` file which maps data files to experiments for analysis.

The script `sxfst/scripts/data_proc2.py` was used to process the data into a large `csv` file containing each raw trace and its metadata. By consolidating the data into a single file, fewer time-consuming operating system calls are made in downstream analysis which decreases the time cost of those tasks.

The script attempted to use the *Echo* logs from each run to calculate the actual volume of each compound dispensed, since towards the end of the screening program shortages of compound made it impossible to fully dispense some experiments. In retrospective this function did not perform properly, since all experiments are later annotated as having the target volumes of each compound. This cannot be the case since it was casually observed that some compounds were completely depleted by the end of the screening program. This issue is significant, leading to false negatives where a lack of response cannot be attributed to either a lack of compound or a true negative response. This coding issue can be solved, but was not due to time constraints.

The downstream analysis in question was done using the script `sxfst/scripts/data_analysis.py` which normalizes the traces, subtracts the control traces from them and calculates the P450 response to each compound, creating a plot in the process. An example of one such plot is in **figure 2.6**.

The traditional response calculation method using $\Delta A_{390} - \Delta A_{420}$ proved ineffective for analysis of this data due to the noise, which would often result in scattering that increased the absorbance at the short wavelengths in ways that were occasionally unpredictable.

Instead, the gradient of each trace was taken by convolution of the kernel $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}$ across the smoothed and normalized traces. The gradients were invariant to total absorbance, which was often perturbed by what appeared to be light scattering, but achieved a similar effect to calculation of $\Delta A_{390} - \Delta A_{420}$. From the gradients, response was calculated as $|\frac{\delta y}{\delta x_{390}}| + |\frac{\delta y}{\delta x_{420}}|$.

Although this proved successful, the automated curve fitting to the response was not, outputting a constant value of 32 μM for each experiment - the lower K_d bound. With extra development this issue can likely be solved, but in order to keep within time constraints, an alternative solution was implemented.

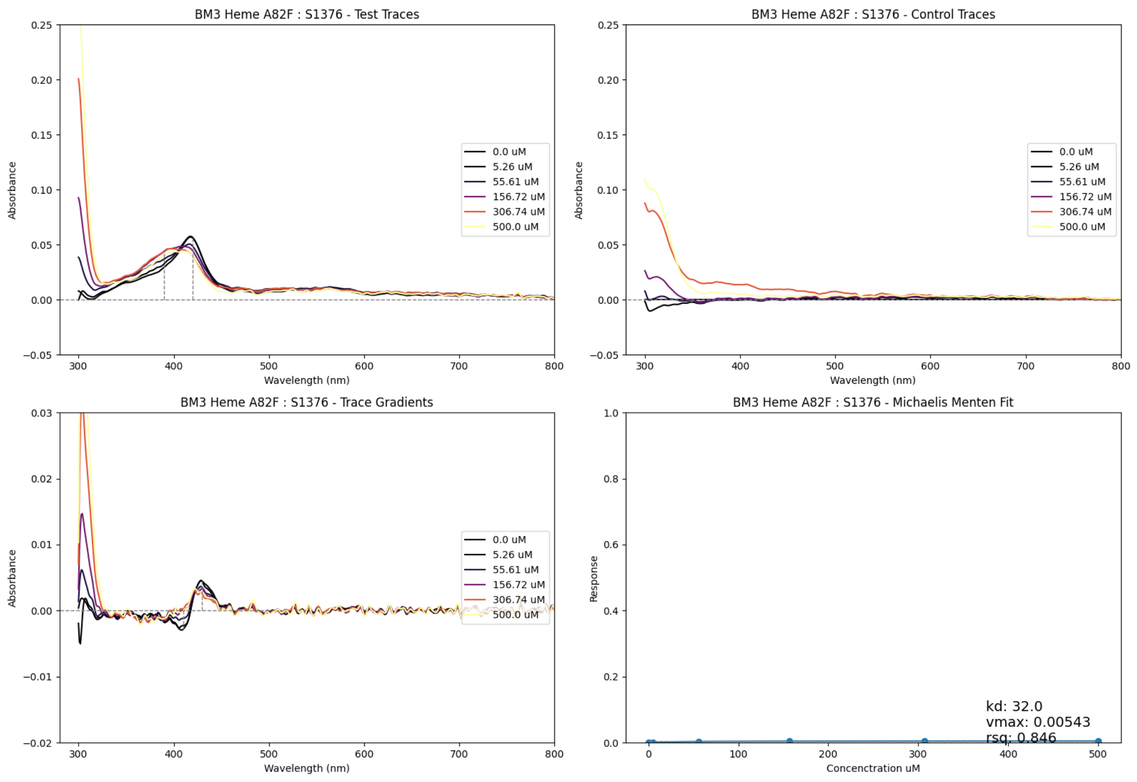
The alternative solution was manual annotation of the plots output by the script. Though crude, this method did yield a list of ostensible *hits* as well as a list of anomalous results that were to be ignored.

Using this approach, 149 hits of the total 4900 compounds were identified, listed in the file `lab/hits.txt` in the repository. These were used to create a `csv` file containing the full sequence and SMILES code for each enzyme compound pair and a boolean value that was true if that experiment was annotated as a *hit*. There are several drawbacks to this that should be noted:

- **Human Error:** Although several passes over the data were done, the order was never randomized and there was only one annotator. Since there are 4900 experiments, even if the annotator averages 1 second per annotation the runtime of an annotation run is 81 minutes, which will likely cause fatigue and subsequent errors. Since the order was not randomized, fatigue-induced error is more likely to occur in the same regions of the dataset in each pass.

- **False Negatives** due to a lack of properly functional compound concentration calculation, which would yield results that appear negative due to a lack of compound without being properly labelled as such. Another source of false negatives is the obfuscation of hits by noise in the traces due to scattering or compound absorbance that in some cases proved hard to correct for.

Figure 2.6: A good example of a positive result or *hit* between BM3 A82F and *Gestodene*. K_d was calculated as 32 μM, which reflects an issue in the curve fitting rather than the actual K_d. **note - rescale axes, move kd text, legend title**



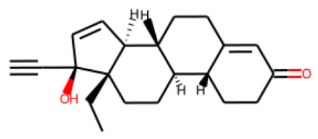


Table 2.6: Parameters for the best performing model in pre-training, designated the identifier *frosty-breeze-83* by the *Weights and Biases* API.

Parameter	**Value**
lr	10^{-5}
esm	esm1_t6_43M_UR50S
cuda	True
load	None
test	False
input	data/o3f.train.csv
epochs	320
batch_size	640
emb_size_fp	512
n_layers_fp	4
stride_pool	3
transformer	True
lstm_hs_pool	1024
emb_size_head	25
n_layers_head	3
kernel_size_pool	9
num_conv_layers_pool	3
num_lstm_layers_pool	2

2.4.2 Model Training

Pre-training

Given constraints on money towards computing resources, a hyper-parameter sweep was not viable. Instead, a small selection of models were trained and evaluated and the best of those was retrained on the screening data. Model architecture and hyper-parameters were directly configurable by supply of arguments to the training script `model/train.py`, from which the best performing model was obtained using the command `./train.py -i data/o3f.train.csv --transformer --cuda --emb_size_head 2560 --n_layers_head 3 --emb_size_fp 512 --n_layers_fp 4 --num_conv_layers_pool 3 --kernel_size_pool 9 --stride_pool 3 --lstm_hs_pool 1024 --lr 1e-5 --batch_size 64 -e 32` , which is consolidated in **table 2.6**.

2.4.3 Model Application

2.5 Discussion and Future Work

- **Improvements (Issues):**
  - **Issues with screening library**
  - **Improvements to data processing - recover compound concentrations**
  - **Improvements to screening dataset partitioning**

Figure 2.7: Model evaluation report after pre-training for the model designated *frosty-breeze-83*. Evaluation was on `o3f.test.csv` - a held-back partition of the pre-training dataset. Depicted are: mean binary cross entropy loss the course of training on `o3f.train.csv`, a confusion matrix, a receiver operator curve (ROC), a precision recall curve, a detection error tradeoff (DET) curve and mean precision and mean binary cross-entropy loss over evaluation.

