DIPARTIMENTO DI
CHIMICA E BIOLOGIA "A. ZAMBELLI"

UNIVERSITÀ DEGLI STUDI
DI SALERNO

AlphaMod Supplementary File 2
User Manual

The Critical Assessment of Protein Structure Prediction Competition (CASP)

The CASP competition provides an independent mechanism for the assessment of methods of protein structure modeling. Specifically for CASP 14 starting from May through August 2020, CASP organizers have been posting on their website sequences of unknown protein structures for modeling. Protein models have been collected from May through mid-September, and evaluated as the experimental coordinates become available. In the summer and fall, the tens of thousands of models submitted by approximately 100 research groups worldwide are processed and evaluated. Independent assessors in each of the prediction categories bring independent insight into their assessment. Tools for viewing, comparison, and analysis of submitted models are available from this website.

Contents

1 Requirements

1.1 Hardware requirements

AlphaMod was tested on a PC with the following specifications:

- **Processor:** Intel® Xeon(R) CPU E5-2630 v4 @ 2.20GHz × 20
- **RAM Memory:** 32GB
- **GPU:** GeForce GTX 1080
- **Disk Capacity:** 2,3 TB
- **Operating System:** Ubuntu 20.04.4 LTS / 64 bit

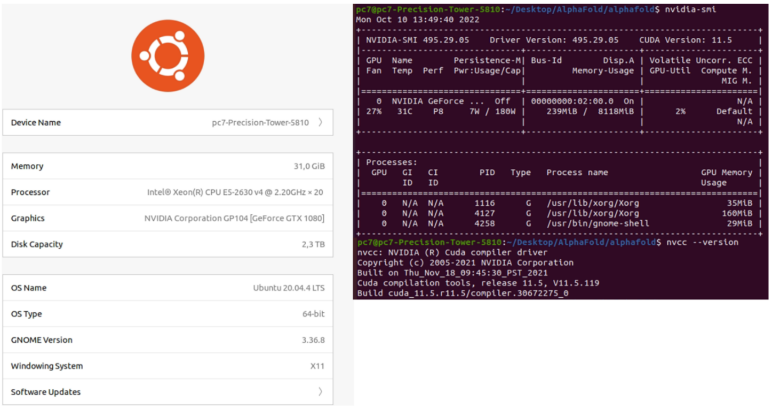


Figure 1: Hardware specifications of the computer we used to run AlphaFold in our research.

1.2 Software requirements

To run ***AlphaMod*** you will need the following required software:

- NVIDIA, CUDA and CuDNN drivers.
- Python3 v3.8.10.
- PyCharm (Optional).
- Docker.
- AlphaFold.
- Modeller (Optional).
- Phenix (Optional).

1.2.1 NVIDIA, CUDA and CuDNN driver installation

AlphaMod uses CUDA Toolkit 11.5.1 (November 2021) as shown in Fig.2.

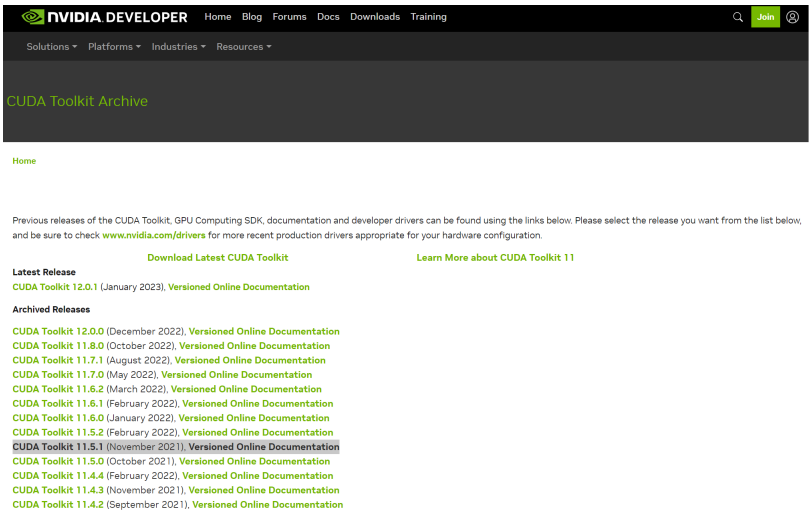


Figure 2: Screenshot of NVIDIA's tool kit archive, highlighting version 11.5.1

Please click on version CUDA Toolkit 11.5.1 (November 2021). According to our described hardware at Section 1, the following options were chosen as shown in Fig.3

- **Operating System:** Linux
- **Architecture:** x86_64
- **Distribution:** Ubuntu
- **Version:** 20.04
- **Installer Type:** runfile (local)

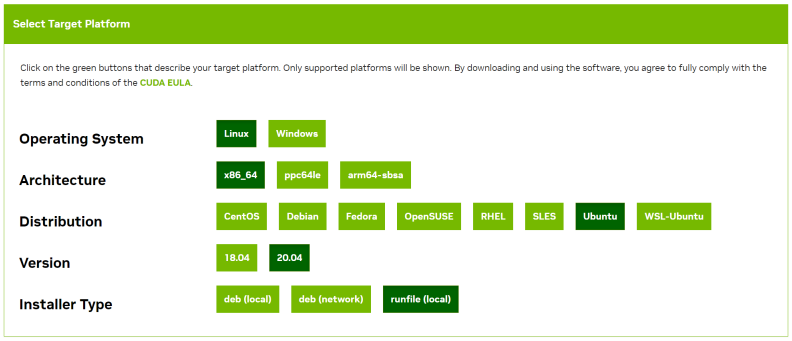


Figure 3: Screenshot of NVIDIA's CUDA Toolkit 11.5.1 (November 2021) selected Target Platform.

The screen shown in Fig.4 will be prompted

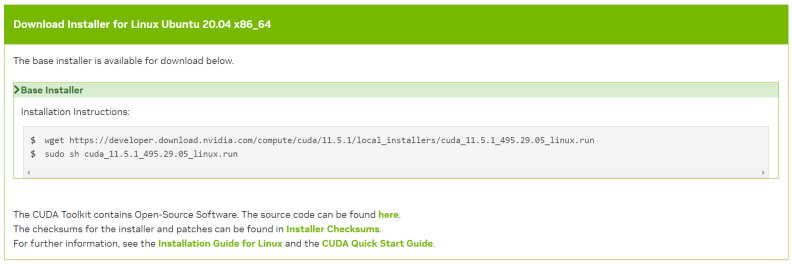


Figure 4: Screenshot of NVIDIA's CUDA Toolkit 11.5.1 (November 2021) installer download screen.

At the desired location (in our example Downloads folder) copy and paste the installation instructions from Fig.4 into your terminal as shown in Fig.5

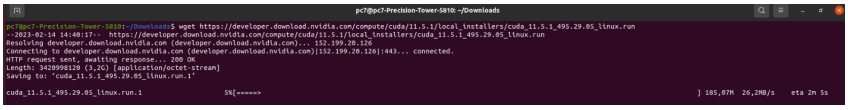


Figure 5: Screenshot of our terminal downloading NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

Once NVIDIA's CUDA Toolkit 11.5.1 (November 2021) has been downloaded, it is necessary to give executing permissions to the file with the following command ***sudo chmod +x cuda_11.5.1_495.29.05_linux.run*** as shown in figure 6.

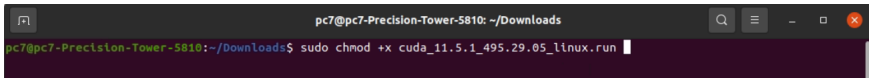


Figure 6: Screenshot of our terminal giving executing permissions to NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

Now, to proceed with the installation of NVIDIA's CUDA Toolkit 11.5.1 (November 2021), please execute the file with the following command ***sudo bash cuda_11.5.1_495.29.05_linux.run*** as shown in figure 7.

