The Final Report of The Summer Research Projects

Hu Tao

September 24, 2024

Chapter 1

Predicting the Antigenic Evolution of SARS-COV-2 with Deep Learning

Keywords: SARS-COV-2, virus, evolution, deep learning

1.1 Introduction

According to Alberto et al. 3.2, SARS-CoV-2 is especially known for destructing the immune systems of hosts. SARS-CoV-2 escapes immunity mainly in two ways which are harassing mitochondria and demaging the antigenic presentation. The later one is the main focus of this paper which has to do with the RBD (receptor binding domain) sequence of the SARS-CoV-2 virus. Wenkai et al. gives us an good way to identify the trend of virus evolution using deep learning methods.

Our research is very important because it can help us to predict the future viral varients and to develop new vaccines and treatments for COVID-19.

1.2 Methods and Data

1.2.1 Hypothesis

1. Under high immune pressure, the virus would tend to escape the antibody neuralization for a short time;

2. The future viral varients tend to have a higher antibody escaping potential without losing much ACE-2 binding ability under high immune pressure.

1.2.2 Methods

ESM-1b Language Model

The ESM-1b Language Model provides a map $\chi^N \to \mathbb{R}^K$, where χ^N is the set of all protein sequences of length N. In other words, the ESM-1b model gives a vector representation of each protein sequence.

This model is trained to predict the binding specificity for ACE-2 and eight antibodies.

The Multi-Task Learning Model

The multi-task learning model is a deep learning model that is trained to predict the binding specificity for ACE-2 and eight antibodies. The model is trained using the ESM-1b language model and the training data from the GISAID database. The model is composed of three modules: the sequence feature extractor, the structure feature extractor and the sets of all nine classification heads.

1. The sequence feature extractor is a deep neural network that takes the protein sequence as input and outputs a dense vector representation of the sequence. The sequence feature extractor can be chosen to be the ESM-1b language model.

2. For the structure feature extractor, we represent the 3D structure as a k-nearest neighbor graph $g = (V, E)$. The we used Structured Transformer to map the transformed graph into a dense vector representation.

3. The classification heads are used to classify the protein sequences. There are nine classification heads, each classification head corresponds to one of the nine classification tasks.

4. The loss function is determined by a certain formula.

The Modified Genetic Algorithm

The genetic algorithm is a global optimization algorithm that is inspired by the process of natural selection. The algorithm maintains a population of candidate solutions and iteratively evolves them using crossover, mutation, and selection operators.

The modified genetic algorithm used by Wenkai et al. [3.2] follows the following steps:

1. Select the initial sequence from the GISAID database;

2. Given a sequence $\vec{x} = (x_1, x_2, \ldots, x_n)$, choose any amino acid x_i randomly and replace it with a neighbor of x_i according to the BLUSUM62 matrix;

3. Using the trained multi-task learning model to calculate the fitness of the new sequence;

4. The crossover operations are performed. In other words, we select the pairs of sequences with the probabilities proportional to their fitness values. Using two samples to generate a new sequence and continue our iteration.

1.2.3 Data

The data used in this study is the SARS-CoV-2 virus sequences from the GISAID database. The GISAID database is a global initiative that provides open access to genomic data of influenza viruses and the SARS-CoV-2 virus. The database contains a large number of SARS-CoV-2 virus sequences from different countries and regions. The sequences are collected from patients with COVID-19 and are used for research purposes.

1.3 Acknowledgements

I would like to thank my instructor, Jian Zu, who has invited me to the fantastic field of Predicting the antigenic evolution with deep learning. He always answers my questions patiently and gives me a lot of help.

I would also like to thank my senior schoolmate, Kexin Guo, who has answered many of my trivial questions. That's a great help to a freshman in this field!

Chapter 2

Trials and Errors in Reproducing the Article

2.1 The First Study Outcome: Learning Some Fundamental Knowledge of Pytorch

The best place to learn Pytorch is the official PyTorch tutorials at

https : //pytorch.org/tutorials/beginner/basics/intro.html

.

Due to constrained time, I only learned the first three chapters of the Pytorch tutorials, which are "Quickstart", "Tensors" and "Datasets and DataLoaders".