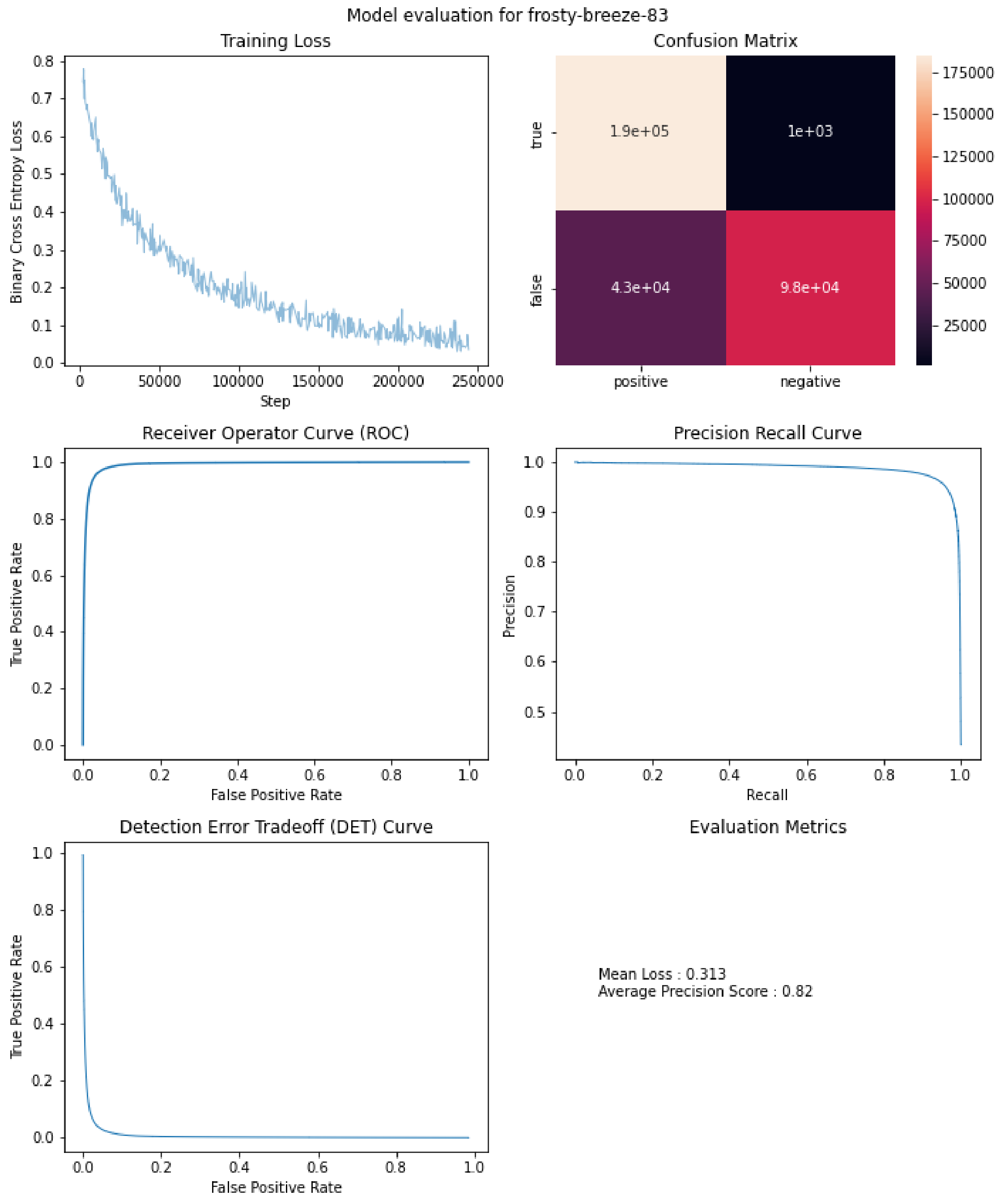
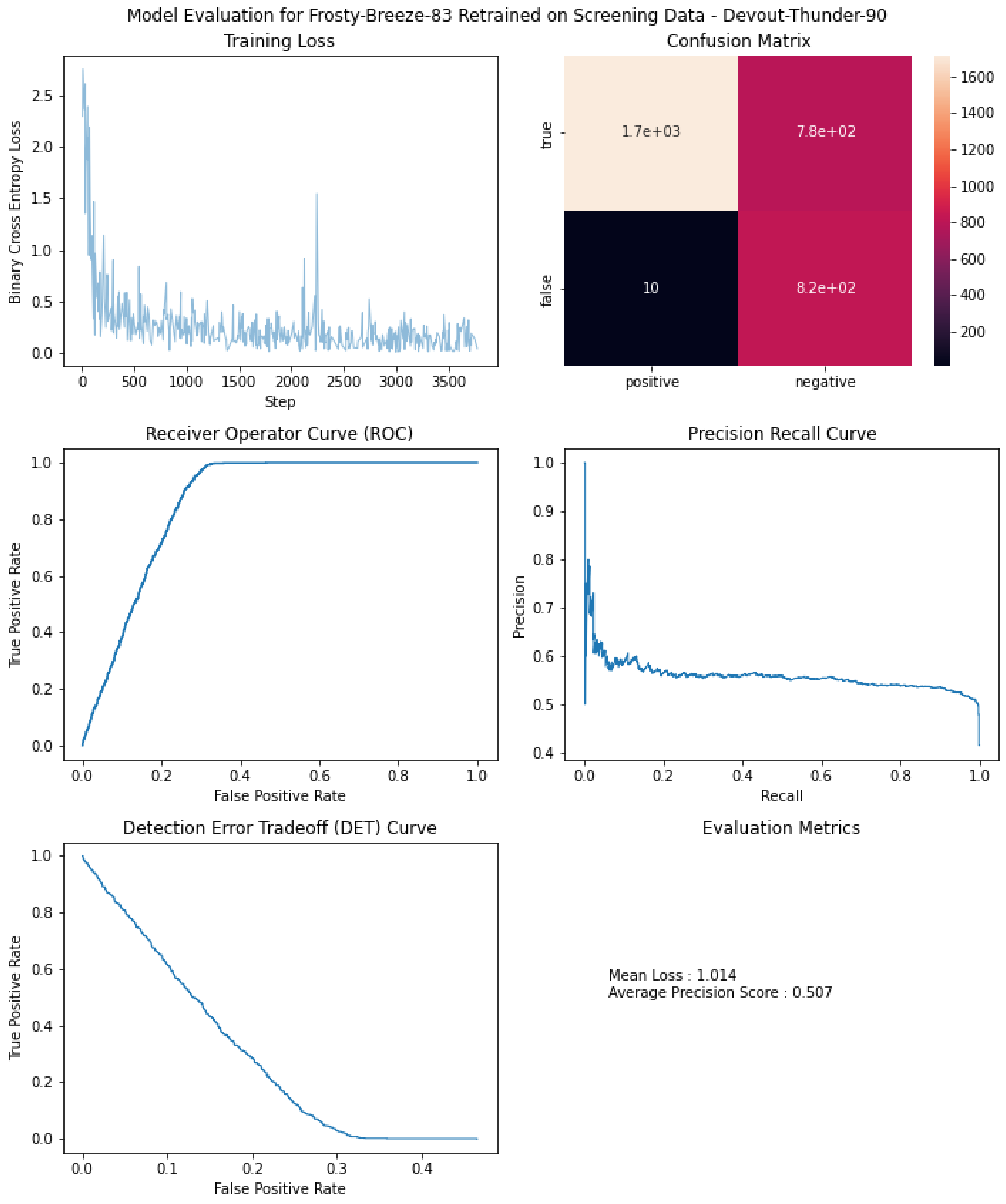
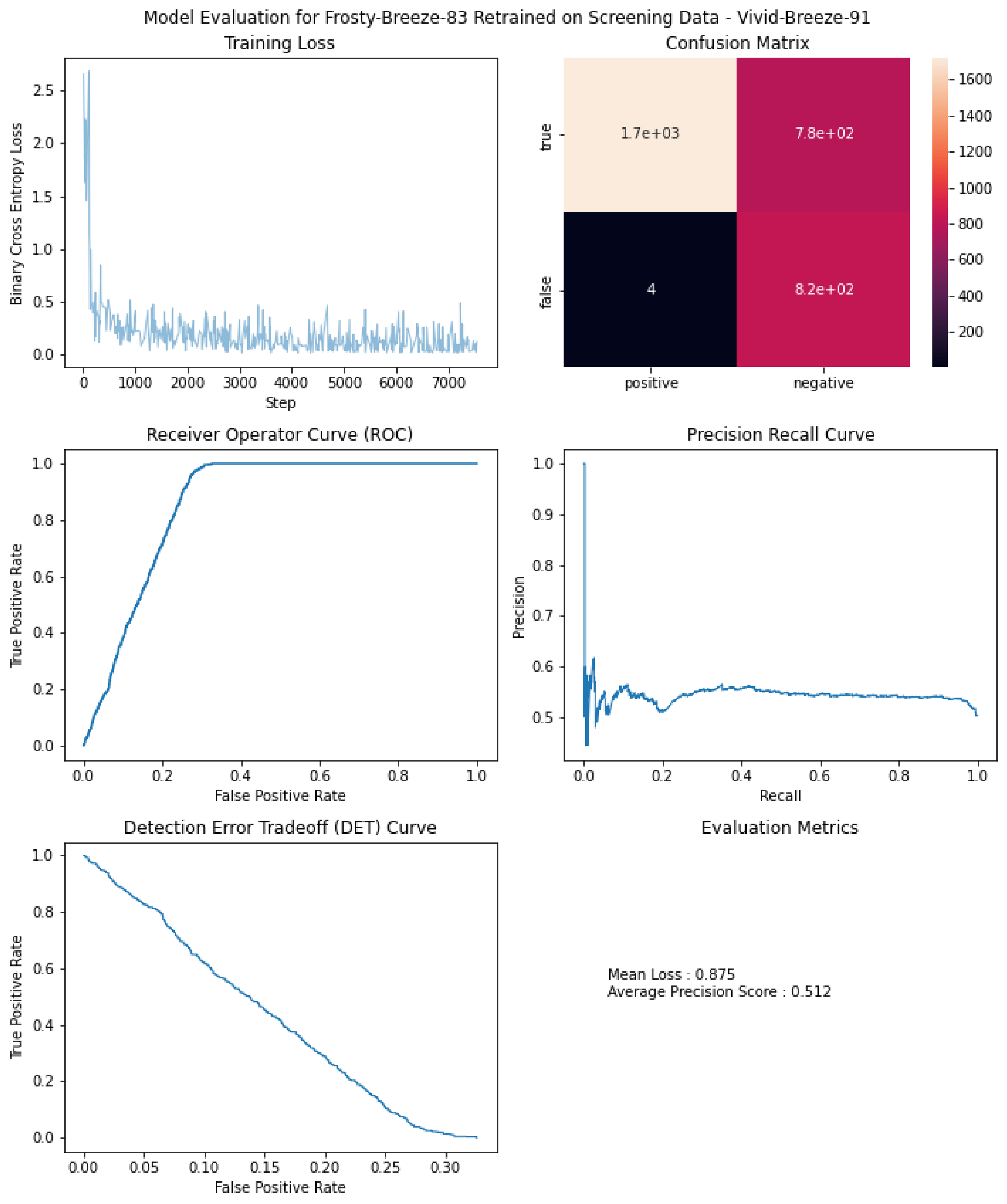


Figure 2.8: Model evaluation for another instance of *frosty-breeze-81*, this time designated *devout-thunder-90*, again retrained on the lab-based screening dataset `screening-data.train.csv` and evaluated on `screening-data.test.csv`. The report uses the same methods as in figure 2.7



Mean Loss : 1.014
Average Precision Score : 0.507

Figure 2.9: Model evaluation report after pre-training for the model designated *vivid-breeze-91*.



Mean Loss : 0.875
Average Precision Score : 0.512

- **Improvements to model application**

Bibliography

[1] Syngenta Participations AG Timothy Robert HawkesBernardus Theodorus Maria Vernooij. “EP 2164320 A2: Cytochrome p450 genes conferring herbicide resistance”. 2007.