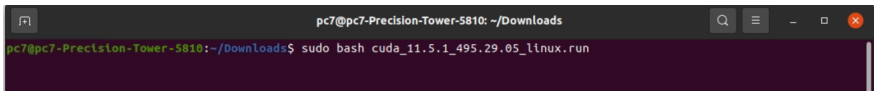


Figure 7: Screenshot of our terminal executing NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

To proceed with the installation it is necessary to ***accept*** the *End User License Agreement*, if you agree type ***accept*** to proceed with the installation as shown in figure 8.

Figure 8: Screenshot of our terminal accepting end user licence of NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

By default NVIDIA's CUDA Toolkit 11.5.1 (November 2021) has drivers and CUDA TOOLKIT for installation, in our research we made the default installation as shown in figure 9.

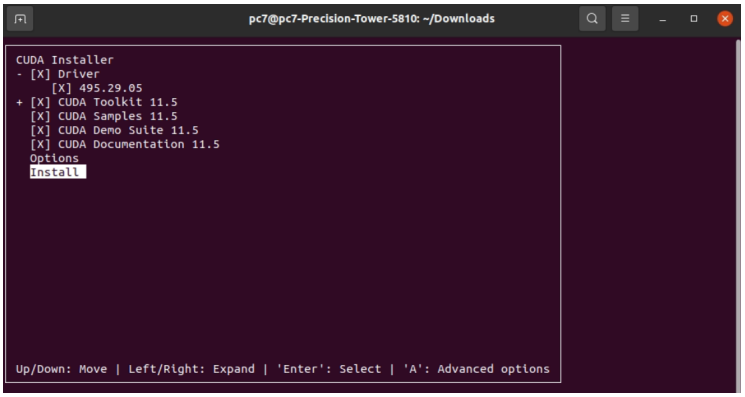


Figure 9: Screenshot of our terminal of the options used in the installation of NVIDIA's CUDA Toolkit 11.5.1 (November 2021)

After rebooting your pc you can confirm your NVIDIA drivers installation with the following command ***nvidia-smi*** as shown in figure 10.

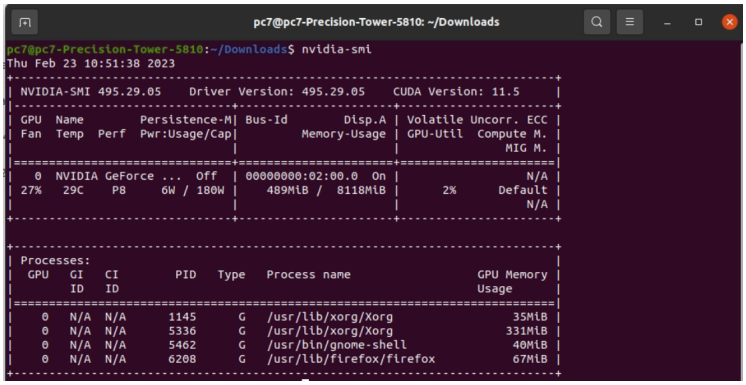


Figure 10: Screenshot of our terminal showing installed NVIDIA drivers

Now we need to activate the ***environmental variables*** for our Linux system to recognize NVIDIA's TOOLKIT. To activate the environmental variables, we will modify the **bashrc** file with the **nano** editor. Please open the **bashrc** file with **nano** text editor using the following command: ***nano /.bashrc*** as shown in figure 11.

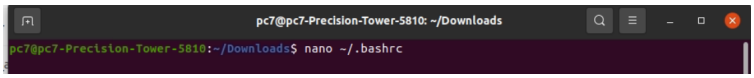


Figure 11: Screenshot of our terminal opening bashrc file with nano text editor

To let our Linux system recognize NVIDIA's TOOLKIT we will add three lines of text to the **bashrc** file using the **nano** editor:

- **EX1:** export PATH=$PATH:/usr/local/cuda-11.5/bin
- **EX2:** export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11.5/lib64
- **EX3:** export CUDADIR=/usr/local/cuda-11.5

At the end of the file **bashrc** please copy the lines **EX1, EX2 and EX3** as shown in figure 12.

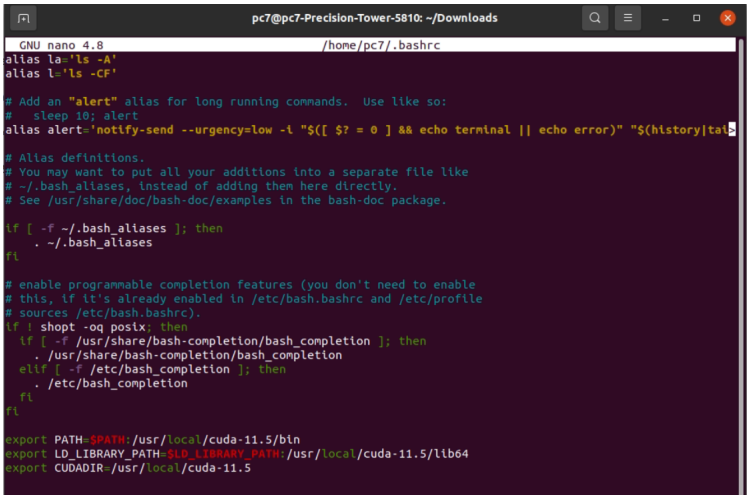


Figure 12: Screenshot of our terminal copying EX1, EX2 and EX3 on our bashrc file with nano text editor

To verify NVIDIA's TOOLKIT installation please type the following command: ***nvcc –version*** as shown in figure 13.

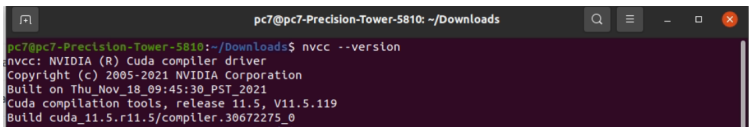


Figure 13: Screenshot of our terminal verifying the installation of NVIDIA's TOOLKIT

Furthermore, we activated cuDNN tools. To activate cuDNN we went to NVIDIA's cuDNN official website as shown in figure 14.

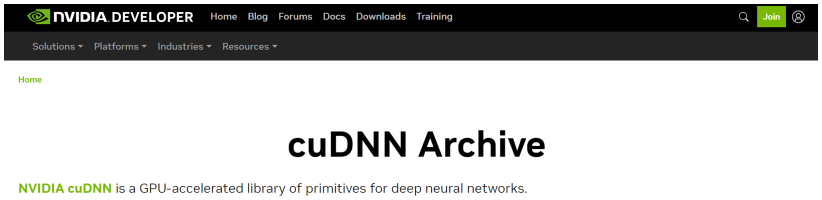


Figure 14: Screenshot of NVIDIA's cuDNN archive.

Please find the associated cuDNN version for your CUDA TOOLKKIT version. In our case we selected it by using the following criteria:

- **C1:** cuDNN v8.3.1 (November 22nd, 2021), for CUDA 11.5
- **C2:** Local Installer for Linux x86_64 (Tar)

An example can be seen in figure 15.

Figure 15: Screenshot of NVIDIA's cuDNN archive selecting criteria C1 and C2.

Once the file has been downloaded please **unzip** the file with the following command: ***tar -xf cudnn-linux-x86_64-8.3.1.22_cuda11.5-archive.tar.xz*** as show in figure 16.

Figure 16: Screenshot of our terminal unzipping cuDNN.

At **Downloads** folder please copy the required files using the following commands:

- **CP1:** sudo cp cudnn-*-archive/include/cudnn*.h /usr/local/cuda/include
- **CP2:** sudo cp -P cudnn-*-archive/lib/libcudnn* /usr/local/cuda/lib64
- **CP3:** sudo chmod a+r /usr/local/cuda/include/cudnn*.h /usr/local/cuda/lib64/libcudnn*

An example can be seen in figure 17.

