Communicating Research to the General Public

The WISL Award for Communicating PhD Research to the Public launched in 2010, and since then over 100 Ph.D. degree recipients have successfully included a chapter in their Ph.D. thesis communicating their research to non-specialists. The goal is to explain the candidate's scholarly research and its significance—as well as their excitement for and journey through their area of study—to a wider audience that includes family members, friends, civic groups, newspaper reporters, program officers at appropriate funding agencies, state legislators, and members of the U.S. Congress.

WISL encourages the inclusion of such chapters in all Ph.D. theses everywhere, through the cooperation of PhD candidates, their mentors, and departments. WISL offers awards of $250 for UW-Madison Ph.D. candidates in science and engineering. Candidates from other institutions may participate, but are not eligible for the cash award. WISL strongly encourages other institutions to launch similar programs.

Wisconsin Initiative for Science Literacy

The dual mission of the Wisconsin Initiative for Science Literacy is to promote literacy in science, mathematics and technology among the general public and to attract future generations to careers in research, teaching and public service.

Contact: Prof. Bassam Z. Shakhashiri

UW-Madison Department of Chemistry

bassam@chem.wisc.edu

www.scifun.org

Machine Learning Based Protein Engineering for Microbial Chemical Production

By

Jonathan C. Greenhalgh

A dissertation submitted in partial fulfillment of the requirements for the degree of:

Doctor of Philosophy

(Chemical and Biological Engineering)

UNIVERSITY OF WISCONSIN-MADISON

2022

Date of Final Oral Examination: 04/08/2022

This dissertation is approved by the following members of the Final Oral Committee:

Philip A. Romero, Assistant Professor, Biochemistry, Chemical and Biological Engineering

Brian F. Pfleger, Professor, Chemical and Biological Engineering

Eric V. Shusta, Professor, Chemical and Biological Engineering

Andrew R. Buller, Assistant Professor, Chemistry

Srivatsan Raman, Assistant Professor, Biochemistry, Chemical and Biological Engineering, Bacteriology

Chapter 6

Application of machine learning-based protein engineering to make an improved acyl-ACP reductase enzyme

Author: Jonathan Greenhalgh

I believe science can be applied in ways that make a difference in people's lives. But that can only happen if science is communicated in a way that people can understand the findings and see ways to apply them. I wrote this chapter to present the results of my scientific work in a more accessible manner so that scientists and non-scientists alike can understand the findings and the process. In some respects, that makes this the most important chapter of my dissertation. I am grateful to the Wisconsin Initiative for Science Literacy (WISL) at UW-Madison for encouraging and enabling communication of science to broader audiences, and I am especially grateful to Professor Bassam Shakhashiri, Elizabeth Reynolds and Cayce Osborne for their support and feedback as I've worked on this chapter.

6.1 Introduction

6.1.1 Background and motivation

Ever since I took biochemistry class in college, I have found the chemistry of proteins fascinating. As a chemical engineering student, I was really interested in how proteins could be used to make chemicals in cells. I joined the lab of Phil Romero who studies protein engineering and machine learning, and collaborated very closely with Brian Pfleger, who studies ways to engineer microbial organisms (like bacteria and yeast) to make valuable chemicals. Combining these two disciplines led to the project that I worked on for most of my graduate studies, which is engineering an enzyme that can be used to make the fatty alcohol molecules that are commonly found in lotions and detergents. Protein engineering is a really exciting field of study and combining protein engineering and machine learning (basically using computers to help engineer proteins better), is even more exciting. In this chapter I'll explain in as simple terms as I can what protein engineering is, why we used it, and how machine learning helps the process using a specific example from my research[1].

6.1.2 Proteins and enzymes

To understand protein engineering, first I must explain a little bit about proteins. Proteins are molecules that are made up of smaller pieces called amino acids. There are twenty common amino acids, each with unique properties. This set of amino acids is like an alphabet; the amino acid alphabet contains twenty letters (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y), where each letter corresponds to a specific amino acid (A for Alanine, C for Cysteine etc., except it's not always the first letter of the amino acids

name). Each amino acid typically only sticks to two other amino acids, so when combined they form long chains that are kind of like words when we write out their one letter abbreviations (things like “WERNLPLDL…”). The order of amino acids in a protein also determines what the protein looks like, and importantly what it does, just like the order of letters in a word determines the word’s meaning. Enzymes are a class of proteins that carry out chemical reactions in cells to convert one molecule to another.

6.1.3 How protein engineering works

Sometimes, for whatever reason, it’s desirable to change what a protein or enzyme does, or make it do a specific task better. Making changes in the protein sequence will usually result in changes to the protein itself (though not always good ones). But how do we change the protein sequence? It turns out there are a lot of ways, but they all involve making changes to DNA in a cell. DNA also has an alphabet (only four letters, A, C, G, and T), but the letters in the DNA alphabet and the letters in the amino acid alphabet are quite different (DNA letters are called nucleotides or bases). DNA contains the instructions for making proteins in cells, and cells make proteins by translating messages from the language of DNA to the language of proteins. In this way, DNA is almost like a coding language, and proteins are kind of like a software application.

Humans have come a long way in terms of understanding DNA, to the point where editing DNA code is becoming much easier to do. Any edits made to instructions for making a protein will result in a modified protein (sometimes called a mutant or variant). There are lots of ways to approach editing the DNA; randomly changing one DNA base at a time, systematically changing very specific bases to target specific amino acids in a

protein, or combining or shuffling large fragments of DNA (this is called recombination)[2–4]. All these methods will affect the protein sequence, and in turn the protein function.

