*< this document is a reduced form of the realization cycle, use for smaller projects and development stuff >*

*< look through the notes and material you’ve collected for this project so far >*

## Why are you doing this project? (is your time better spent somewhere else?)

### 0.1 What would you gain from completing this project?

A batchelors, skills, possible publication

### 0.2 Is this the right time for this project?

Need to be done now

### 0.3 How is this project connected to your overarching goals?

It will build a more secure foundation for me and possible enable me to find a more fitting job

### 0.4 How is this project connected to other projects?

It will give me skills to tackle machine learning and python

### 0.5 Does this project hog resources better spent on other projects?

Yes, but I’ll have to prioritize to be able to get into a masters this autumn

### 0.6 How does this project benefit others?

An improved Sweetnet could possibly allow others to make better prediction models applied to specific glycobiology projects

### 0.7 How does this project improve the world?

Better glycan property prediction could improve treatments for diseases connected to the glycobiome, stuff like stomach bugs and virus binding

*< if the reasons are insufficient, drop the project for now >*

## Define the project scope (what are the boundaries of the project?)

### 1.1 Describe the project (just write, don’t overthink)

I am going to try to see if I can improve the performance of the SweetNet Graph Neural network by using the pre-trained values of embeddings of a Glycan Language Model as features at the start of training to see if the relationships they have captured through training improves the predictive capabilities of the new hybrid model.

### 1.2 What are the top-three objectives of the project?

1. Using Sweetnet to predict a specific property (to test the base system, and all iterations)
2. A Sweetnet with glm embeddings that I can benchmark and compare to the base Sweetnet
3. A Hyperparameter optimized Sweetnet that outperforms the original (with or without embeddings)

### 1.3 What is the goal of the project? (Why are you doing it?)

To Improve SweetNet using GLM embeddings, or prove that this technique isn’t useful

### 1.4 What assumptions does the project rest on?

* That the embeddings taking from the pre-trained GLM has captured something about the structure and chemistry of the glycan
* That starting with non-random features improves the model after training
* That this method doesn’t lead to over-fitting

### 1.5 What is the central concept of the project

GLM embeddings will improve a glycan GNN

### 1.6 What is a possible final deutrotype of the project?

A commit to the glycoworks repo with my improved sweetnet (or even a new tool entirely)

### 1.7 What do you need to understand before going further?

### 1.8 What is the deadline of the project?

*2024-05-29*

*< delve to gain that understanding before moving on to the first prototype >*

*<limit this to a day or two, more for deeper unknowns that require a book to be read>*

## Define the first prototype (prove the concept before moving on)

*<look through your notes from the delving before starting>*

### Notes:

### 2.1 Can the prototype be broken into stand-alone prototype shards? (MoSCoWize)

2.1.1 Must have:

1. !Getting SweetNet Up and running (2025-04-11)
2. Infusing SweetNet with GLM Embeddings (2025-04-25)
3. Test And compare infused model to original (2025-05-02)

2.1.2 Should have:

1. Hyperparameter Optimized Sweetnet
2. Test other GLMs (Roman has at least 19)
3. Applying SweetNet to specific problem

2.1.3 Could have:

1. Tinkering with Sweetnet Structure (birthing new model)

2.1.2 Won’t have at this time (for future projects):

1. s

*< select one shard to start with (mark with!), tackle the next in a separate iteration and so on >*

*< add approximate deadlines for each shard >*

*< prototype shards are not set in stone, they are only guidelines, the next prototype may be something else >*

### 2.2 What is the goal(s) of the prototype?

To Familiarize myself with glycowork, python, and git, as well as building a starting point to iterate on for future experiments

### 2.3 What do I want to do with the prototype?

Tinker with it to get to know it better. Run test valuation to get a baseline, get familiar with git.