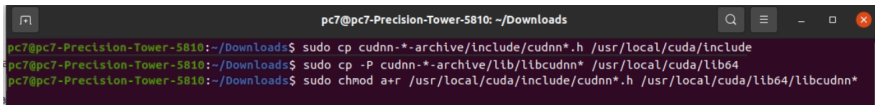


Figure 17: Screenshot of our terminal executing **CP1, CP2 and CP3**.

To verify cuDNN's installation go to **NVIDIA_CUDA-11.5_Samples** located at **home** as seen in figure 18.

Figure 18: Screenshot of our **home** directory with focus on **NVIDIA_CUDA-11.5_Samples** folder.

To create the test examples to confirm cuDNN installation we run the following command: ***sudo make -k*** as shown in figure

Figure 19: Screenshot of our terminal executing **sudo make -k** command to create test samples for cuDNN.

Once the **sudo make -k** command has finished executing, we proceed to enter the **release** folder with the following command: ***cd bin/x86_64/linux/release/*** and then we execute the test query with ***./deviceQuery*** as show in in figure 20.

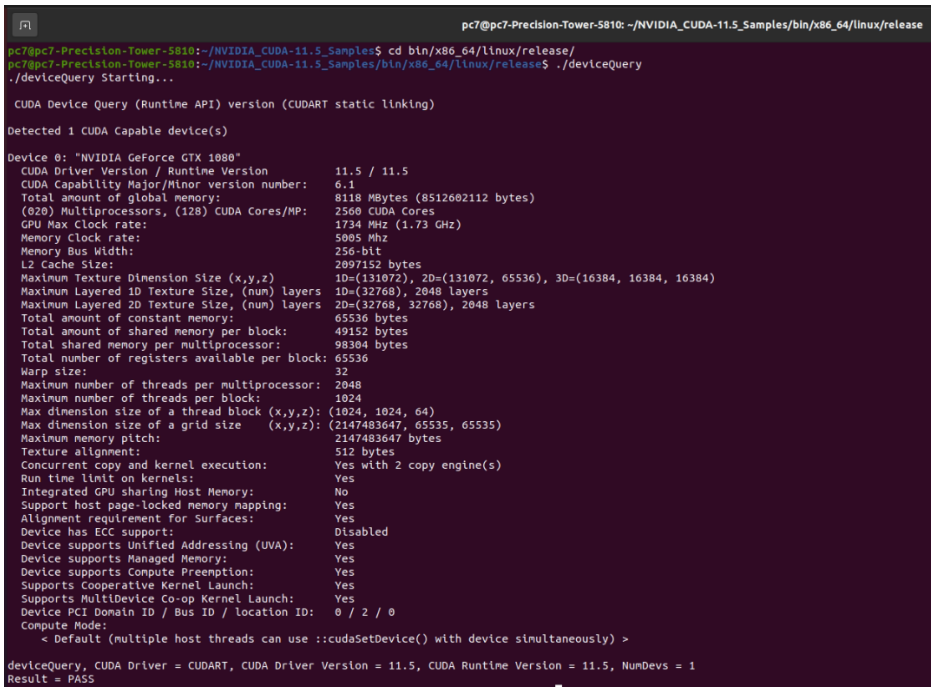


Figure 20: Screenshot of our terminal executing cuDNN test query.

The last row of the execution states: ***Result = PASS***, confirming that our installation was done correctly.

1.2.2 Python3 installation

Before installing Python3 we can verify if the system has it already installed with the following command: ***python3 –version*** as shown in figure 21.

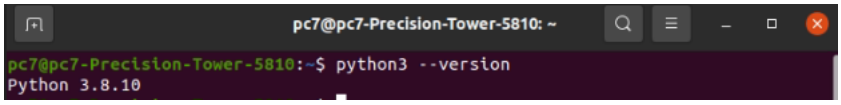


Figure 21: Screenshot of our terminal checking python3's version.

If python3 version 3.8.10 is installed in your local machine we can proceed to 1.2.3, otherwise we will start with the installation process.

First, we will **update and refresh repository lists** with the following command: ***sudo apt update*** as shown in figure 22

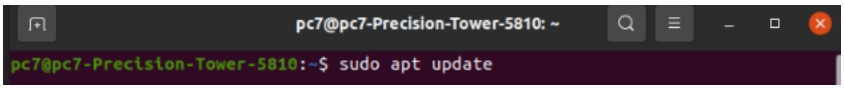


Figure 22: Screenshot of our terminal updating and refreshing repository lists.

Followed, by the python3 installation command s***sudo apt install python3.8.10*** as shown in figure 23.

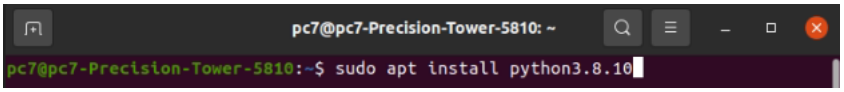


Figure 23: Screenshot of our terminal installing Python v3.8.10.

1.2.3 Pycharm Installation (Optional)

To install PyCharm, please open a terminal and prompt the following command: ***sudo snap install pycharm-community –classic*** as shown in figure 24.

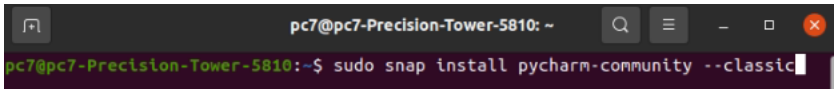


Figure 24: Screenshot of our terminal installing PyCharm community edition.

1.2.4 Docker Installation

For the installation of Docker we will use a bash script. First, we will create the bash script file with nano editor by prompting the following command into the terminal: ***nano docker_install.sh*** as shown in figure 25

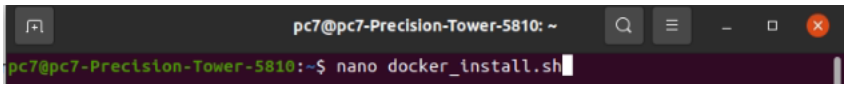


Figure 25: Screenshot of our terminal creating the docker installation bash script file.

Copy the following set of commands:

```
#!/bin/bash

sudo apt-get update

sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable" — sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

sudo docker run hello-world
```

You may run the created script by typing bash **docker_install.sh** as shown in figure 26.

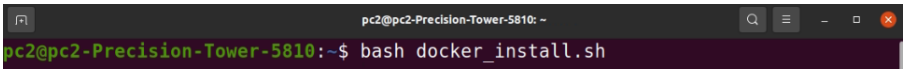


Figure 26: Screenshot of our terminal running docker installation bash script file.

You may confirm the installation was a success if you see the *"Hello World from docker"* as shown in figure 27.

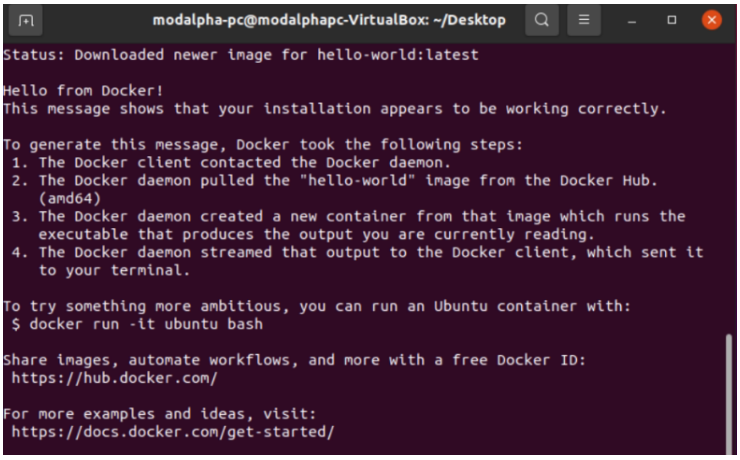


Figure 27: Screenshot of our terminal running docker "Hello World".