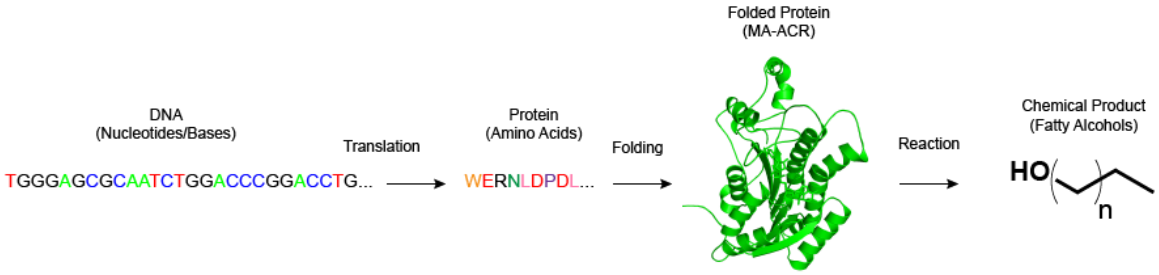

Figure 6.1: DNA determines a protein's sequence, which determines what the protein does or makes. DNA is transcribed to a similar molecule, RNA, which is then translated to make the proteins. Then a protein will fold into a 3D structure, which is uniquely suited for carrying out its designated function (for enzymes this would be carrying out a chemical reaction).

Once a strategy has been selected for editing the protein sequence (via editing the DNA), there are multiple ways to approach engineering. The Nobel Prize in 2018 was awarded to Frances Arnold for developing an approach called directed evolution. Directed evolution is a way to engineer proteins by copying how evolution in nature works[5]. Large sets of altered proteins are tested for a specific trait (for example, heat tolerance, ability to use a specific chemical as a starting material, or reaction speed) and the best one is used as the starting template for the next round. This process is repeated over and over and can result in new protein sequences with massive improvements. Another strategy is to use information about a protein's shape, or structure, and pick specific amino acids to change to accomplish a specific goal. This strategy is called rational design. Both directed evolution and rational design can be used to alter similar properties, the choice of which method to use really depends on how much is known about the structure (more

knowledge favors rational design), or whether it is easy to test large numbers of proteins quickly (the rough number of sequences that can be tested is called throughput; high throughput favors directed evolution).

Though rational design and directed evolution are the main ways to engineer proteins, they have drawbacks too. Rational design requires accurate models of protein structures, which are not always available. Directed evolution requires being able to test a huge number of proteins, tens of thousands to millions (usually all at once, though some very fast technologies enable individual testing), so it is limited by the capacity and speed of the test. However, new methods are emerging to use machine learning to accelerate protein engineering. Machine learning can overcome bottlenecks or shortcomings of directed evolution and rational design to help design better protein sequences in a more efficient manner[6].

6.1.4 Protein engineering for chemical production

Protein engineering has a lot of uses, ranging from therapeutic antibodies, which can be used to treat viral infections, to improving the enzymes used in liquid laundry detergent. In my studies, I am using protein engineering to increase chemical production of fatty alcohols (a kind of chemical commonly used in detergents, cosmetics and flavorings)[7] in cells. Making a chemical product in cells is similar to navigating from a location to a destination on a map. The location is the starting material (usually for cells this would be a sugar like glucose) and the destination is the product (in this case, fatty alcohols). Each step in the path is carried out by an enzyme. The enzyme controls how fast and how well the reaction occurs; this is kind of like controlling what kind of road the

path is and enforcing a speed limit. Sometimes the most direct route to the product goes through a slow enzyme, and so protein engineering can be used to speed that step up.