### 2.4 What are four or more bad ideas? (just write, don’t analyze)

1. An implementation of Sweetnet in Houdini
2. Just copying the sweetnet repo without getting to know it
3. Sweetnet made as a physical system
4. Just reading the paper and trying to implement my own sweetnet from scratch
5. Iterating my basic machine learning experiment into something sweetnet-esque

### 2.5 What is the Prototype?

A branch of forked Git repo made local and with added folder structure for tis project to live in

### 2.6 What are the top-level MoSCoW requirements of the prototype? (max 10 initially, be concrete)

#### 2.6.1 Must haves (60% of effort max)

1. +Fork Git Repo and make branch, clone branch to laptop
2. +Experimentation jupyter notebook (should the whole project live here?)
   1. +GLM-infused Sweetnet development system
   2. +Set up dependencies (If needed)
3. ~~Basic Evaluation system (make a class)~~
   1. ~~To iterate later~~
   2. ~~Check for stuff already in glycowork~~
4. ~~Pre-trained sweetnet implementation~~

#### 2.6.2 Should haves

1. +Training my own Sweetnet

#### 2.6.3 Could haves (20% of effort)

1. +Better name
2. Tinkering with my own sweetnet
3. More knowledge

#### 2.6.4 Won’t have at this time (for future iterations)

1. Embeddings

*< prefix the riskiest features of the prototype with ® in the list above (do these first)>*

### 2.7 What is the done definition of each requirement?

1. Cloned locally and connected to main
2. A well-placed jupyter notebook file in my branch of my fork of the glycowork repo
3. The evanluation gives me a value of how well the model works with new data
4. A cell in the experimentation notebook that loads the pretrained model
5. I Have a sweetnet model that works as well as the pretrained model

#### 2.7.1 How could each requirement be tested?

1. Run some basic functions locally glycowork after setting up experiment notebook
2. This is tested by the test of 1’
3. Test by using on pretrained sweetnet (implement that first)compare to expected data from sweetnet paper and ask roman
4. Does it do predictions? Test with basic evaluator
5. Goto 3, and then compare to pretrained

### 2.8 What is the done definition of the prototype?

When I have run the tests to compare the pre-trained and the custom and they show similar results I can move on

### 2.9 How should I test that the prototype functions as expected?

If all requirement tests are positive the whole prototype should be fully functional

### 2.10 Describe the prototype fully (just write, don’t overthink)

I’m going to set up my development environment for project GLM-infused Sweetnet, this includes getting a git workflow up and running, cloning my own fork with the custom branch of the glycowork repo I will be working on, making an experimental jupyter notebook for this project to live in (or perhaps a folder with several ones including utility files) Then I’ll try using the pretrained sweetnet from glycowork and build an evaluation function that will eventually live in its own utility file

*< select a requirement (starting with ®’s) >*

### 2.11 Define the requirement

#### 2.11.1 What is the requirement? (! Prefix in 2.6)

#### 2.11.2 Are there requirements to the requirement? (add one level down below selected requirement in 2.6)

*<prefix the riskiest requirements with ®>*

#### 2.11.3 What is the done definition of each requirement? (add one level down below selected requirement in 2.7)

#### 2.11.4 How should I test that the requirement functions as expected? add one level down below selected requirement in 2.7.1)

#### 2.11.5 What are the milestones to the requirement? (cross out the past milestones, add new beneath)

*< did the requirement have requirements? select a requirement (starting with ®’s) & goto 2.11>*

*< when the bottom is reached, implement the requirement in chaoskampf >*

*< check the finished requirement against done definition & test it >*

*< if it passes, change ! prefix to +, else back to chaoskampf >*

*< select the next requirement, goto 2.10 >*

*< if all requirements have been implemented, test the prototype itself >*

*< check against done definition >*

*< done? test the finished prototype >*

*< tests passed? mimimize & add line to proximate ideas --- >*

*< check project definition & decide if another iteration is needed >*

*< another iteration? goto 3 >*

*< no iteration? goto 4 >*

## Iteration [1] (look through unfinished <requirements> of past iteration and reprioritize as needed)) [copy this section below for further iterations before using]

### Notes:

* Skipped pretrained sweetnet, I spent way longer to get that to work than just running my own training took
* Didn’t need evaluator, just using the data I get from training

### 3.i What did you learn last iteration?

A lot of git, vc code, and much more. Glycowork is both easier and harder to work with than I thought. Training your own models is easy.