Finally we want to configure **docker as a non root user**, we can do it by executing the following commands:

```
#!/bin/bash

sudo groupadd docker

sudo usermod -aG docker $USER

newgrp docker
```

1.2.5 AlphaFold Installation

According to AlphaFold's official website, we have already installed docker.

The next step is to install the **NVIDIA Container Toolkit**.

To install **NVIDIA Container Toolkit** create a bash script file as shown in figure 28.

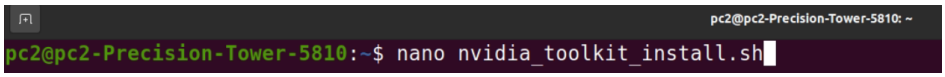


Figure 28: Screenshot of our terminal creating nvidia_toolkit_install.sh bash script.

Inside the file **nvidia_toolkit_install.sh** please copy and paste the following script:

```
#!/bin/bash

distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
&& curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey — sudo apt-key add - \
&& curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list — sudo tee /etc/apt/sources.list.d/nvidia-docker.list

sudo apt-get update

sudo apt-get install -y nvidia-docker2

sudo systemctl restart docker

sudo docker run –rm –gpus all nvidia/cuda:11.0-base nvidia-smi
```

Finally, install **nvidia_toolkit_install.sh** by executing the script as shown in figure 29.

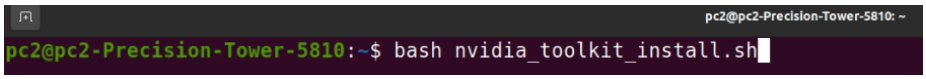


Figure 29: Screenshot of our terminal executing nvidia_toolkit_install.sh bash script.

According to AlphaFold's official website you may run AlphaFold with the instructions shown in figure 30.

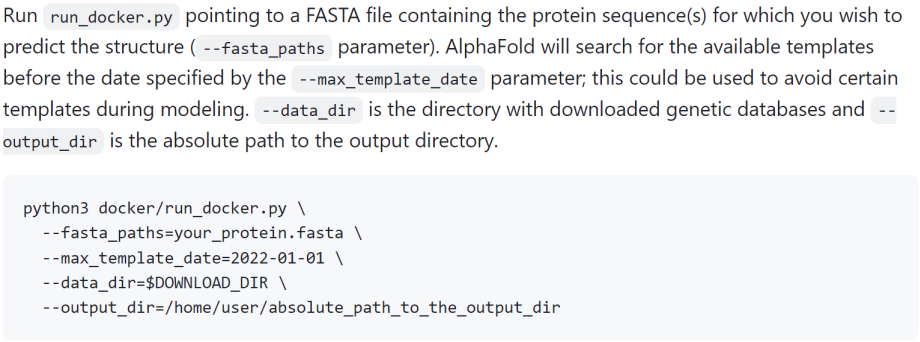


Figure 30: Screenshot of AlphaFold's official website. Instructions to run AlphaFold are shown.

1.2.6 Modeller Installation (Optional)

At Modeller's official website you may find the installation guide in case you find it necessary. **The code we have released includes a ready to use installation of Modeller, thus it is not necessary to install Modeller to run AlphaMod.**

1.2.7 Phenix Installation (Optional: Required to calculate RMSD)

You may follow Phenix's Official website for downloading and installing Phenix software.

2 Fasta Files

In this section we will describe how to acquire the same CASP14 fasta files we used in our research. These fasta files contain a string of characters representing the amino acid sequence of a protein structure.

2.1 CASP14 Target List

At the official CASP website proceed to the CASP14 Target List section by clicking as shown in Fig.31.

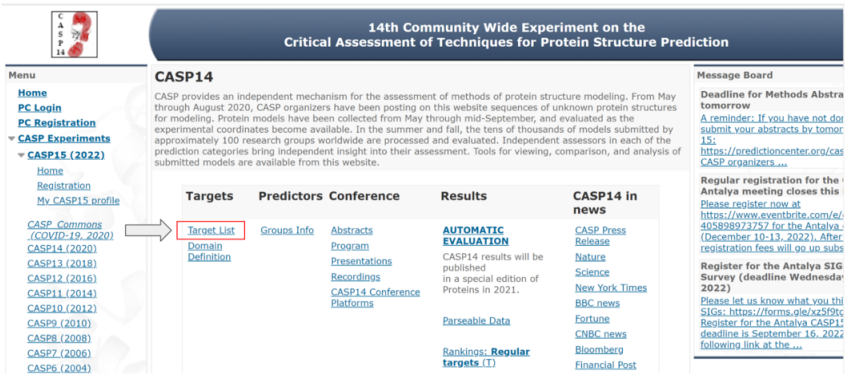


Figure 31: Screenshot of the official Critical Assessment of Techniques for Protein Structure Prediction (CASP), clicking on Target List

2.2 Downloading Aminoacid Sequences

At the Target List page, proceed to select the desired target by clicking as shown in Fig.32

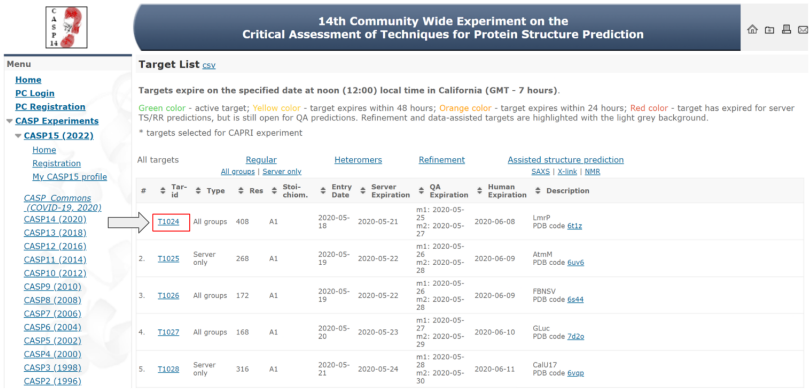


Figure 32: Screenshot of the official Critical Assessment of Techniques for Protein Structure Prediction (CASP), Selecting a Target, specifically T1024, the same approach can be used for the other targets.

At the specific target page (in this example T1024) you will find the **amino acid sequence** and its corresponding **template** as shown in Fig.33.