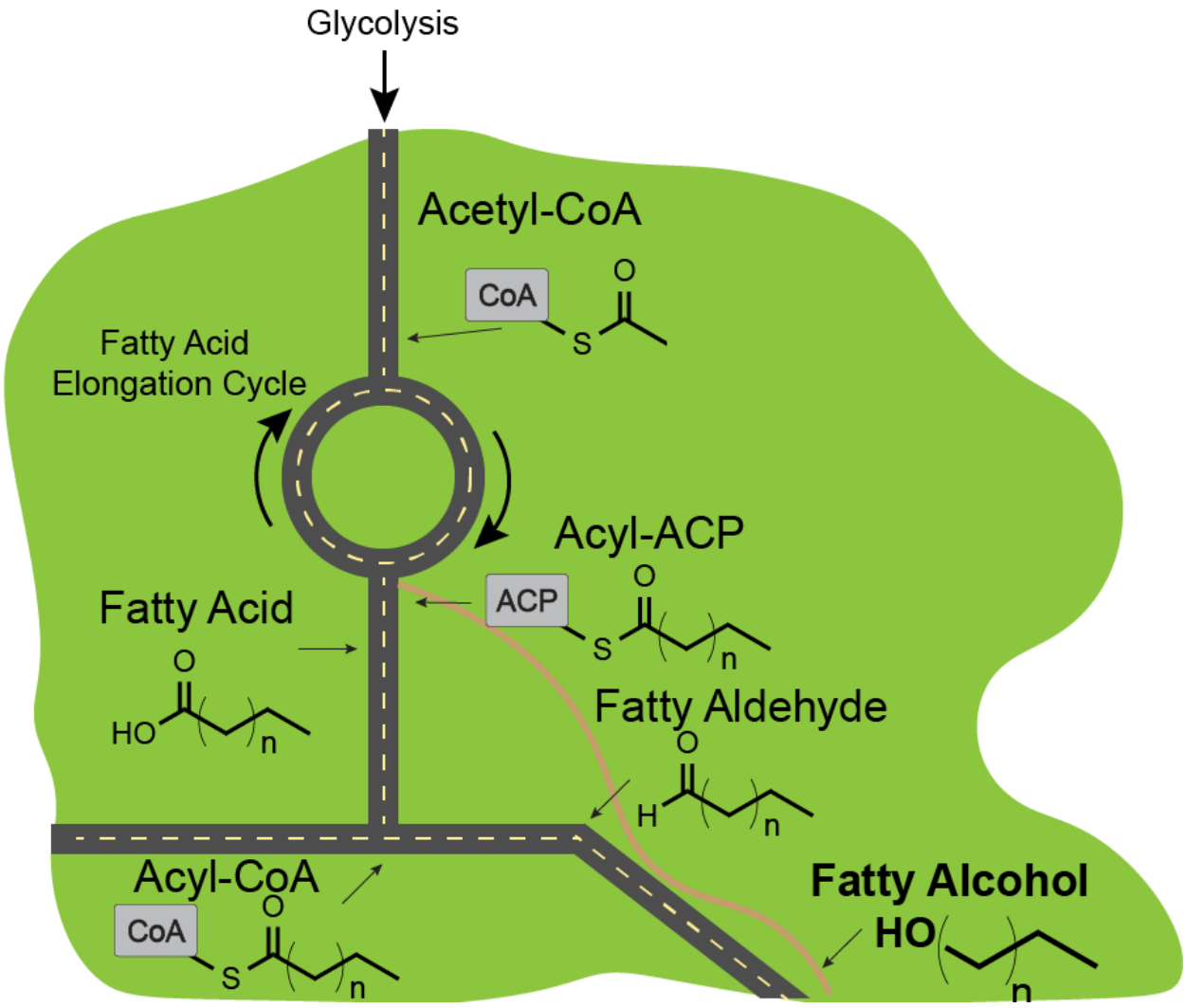

Figure 6.2: The roadmap to fatty alcohols in cells. Glycolysis is the process that breaks down sugars, fatty acid elongation is how fatty acyl-ACPs of different sizes get made. There are multiple possible routes to get from sugars to fatty alcohols. The direct route from acyl-ACPs (shown as a narrow path) is currently less used, but potentially more efficient. More commonly, the longer path through fatty acids and acyl-CoAs is used, which requires more energy.

6.1.5 The acyl-CoA route

There are multiple routes to fatty alcohols on our map. The most used routes go through a specific kind of intermediate called acyl-CoA (ay-seel-co-ay)[7–9]. Acyl-CoAs are

a lot like fatty alcohols in that they come in a range of lengths. They consist of two parts, an acyl chain, and a large molecule called coenzyme A or CoA, (which is a very important molecule elsewhere in metabolism too) linked together. A type of enzyme called an acyl-CoA reductase (or ACR) basically breaks the link between the acyl chain and the CoA. When the link breaks, the acyl chain gets converted first to a fatty aldehyde, and then the ACR transforms the fatty aldehyde to a fatty alcohol (in a reaction called a reduction)[10].

6.1.6 The acyl-ACP route

Acyl-CoA reductases are very good at converting acyl-CoAs to fatty alcohols, but acyl-CoAs aren't necessarily the most direct route to the destination. There is another kind of intermediate called acyl-acyl-carrier proteins (acyl-ACPs) that can be converted to fatty alcohols, and that could potentially be more efficient. Acyl-ACPs are like acyl-CoAs, the acyl chain is just attached to a small protein (ACP) instead of CoA. Because in many cases, the cells need to make acyl-ACPs to get to CoAs anyway, having an ACR that can make fatty alcohols from acyl-ACPs could save the cell energy and resources and enable better results. Some ACRs can also convert acyl-ACPs to fatty alcohols, but they are typically very bad at it[11], so we decided to engineer an ACR to be able to do it better. The ACR called MA-ACR from a species of bacteria called *Marinobacter Aqueolei* showed a lot of promise in scientific work done by others[7–9], so we decided to use it as the starting point for our engineering effort.

6.2 Engineering ACRs to Acyl-ACP Reductases: design, build, test, learn

6.2.1 Designing the library

To engineer MA-ACR, we started by designing a sequence library. Just like a physical library is a place where documents or books are stored, a protein library is a place where a large set of potential protein sequences are gathered. Each protein sequence in the library is like an individual book or document. This metaphor can extend even further, chapters in the book could be like important motifs or modules in the protein sequence and the words could be the individual amino acids. But the library is where we figure out what sequences (or books) are available.