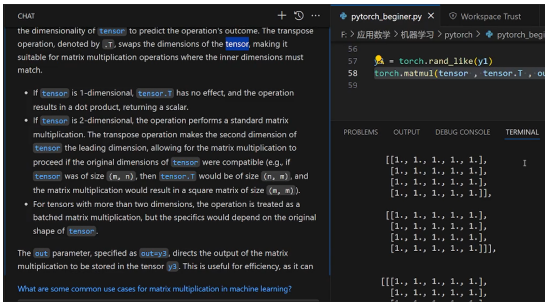

Figure 2.1: How I learned Pytorch

2.2 The First Obstacle: Find the Data Source Provided by Han et al. and Download the Data

It is mentioned by the Author that the data source is the GISAID database. However, it takes a GISAID account to download the data. In order to download the data, I have to apply for a GISAID account, which is not easy. Therefore, I found a QQ chat group that shares GISAID accounts. I borrowed an account from the group.

The GISAID database only allows one user to download at most 10000 sequences at a time. However, the author mentioned that he used 5,483,918 sequences. The author also mentioned that the RBD sequences he used are available at the website

$$https://doi.org/10.55876/gis8.230510mg$$

. However, this website does not give direct information about the exact RBD sequence.

Thus, I would have to find another way to download the data. I used the python crawler to download the data.

There used to be python crawlers that can download the data from the GISAID database. However, the GISAID database has updated its security system and the old python crawlers are no longer useful. I have to write a new python crawler to download the data.

By communicating with students who are major in computer science in XJTU and USTC, I learned that the best way to write a python crawler for me is to use the "selenium" package. The "selenium" package is a package that can be used to automate web browsers. I learned some basis operations of selenium from the youtube video "Python Selenium Tutorial" by Tech With Tim. I also recaped something about the HTML language.

I tackled with the issues of the function "WebDriverWait" and finally wrote a python crawler that can reach the download page. However, in the download page, there is a frame that cannot be accessed by the python crawler. I have to find another way to download the data.

In first figure of 2.2, I was not able to access to the frame and tick the checkbox at first. I successed after I learned something about iframe from a student major in energy and power engineering in XJTU. However, I was not able to access to the second frame and tick the checkbox in the second figure of 2.2, although the problem appears to be the same, the solution is not the same. Until now, I didn't find a way to download the data.

```
        from selenium import webdriver
        from selenium.webdriver.chrome.service import Service
        from selenium.webdriver.common.by import By
        from selenium.webdriver.common.keys import Keys
        from selenium.webdriver.support.ui import WebDriverWait
        from selenium.webdriver.support import expected_conditions as
            EC
        # from webdriver_helper import debug , get_webdriver
        from selenium.webdriver.chrome.options import Options
        import time

        opt = Options()
        opt.add_experimental_option('excludeSwitches' , ['enable-
            automation'])
```