[2] Andrew W Munro et al. “P450 BM3: the very model of a modern flavocytochrome”. In: *Trends in biochemical sciences* 27.5 (2002), pp. 250–257.

[3] Jesse D Bloom et al. “Protein stability promotes evolvability”. In: *Proceedings of the National Academy of Sciences* 103.15 (2006), pp. 5869–5874.

[4] Christopher JC Whitehouse, Stephen G Bell, and Luet-Lok Wong. “P450 BM3 (CYP102A1): connecting the dots”. In: *Chemical Society Reviews* 41.3 (2012), pp. 1218–1260.

[5] Christopher F Butler et al. “Key mutations alter the cytochrome P450 BM3 conformational landscape and remove inherent substrate bias”. In: *Journal of Biological Chemistry* 288.35 (2013), pp. 25387–25399.

[6] Andrew Ng. “Nuts and bolts of building AI applications using Deep Learning”. In: *NIPS Keynote Talk* (2016).

[7] Roshan Rao et al. “Evaluating Protein Transfer Learning with TAPE”. In: *bioarxiv* (2019).

[8] Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J Gray. “PyRosetta: a script-based interface for implementing molecular modeling algorithms using Rosetta”. In: *Bioinformatics* 26.5 (2010), pp. 689–691.

[9] Oleg Trott and Arthur J Olson. “AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading”. In: *Journal of computational chemistry* 31.2 (2010), pp. 455–461.

[10] Noel M O’Boyle et al. “Open Babel: An open chemical toolbox”. In: *Journal of cheminformatics* 3.1 (2011), pp. 1–14.

[11] Roland L Dunbrack Jr and Martin Karplus. “Backbone-dependent rotamer library for proteins application to side-chain prediction”. In: *Journal of molecular biology* 230.2 (1993), pp. 543–574.

[12] Christopher C Moser et al. “Distance metrics for heme protein electron tunneling”. In: *Biochimica et Biophysica Acta (BBA)-Bioenergetics* 1777.7-8 (2008), pp. 1032–1037.

[13] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[14] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[15] Adrian A Canutescu and Roland L Dunbrack Jr. “Cyclic coordinate descent: A robotics algorithm for protein loop closure”. In: *Protein science* 12.5 (2003), pp. 963–972.

[16] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589.

[17] Stefan Doerr et al. “Torchmd: A deep learning framework for molecular simulations”. In: *Journal of chemical theory and computation* 17.4 (2021), pp. 2355–2363.

[18] Junmei Wang et al. “Development and testing of a general amber force field”. In: *Journal of computational chemistry* 25.9 (2004), pp. 1157–1174.

[19] Diogo Santos-Martins et al. “Accelerating AutoDock4 with GPUs and gradient-based local search”. In: *Journal of chemical theory and computation* 17.2 (2021), pp. 1060–1073.

[20] Zhihai Liu et al. “PDB-wide collection of binding data: current status of the PDBbind database”. In: *Bioinformatics* 31.3 (2015), pp. 405–412.

Chapter 3

Virtual Directed Evolution

3.1 Abstract

3.2 Introduction

3.2.1 Background

Herbicide Resistant Crops

Herbicide-resistant crops are important for global agriculture because they mitigate yield losses due to weeds and give farmers extra flexibility in their herbicide application programs, which is important to suppress emergence of herbicide-resistant weeds.

Herbicides kill plants by inhibiting key metabolic processes and their species-specificity is determined by susceptibility of herbicide target and their ability to metabolize the herbicide. HPPD inhibitors are a key herbicide class that cause leaf bleaching and death in susceptible plants. HPPD inhibition disrupts tyrosine catabolism which disrupts UV-protection via carotenoid production and photosynthetic electron shuttling via plastoquinone, leading to death by UV damage and radical toxicity.

Engineering HPPD-inhibitor resistance into plants have used the HPPD and metabolic enzymes from naturally resistant species like *Avena fatua*, which employs cytochrome P450 Cyp72A1 to initiate metabolism of mesotrione by ring hydroxylation at C_5. In this case, the C_5 hydroxylation acts as a target site for glutathione-S-transferases which conjugate glutathione to xenobiotics. The glutathione conjugate tags the xenobiotic for sequestration in the cell vacuole, which neutralises the threat.

Engineered Cyp72A1 has been explored as a means of HPPD herbicide in soybean, which is an important target recipient for HPPD resistance traits.

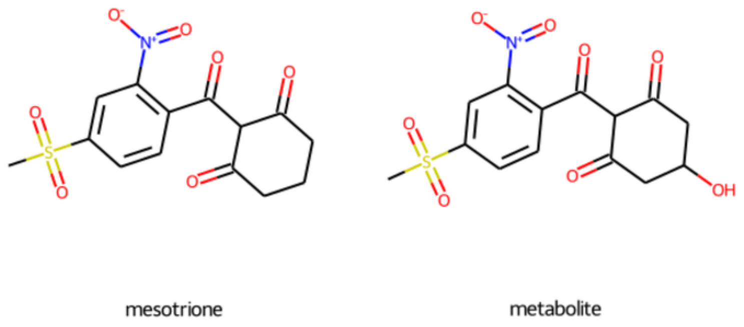


Figure 3.1: The HPPD inhibiting herbicide mesotrione and its primary metabolite 5-hydroxy-mesotrione in resistant strains of *A. fatua*.

Cytochrome P450s

Cytochrome P450s are a ubiquitous class of heme-dependent oxido-reductases that are frequently involved in xenobiotic metabolism. Bacterial P450s have been engineered to catalyse a range of xenobiotic biotransformations. The bacterial P450 BM3 from *Bacillus megaterium* is one such bacterial P450 whose structure has been studied extensively. The A82F/F87V mutant has a broad substrate specificity, however it has no activity towards the HPPD herbicide mesotrione.

Virtual Directed Evolution

Enzymes can be designed computationally using a genetic algorithm that evaluates the fitness of mutants by simulating the interaction between a target substrate and the predicted structure of the mutant.

The structure of a mutant can be predicted based on a template using techniques such as side-chain repacking by stochastic sampling from rotamer libraries and loop remodelling by cyclic coordinate descent.

Binding site interaction can be predicted using molecular docking, which attempts to predict likely protein-ligand binding conformations. A combination of the energy score and the conformation of docked molecules can be used to estimate likelihood of desirable reaction and therefore the fitness of a mutant. In rounds of selection within a genetic algorithm, the fitness of a batch of mutants is evaluated by scoring desirability of protein-ligand binding, the fittest mutants are selected for breeding, in which mutants have elements of their genes recombined are further mutated, then the cycle repeats.

3.2.2 Technologies Used

Directed Evolution

Structure-Based Design

Protein Structure Prediction

Docking

Sequence Optimization Algorithms

Overview of this work

Engineering Problem

3.2.3 Overview of this Work

Here, in attempt to engineer a mutant of the Cytochrome P450 BM3 to hydroxylate mesotrione at the C_5 position is made by developing a *VDE* system, deploying it at scale on cloud infrastructure and identification on clusters of putatively active mutants.

3.3 Methods

The project was operated as a `git` repository which can be found at:

https://github.com/jamesengleback/vde. The structure of the directory is:

- **docs/:** write up for this document and markdown documentation
- **nb/:** jupyter notebooks used for data analysis
- **scripts/:** scripts to create and configure cloud machines to run algorithm on
- **vde/:** the vde algorithm configured to optimize BM3 for desirable mesotrione binding

This section details the implementation of this project:

- The project is dependent on a `python` package `enz`, developed here for protein structure prediction and molecular docking to predict the behaviour of mutants; described in 3.3.1.
- A score function that attempts to predict the likelihood of a C_5 hydroxylation of mesotrione is described in section 3.3.2.
- A genetic algorithm to optimize the sequence of BM3 mutants is discussed in section 3.3.3
- Section 3.3.5 describes execution of the algorithm at scale on cloud infrastructure.

Figure 3.2: A short program that uses `enz` to predict the structure of a BM3 mutant, dock a fatty acid to it and save the result.

```
import enz
sequence = 'MKTIKEM...'
p = enz.protein('4KEY.pdb',
                seq=sequence,
                cofactors=['HEM']) # 1. - initialization
p.mutate(181, 'S') # 2. mutation
p.mutate(87, 'A')
p.refold() # 3. refolding
results = p.dock('CCCCCCCCCCCC=O',
                 target_residues=[49, 75, 87, 181, 263, 330, 400]) # 4. docking
results.save('docking-run-fatty-acid') # save docking results
```

3.3.1 `enz`

Abstraction and modularization of protein structure prediction and molecular docking was important for reducing complexity of experiments and developability of the algorithm. To this end the `python` package `enz` was created, an *Application Program Interface (API)* wrapper around the *PyRosetta* [8] protein structure prediction software and the *Autodock VINA* [9] binary, as well as utilities to handle file format conversion using *OpenBabel* [10]. The package is modular enough to allow replacement of its functionality-providing back-ends according to a users requirements and is hosted at https://github.com/jamesengleback/enz.

`enz` performs the following functions:

- **Protein Structure Prediction:** `enz` uses side chain repacking [11] functionality from *PyRosetta* for template-based structure prediction. This functionality is provided by *Pyrosetta*.
- **Docking:**
- **Return new atomic coordinates:** via `pandas` DataFrames, which can be used to score pose configurations.

The user-exposed command set is minimal so programs written using `enz` can be short. Figure 3.2 shows a short `python` program using `enz` to predict the structure of a new BM3 mutant, dock a fatty acid to it and save the results. A key aspect not shown in 3.2 is the accessibility of molecular data like coordinates, which are essential in this work for calculating distances between ligand and protein atoms and determining the score of the mutant.

3.3.2 Score function

Given the aim of engineering a BM3 mutant capable of C_5 hydroxylation of mesotrione, and given that likelihood of electron transfer is proportional to $\frac{1}{d^6}$ [12], the objective of the score function is to select for mutants that promote a mesotrione binding conformation where C_5 is close to the heme iron with a high affinity. The C_5-heme iron distance is noted as d measured in Å. Poses with a low d should also be stably held with a low ΔG, which is estimated for each pose by *VINA*.