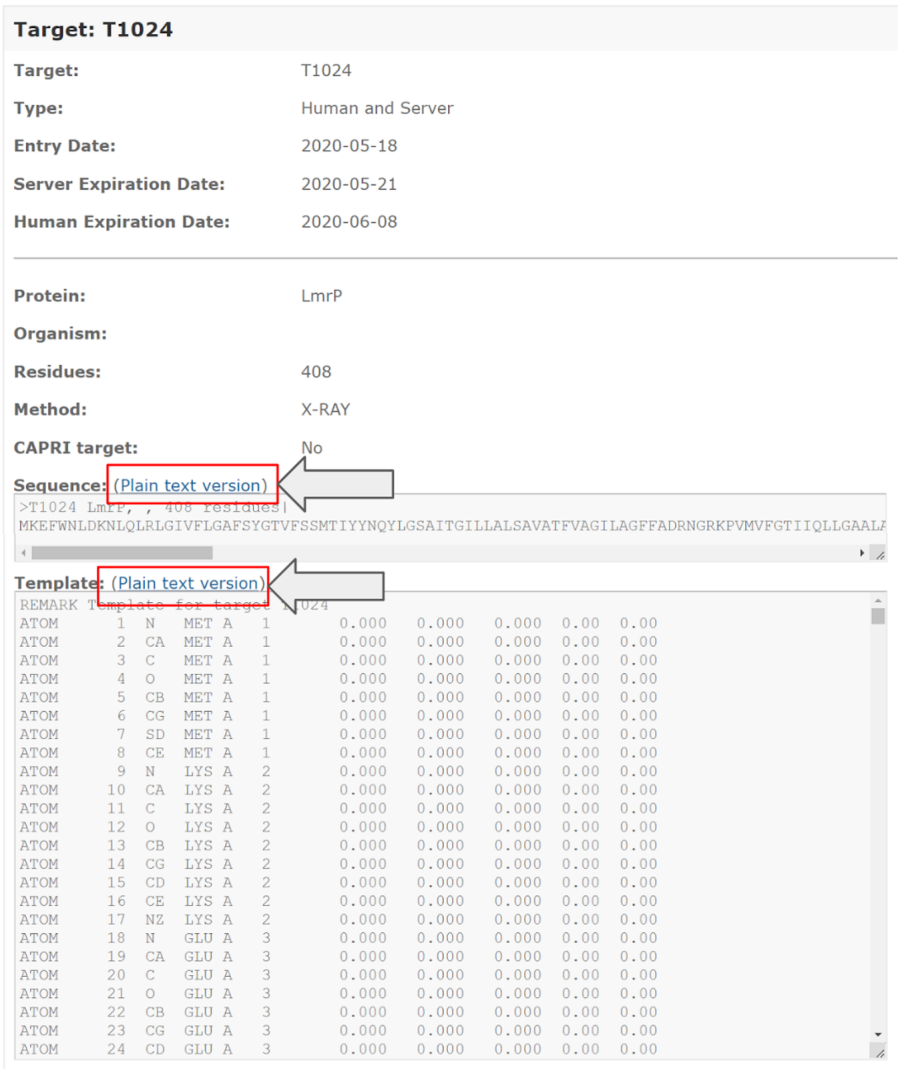


Figure 33: Screenshot of Target T1024 displaying its sequence and the template, the arrows and red boxes shows that "Plain text version" is a clickable link to its raw version.

2.3 Saving as fasta format

After clicking on “plain text version” you will be prompted to the raw version, select the whole amino acid sequence (CTRL + A), open a text editor of your preference and paste the sequence (CTRL + V) as shown in Fig.34, save it with fasta extension as shown in Fig.35.

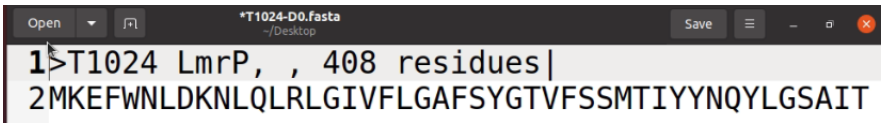


Figure 34: Process for acquiring the raw text information of the Target T1024 copy its information and pasting it on a text editor.

Figure 35: Process for saving Target T1024 with fasta extension.

2.4 Adjusting FastaFile to comply with AlphaMod guidelines

Open the saved FastaFile, in this example T1024-D0 as shown in figures 36.

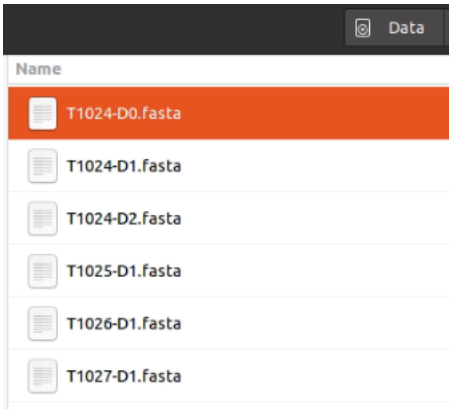


Figure 36: Opening Target T1024-D0.

We want to change the first line shown in figure 37.

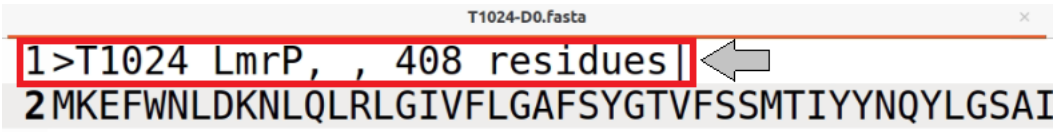


Figure 37: Opening Target T1024-D0.

We want to change the first line with the correct number of residues that were evaluated by the GDT_TS algoritm, also to put the correct extension (-D0, -D1, -D2 etc) and the word *Domain* all of this separated by commas. To do so, we go to the CASP14 Results - Automatic Evaluation official website, and we want to look for our protein target, in this case T1024-D0 as shown in figure 38 and click on it.

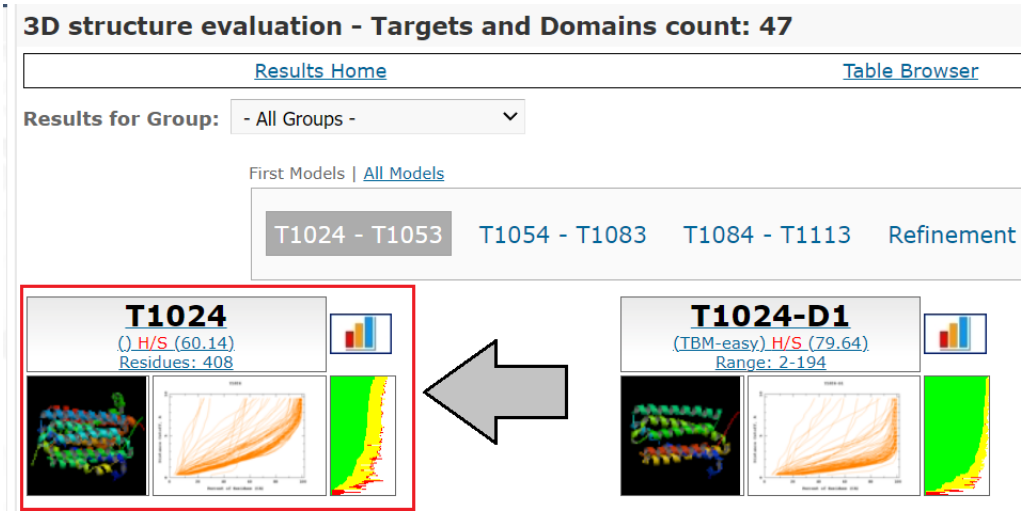
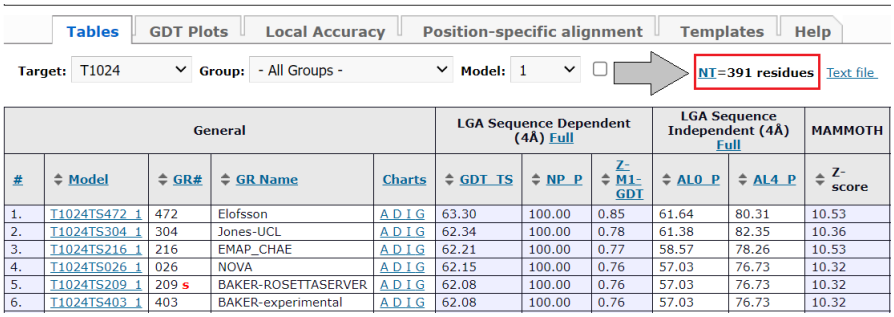


Figure 38: Screenshot of Target T1024-D0 at CASP's official website.

After clicking on the protein you will be prompted a list with the names of the competing groups and their respective scores of different metrics, nevertheless, **we are interested in the value of NT** as shown in figure 39.



General					LGA Sequence Dependent (4Å) Full			LGA Sequence Independent (4Å) Full		MAMMOTH
#	Model	GR#	GR Name	Charts	GDT_TS	NP_P	Z-M1-GDT	AL0_P	AL4_P	Z-score
1.	T1024TS472_1	472	Elofsson	A D I G	63.30	100.00	0.85	61.64	80.31	10.53
2.	T1024TS304_1	304	Jones-UCL	A D I G	62.34	100.00	0.78	61.38	82.35	10.36
3.	T1024TS216_1	216	EMAP_CHAE	A D I G	62.21	100.00	0.77	58.57	78.26	10.53
4.	T1024TS026_1	026	NOVA	A D I G	62.15	100.00	0.76	57.03	76.73	10.32
5.	T1024TS209_1	209 s	BAKER-ROSETTASERVER	A D I G	62.08	100.00	0.76	57.03	76.73	10.32
6.	T1024TS403_1	403	BAKER-experimental	A D I G	62.08	100.00	0.76	57.03	76.73	10.32

Figure 39: List of results for Target T1024-D0 at CASP14's official website.