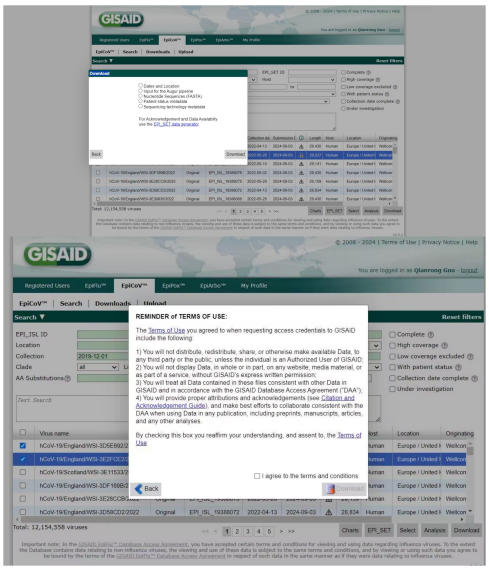

Figure 2.2: I Tried and Successfully Accessed the First Frame But Failed to Access the Second Frame

```
13
14          service = Service(executable_path = 'chromedriver.exe')
15          driver = webdriver.Chrome(service = service , options = opt)
16          webdriver.Chrome(service = service)
17
18          driver.get('https://gisaid.org/')
19
20          time.sleep(5)
21
22          WebDriverWait(driver, 5).until(EC.presence_of_element_located((
                By.CLASS_NAME , "menuicon")))
23          menuicon = driver.find_element(By.CLASS_NAME , "menuicon")
24          menuicon.click()
25
26          WebDriverWait(driver, 5).until(EC.presence_of_element_located((
                By.PARTIAL_LINK_TEXT , "Login")))
27
28          link = driver.find_element(By.PARTIAL_LINK_TEXT , "Login")
29          link.click()
30          # If link is found successfully, print "yes", else "no"
31          print("Found the login icon!")
32
33          WebDriverWait(driver, 5).until(EC.presence_of_element_located((
```

```
            By.ID , "elogin")))

        input_username = driver.find_element(By.ID , "elogin")
        input_username.clear()
        input_username.send_keys("qrguo")

        input_password = driver.find_element(By.ID , "epassword")
        input_password.clear()
        input_password.send_keys("Db2BXpWx" + Keys.ENTER)

        time.sleep(5)

        WebDriverWait(driver, 5).until(EC.presence_of_element_located((
            By.XPATH , "/html/body/form/div[5]/div/div[2]/div/div[1]/div
            /div/div[3]")))

        search_link = driver.find_element(By.XPATH , "/html/body/form/
            div[5]/div/div[2]/div/div[1]/div/div/div[3]")
        search_link.click()
        #print(driver.page_source)
        print("Found the search icon!")

        time.sleep(5)

        XPATH_start_date = "/html/body/form/div[5]/div/div[2]/div/div
            [2]/div[1]/div/table/tbody/tr[4]/td[2]/div[1]/div/div[1]/
            input"
        WebDriverWait(driver, 5).until(EC.presence_of_element_located((
            By.XPATH , XPATH_start_date)))
        input_start_date = driver.find_element(By.XPATH ,
            XPATH_start_date)
        input_start_date.clear()
        input_start_date.send_keys("2019-12-01")

        XPATH_end_date = "/html/body/form/div[5]/div/div[2]/div/div[2]/
            div[1]/div/table/tbody/tr[4]/td[2]/div[3]/div/div[1]/input"
        input_end_date = driver.find_element(By.XPATH , XPATH_end_date)
        input_end_date.clear()
        input_end_date.send_keys("2022-05-31" + Keys.ENTER)

        # Find patterns of the XPATHs of the checkboxes
        # The first on the first page: /html/body/form/div[5]/div/div
            [2]/div/div[2]/div[2]/div[1]/div[3]/table/tbody[2]/tr[1]/td
            [1]/div/input
        # The second on the first page: /html/body/form/div[5]/div/div
            [2]/div/div[2]/div[2]/div[1]/div[3]/table/tbody[2]/tr[2]/td
```

```
    [1]/div/input
# Turn from the first page to the second page:  /html/body/form
    /div[5]/div/div[2]/div/div[2]/div[2]/div[2]/div[2]/div/span
    [3]/a[1]
# The first on the second page: /html/body/form/div[5]/div/div
    [2]/div/div[2]/div[2]/div[1]/div[3]/table/tbody[2]/tr[1]/td
    [1]/div/input
# The first on the seventh page: /html/body/form/div[5]/div/div
    [2]/div/div[2]/div[2]/div[1]/div[3]/table/tbody[2]/tr[1]/td
    [1]/div/input
# Turn from the sixth page to the seventh page: /html/body/form
    /div[5]/div/div[2]/div/div[2]/div[2]/div[2]/div[2]/div/span/
    a[3]
# Turn from the second page to the third page: /html/body/form/
    div[5]/div/div[2]/div/div[2]/div[2]/div[2]/div[2]/div/a[3]
# Turn from the third page to the fourth page: /html/body/form/
    div[5]/div/div[2]/div/div[2]/div[2]/div[2]/div[2]/div/a[3]

items_per_page = 50 # The number of items per page
pages_per_iteration = 100 # The number of pages per iteration

"""
XPATH_next_page = "/html/body/form/div[5]/div/div[2]/div/div
    [2]/div[2]/div[2]/div[2]/div/a[1]"
#                   /html/body/form/div[5]/div/div[2]/div/div
    [2]/div[2]/div[2]/div[2]/div/a[1] 1st to 2nd