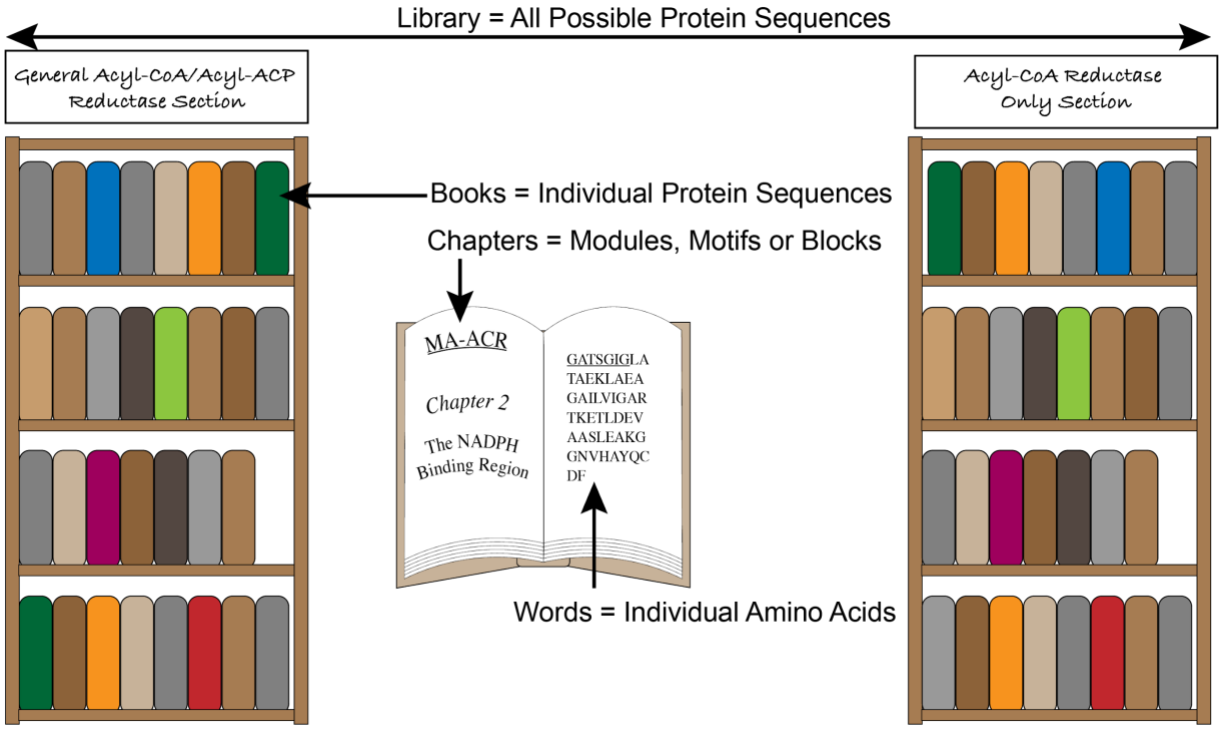

Figure 6.3: Large groups of protein sequences are often referred to as “libraries” in the scientific literature. In this analogy, the library refers to the place where all the sequences are stored, and each protein sequence can be thought of as an individual document. Though literature examples typically don’t extend the library metaphor further, we can think of the parts that make up the protein (modules or motifs), as chapters and the amino acids as words.

The way we chose to design our ACR library was using a trick called recombination[4]. The idea of recombination is that proteins are modular and have modules or blocks of amino acids that can be interchanged for other similar blocks. Because different kinds of organisms often have slightly different versions of the same proteins, these modules can be swapped out to generate new enzymes that work better, and because evolution tends to generate proteins that actually work, it is more likely that changing the protein sequence in this way will result in a functional protein than by just making random changes (a technique that is actually very commonly used). First, since we already knew that each ACR had two parts (or domains)[10], we selected three ACR sequences from different bacteria (MA-ACR, MB-ACR and MT-ACR), and made all nine combinations of the pieces. The results were surprising. First, we found that MB-ACR (BB in Figure 6.4) actually worked better than MA-ACR (AA in Figure 6.4). Second, and more surprisingly, when we made an enzyme that was half MA-ACR and half MB-ACR (AB in Figure 6.4), it worked even better than both of them. It was really exciting and encouraging to have a positive result so early.

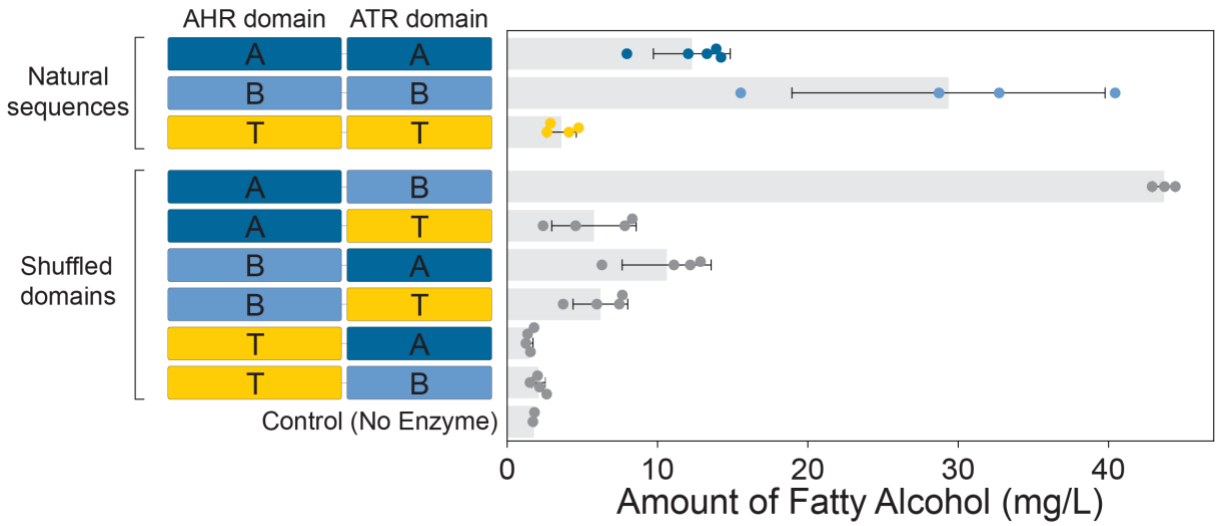

Figure 6.4: Recombining parts from natural enzymes resulted in even better enzymes (compare AB to AA and BB). Note, AA is MA-ACR and BB is MB-ACR.

Next, we started to work on studying the half of the enzyme that carries out the first reaction step (the conversion of acyl-ACP to a fatty aldehyde). To do this, we kept the other half of the enzyme constant and chose the three enzymes that had it as "parents" for recombination (I'll refer to the versions we used as Parent A, B and T), and then used a mathematical algorithm to help us identify a set of eight blocks from each sequence that line up with the other sequences[12]. This would normally give 3^8 = 6,561 possible combinations of blocks, but in this case because two blocks were exactly the same there are 4,374.

