Installation of AlphaFold 2 and AlphaFold-Multimer to faciliate studies of protein

Final report submitted in partial fulfilment of the requirements of course BIO301

Cyrus Pellet, Nguyen Doan Dai

January 3, 2022

1. Introduction

Originally a task given to us in the course BIO301, we were asked to install AlphaFold 2 and AlphaFold-Multimer (hereinafter together referred to as AlphaFold), and all the necessary additional tools to study the proteins related to mutations in DNA of archaea. Unfortunately, during the course of the project, there have been many complications due to various causes, which would be the subject of discussion further below.

In short, imperfect conditions have raised the technical complexity and forced us to come up with some ad-hoc engineering solutions, but also forbidden us from using the tools installed to initiate any meaningful scientific studies. Nevertheless, for this exact reason, we also feel the need to write this technical report, serving both as a documentation of the project and an user's/developer's guide, with our hope that it shall help future maintenance.

2. Preliminary

First, we shall give some background, which will then explain our choice of tools.

2.1. **Protein structure prediction.** Whilst it is true that our task was to specifically install AlphaFold, all such decisions must be justified, and for this, we were asking if AlphaFold would be the most appropriate choice, for apparently, it is not the only Machine Learning model addressing the problem of protein folding. The question we wished to answer before the start of the project was two-fold:

1. AlphaFold-Multimer has the advantage that it also predicts structures of multimeric protein complexes, and furthermore, at the moment of writing,

it is the only deep learning method designed for the task [1]. But is it necessary for our task?

2. For the problem of *monomer* protein folding, is AlphaFold 2 the most suitable solution?

For the first question, although we must admit that we have only little knowledge of the proteins which shall be studied after, it has come to us that the installation and/or creation of these tools shall take into consideration future studies on different subjects, far beyond the scope of archaea. And in this sense, undeniable is the existence and the importance of dimer and multimer proteins in DNA and RNA handling [2], selection [3], regulation [4]. With recent developments in epigenetics and studies of histone, one may expect more applications in the future, and thus the need to study multimeric protein-DNA complexes. Thus, we deem that the solving of multimeric protein complexes be of indisputable necessity.

For the second question, since DeepMind's breakthrough of AlphaFold and publication of AlphaFold 2 [5], there have been many deep learning methods to the problem of monomeric protein structure prediction which may or may not based on AlphaFold itself, e.g. D-I-TASSER [6], RaptorX [7], ColabFold [8], RoseTTAFold [9], trRosetta [10], DMPfold [11], EVfold [12], SAINT2 [13], and most recently, EMSFold [14] and OmegaFold [15] (an incomplete review is given at [16]). In short, given the wide scope, large depth, and complexity of the protein folding problem, it is unfathomable to have a perfect, one-size-fit-all solution, and the choice of the model depends on the set of proteins of interest and the available resources, for there is a trade-off between accuracy and speed. But, given that in this project, speed is of little concern and accuracy is given much emphasis, as well as our wish to facilitate future studies to the best of our capability, the choice of AlphaFold is thus natural, as the most general, accurate, and well-documented method. The necessity of solving multimeric protein folding problem as explained above, and available methods for the protein-DNA binding site prediction problem, which shall be discussed below, also motivate this decision.

2.2. **Protein-DNA binding site prediction.** Another goal of the project is to study protein-DNA complexes, and in particular the possible binding sites of proteins on DNA. One may argue that as the geometry of the protein will play a role in how it will interact with DNA, it is essential to incorporate the structure of the protein into the prediction, and indeed, the breakthrough of AlphaFold in predicting structures of proteins has allowed similar progresses in protein-DNA binding site problem. Given that we install AlphaFold, we look for a way to incorporate that into the solving of protein-DNA binding site, and recently reported is such a method, called GraphSite [17] (a short literature review and comparisons with other state-of-the-art methods are also available in [17]). Of course, there are other methods which do not depend upon AlphaFold or any other methods for the problem of protein structure prediction (for a review, see

[18, 19]), but whilst there exist no standard datasets to serve as a benchmark, comparisons with other methods suggest that GraphSite is amongst the most accurate solution at the moment of writing. Thus, we decide to install GraphSite after AlphaFold.

3. Installation of AlphaFold

3.1. **Notes on installation.** In general, we follow the guide published at `https://github.com/deepmind/alphafold/`, but during the installation, there were complications that forced us to devise ad-hoc methods to overcome, either known existing issues with AlphaFold and/or its database, or limitation of the server on which we installed AlphaFold, that we believe it will be beneficial to document as it will help future maintenance.