#                   /html/body/form/div[5]/div/div[2]/div/div
    [2]/div[2]/div[2]/div[2]/div/a[3] 2nd to 3rd
#                   /html/body/form/div[5]/div/div[2]/div/div
    [2]/div[2]/div[2]/div[2]/div/a[3] 3rd to 4th
time.sleep(10)
WebDriverWait(driver, 5).until(EC.presence_of_element_located((
    By.XPATH , XPATH_next_page)))
next_page = driver.find_element(By.XPATH , XPATH_next_page)
print("Found the next page icon!")
next_page.click()

Testing the function to flip the page
"""


for i in range(0, pages_per_iteration):
    for j in range(0, items_per_page // 25):
        XPATH_checkbox = "/html/body/form/div[5]/div/div[2]/div
            /div[2]/div[2]/div[1]/div[3]/table/tbody[2]/tr[" +
            str(j + 1) + "]/td[1]/div/input"
```

```
96                  #                   /html/body/form/div[5]/div/div[2]/div
                        /div[2]/div[2]/div[1]/div[3]/table/tbody[2]/tr["+
                        str(j + 1) + "]/td[1]/div/input
97                  #                   /html/body/form/div[5]/div/div[2]/div
                        /div[2]/div[2]/div[1]/div[3]/table/tbody[2]/tr[1]/td
                        [1]/div/input
98                  #WebDriverWait(driver, 5).until(EC.
                        presence_of_element_located((By.XPATH ,
                        XPATH_checkbox)))
99                  time.sleep(5)
100                 print(f"Found the {j + 1} checkbox!")
101                 checkbox = driver.find_element(By.XPATH ,
                        XPATH_checkbox)
102                 checkbox.click()
103
104             #time.sleep(8)
105
106             print(f"Ticked all items on page {i + 1}!")
107
108             XPATH_download = "/html/body/form/div[5]/div/div[2]/div/div
                    [2]/div[2]/div[2]/div[3]/button[5]"
109             WebDriverWait(driver , 20).until(EC.
                    presence_of_element_located((By.XPATH , XPATH_download))
                    )
110             button_download = driver.find_element(By.XPATH ,
                    XPATH_download)
111             button_download.click()
112             time.sleep(5)
113
114
115
116             # driver.switch_to.window(driver.window_handles[1]) #
                    Switch to the new tab
117             with open("page_source_WANTING_AUGAR_PIPELINE.txt", "w",
                    encoding="utf-8") as file:
118                  file.write(driver.page_source)
119
120             #input('Press Enter to continue...') #
121             iframe_id = "sysoverlay-wid_sjdx3j_1jyx"
122             #           "sysoverlay-wid_sjdx3j_1i8f"
123             iframe_class_name = "sys-overlay-style"
124             download_frame = driver.find_element(By.CLASS_NAME ,
                    iframe_class_name) # The frame for the download
125             driver.switch_to.frame(download_frame) # switch to the
                    frame for the download so that the download button can
                    be clicked
```

```
126             print("go to frame for download")
127
128
129             XPATH_CHOSEN = '//*[@id="ce_sjlgo0_oc_2"]'
130             #               '//*[@id="ce_sjdx3j_on_2"]'
131             #               "//input[@type='radio' and @value='
                    augar_pipeline']"
132             #                //*[@id="ce_sjdx3j_on_2"]
133             #                /html/body/form/div[5]/div/div[1]/div/div/
                    table[1]/tbody/tr/td[2]/div/div[1]/div[2]/div[2]/input
134             #               "/html/body/form/div[5]/div/div[1]/div/div/
                    table[1]/tbody/tr/td[2]/div/div[1]/div[2]/div[2]/input"
                    # input for the augar pipeline
135             #                /html/body/form/div[5]/div/div[1]/div/div/
                    table[1]/tbody/tr/td[2]/div/div[1]/div[2]/div[2]/input
136             #                /html/body/form/div[5]/div/div[1]/div/div/
                    table[1]/tbody/tr/td[2]/div/div[1]/div[2]/div[2]/input
137             #WebDriverWait(driver , 15).until(EC.
                    presence_of_element_located((By.XPATH , XPATH_CHOSEN)))
138             ID_CHOSEN = "ce_sjdx3j_on_2"
139             time.sleep(10)
140             # input('Press Enter to continue...') #