### 3.1 Can the prototype be broken into stand-alone prototype shards? (List them)

*< if this prototype is a shard, copy the shards from the original prototype, mark the done with +)*

2.1.1 Must have:

1. +Getting SweetNet Up and running (~~2025-04-11~~ 2025-04-16)
2. !Infusing SweetNet with GLM Embeddings and comparing it (2025-04-25)
3. Through testing and statistics to quantify how the infusion influenced the model (2025-05-02)
4. Write a thesis that conveys my findings and results (2024-05-29)

2.1.2 Should have:

1. Applying SweetNet to specific problem
2. Test embeddings from other GLMs (Roman has at least 19)
3. Hyperparameter Optimized Sweetnet (Evolutionary algorithm?)
4. Fix glycowork bugs
5. Improve glycowork documentation

2.1.3 Could have:

1. Tinkering with Sweetnet Structure (birthing new model)

2.1.2 Won’t have at this time (for future projects):

*< select one shard to start with (mark with!), tackle the next in a separate iteration and so on >*

*< prototype shards are not set in stone, they are only guidelines, the next iteration may be something else >*

### 3.2 What is the goal(s) of this prototype?

To modify the base SweetNet to use the GLM embeddings I got from roman

### 3.3 What do I want to do with this prototype?

Make better predictions than the base model. Look whether pre-trained embeddings improve things or make them worse.

### 3.4 What are four or more bad ideas? (just write, don’t analyze)

1. Go in blindly and just try to get the data from roman in somehow
2. Edit glycowork sweetnet rather than working with copy in notebook
3. Do a completely different prediction task this time
4. Switch to another language for fun

### 3.5 What is the Prototype?

A modified version of SweetNet that allows it to take pre-trained embeddings as input. To get there I need a way to take the embeddings I’ve gotten from roman and transform them into nice inputs for the model, and a way to set the initial features using these inputs.

### 3.6 What are the MoSCoW reuirements of the prototype? (10 max initially, be concrete)

#### 3.6.1 Must haves (60% of effort max)

1. Jog memory by rereading DeepRank paper
2. Explore Embedding data
3. Copy SweetNet Code to new Jupyter notebook for experimentation
4. Filter and Transform embedding data to a format usable in SweetNet
5. Modify SweetNet to accept embedding data
   1. Run training and compare
   2. If results don’t show improvement, look for errors and try other embeddings

#### 3.6.2 Should haves

1. Add new SweetNet functionality to glycowork
2. Test other GLM embeddings (if the first one fails)

#### 3.6.3 Could haves (20% of effort)

#### 3.6.4 Won’t have at this time (for future iterations)

*< prefix the riskiest features of the prototype with ® in the list above (do these first)>*

### 3.7 What is the done definition of each requirement?

1. I’ve read the text again
2. I understand the structure and contents of the data
3. I can run the local experiment and get the same results as the imported version
4. I get a fitting datastructure that I can pipe into SweetNet
5. I can run training using the modified SweetNet and get an accuracy score
6. I port my experimental code into the models.py file of glycoworks without breaking anything
7. See if any of the other embeddings give different results

#### 3.7.1 How could each requirement be tested?

1. Do I Understand what I need to be doing?
2. What is the data I need? How might I add it?
3. Running the kingdom test with copied sweetnet, see that it behaves similarly
4. I get data that works in the modified sweetnet
5. Training works (I get convergence and accuracy scores)
6. Training works when running from local glycowork,
   1. test with other settings to make sure nothing broke

### 3.8 What is the done definition of the prototype?

I have a GLM-infused Sweetnet that trains as it should and reports an accuracy score