For a set of mutant-bound poses of mesotrione, their collective score could be:

$$score = \frac{1}{n}\sum_{i \in n}^{n} \Delta G_i \times d_i \tag{3.1}$$

Another important factor in the score function is the *Hamming Distance* h between the sequence of the mutant being scored and the template sequence. Low h can make DNA synthesis by a set of site directed mutagenesis reactions possible, or reduce the size of degenerate codon libraries by reducing the number of mutation sites. A low h is also important here because the structure prediction is purely template-based and a high h could result in a very different structure in actuality. So h was added to the score function, which became score function A to be used in experiment A (equation 3.4).

$$score = \frac{1}{n}\sum_{i \in n}^{n} d - \log|\Delta G_i| - h \tag{3.2}$$

Low ΔG poses represent those more likely to occur, so their d should be weighted according to ΔG.

The heuristic currently employed to estimate the desirability of each set of m docking results is described in equation 3.3:

$$score = \frac{1}{n}\sum_{i \in m}^{n} \Delta G_i \times d_i \tag{3.3}$$

where ΔG is a free energy estimation of the interaction calculated by *Autodock VINA* (given in *kcal/mol*) and d is the distance between the heme iron and the C_5 of mesotrione for each of m binding poses **(figure 3.3)**.

3.3.3 Genetic Algorithm

A simple genetic algorithm *(GA)* was used for sequence optimization during *VDE*. The *GA* was implemented in pure `python` and its built-in modules.

In this case, the *GA* repeated the following steps in each iteration:

1. **Initialize mutant population:** From the template sequence, generate p mutants each with one random point mutation.
2. **For n Iterations:**
   (a) **Evaluate *fitness* of each mutant:** Using multiprocessing, evaluate the score for each mutant in parallel, returning a mapping of sequences to respective scores.
   (b) **Select for best $\frac{1}{m}$ mutants:** where $\frac{1}{m}$ is the survival rate in each iteration.

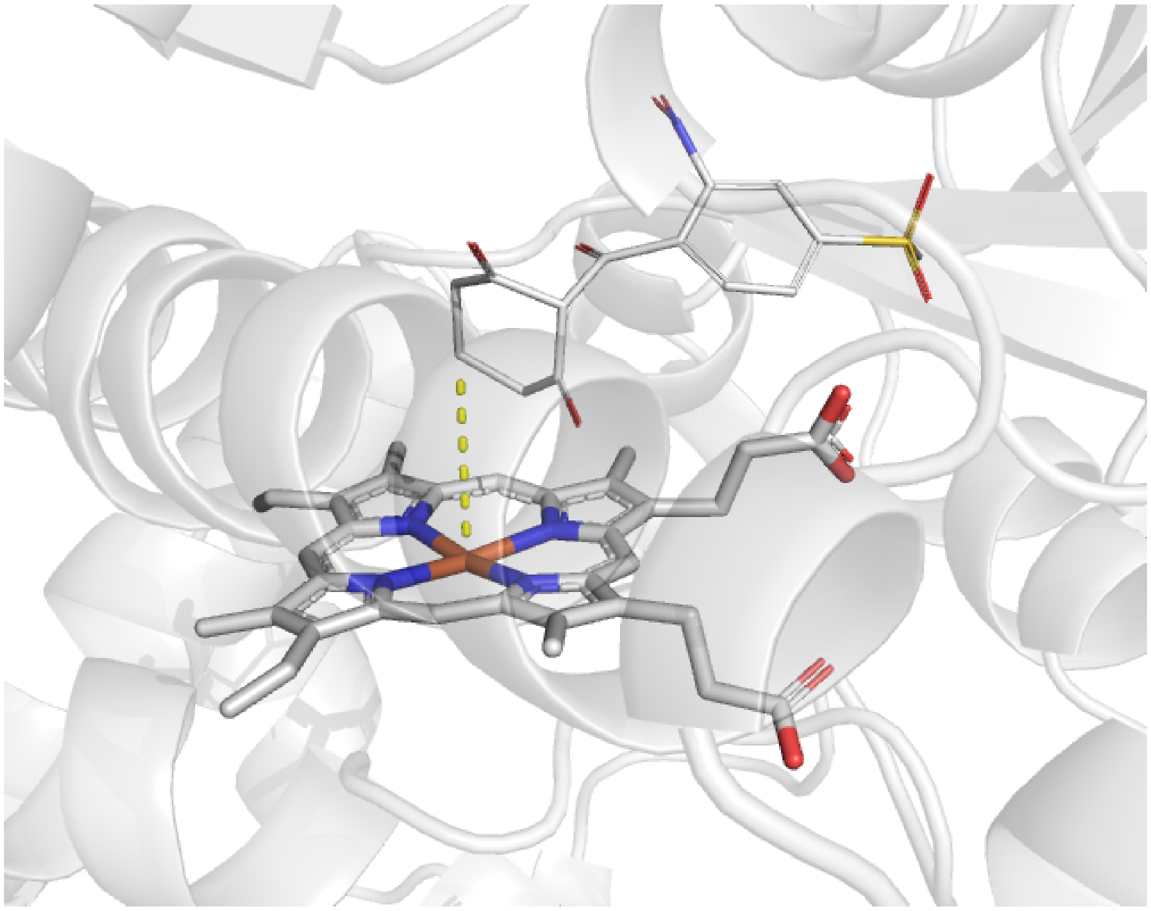


Figure 3.3: - Distance d between carbon C_5 of mesotrione and the heme iron of BM3, used in the fitness score (Å) marked by a yellow dashed line.

(c) **Repopulate gene pool by crossover and point mutation of selected mutants:** where two random members of the surviving mutants a and b are crossed by recombining sequences at a random cut point and introducing additional random point mutation. Repeat p times.

Algorithm 1 : A genetic algorithm

```
procedure GA($sequence, n_{mutants}, n_{iter}, n_{survivors}$)
    for $i := 1; i < n_{mutants}; i_{++}$ do                ▷ Initialize a population of size $n_{mutants}$
        $genePool_i := mutate(sequence)$                 ▷ Random substitution at random position
    end for
    for $i := 1; i < n_{iter};$ do                                ▷ For each generation
        for all $mutant_j \in genePool$ do                ▷ Map fitness function to each mutant
            $fitness_j := fn(mutant_j)$                         ▷ where $fitness$ is mappable
        end for
        $genePool := nlargest(fitness, n_{survivors}))$        ▷ Select $n_{survivors}$ best sequences
        for $i := 1; i < n_{mutants}; i_{++}$ do            ▷ Repopulate gene pool with new mutants
            $newGenePool_i := mutate(crossover(genePool_{random}, genePool_{random}))$
        end for
        $genePool := newGenePool$
    end for
end procedure
```

Algorithm 1 is implemented in `python` in the file `vde/ga.py` and makes use of multiprocessing to parallelise evaluations of a function.

3.3.4 Main Function

The program in `vde/main.py` executes the main functionality of *VDE*. It executes iterations of the genetic algorithm 3.3.3 where the evaluation function for a $sequence$ is:

Algorithm 2 : One fitness evaluation

```
procedure EVALUATE MUTANT($sequence$)
    structure = map_refold(sequence, pdb=4KEY.pdb)    ▷ Predict mutant structure [11] [8].
    docking poses = dock(structure, mesotrione)                 ▷ [9]
    fitness = score(docking poses)                              ▷ Using score 3.3.2
    Return fitness
end procedure
```

Two instances of the *VDE* program were run, each with a different score function and labeled A and B. The score function for A was as in equation 3.4:

$$fn_A(a, d, h) = d - \log|a| - h \tag{3.4}$$

Where variables as in section 3.3.2. The score function for B is described in equation 3.5:

Table 3.1: The parameters used in experiments A and B.

Parameter	**Value**
Template Structure	*4KEY* [5]
Mutation Sites	47, 49, 51, 75, 78, 88, 94, 138, 142, 175, 178, 184, 188, 205, 226, 252, 255, 260, 263, 290, 295, 328, 330, 350, 353
Population Size	128
Survival Rate	$\frac{1}{4}$
Docking Exhaustiveness	16
Number of Generations	32
Number of Repeats	8

$$fn_B(a, d, h) = (\frac{1}{n} \sum_{n}^{i} softmax(\log |a|_i) \times d_i) - h \tag{3.5}$$

Asides from the differing score functions, the two experiments were set up to be run with the configuration in table 3.1:

- The *Mutation Sites* were chosen manually from the crystal structure of P450 BM3 A82F/F87V mutant `4KEY` [5], the template structure for these experiments.
- *Docking Exhaustiveness* is 16: the maximum value allowed by *VINA*, which is computationally intensive but yields results with lower variability.
- *Population Size* is 128, this was chosen because monitoring CPU usage was observed to be low in instances run with large *Population Sizes* in a single process, so to maximize CPU usage, eight replicates were executed in separate parallel processes instead.
- *Survival Rate* is $\frac{1}{4}$, where the top scoring fraction survive to repopulate the gene pool.

Experiment A was run using code in the `git` branch `x2` on commit `????`, and experiment B was run on branch `x3` and commit `e7af345fba8b8be36bea25982af0b978df8267e8`.

Each was run in eight parallel instances using the script `vde/vde.sh`, which launched `vde/main.py` with the command `python main.py -fn b -p 128 -e 16 -n 32 -s 0.25 &`, which invokes the parameters in table 3.1.

3.3.5 Cloud Deployment

Each `g6-dedicated-50` was provisioned with `evo/cloud/build-linode.sh` and the configuration scripts in `evo/scripts/config`. The *Linode* instance had the virtual hardware specifications showing in table 3.2.

Table 3.2: Virtual hardware specifications of the `g6-dedicated-50` server hired for experiments A and B.

Item	**Specifications**
CPU cores	50
RAM	128 GB
Disk	250 GB
Cost per hour	$1.44

Table 3.3: Virtual hardware specifications of the `g6-dedicated-8` server hired for analysis of experiments A and B.

Item	**Specifications**
CPU cores	8
RAM	16 GB
Disk	325 GB
Cost per hour	$0.18

Each experiment lasted about 3 days and cost $103 at a rate of $1.44 per hour. Data generated was structures, docking poses and scores for each mutant - it was compressed into a `.tar.gz` archive and uploaded to a *Linode* bucket and the machines were terminated.