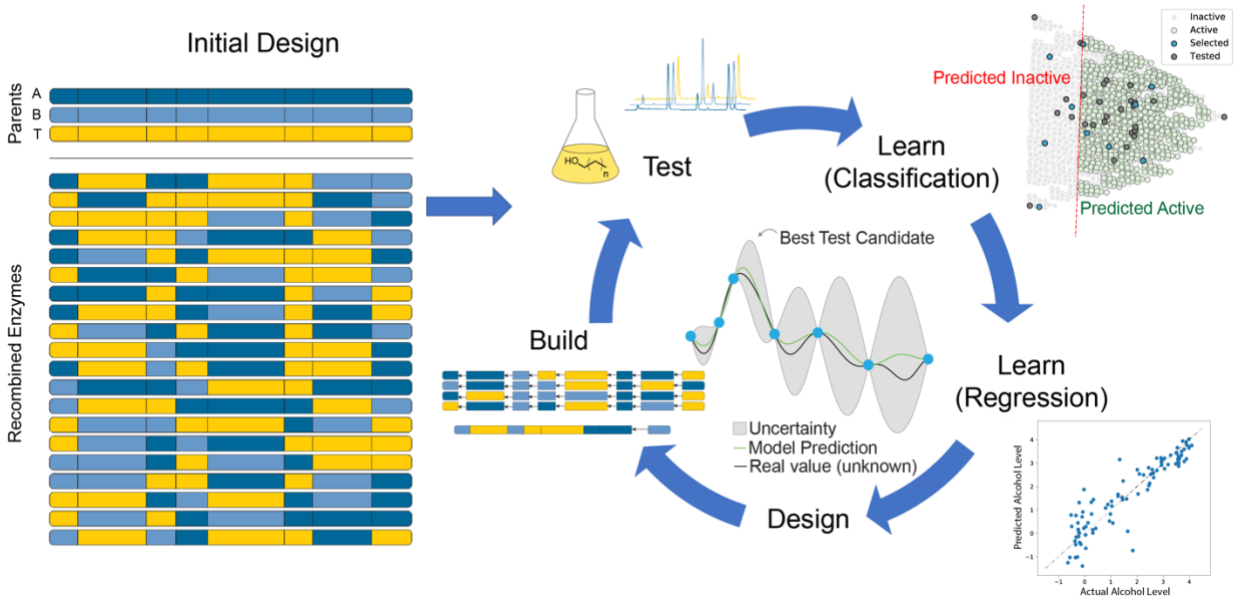

Figure 6.5: The design-build-test-learn cycle. We designed a test set to try as many combinations of the three parent enzymes as possible. Then we tested the sequences experimentally to figure out how much fatty alcohol each one made. After that we used two different kinds of machine learning models to help design sequences to study in future iterations of the cycle. Finally, we assembled the sequences using pieces of DNA and repeated the process.

Normally, with this many potential combinations, we would have to test all the sequences, either individually (which would be prohibitive with our detection methods), or all at once (by linking alcohol production to cell survival or growth). However, because we didn't have reasonable ways to use either of these approaches, we chose to use machine learning to help us learn about these protein sequences more efficiently and bypass the bottleneck in the number of sequences that we could test experimentally.

6.2.2 Machine learning compliments protein engineering

Just like mapping apps can use data and algorithms to help us find directions, similar strategies can be applied to protein engineering. Machine learning is the term for computer algorithms that learn from data. Examples of machine learning in everyday life

are abundant; one common and important example is the spam filter that detects junk emails. Machine learning has the potential to drastically improve efforts to engineer proteins and biological pathways. Biological pathways are complex, and the number of mathematical variables involved in engineering them can be staggering. But machine learning models can deal with all those variables and can help us identify the most important ones. Also, just like machine learning tools can be used in advertisements to recommend specific products for specific people, machine learning can be used to recommend specific “books” or sequences in our library of ACRs that could be good for making fatty alcohols. This means that we don’t have to test all four thousand sequences to find the best enzymes, instead we can test a smaller set and use what we learn to improve our models and find better enzymes.

6.2.3 Building the sequences

From our set of 4,374 sequences, I identified twenty sequences that as a set, gave me the most information about all the potential variables (using statistics). Then, I started working on building them. This is kind of like building Legos, just instead of plastic blocks, we use tiny fragments of DNA and a special enzyme (called ligase) that sticks them together[13]. I found this process fascinating. The idea that I could get on my computer, design a large circular piece of DNA (called a plasmid), and then use that DNA to build something else was very exciting. Of course, it is not quite that easy, and I had my share of troubles getting it to work, but at the end of the day I figured things out and was able to successfully build all twenty sequences I needed for initial testing. I would use the same

method again later once we started learning more about which sequences were good at making fatty alcohols.

6.2.4 Testing

To test our ACR sequences, we would put the DNA sequences into *E. coli* and grow the *E. coli* for about 18 hours. In order to signal the cells to make more ACR protein (i.e. expressing ACR), we would add a chemical called IPTG, which signals the cells to start making the protein, and to make sure the cells had enough resources to make extra fatty alcohols we added glycerol (which serves as a source of carbon, similar to the sugar glucose). After turning on protein production, the cells would naturally start making fatty alcohols as well. We then used an instrument called a Gas Chromatograph to measure the amount of fatty alcohol produced by the cells (the amount or concentration of fatty alcohols in cells is also called the titer).

6.2.5 Learning

After figuring out how much fatty alcohol each of our ACRs made, we then started using machine learning models to learn as much as we could about what sequences worked best. The machine learning models take the protein's amino acid sequences as an input and try to find a mathematical relationship between the sequence and the fatty alcohol levels. This process is called model training. Once a model is trained, it can also be used to make predictions about other sequences, even sequences that have not been tested yet.

I used two kinds of models. First, I used a classifier to learn the difference between ACRs that worked and ACRs that didn't. The classifier input is the protein sequences I tested and whether they were active in my experiments, and it produces a binary output or prediction (active/inactive) for all the sequences that I hadn't tested yet. The classifier is kind of similar to how a spam filter works (predicting spam/not spam based on content of an email). Using the classifier helped me pre-filter out bad sequences so that I wouldn't have to worry about them for the next modeling step.

The next step was using regression models. Unlike the classifier, which outputs categories, regression has a continuous output, meaning that it outputs numerical values. For example, a classifier can be used to classify based on meteorological data, whether it will be hot or cold, but a regression model can be used to predict the temperature. Regression allowed me to make predictions about which ACR sequences would work, and how well they would work compared to others. Additionally, the regression model that I used also outputs estimates of uncertainty[14]. This was very useful, because it allowed me to use the model to suggest (or design) new sequences to build that would be both rich in information and likely to work well at the same time.