Finally, we update our file as shown in figure 40.

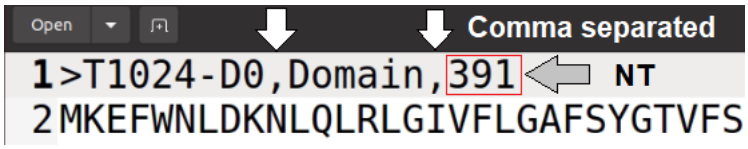


Figure 40: Screenshot of updated Target T1024-D0 fasta file to comply with AlphaMod guidelines.

3 AlphaMod

3.1 Pre-requisites

At section 1 (requirements) we mentioned that we must have installed:

- Nvidia drivers, CUDA and CuDNN installed in our local machine.
  - *Further information can be found at NVIDIA's official website.*
- AlphaFold.
  - *For installation instructions please refer to AlphaFold's GitHub*
- Modeller
  - *For installation instructions please refer to Modeller's official website* (Modeller's Installation is optional as our GitHub repository includes Modeller installed already.)
- Phenix
  - *For installation instructions please refer to Phenix's official website* (Phenix installation is optional, you may install it if you want to calculate RMSD)

3.2 Downloading AlphaMod's GitHub Repository

At AlphaMod's official repository click on *code*, at *local*, *SSH* click on the two squares on the right to copy the link as shown on figures 41 and 42.

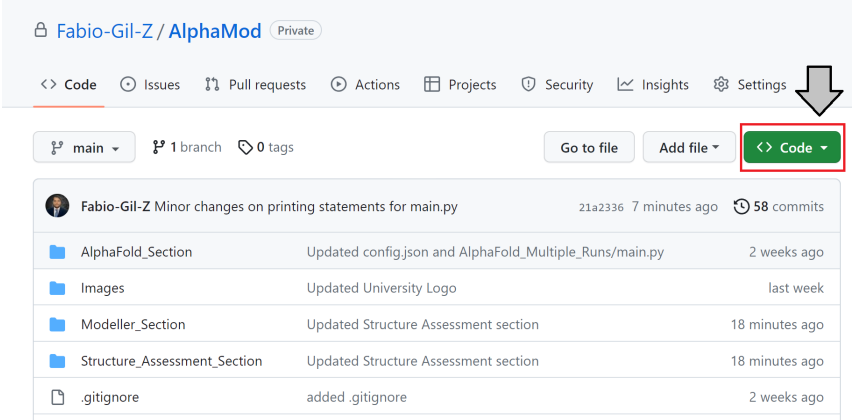


Figure 41: Screenshot of AlphaMod GitHub repository.

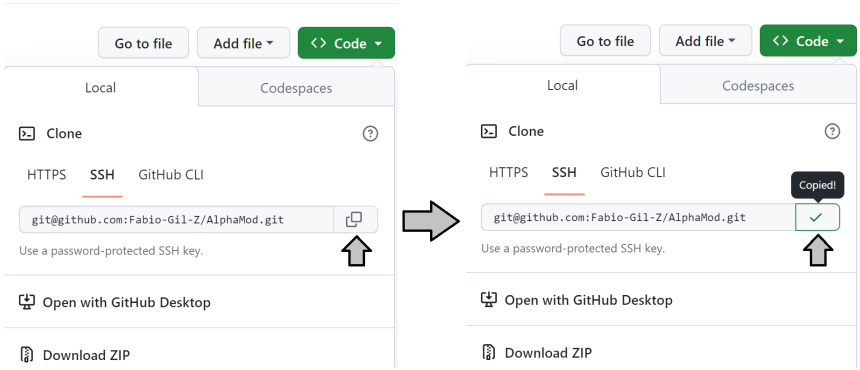


Figure 42: Screenshot of AlphaMod GitHub repository, copying link.

On your computer at the desired destination open a terminal and write:
git clone git@github.com:Fabio-Gil-Z/AlphaMod.git
As shown in figure 43.

Figure 43: Screenshot of AlphaMod GitHub repository, copying link.

3.3 Managing AlphaMod's configuration (config.json)

The file **config.json** is the file that will control *AlphaMod's* output. An overview of **config.json** can be seen in figure 44.

3.3.1 AlphaFold_Prediction_List

This part of the file contains the names of the sequences we want to calculate. Please notice the format, the first sequence is **T1024-D0**, also it is inside quotation marks "T1024-D0" followed by a colon : and then also in quotation marks the number of residues (the **NT** explained at section 2.4). That would be for a single sequence, if we need to calculate multiple sequences then we add a comma (,) and then we proceed to repeat the process with the information of another sequence in figure 44 the second sequence we want to calculate is T1024-D1 (remember -D0 stands for the whole protein, while -D1 stands for a domain of the protein as it is presented on CASP14 website). Finally, the last sequence we will calculate is T1030-D2, notice it is in the last row and **the last row does not have a comma**.

```
{
    "AlphaFold_Prediction_List": {
        "T1024-D0": "391",
        "T1024-D1": "193",
        "T1024-D2": "204",
        "T1025-D1": "257",
        "T1026-D1": "146",
        "T1027-D1": "99",
        "T1028-D1": "292",
        "T1029-D1": "125",
        "T1030-D0": "273",
        "T1030-D1": "154",
        "T1030-D2": "119"
    },
    "StructureAssessment": {
        "pLDDT": "TRUE",
        "GDT_TS": "TRUE",
        "DOPESCORE": "TRUE",
        "QMEAN": "TRUE",
        "PROSA": "TRUE",
        "MOLPROBITY": "TRUE",
        "PROCHECK": "TRUE",
        "AlphaFold_Models": "TRUE",
        "2best_supervised": "TRUE",
        "2best_unsupervised": "TRUE",
        "5ranked_unsupervised": "TRUE",
        "first_run_flag": "TRUE"
    },
    "Install_Requirements": "TRUE",
    "AlphaFold_Databases" : "/media/pc7/SeagateBasic/AlphaFold/alphafold_databases"
}
```


Figure 44: Screenshot of AlphaMod **config.json** file.

3.3.2 StructureAssessment

This part of the file controls two things:

- The metrics you want to calculate such as: pLDDT, GDT_TS, QMEAN.
- How you want to generate the models, supervised or unsupervised

It is important to note, that certain models require specific metrics to be calculated. For instance, to be able to calculate 2best_supervised we need the supervised metric GDT_TS, otherwise would be impossible to calculate it. In a similar way, 2best_unsupervised requires pLDDT and QMEAN to select the best models, whereas 5ranked_unsupervised does not need anything as it uses all ranked models given as output by AlphaFold. A table of the required metrics in order to calculate each of the models is shown in figure 45.

	pLDDT	GDT_TS	DOPESCORE	QMEAN	PROSA	MOLPROBITY	PROCHECK
AlpaFold_Models	⊗	⊗	⊗	⊗	⊗	⊗	⊗
2best_supervised	⊗	✓	⊗	⊗	⊗	⊗	⊗
2best_unsupervised	✓	⊗	⊗	✓	⊗	⊗	⊗
5ranked_unsupervised	⊗	⊗	⊗	⊗	⊗	⊗	⊗

Figure 45: Metrics required to calculate, 2best_supervised and 2best_unsupervised, AlphaFold's models and 5ranked_unsupervised have no requirements.