The eight replicates were run in parallel, generating 32,000 mutants for each experiment.

Data was pulled down from the bucket storage onto a separate server for analysis with specifications listed in table 3.3.

3.4 Results

Each experiment A and B yielded near to 32,000 unique mutants. In each run, metrics were saved to a `csv` file containing the following headers:

- **gene:** The mutant sequence represented only by the amino acids at the *Mutation Sites* in table 3.1.
- **score:** The score A or B depending on which experiment is in question.
- **dist_mean:** The mean distance between the C_5 carbon of a mesotrione pose and the heme iron, for all poses.
- **aff_mean:** The mean binding energy estimate provided by *VINA* for all poses, given in *kcal/mol*
- **ham:** The Hamming distance, h between a given mutant and the template sequence.
- **uid:** A unique ID.

The distributions for A and B are compared in figure 3.4. For both, there are many mutants for whom $d < 8$ which may be suitable for the project aim.

Table 3.4: The mutations converged upon using *Virtual Directed Evolution*

Mutation
G47R
V49T
R51Y
A75L
F78V
V88T
E94K
E138H
V142P
I175T
M178V
E184A
K188L
Q205F
A226S
D252E
I255R
I260T
L263I
K290A
A295A
T328A
P330A
K350G
E353L

Between the two experiments, the distance d between carbon C_5 and the heme iron is largely similar, so too are the calculated affinities (ΔG) and Hamming distance to the wild type, h. Hamming distance h rarely exceeded 8, likely due to the constraint on h incorporated into both score functions. Docking binding energy estimates ΔG cluster at a set of values for both experiments which may indicate the same number of pose clusters with similar energies. For both, distance d was a skewed distribution that leans towards lower values, each with a small shoulder at around 15 Å, which may reflect either a cluster of binding poses or a cluster of mutants for which binding of mesotrione to the active site is blocked.

The repeats of each experiment converged towards a preferred mutation at each position. Figure 3.5a is a sequence logo which shows the amino acids at each mutation site for the template sequence BM3 A82F/F87V, whilst 3.5b and 3.5c represent the overall frequencies of different amino acids across experiments A and B respectively. These mutations are shown in table 3.4.

The exact set of 25 mutations in table 3.4 are never reached, since figure 3.4 shows that the *Hamming Distance* h never exceeded eight. **Mutation Correlation**

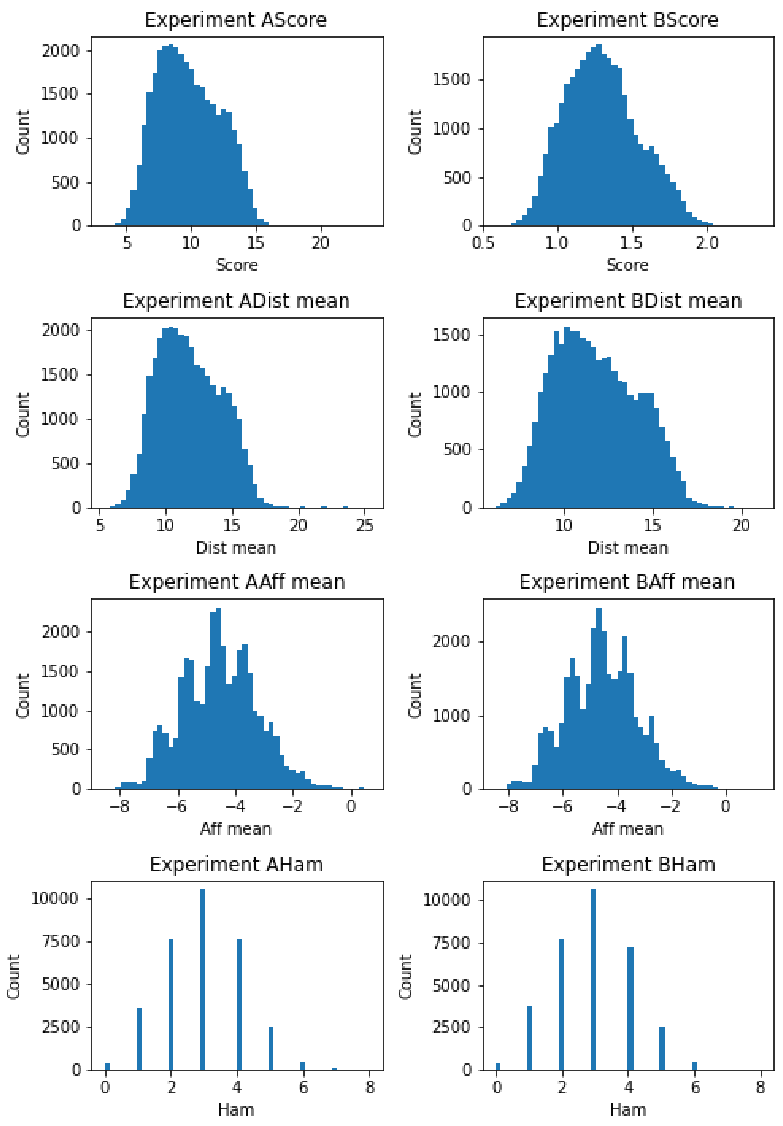
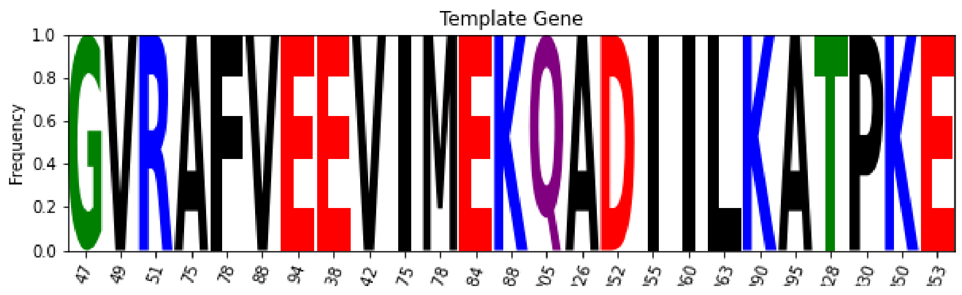
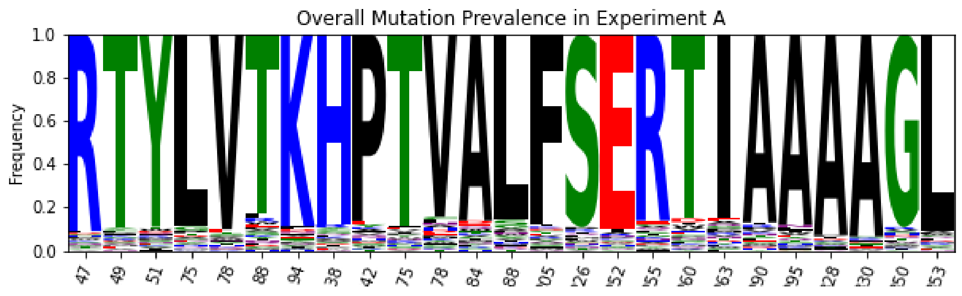


Figure 3.4: The distributions of each recorded metric across the entire experiment for both experiments A and B.

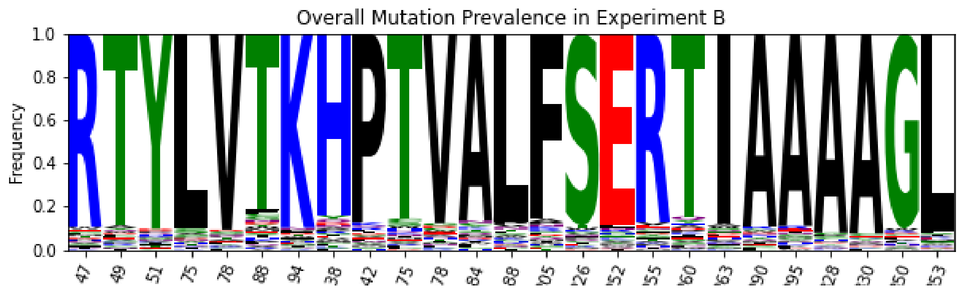
Figure 3.5: Sequence logos.



(a) Sequence logo showing the original amino acids at mutation positions in experiments A and B.



(b) Sequence logo showing the overall amino acid frequencies in mutants in A.



(c) Sequence logo showing the overall amino acid frequencies in mutants in B.

Several clusters of mutants were identified using dimensionality reduction and clustering algorithms. Mutant sequences were reduced in dimensionality using both the *t-SNE* [13] and *UMAP* [14] algorithms to decompose an input of the binary encoded amino acids at each mutation position. The reduced sequence information was plotted in two ways:

1. Colour mapping score to each point, in an attempt to identify clusters with an especially high score (figure 3.6).
2. Colour by point density (figure 3.7).

At the center of each dimensionality reduction map are the sequences that were repeatedly converged upon, visible by an area of low score (desirable). When colored by point density the same low scoring centers of the mapping are also densely populated. In order to confirm that the solutions in the centre were converged upon a generation number color map would be necessary, though was not possible since generation numbers were not recorded.

Whilst it is possible that the converged mutations are sufficient for creating a BM3 mutant with target activity, it is also possible that optimizing sequences for these scores exploited an inaccuracy in the model which would result in unintended consequences, such as large structural changes that can not be predicted by the model. To mitigate the risk of a failed round of lab screening, it is prudent to identify other clusters with activity in order to introduce diversity into the screening set. Clusters were made directly from the reduced dimensions using the *DBSCAN* algorithm, exemplified in figure 3.8.

From these clusters, the best scoring eight were selected, their amino acid frequencies calculated and degenerate codons designed to reflect that frequency using the *Codon Genie API*. Reports for each were generated and an example is shown in figure 3.9a.

The codons designed for exemplary cluster 314 are in table 3.5. In this manner codons were designed for clusters in experiments A and B, the data and reports for which are stored in `nb/codons/` of the project `git` repository. The total size of each library yielded by the codon design varies from four to several thousand, the latter of which are not suitable for use owing to the extreme numbers of potential variants. Some libraries also inadvertently encode a *Stop* codon, which should also be avoided.