141             tick_chosen = driver.find_element(By.XPATH , XPATH_CHOSEN)
142             print("Found the chosen tick!")
143             tick_chosen.click()
144
145
146             time.sleep(15)
147             #input('Press Enter to continue...')
148             XPATH_DOWNLOAD_CONFIRMATION = '//*[@id="ce_sjlgo0_oi"]/div/
                    button'
149             #                             '//*[@id="ce_sjdx3j_ot"]/div/
                    button'
150             #                              /html/body/form/div[5]/div/
                    div[2]/div/div/div[2]/div/button
151             #                              /html/body/form/div[5]/div/
                    div[2]/div/div/div[2]/div/button
152             WebDriverWait(driver , 20).until(EC.
                    presence_of_element_located((By.XPATH ,
                    XPATH_DOWNLOAD_CONFIRMATION)))
153             download_confirmation = driver.find_element(By.XPATH ,
                    XPATH_DOWNLOAD_CONFIRMATION)
154             download_confirmation.click()
155
156             driver.switch_to.parent_frame()
157
```

```
158             iframe_id = "sysoverlay-wid_sjdx3j_1jyx"
159             #           "sysoverlay-wid_sjdx3j_1i8f"
160             iframe_class_name = "sys-overlay-style"
161             WebDriverWait(driver , 20).until(EC.
                    presence_of_element_located((By.CLASS_NAME ,
                    iframe_class_name)))
162             download_frame = driver.find_element(By.CLASS_NAME ,
                    iframe_class_name) # The frame for the download
163             driver.switch_to.frame(download_frame) # switch to the
                    frame for the download so that the download button can
                    be clicked
164             print("go to the 2nd frame for download")
165
166             time.sleep(15)
167             input('Press Enter to continue...')
168             XPATH_TERM_OF_USE = '//*[@id="ce_sjlgo0_t4"]/div[1]/div/
                    input'
169             # '//*[@id="ce_sjlgo0_t4"]/div[1]/div/input'
170             # '//*[@id="ce_sjdx3j_tf"]/div[1]/div/input'
171             # '//*[@id="ce_sjdx3j_tf"]/div[1]/div/input'
172             # WebDriverWait(driver , 20).until(EC.
                    presence_of_element_located((By.XPATH ,
                    XPATH_TERM_OF_USE)))
173             # The radio button were not allowed to be used with
                    WebDriverWait
174             radio_term_of_use = driver.find_element(By.XPATH ,
                    XPATH_TERM_OF_USE)
175             radio_term_of_use.click()
176
177             input('Press Enter to continue...')
178
179             time.sleep(15)
180             XPATH_LAST_FOR_DOWNLOAD = '//*[@id="ce_sjlgo0_t7"]/div/
                    button' # The last character plus 3
181             #                         '//*[@id="ce_sjdx3j_ti"]/div/
                    button'
182             last_buttion_for_download = WebDriverWait(driver , 20).
                    until(EC.presence_of_element_located((By.XPATH ,
                    XPATH_LAST_FOR_DOWNLOAD)))
183             last_buttion_for_download.click()
184
185             input("Press Enter to continue...")
186
187             # driver.switch_to.window(main_window) # Switch back to the
                     main window
188             driver.switch_to.parent_frame()
```

```
189
190             if i < pages_per_iteration - 1:
191                 XPATH_next_page = "/html/body/form/div[5]/div/div[2]/
                        div/div[2]/div[2]/div[2]/div[2]/div/a[3]"
192                 if i == 0:
193                     XPATH_next_page = "/html/body/form/div[5]/div/div
                            [2]/div/div[2]/div[2]/div[2]/div[2]/div/a[1]"
194                 time.sleep(20)
195                 WebDriverWait(driver, 5).until(EC.
                        presence_of_element_located((By.XPATH ,
                        XPATH_next_page)))
196                 next_page = driver.find_element(By.XPATH ,
                        XPATH_next_page)
197                 next_page.click()
198
199
200
201         time.sleep(10000)
202
203         driver.quit()
```