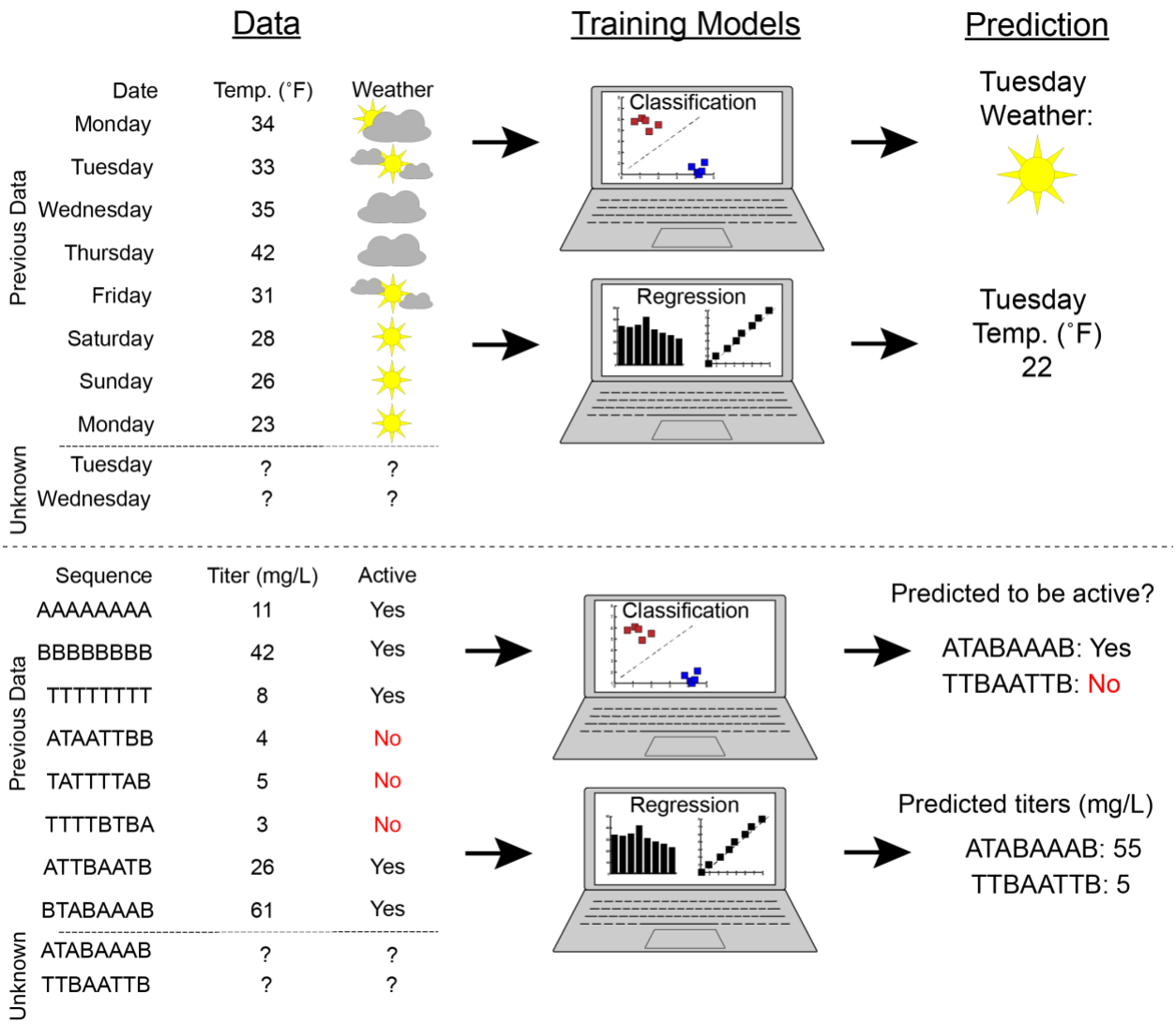

Figure 6.6: Schematic depicting how different kinds of machine learning models work (regression and classification). The top half shows an example of how each kind of model could be used to predict different weather-related properties. The bottom example is an example of how the models could be used to predict whether an enzyme works (i.e., whether it is active) and how much fatty alcohol it can make (i.e. it's titer).

6.2.6 Completing the cycle

Searching for the best enzyme is kind of like hiking to the top of a mountain. From the base of the mountain, the top is not always visible, so we can make an estimate (like training a model and making a prediction) to guess where it might be. As we climb higher and higher, we test out our theory and gain more information. This helps us get a clearer picture of where the summit is (like updating a model). Occasionally this journey leads to the top of false summits or secondary peaks, but if we keep updating our objective as we learn from the landscape, we can make it to the top.

Finding the best ACR sequence was very similar, the first batch of sequences that were suggested by the machine learning models didn't work very well (we had no idea where the summit was). But that was ok. We never intended to stop after the first attempt, and when it comes to machine learning, any data we could get would be useful for future rounds. After the sequences in the first round failed, we just updated the machine learning models and tried again. We repeated the process of building and testing sequences, training machine learning models on the data, and using the models to design new sequences, over and over until we finally achieved ACR sequences that worked much better than the original enzymes we started with. All in all, we tested about 96 ACR sequences over ten turns (or rounds) of this iterative design-build-test-learn cycle.

Each full cycle took about two weeks: one week to build the sequences and verify that they were correct, and another week to grow the cells containing the sequences and test them. Updating the machine learning models was the fastest part of the workflow. It only took a few minutes to update the models and make new predictions. Over time, as

there was more and more data to train the models, the models took slightly longer to train, but on the flipside, they got much more accurate too.