3.5 Discussion and Future Work

Here, a working *Virtual Directed Evolution* system was developed, including the simulation back end `enz`. The system used a genetic algorithm to optimize a set of mutation points in the Cytochrome P450 BM3 for a mesotrione binding metric that aimed to select for mutants that dock mesotrione with the C_5 carbon of mesotrione held stably near the BM3 heme iron. It was run on cloud infrastructure at a scale of 64,000 mutants over a three day period. From the output of the algorithm a set of mutations with potentially desirable effects were identified, given the aim of engineering a BM3 mutant capable of mesotrione 5-hydroxylation. Additionally, mutant clusters were found amongst the results of the algorithm, for the most active of which sets of degenerate codons were designed.

The system was designed to be modular enough so that components can be modified independently of one another. As such it is important to review these components with a view to

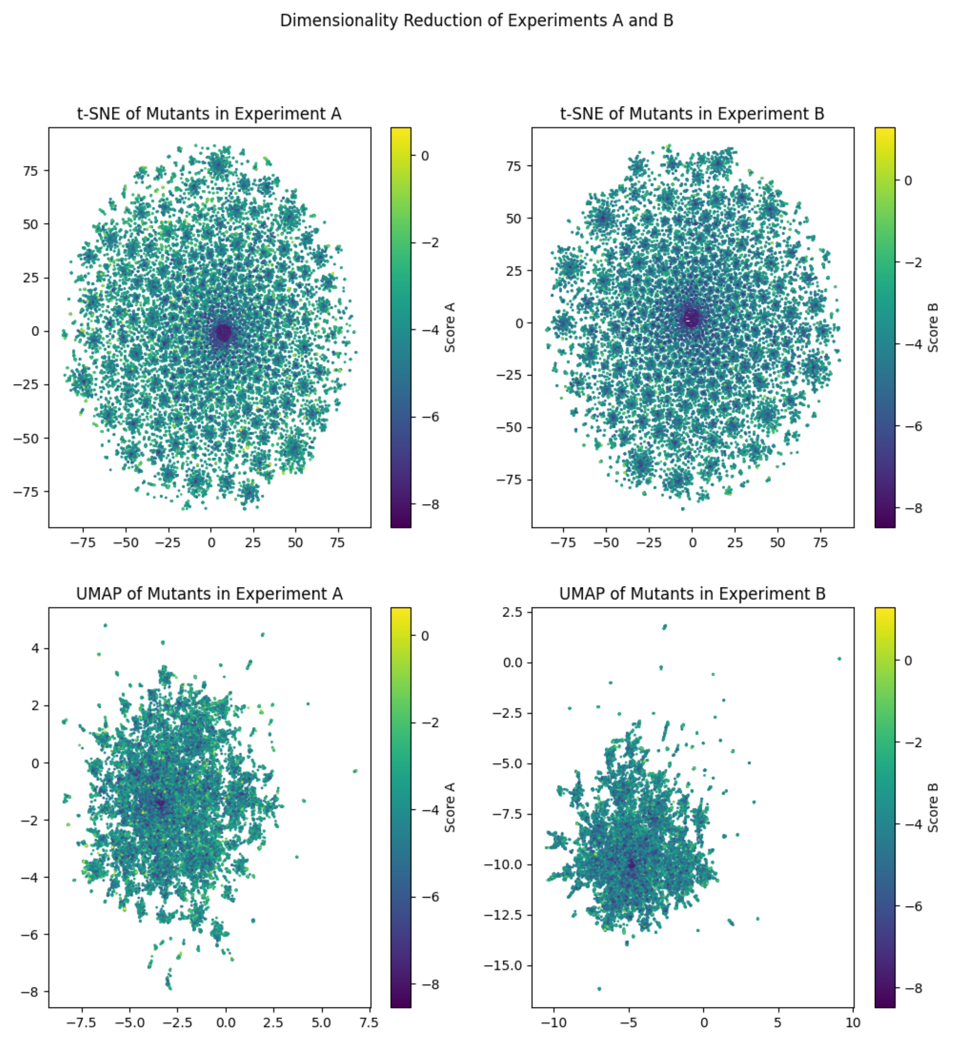

Dimensionality Reduction of Experiments A and B
t-SNE of Mutants in Experiment A
t-SNE of Mutants in Experiment B
UMAP of Mutants in Experiment A
UMAP of Mutants in Experiment B
Score A
Score B

Figure 3.6

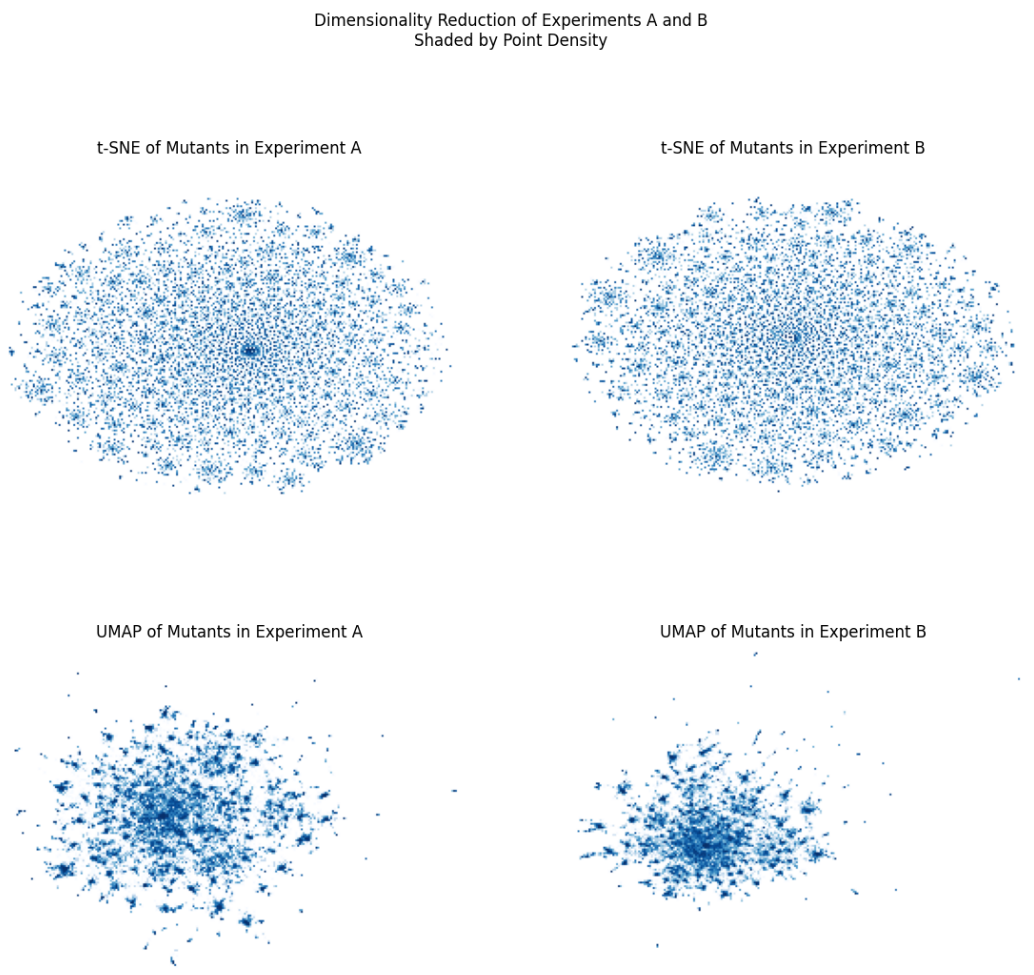

Dimensionality Reduction of Experiments A and B
Shaded by Point Density
t-SNE of Mutants in Experiment A
t-SNE of Mutants in Experiment B
UMAP of Mutants in Experiment A
UMAP of Mutants in Experiment B