Therefore, if we want to calculate only AlphaFold's models we set:

- "AlphaFold_Models": "TRUE"
- "2best_supervised": "FALSE"
- "2best_unsupervised": "FALSE"
- "5ranked_unsupervised": "FALSE"

If also, we want to calculate AlphaMod's 2best_supervised, we set:

- "AlphaFold_Models": "TRUE"
- "2best_supervised": "TRUE"
- "2best_unsupervised": "FALSE"
- "5ranked_unsupervised": "FALSE"

Nevertheless, according to figure 45 to be able to calculate 2best_supervised we need to also activate GDT_TS. So we set, GDT_TS to "TRUE" as shown in figure 46.

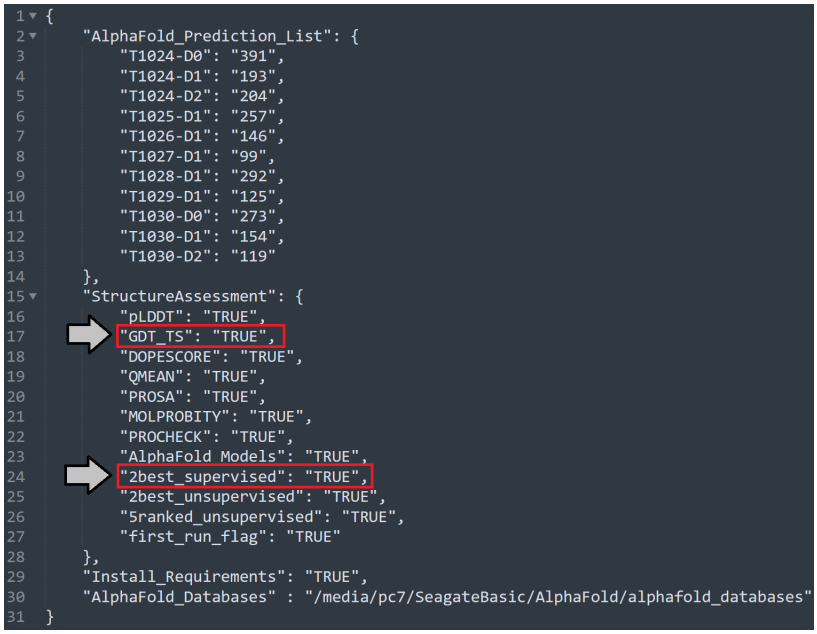


Figure 46: Screenshot of config.json configuration. AlphaFold's and AlphaMod (2best_supervised) will be calculated. The metric GDT_TS is set to "TRUE" as it is required to calculate supervised models.

Finnaly, according to figure 45 to calculate 2best_unsupervised we need pLDDT and QMEAN, the configuration is shown in figure 47.

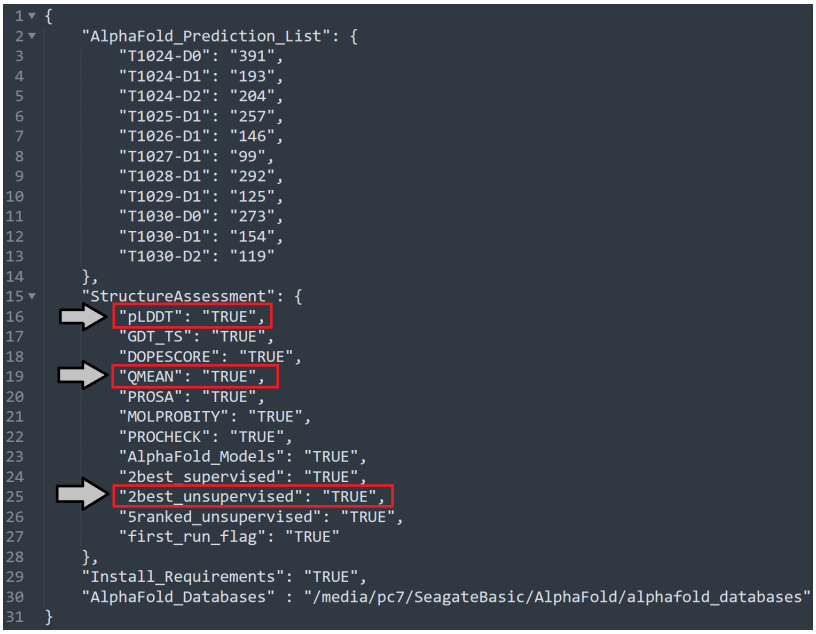


Figure 47: Screenshot of config.json configuration. AlphaFold's, AlphaMod (2best_supervised) and AlphaMod (2best_unsupervised) will be calculated. The metrics pLDDT and QMEAN are set to "TRUE" as it is required to calculate unsupervised models.

The other unsupervised metrics:

- DOPESCORE
- PROSA
- MOLPROBITY
- PROCHECK

These metrics are optional and not required to calculate the models. They are used to provide additional information about the predicted models.

The row *first_run_flag* is used as flow control inside the program and it is recommended to leave it as "TRUE" if it is the first time you are running the program. In general do not change it.

The row *"Install_Requirements"* reads *requirements.txt* text file and installs all the packages needed via pip if "TRUE".

Finally, the row *"AlphaFold_Databases"* is asking for the AF2 databases path, these are the databases downloaded acoording to AlphaFold's official website. In our case we stored the databases in a folder called "alphafoldatabases".

3.4 Launching AlphaMod

Once **config.json** has been configured according to sections 3.3.1 and 3.3.2, we are ready to launch AlphaMod.

First, open a terminal at the top folder (AlphaMod). Followed by the command **python3 main.py** as shown in figure 48. In our example, our base folder is located at */Desktop/AlphaMod*.

Figure 48: Screenshot of our terminal initializing AlphaMod.

The configuration we will use in this example is shown in figure 49. Additionally, if it is the first time you are launching AlphaMod you will be prompted with the installation of the required packages.

```
(venv) pc7@pc7-Precision-Tower-5810:/media/pc7/Data/AlphaMod$ python3 main.py
################################################
###              CONFIGURATION               ###
################################################
{
    "AlphaFold_Prediction_List": {
        "T1024-D0": "391",
        "T1024-D1": "193",
        "T1024-D2": "204",
        "T1025-D1": "257",
        "T1026-D1": "146",
        "T1027-D1": "99",
        "T1028-D1": "292",
        "T1029-D1": "125",
        "T1030-D0": "273",
        "T1030-D1": "154",
        "T1030-D2": "119"
    },
    "StructureAssessment": {
        "pLDDT": "TRUE",
        "GDT_TS": "TRUE",
        "DOPESCORE": "TRUE",
        "QMEAN": "TRUE",
        "PROSA": "TRUE",
        "MOLPROBITY": "TRUE",
        "PROCHECK": "TRUE",
        "AlphaFold_Models": "TRUE",
        "2best_supervised": "TRUE",
        "2best_unsupervised": "TRUE",
        "5ranked_unsupervised": "TRUE",
        "first_run_flag": "TRUE"
    },
    "Install_Requirements": "TRUE",
    "AlphaFold_Databases": "/media/pc7/SeagateBasic/AlphaFold/alphafold_databases"
}
First time launching AlphaMod, requirements will be installed...
Please press Enter...
```

Figure 49: Screenshot of our terminal executing AlphaMod for the first time.

Once all the packages have been installed you will be given a message that all the requirements have been installed as shown in figure 50.

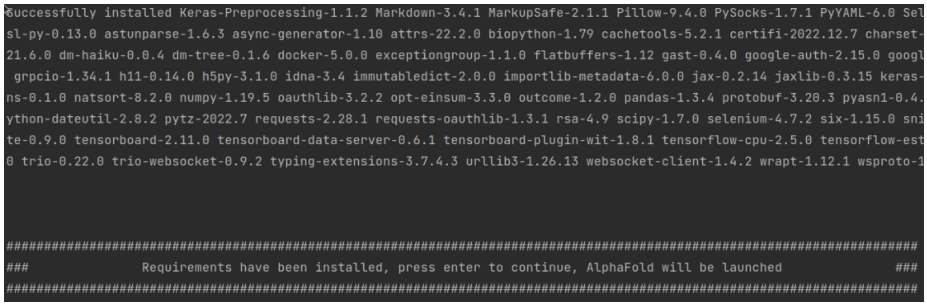


Figure 50: Screenshot of our terminal at the end of the installation of packages.