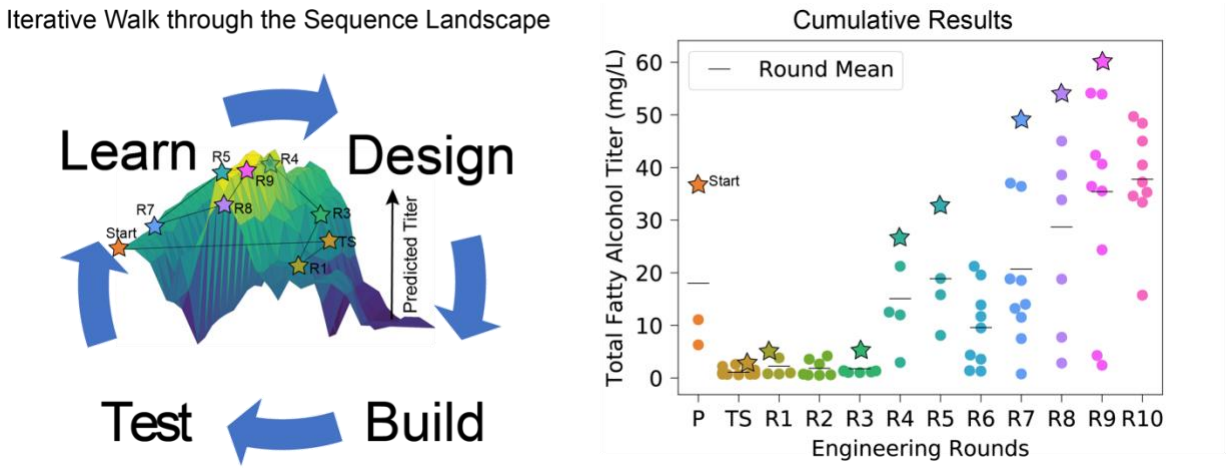

Figure 6.7: The search for the optimal enzyme can be thought of as a hike. By gradually working your way uphill, you can make your way to the summit. Similarly, by gradually searching for better and better enzymes you can progress towards the best one. The left side of this figure shows a visualization of the sequence landscape in 3D, where the "elevation" is like the predicted titer from the models. The stars show key points along the way (basically top sequence in every round where there was an improvement). Further distances between sequences suggest that the sequences are more different from each other. The figure on the right shows the amounts of fatty alcohols in each round (each round is shown as a different color). P is the parents (AA, AB, and AT) and TS is the initial test set of twenty ACRs. The best sequence was identified in round 9. The black bars in this figure show the average titer for all the sequences in that round.

6.3 Conclusions

Our best ACR sequence (which we called ATR-83) made about twice as much fatty alcohol as the best natural enzyme that we studied. Experiments in test tubes validated these results and verified that they were due to the enzyme being a better catalyst, rather than simply being easier for the cell to make. The next question we wanted to answer was why it worked better. Again, we turned to machine learning models. This time instead of making predictions about sequences we hadn't tested yet; we used the models to try to understand as much of the sequence as we could. The outputs of the

models suggested that a few very specific blocks were very important. Some of these were expected to play a big role (the blocks that actually carry out the reaction), but some of the important blocks were surprising. As we studied the protein sequence further, we found that there were a large number of positively charged amino acid residues near the site where we believed the acyl-ACP should bind. Interestingly, ACP has a lot of negatively charged amino acids, so the two proteins can stick together similar to a pair of oppositely charged magnets[15]. ATR-83, our best sequence, had even more positively charged amino acids than the natural proteins.

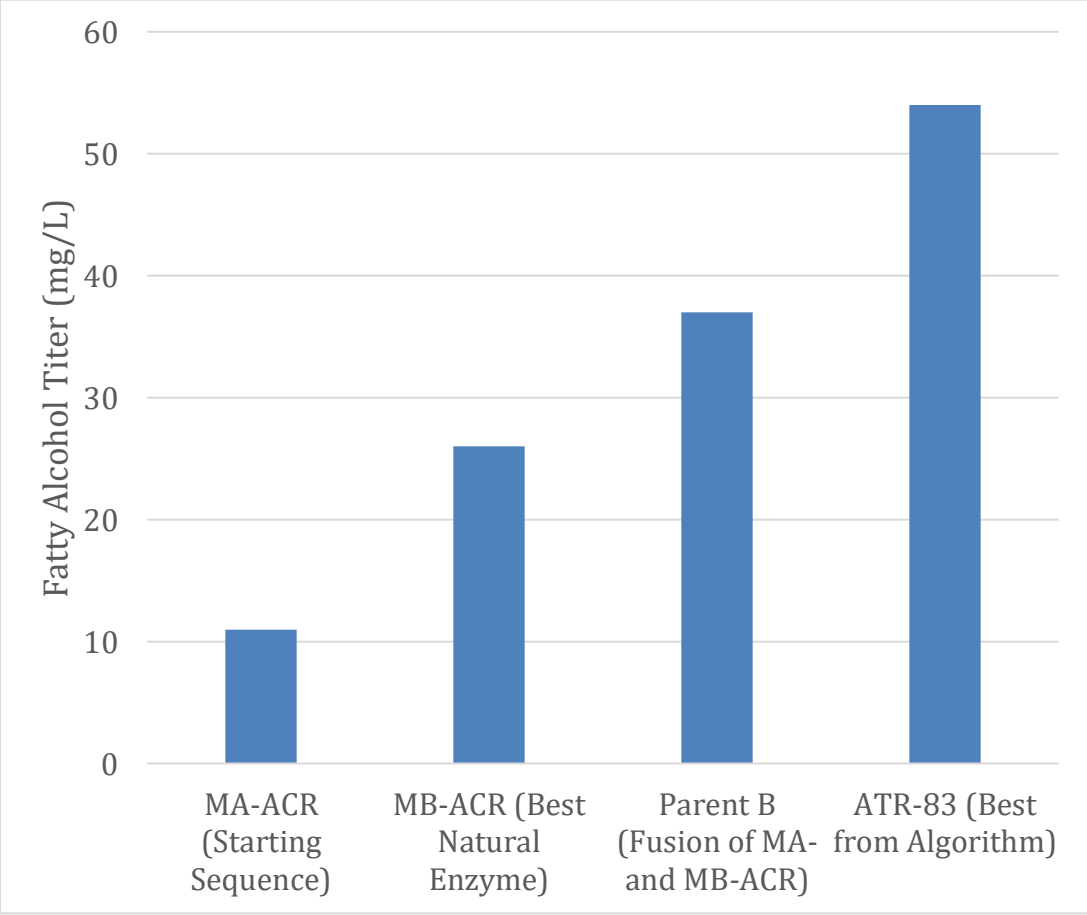

Figure 6.8: Summary of Improvements to ACRs. Testing out new natural enzymes helped us find a better ACR for converting acyl-ACPs to alcohols. Recombining that sequence with our starting sequence resulted in another boost in activity and using our machine learning-guided approach we were able to design a sequence (ATR-83) that could make even more fatty alcohols.