### 3.9 How should I test that the prototype functions as expected?

Try to vary the settings of the model to make sure that nothing is broken

### 3.10 Describe the prototype fully (just write, don’t overthink)

I will learn a bit about the technique I will employ to use embeddings from a language model to infuse a GNN. I’ll set up a new jupyter notebook where I’ll first explore a GLM embedding data file to figure out how they are represented and in what structure, then I’ll simply paste over the code defining SweetNet from models.py so that I can experiment with it freely. After verifying that the copied version functions as expected I’ll explore the code and figure out how it handles the initial featurization to see in what form and how I might get in the embeddings. Then I will get to work to transform the embedding data into something that I can work with once I start to work on the pipeline to get them into SweetNet. Once I have infused SweetNet with the embeddings I’ll run the same training steps that I did in iteration 0. If It converges and I get an accuracy score I am golden (unlikely). If it doesn’t work I will troubleshoot and try to figure out why it isn’t working, iteratively trying to get to the core of the problem. If it works and I don’t get better results I can try to use another prediction class, or even different GLM embeddings. This is still a result though, but I will need to verify that my implementation isn’t the problem and that it is generalizable. If I have time (and it gives a real improvement) I will take my experimental SweetNet and add it to my branch of the glycowork repo (conserving the normal sweetnet, perhaps allowing you to use the GLM-infused version with a flag or something) perhaps this is bnetter left for a future iteration.

*< select a requirement (starting with ®’s) >*

### 3.11 Define the requirement

#### 3.11.1 What is the requirement? (! Prefix in 3.6)

#### 3.11.2 Are there requirements to the requirement? (add one level down below selected requirement in 3.6)

*<prefix the riskiest requirements with ®>*

#### 3.11.3 What is the done definition of each requirement? (add one level down below selected requirement in 3.7)

#### 3.11.4 How should I test that the requirement functions as expected? add one level down below selected requirement in 3.7.1)

#### 3.11.5 What are the milestones to the requirement? (cross out the past milestones, add new beneath)

*< did the requirement have requirements? select a requirement (starting with ®’s) & goto 3.11 >*

*< when the bottom is reached, implement the requirement in chaoskampf >*

*< check the finished requirement against done definition & test it >*

*< if it passes, change ! prefix to +, else back to chaoskampf >*

*< select the next requirement, goto 3.10 >*

*< if all requirements have been implemented, test the prototype itself >< if all features have been implemented, test the prototype itself >*

*< check against done definition >*

*< done? test the finished prototype >*

*< tests passed? mimimize & add line to proximate ideas --- >*

*< check project definition & decide if another iteration is needed >*

*< another iteration? copy this section below, increment, remove answers & goto it >*

*< no iteration? goto next >*

## Iteration [2] (look through unfinished <requirements> of past iteration and reprioritize as needed)) [copy this section below for further iterations before using]

### Notes:

### 3.i What did you learn last iteration?

### 3.1 Can the prototype be broken into stand-alone prototype shards? (List them)

*< if this prototype is a shard, copy the shards from the original prototype, mark the done with +)*

2.1.1 Must have:

1. +Getting SweetNet Up and running (~~2025-04-11~~ 2025-04-16)
2. !Infusing SweetNet with GLM Embeddings and comparing it (2025-04-25)
3. Through testing and statistics to quantify how the infusion influenced the model (2025-05-02)
4. Write a thesis that conveys my findings and results (2024-05-29)

2.1.2 Should have:

1. Applying SweetNet to specific problem
2. Test embeddings from other GLMs (Roman has at least 19)
3. Hyperparameter Optimized Sweetnet (Evolutionary algorithm?)
4. Fix glycowork bugs
5. Improve glycowork documentation

2.1.3 Could have:

1. Tinkering with Sweetnet Structure (birthing new model)

2.1.2 Won’t have at this time (for future projects):

*< select one shard to start with (mark with!), tackle the next in a separate iteration and so on >*

*< prototype shards are not set in stone, they are only guidelines, the next iteration may be something else >*

### 3.2 What is the goal(s) of this prototype?

### 3.3 What do I want to do with this prototype?

### 3.4 What are four or more bad ideas? (just write, don’t analyze)

### 3.5 What is the Prototype?

### 3.6 What are the MoSCoW reuirements of the prototype? (10 max initially, be concrete)

#### 3.6.1 Must haves (60% of effort max)

#### 3.6.2 Should haves

#### 3.6.3 Could haves (20% of effort)

#### 3.6.4 Won’t have at this time (for future iterations)

*< prefix the riskiest features of the prototype with ® in the list above (do these first)>*