Moreover, if you are going to use *2best_supervised* it is necessary to set GDT_TS as "TRUE" and thus we will need to calculate the GDT_TS score. The scripts necessary to calculate the GDT_TS score needs administrator rights for execution. You will be informed which files are requesting this permission and it's location as shown in figure 51. Once you have entered your password AlphaMod will proceed to launch AlphaFold.

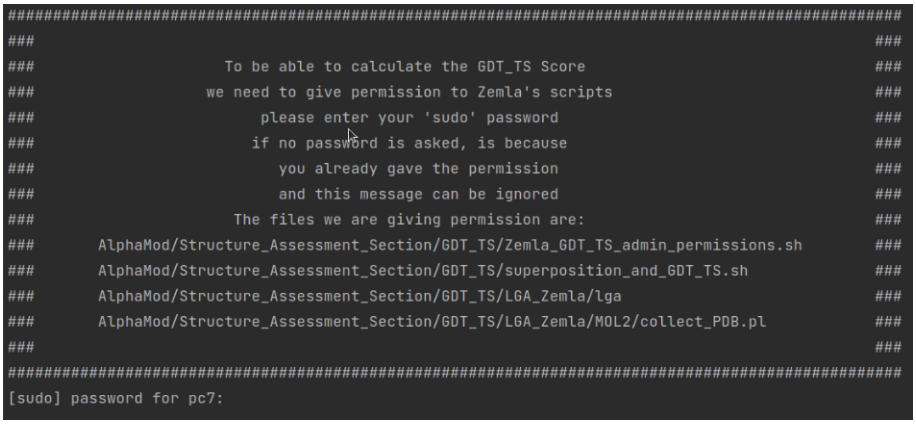


Figure 51: Screenshot of our terminal entering administrator password for GDT_TS scripts.

After giving administrator rights to Zemlas's scripts AlphaFold will be launched as shown in figure 52.

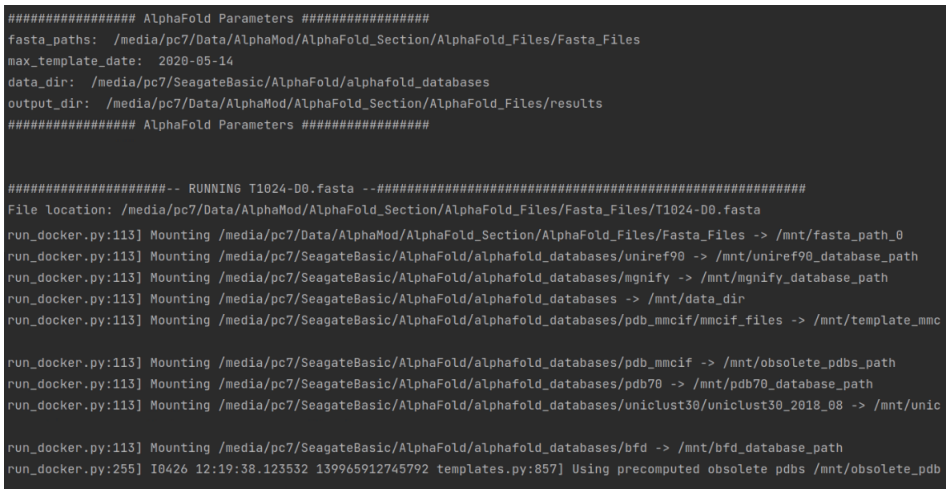


Figure 52: Screenshot of our terminal running AlphaFold.

After AlphaFold2 has finished doing the prediction of the protein, the next step is to assess the quality of the predicted structure, be it on a supervised or unsupervised manner. In our example we selected *2best_supervised*. Therefore, according to figure 45 we need the GDT_TS score to calculate supervised models, the calculation of the GDT_TS score is shown in figure 53.

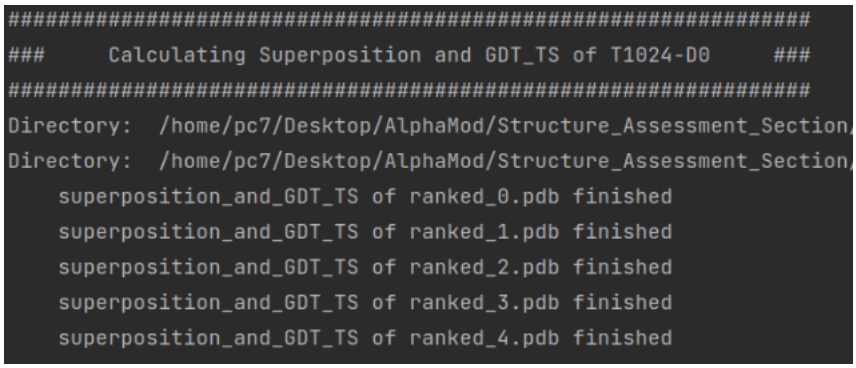


Figure 53: Screenshot of our terminal calculating the GDT_TS score for Target T1024-D0.

In our example we have also selected *2best_unsupervised*, for the unsupervised route we need to extract the *pLDDT* values and calculate *QMEAN* score. The pLDDT extraction is done in the background while the QMEAN score calculation is shown in figure 54.

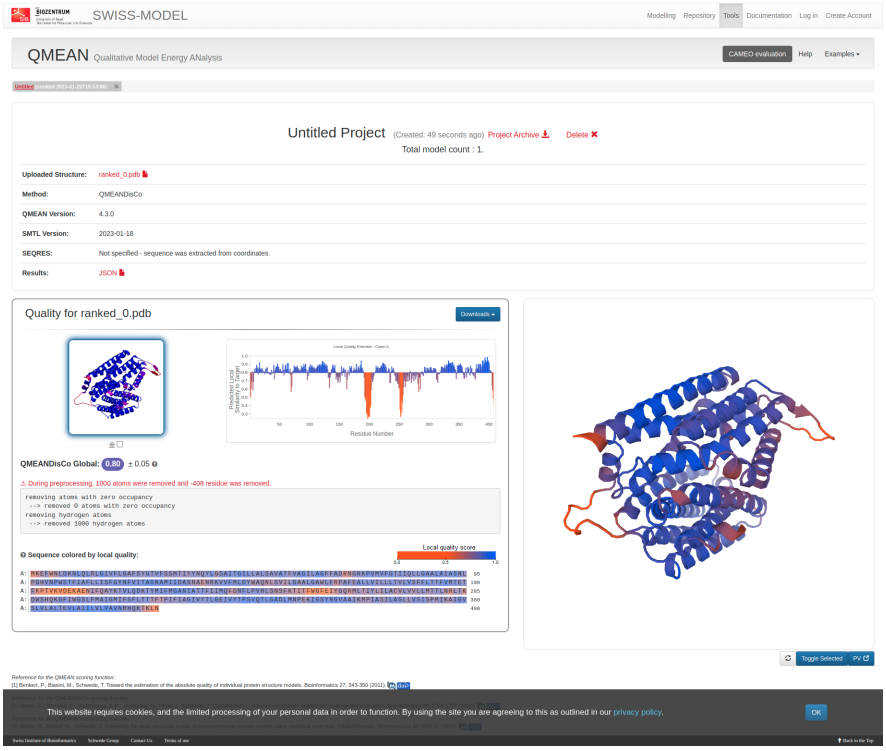


Figure 54: Screenshot of our terminal running python Ad-Hoc web-crawler. Shows QMEAN calculation for Target T1024, AF2 structure ranked_0.pdb .