Figure 3.7

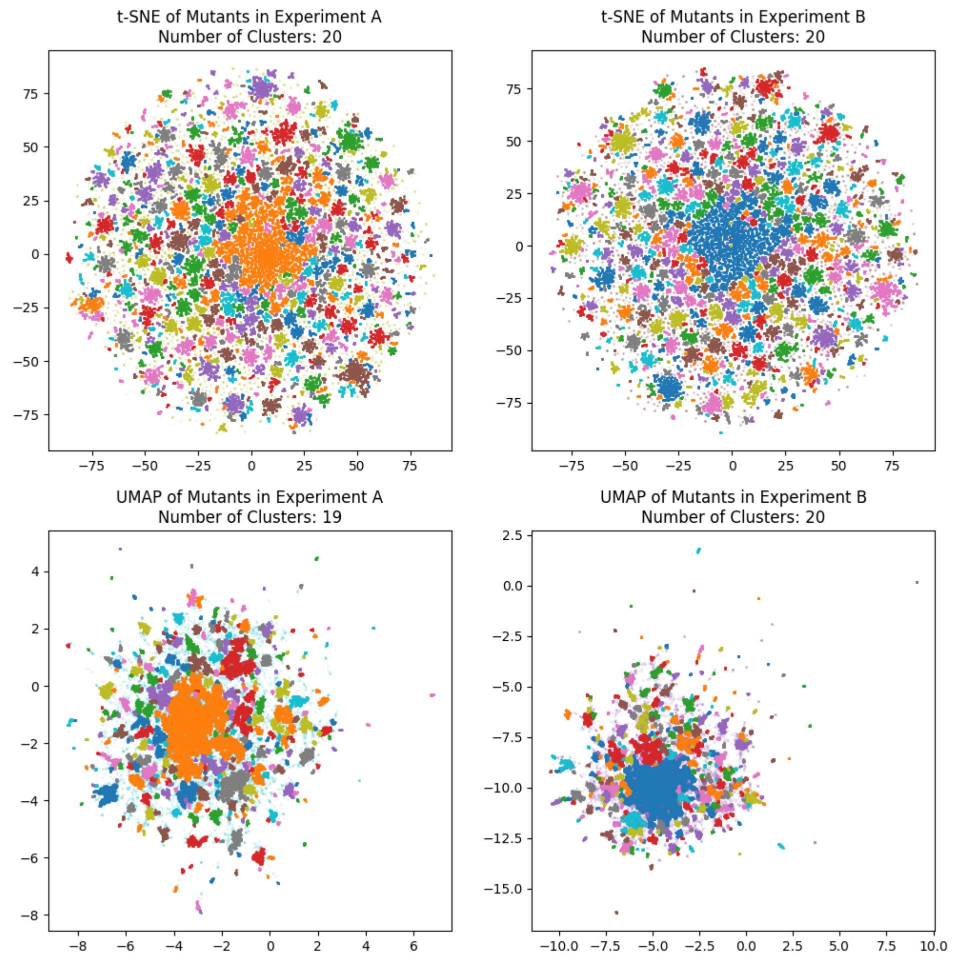
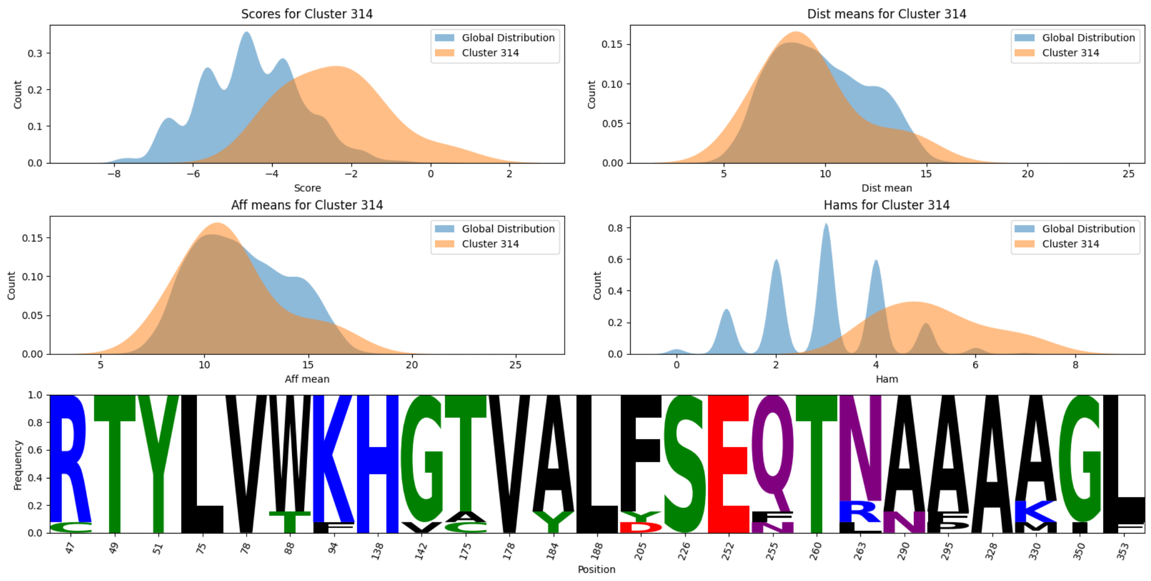


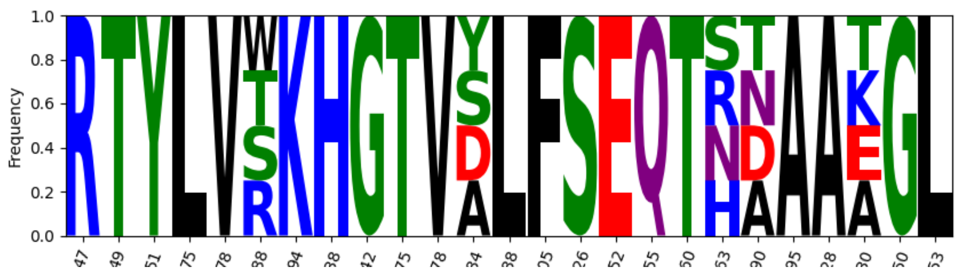
Figure 3.8: Clusters of dimensionality-reduced mutants. Color codes to clusters are not shown.

Table 3.5: An example of the codons auto-generated for mutant cluster 314, which encodes few variants. A total of 1024 variants are encoded by this set of codons.

Position	Codon	Expansion	Number of Encoded Variants	Amino Acids
47	CGT	CGT	1	R
49	ACC	ACC	1	T
51	TAT	TAT	1	Y
75	CTG	CTG	1	L
78	GTT	GTT	1	V
88	WSG	ACG, AGG, TCG, TGG	4	T, W, R, S
94	AAA	AAA	1	K
138	CAT	CAT	1	H
142	GGT	GGT	1	G
175	ACC	ACC	1	T
178	GTT	GTT	1	V
184	KMT	GAT, GCT, TAT, TCT	4	A, Y, D, S
188	CTG	CTG	1	L
205	TTT	TTT	1	F
226	AGC	AGC	1	S
252	GAA	GAA	1	E
255	CAG	CAG	1	Q
260	ACC	ACC	1	T
263	MRT	AAT, AGT, CAT, CGT	4	N, R, H, S
290	RMT	AAT, ACT, GAT, GCT	4	A, N, D, T
295	GCA	GCA	1	A
328	GCA	GCA	1	A
330	RMA	AAA, ACA, GAA, GCA	4	A, K, E, T
350	GGT	GGT	1	G
353	CTG	CTG	1	L



(a) A codon report for cluster 314 derived from a *UMAP* of the sequences in experiment A. Shown are the cluster distributions of each recorded metric relative to the global distribution. The sequence logo represents the mutation frequency at each position.



(b) The amino acid coverage of the degenerate codons designed for cluster 314, which include off-target amino acids at several sites.

fixing and upgrading them.

The most critical uncertainty and potential shortcoming of this implementation of *Virtual Directed Evolution* is whether the simulation evaluation used is a meaningful metric to optimize for. The accuracy of the simulation in portraying relevant aspects of the enzyme-ligand interaction is critical to be applicable to the world outside the simulation. So too is the meaningfulness of the score function which is to be optimized for, which in this case is only designed as a proxy to desired chemical activity.

Less critical, though with operational significance is the efficacy and efficiency of the sequence optimization algorithm used. Large gene pools improve the performance of genetic algorithms, so scale is important. Slightly out of scope of *Virtual Directed Evolution* is codon design, which must yield libraries that are small enough to screening and rich enough in activity to offset the high cost of making and testing the mutants.

These key parts of the project are dissected further for potential improvements:

Simulation Accuracy

In as far as a simulation is a time series approximation of a real world situation, the simulation here is a misnomer for referring to prediction of mutant structures and likely binding poses between them and mesotrione before being subject to some score function. That said, the simulation used in this work is crude owing to its lack of protein backbone movement during structure prediction and lack of interaction dynamics between protein and ligand. These two functions, protein structure prediction and docking were contained by `enz`, which abstracts these two functions in its *API* and therefore the scope of improvements to simulation accuracy are within `enz`.

There are several aspects of `enz` that can be improved over different time scales. One short and simple improvement is unit testing of all code in the module to ensure that the code behaves as expected. There is currently no unit testing in `enz` which limits the ability to diagnose problems with it.

Since `enz` itself is a simple wrapper around protein structure prediction and docking programs, those programs can be replaced or modified. *PyRosetta* proved problematic in deployment owing to its licensing and package size, which necessitates distribution of *PyRosetta* to target machines via direct file transfer as opposed to download directly from the world wide web. This was done here by storage of a copy in a *Linode* bucket storage in Hamburg which itself required authentication with the `linode-cli` interface, a step that would require careful planning were it to be fully automated given that access tokens would be in play.

PyRosetta contains lots of functionality, but only a single function - side chain repacking [11] is used. Cyclic coordinate descent as a means of predicting loop structure [15] was investigated but subsequently dropped owing to the difficulty of the software implementation, which would often result in *Segmentation Faults* that crash the process by attempting to access out of bounds memory, something inherent to *PyRosetta* itself. *Segmentation Faults* were commonplace in investigating other folding methods, which combined with sparse documentation coverage and a license which incurs an annual cost to non-academic users raises a case for exploring other protein folding methods.

Recent advances in the use of machine learning in protein structure prediction [16] have outperformed their non-learning counterparts. Though the *de-novo* structure prediction of *Alphafold* [16] outperforms all known structure prediction methods to date, since it is such a large model it requires GPU or TPU cores to function and may take some hours on a single prediction. *De-novo* protein structure prediction may be unnecessary for this application, since the protein backbone chain may be similar among mutants.

Template-based protein structure prediction using machine learning methods may on the other hand be more viable, given the shorter prediction times involved. An additional advantage of using a lightweight machine learning-based method is the prospect that inference can be accelerated significantly using a GPU or TPU. One candidate for replacement of the *PyRosetta* back end is `torchmd` [17]: an open-source *PyTorch*-based molecular dynamics package which would allow side chain replacement and subsequent repacking via either energy minimization using a force field such as *Amber*[18]. Being based on *PyTorch*, there is scope for accelerated calculations using GPUs, and it also offers compatibility with other *PyTorch*-implemented protein structure machine learning methods.

Docking using *VINA* is CPU-bound and the most time-consuming step of each round of the *Virtual Directed Evolution*. This can be accelerated with more CPU cores, or employment of a GPU-based docking program like *Autodock GPU*[19], which is free and open source. Alternatively, emerging docking methods based on machine learning may be worth consideration given their performance [17]; should these methods be implemented in *PyTorch* then the entire protein structure prediction and docking pipeline could be carried out on a GPU without writing to disk or converting file formants.

The structure of the `enz` wrapper allows for drop in replacements of each component, whilst also providing a coherent data structure for user interaction and analysis which is essential for scoring. The key priorities for improvements to `enz` are:

- **Accuracy:** To assess accuracy, a task-relevant baseline needs to be established. In the context of an enzyme engineering project, this could be the results of a lab screen of the mutants generated here. In that case, correlation between simulation predictions and lab metrics can be quantified using an appropriate metric like Pearson's correlation coefficient. With a metric to optimize towards, changes can be made to the underlying simulation mechanisms of `enz`. An alternative, cheaper source of baseline data is the *PDBBind*[20] dataset - a collection of ligand bound protein structures for which the task would be to replicate these bound configurations using docking. Differences between poses generated by docking and those in the *PDBBind* dataset can be quantified using a metric like root mean squared difference (RMSD) between each atom.