### 3.7 What is the done definition of each requirement?

#### 3.7.1 How could each requirement be tested?

### 3.8 What is the done definition of the prototype?

### 3.9 How should I test that the prototype functions as expected?

### 3.10 Describe the prototype fully (just write, don’t overthink)

*< select a requirement (starting with ®’s) >*

### 3.11 Define the requirement

#### 3.11.1 What is the requirement? (! Prefix in 3.6)

#### 3.11.2 Are there requirements to the requirement? (add one level down below selected requirement in 3.6)

*<prefix the riskiest requirements with ®>*

#### 3.11.3 What is the done definition of each requirement? (add one level down below selected requirement in 3.7)

#### 3.11.4 How should I test that the requirement functions as expected? add one level down below selected requirement in 3.7.1)

#### 3.11.5 What are the milestones to the requirement? (cross out the past milestones, add new beneath)

*< did the requirement have requirements? select a requirement (starting with ®’s) & goto 3.11 >*

*< when the bottom is reached, implement the requirement in chaoskampf >*

*< check the finished requirement against done definition & test it >*

*< if it passes, change ! prefix to +, else back to chaoskampf >*

*< select the next requirement, goto 3.10 >*

*< if all requirements have been implemented, test the prototype itself >< if all features have been implemented, test the prototype itself >*

*< check against done definition >*

*< done? test the finished prototype >*

*< tests passed? mimimize & add line to proximate ideas --- >*

*< check project definition & decide if another iteration is needed >*

*< another iteration? copy this section below, increment, remove answers & goto it >*

*< no iteration? goto next >*

## Define the deutrotype (final aestetic pass & creation of advertising materials)

### Notes:

*< Do you actually need a deutrotype? If not, goto next section. >*

### 4.1 What are the core defining features of the prototype(s)?

### 4.2 What metaphor(s) captures what the prototype(s) are all about?

### 4.3 What is the feeling(s) that defines the prototype(s)?

### 4.4 What is the main story of the prototype(s)?

*< use the R&I heuristic to find images to guide the visual development >*

### 4.5 What is the sum of all prototypes?

### 4.6 What is the deutrotype? (Embed 4.1-4.5 and R&I)

*< Dive into chaoskampf to make the deutrotype >*

### 4.7 Describe the finished deutrotype fully

*< make advertising material as needed>*

*< release the deutrotype >*

## Evaluation

*< go through this whole document from the start >*

### 5.1 What was the path to the deutrotype? (Hurdles and interesting developments)

### 5.2 What did you learn? (if you did)

### 5.3 Did you uncover any new knowledge? (Publish)

### 5.4 What should be different next time? (iterate this document)

### 5.5 How did this project further your goals?

### 5.6 What future projects did this project unlock?

*< close project and move project folder to the archive >*

The point of being done is not to finish but to get other things done

**[End date]**

|  |  |
| --- | --- |
| Proximate Ideas & thoughts: | Distal Ideas: |

# Iterative development.

## Things to add in future iterations (ideas):

* I need an even more reduced form of the cycle for subprojects
* I need better control systems for keeping me from letting projects grow, the done definition needs to be stricter
* A much stricter structure. I keep getting stuck in feature kreeps, I ned to implement the iterative nature in reality instead of just theorizing about it
* I need to be more realistic and constrained in my done definition.
* Have a way to return to this document to develop a closed project further in the future.
* It might be useful to implement milestones somehow
* Sometimes I may need to delve before starting the next iteration

## Things to vary/experiment with:

* Do I need to describe the prototype or is the list of features enough?
* Should each milestone have a cycle of delving?
* Do I need to test each feature?
* Should I implement prototype shards that are dealt with as if they where stand-alone prototypes?
* It seems as if shards could be very useful, a sort of step between delusion and prototype,
* Features or milestones?
* Perhaps each iteration should have its own idea dump

## Hypotheses:

## Points of friction (not necessarily bad):