2.3 The Second Obstacle: Determine the Omitted Files in the Code Repository Provided by Han et al.

The author mentioned that the code repository is available at the website

$$https://github.com/WHan-alter/MLAEP$$

. However, the code repository does not contain the files that are necessary for the codes to run.

I collected all the files that are necessary from the codes but omitted in the code repository ("OK" means the files that can be generated).

```
1         prediction.csv
2         prediction.npy
3         embedding.npy
4             ../data/covid_model.csv
5         "../data/RBD0308.fasta"
6         '../data/supp_4.xlsx'
7         '../result/target/ev_cache/{namespace}_rand_idx.txt'
8         '../result/target/ev_cache/cov_seqs.pkl'----------------------------
              -------OK
9         '../data/unique_GISAID.csv'
10        '../result/gisaid_embedding.npy'
11        '../data/gisaid_score.csv'
12        '../result/target/ev_cache/cov_adata.h5ad'
13        '../result/gisaid_synthetic_score.csv'
```

```
        '../result/embedding_synthetic.npy'
        '../data/init_seq_2022-01-01_0.8.csv'
```

I also collected the files that can be generated by the code is the code could run:

```
        '../result/target/ev_cache/cov_seqs.pkl'
        '../result/target/ev_cache/cov_adata.h5ad'
        '../result/target/ev_cache/{namespace}_knn30'
        '../result/target/ev_cache/cov_knn30'
        'figures/combined_embedding_pc2.png'
        'figures/All_site_%s_%s_%s_seqlogo_add%s.png'%(mode,"top"+str(
            top),measure,str(add))
        embed_fname = ('target/{}/embedding/{}_{}.npy'.format(args.
            namespace, args.model_name, args.dim))
        embed_fname = embed_prefix + '.{}.npy'.format(batchi)
        dirname = ('target/{}/semantics/cache'.format(args.namespace))
        ofname = dirname + '/{}_mutations.txt'.format(args.namespace)
        cache_fname = dirname + ('/analyze_semantics_{}_{}_{}.txt'.
            format(plot_namespace, args.model_name, args.dim))
        dirname = ('target/{}/reinfection/cache'.format(args.namespace)
            )
        mkdir_p(dirname)
        fname = dirname + '/{}_mut_{}.txt'.format(args.namespace,
            namespace)
```

Many files in this code are omitted in the code repository and cannot be generated by the code. Stil, I cannot find a way to fill up the data needed.

2.4 The Second Study Outcome: Learning to Use the Virtual Machine

At the beginning, when I was running the command line "conda env create -f environment.yml" on anaconda prompt, I found that many of the files cannot be downloaded and I worried that my python version was not updated completely and this code need to run on a Linux environment instead of a Windows environment. In this way, I learned to use the virtual machine. It is suggested that the ideal virtual machine is the VMware work station. I downloaded CENTOS7 and used CentOS Linux release 7.9.2009 as the virtual machine(see Figure **??**). I also learned to use the command line in the virtual machine. Then, I learned how to surf the internet with the virtual machine. Through the internet, I downloaded a new version of anaconda.

However, there is still inconvenience to use the virtual machine. Hence, I return to the Windows environment and try to find a way to run the code on the Windows environment. Later, I discovered that I have no way to prove that the problems all come from the environment. My senior schoolmate, Kexin Guo, told me that she succeeded without using a virtual machine.

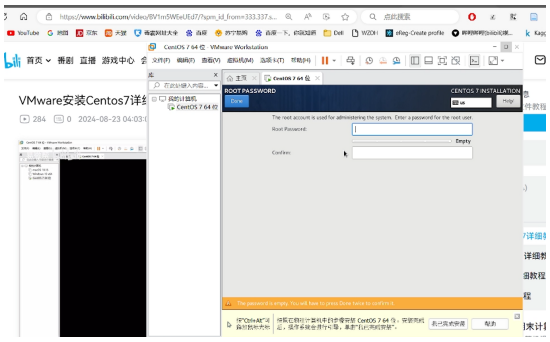

Figure 2.3: I Used the Virtual Machine