- **Performance:** Both *PyRosetta* and *VINA* are CPU-bound, which limits the speed at which mutants can be evaluated because of the inherently linear nature of CPU processing. Should alternatives for both be found they would ideally run with GPU acceleration, which is inherently parallel. *Autodock GPU*[19] is an attractive short-term replacement for *VINA* given its performance. Beyond short term changes, structure prediction and docking algorithms implemented in *PyTorch* are attractive because of the prospective GPU acceleration and compatibility with machine learning algorithms.

- **Portability:** Installing the required environment on the target machine with minimal intervention is important for scaling to multiple machines. Therefore, the *portability* of the requirements - in terms of ease of data automated data transfer and setup is important. *PyRosetta* is not free for non-academic users, which makes it more difficult to find on the clear web and is a reason to not use it in a commercial setting. It also relies on distribution of compiled binaries that may not be suitable for a particular CPU architecture or operating system. It is also large - with the compressed release reaching 1.3 GB, which is uncomfortably large given that only a small subset of its functionality are employed.

Score Relevance

The score function optimized towards during runtime is critical for output of mutants with the desired activity in the real world. It uses a set of assumptions as a proxy metric for likelihood of C_5 mesotrione hydroxylation:

1. That the C_5-heme iron distance d must be minimized in order to maximize the likelihood of the desired reaction. This is based on the assumption that C_5-heme iron proximity will drive the target reaction. It also assumes that the chemical potential between the two atoms is sufficient to drive electron transfer from the BM3 reductase domain to C_5.
2. That the estimated binding ΔG energy of a pose to BM3 relates to the likelihood of that pose occurring.
3. That a low Hamming distance h between a mutant sequence and the template indicates that the mutant structure would not be so different from the template that the protein structure prediction methods used are sufficiently accurate.

If these assumptions hold, then the score function may be sufficient as is in experiments A and B. The ideal scenario for evaluating the efficacy of the score function is to compare it to lab data containing BM3 mutants and data on the products formed on reaction with mesotrione, if any and the rate at which they are formed.

Since the score is only a simple proxy in its current states, in future design iterations in conjunction with lab data to mimic it will likely grow in complexity.

Sequence Optimization

Though a genetic algorithm was used as a sequence optimizer in this work, others can be used and the genetic algorithm itself stands to be modified to improve efficiency. A genetic algorithm was implemented because they are simple and inherently parallelisable, which suits horizontal scaling of processes. Many evolutionary algorithms have been studied in other work, which represents a rich field from which to harvest upgrades.

Some changes that can be made include:

- **Tournament Selection:** Rather than selection of the n best performing mutants, random pairs are compared for score and the best performing survives. This introduces an element of randomness which can enhance diversity and avoid *local minima trapping*, where a good solution is found in a small subset of sequence space but is far from the global optimum for the given constraints.

- **Randomness:** As with tournament selection, an element of randomness in the selection step can help avoid *local minima trapping* in a similar manner to tournament selection.
- **Generation Persistence:** The implementation used here only carries the current generation in memory, so if a cluster of good mutants are found in an earlier round there is a risk of *forgetting* by evolutionary divergence from those sequences. A solution to this could be that the best performing mutants within and between experiments are cached and repopulation stems from this pool.

Again, a benchmark for performance of these algorithms must be established to update and improve it. The benchmark must be task relevant in that it approximates the type of fitness landscape expected for protein sequence optimization, so it may be best done with the current simulation method. In this case, the metric for improvement can be some formulation of:

- Rate of fitness improvement $\frac{\delta f}{\delta t}$, where the area under an curve interpolated through points of generation numbers against fitness scores can represent this rate. Alternatively, the initial gradient of the interpolated curve may also be a suitable metric for rate of fitness improvement.
- Maximum fitness attained by the algorithm may be a good indicator of whether the algorithm is subject to *local minima trapping*, which itself may be a good proxy metric for sequence exploration.
- Exploration of new sequences is important to avoid *local minima trapping* and to avoid re-visiting sequences. This can be quantified using a sequence diversity metric of all mutants tested over the course of an experiment, such as the sum of pairwise h between all sequences.

Many improvements within the scope of the implemented genetic algorithm stand to be investigated, other sequence optimization algorithms are worth investigation. Namely, *Bayesian Optimization* is an attractive candidate for this given its efficiency and applicability to sequence optimization. *Bayesian Optimization* uses an internal *Bayesian* model of response to action space input in order to generate new inputs based on their expected fitness $\mathbb{E}f$ and expected information gain $\mathbb{E}IG$.

Scale

Scale is important to evolutionary algorithms, where a large gene pool in each generation leads to more diversity explored. Even using *Bayesian Optimization*, a large batch of candidates evaluated in parallel will likely yield a solution faster than sequential evaluations. Therefore, scaling to a large number of parallel evaluations is important. Using cloud resources is ideal for rapid and elastic scaling, where instances are provisioned for the duration of the experiment and then terminated, reducing costs massively compared to the overhead cost of purchasing and maintaining the equivalent computing hardware. It also allows immediate scale to n machines with m computing cores.

That said, in this work the algorithm was run on a single, large machine which tool some semi manual setup using a set of scripts included in this repository. Scripts were required because inclusion of *PyRosetta* in the project would push the size of a disk image of the machine above the upper limit of that allowed to be saved, stored and distributed using *Linode*. Were this process able to boot directly from a disk image, setup could be fast and automated.

To increase scale beyond that used in this project, each round of evaluations may have to be distributed among multiple machines. This can be done using a hub and spokes model where the sequence optimization algorithm resides in the hub node and the more expensive evaluations are sent to the spoke nodes. This can be coordinated using the *Kubernetes* engine, which is designed to orchestrate and distribute jobs amongst an array of virtual machines. *Kubernetes* can then be used for arbitrary horizontal scaling and coordination of clusters of machines.

Codon Design

Though slightly out of scope of *Virtual Directed Evolution* itself, efficient codon design is essential for downstream applications. Here several sets of codons were designed for downstream experiments in the lab, the results of which can be used to improve the algorithm. Codons were designed as degenerates, each potentially encoding a selection of amino acids.

An important consideration in codon design are the library size, which can be calculated by taking the product of the number of amino acids that each degenerate codon encodes. Library size constraints are determined by the capacity of downstream screening or selection processes. If screening based on colonies picked from the transformed libraries is used then the cost of the screen scales with the number of mutants, in which case a library size on the order of hundreds may be advisable. On the other hand if the downstream process is a selection screen, then constraint on library size may be imposed by the transformation efficacy into the selection cells, which may be on the order of 10^4, though repeating transformations and subsequent selection steps can be cheap, so a library size on the order of 10^5 may be suitable.

Although the codons themselves are designed here, no additional designs towards DNA assembly are made, for example primer design. Should this functionality be improved, it represents project creep and will best be compartmentalized within another project.

Bibliography

[1] Syngenta Participations AG Timothy Robert HawkesBernardus Theodorus Maria Vernooij. “EP 2164320 A2: Cytochrome p450 genes conferring herbicide resistance”. 2007.

[2] Andrew W Munro et al. “P450 BM3: the very model of a modern flavocytochrome”. In: *Trends in biochemical sciences* 27.5 (2002), pp. 250–257.

[3] Jesse D Bloom et al. “Protein stability promotes evolvability”. In: *Proceedings of the National Academy of Sciences* 103.15 (2006), pp. 5869–5874.

[4] Christopher JC Whitehouse, Stephen G Bell, and Luet-Lok Wong. “P450 BM3 (CYP102A1): connecting the dots”. In: *Chemical Society Reviews* 41.3 (2012), pp. 1218–1260.

[5] Christopher F Butler et al. “Key mutations alter the cytochrome P450 BM3 conformational landscape and remove inherent substrate bias”. In: *Journal of Biological Chemistry* 288.35 (2013), pp. 25387–25399.

[6] Andrew Ng. “Nuts and bolts of building AI applications using Deep Learning”. In: *NIPS Keynote Talk* (2016).

[7] Roshan Rao et al. “Evaluating Protein Transfer Learning with TAPE”. In: *bioarxiv* (2019).

[8] Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J Gray. “PyRosetta: a script-based interface for implementing molecular modeling algorithms using Rosetta”. In: *Bioinformatics* 26.5 (2010), pp. 689–691.

[9] Oleg Trott and Arthur J Olson. “AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading”. In: *Journal of computational chemistry* 31.2 (2010), pp. 455–461.

[10] Noel M O’Boyle et al. “Open Babel: An open chemical toolbox”. In: *Journal of cheminformatics* 3.1 (2011), pp. 1–14.

[11] Roland L Dunbrack Jr and Martin Karplus. “Backbone-dependent rotamer library for proteins application to side-chain prediction”. In: *Journal of molecular biology* 230.2 (1993), pp. 543–574.

[12] Christopher C Moser et al. “Distance metrics for heme protein electron tunneling”. In: *Biochimica et Biophysica Acta (BBA)-Bioenergetics* 1777.7-8 (2008), pp. 1032–1037.

[13] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[14] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[15] Adrian A Canutescu and Roland L Dunbrack Jr. “Cyclic coordinate descent: A robotics algorithm for protein loop closure”. In: *Protein science* 12.5 (2003), pp. 963–972.

[16] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589.

[17] Stefan Doerr et al. “Torchmd: A deep learning framework for molecular simulations”. In: *Journal of chemical theory and computation* 17.4 (2021), pp. 2355–2363.

[18] Junmei Wang et al. “Development and testing of a general amber force field”. In: *Journal of computational chemistry* 25.9 (2004), pp. 1157–1174.

[19] Diogo Santos-Martins et al. “Accelerating AutoDock4 with GPUs and gradient-based local search”. In: *Journal of chemical theory and computation* 17.2 (2021), pp. 1060–1073.

[20] Zhihai Liu et al. “PDB-wide collection of binding data: current status of the PDBbind database”. In: *Bioinformatics* 31.3 (2015), pp. 405–412.