In summary, we were able to over double the amount (or titer) of fatty alcohols that we produced in cells by using our designed ACR sequence, ATR-83. This result shows that machine learning can be used to help engineer proteins without needing to test thousands of sequences or without knowing a precise structure. The most exciting thing about this approach is that it can be used to engineer almost any enzyme, as long as there is a way to test the enzyme's function. Hopefully this result will not only enable further improvements in production of chemicals in cells but accelerate the use of machine learning in protein engineering workflows.

References:

(1) Greenhalgh, J. C.; Fahlberg, S. A.; Pfleger, B. F.; Romero, P. A. Machine Learning-Guided Acyl-ACP Reductase Engineering for Improved in Vivo Fatty Alcohol Production. *Nat. Commun.* **2021**, *12* (1), 1–10. https://doi.org/10.1038/s41467-021-25831-w.

(2) Coco, W. M.; Levinson, W. E.; Crist, M. J.; Hektor, H. J.; Darzins, A.; Pienkos, P. T.; Squires, C. H.; Monticello, D. J. DNA Shuffling Method for Generating Highly Recombined Genes and Evolved Enzymes. *Nat. Biotechnol.* **2001**, *19* (4), 354–359. https://doi.org/10.1038/86744.

(3) Stemmer, W. P. Rapid Evolution of a Protein in Vitro by DNA Shuffling. *Nature*. 1994, pp 389–391. https://doi.org/10.1038/370389a0.

(4) Silberg, J. J.; Endelman, J. B.; Arnold, F. H. SCHEMA-Guided Protein Recombination. *Methods Enzymol.* **2004**, *388* (2003), 35–42. https://doi.org/10.1016/S0076-6879(04)88004-2.

(5) Arnold, F. H.; Volkov, A. A. Directed Evolution of Biocatalysts. *Curr. Opin. Chem. Biol.* **1999**, *3* (1), 54–59. https://doi.org/10.1016/S1367-5931(99)80010-6.

(6) Yang, K. K.; Wu, Z.; Arnold, F. H. Machine-Learning-Guided Directed Evolution for Protein Engineering. *Nature Methods*. Nature Publishing Group August 1, 2019, pp 687–694. https://doi.org/10.1038/s41592-019-0496-6.

(7) Hernández Lozada, N. J.; Simmons, T. R.; Xu, K.; Jindra, M. A.; Pfleger, B. F. Production of 1-Octanol in Escherichia Coli by a High Flux Thioesterase Route. *Metab. Eng.* **2020**, *61*, 352–359. https://doi.org/10.1016/j.ymben.2020.07.004.

(8) Youngquist, J. T.; Schumacher, M. H.; Rose, J. P.; Raines, T. C.; Politz, M. C.; Copeland, M. F.; Pfleger, B. F. Production of Medium Chain Length Fatty Alcohols from Glucose in Escherichia Coli. *Metab. Eng.* **2013**, *20*, 177–186. https://doi.org/10.1016/j.ymben.2013.10.006.

(9) Mehrer, C. R.; Incha, M. R.; Politz, M. C.; Pfleger, B. F. Anaerobic Production of Medium-Chain Fatty Alcohols via a β-Reduction Pathway. *Metab. Eng.* **2018**, *48* (April), 63–71. https://doi.org/10.1016/j.ymben.2018.05.011.

(10) Willis, R. M.; Wahlen, B. D.; Seefeldt, L. C.; Barney, B. M. Characterization of a Fatty Acyl-CoA Reductase from Marinobacter Aquaeolei VT8: A Bacterial Enzyme Catalyzing the Reduction of Fatty Acyl-CoA to Fatty Alcohol. *Biochemistry* **2011**, *50* (48), 10550–10558. https://doi.org/10.1021/bi2008646.

(11) Opgenorth, P.; Costello, Z.; Okada, T.; Goyal, G.; Chen, Y.; Gin, J.; Benites, V.; De Raad, M.; Northen, T. R.; Deng, K.; et al. Lessons from Two Design-Build-Test-Learn Cycles of Dodecanol Production in Escherichia Coli Aided by Machine Learning. *ACS Synth. Biol.* **2019**, *8* (6), 1337–1351.

https://doi.org/10.1021/acssynbio.9b00020.

(12) Endelman, J. B.; Silberg, J. J.; Wang, Z.; Arnold, F. H. Site-Directed Protein Recombination as a Shortest-Path Problem. *Protein Eng. Des. Sel.* **2004**, *17* (7), 589–594. https://doi.org/10.1093/protein/gzh067.

(13) Engler, C.; Gruetzner, R.; Kandzia, R.; Marillonnet, S. Golden Gate Shuffling: A One-Pot DNA Shuffling Method Based on Type Ils Restriction Enzymes. *PLoS One* **2009**, *4* (5). https://doi.org/10.1371/journal.pone.0005553.

(14) Romero, P. a; Krause, A.; Arnold, F. H. Navigating the Protein Fitness Landscape with Gaussian Processes. *Proc. Natl. Acad. Sci. U. S. A.* **2013**, *110* (3), E193-201. https://doi.org/10.1073/pnas.1215251110.

(15) Sarria, S.; Bartholow, T. G.; Verga, A.; Burkart, M. D.; Peralta-Yahya, P. Matching Protein Interfaces for Improved Medium-Chain Fatty Acid Production. *ACS Synth. Biol.* **2018**, *7* (5), 1179–1187. https://doi.org/10.1021/acssynbio.7b00334.