3.1.1. *Issue with /pdb_seqres/pdb_seqres.txt.* During our first run of AlphaFold, we encountered the following error

```
I1026 15:27:03.974704 140459173607232 run_docker.py:255]
Parse failed (sequence file
    /mnt/pdb_seqres_database_path/pdb_seqres.txt):
I1026 15:27:03.974859 140459173607232 run_docker.py:255]
Line 1364612: illegal character 0
```

Further reading shows that this is a known problem (`https://github.com/deepmind/alphafold/issues/569`), and it is intrinsic of the file `pdb_seqres.txt`, thus it will occur every time one updates the database, which includes running the script `scripts/download_pdb_seqres.sh`. As of the moment of writing, the issue has been addressed in the release of version 2.3.0 of AlphaFold. Nonetheless, it was not the case at the start of the project, and we decide that it is necessary that we mention this issue in this report, in case it occurs again in the future.

3.1.2. *Issue with insufficient storage.* The database of AlphaFold calls for a continuous storage space of approximately 2.62 GB. At the time of writing, we were allocated two storage directories, at `/home/alphafold` and at `/media/disk1/alphafold`, respectively. Given the large storage space available at `/media/disk1/alpafold`, it was decided that this would be where the database would be stored, but over the course of project, it occurred to us that there was not sufficient space: although we used 98% of the storage, meaning there were only 78 GB available, we would still need to store the `uniref30` dataset of size 206 GB. And, it is not possible to run AlphaFold without this dataset, as both attempts to do so with monomeric and multimeric version were unsuccessful.

One possible solution is to set up a virtual file system and combine multiple physical storage spaces as a single virtual one, which proved to be too complicated for a problem that can be resolved in the future just by adding a new hard disk. Nevertheless, for the time being, we still need a way to load `uniref30` dataset. Thus, we opted for an ad-hoc solution, which goes against principles in software development and will fail if one decides to update the code base of AlphaFold. Closer inspection of the code base of AlphaFold shows that the database is loaded during the execution of the script `docker/run_docker.py`, and specifically, the links to the file in the database are described from line 126 to line 162 (cf. the file `docker/run_docker.py` up to the commit with hash `54f127a51778ba12407d96d7d23cc66d24e2cf58`). Thus, by modifying these lines, one can point to any directories, and there will be no need for a single continuous storage space for the database. In particular, we modified the line 148-149, from

```
    uniref30_database_path = os.path.join(
      FLAGS.data_dir, 'uniref30', 'UniRef30_2021_03')
```

to

```
    uniref30_database_path = os.path.join(
      '/home/alphafold/dbs_extended, 'uniref30', 'UniRef30_2021_03')
```

Finally, the final directory structure is given below.

```
home/alphafold
└── dbs_extended
    └── uniref39
        └── # 7 files
media/disk1/alphafold/dbs
├── bfd
│   └── # 6 files
├── mgnify
│   └── mgy_clusters_2022_05.fa
├── params
│   └── # 16 files
├── pdb70
│   └── 9 files
├── pdb_mmcif
│   ├── mmcif_files
│   │   └── # About 199,000 .cif files
│   └── obsolete.dat
├── pdb_seqres
│   └── pdb_seqres.txt
├── small_bfd
│   └── bfd-first_non_consensus_sequences.fasta
├── uniref90
│   └── uniref90.fasta
└── uniprot
    └── uniprot.fasta
```

Future maintainers of the installed AlphaFold are thus advised to take this change into consideration.

3.1.3. *Issue with installation of* `docker`. After an incident occurred in which the `NVIDIA` driver on `iss` machine stopped working, it has appeared to us that `docker` was reinstalled with `snap` instead of `apt`. Whilst there seemed to be no differences between the two version, it is generally considered that it is better to install a program on Linux with `apt` than with `snap`. And in this case, it made a difference, as we encountered the following error

```
Traceback (most recent call last):
File "docker/run_docker.py", line 264, in
    app.run(main)
File "/usr/local/lib/python3.8/dist-packages/absl/app.py", line 312, in
     run
    _run_main(main, args)
File "/usr/local/lib/python3.8/dist-packages/absl/app.py", line 258, in
     _run_main
    sys.exit(main(argv))
File "docker/run_docker.py", line 234, in main
    container = client.containers.run(
File
     "/usr/local/lib/python3.8/dist-packages/docker/models/containers.py",
     line 818, in run
    container.start()
File
     "/usr/local/lib/python3.8/dist-packages/docker/models/containers.py",
     line 404, in start
    return self.client.api.start(self.id, **kwargs)
File
     "/usr/local/lib/python3.8/dist-packages/docker/utils/decorators.py",
     line 19, in wrapped
    return f(self, resource_id, *args, **kwargs)
File "/usr/local/lib/python3.8/dist-packages/docker/api/container.py",
     line 1111, in start
    self._raise_for_status(res)
File "/usr/local/lib/python3.8/dist-packages/docker/api/client.py", line
     270, in _raise_for_status
    raise create_api_error_from_http_exception(e)
File "/usr/local/lib/python3.8/dist-packages/docker/errors.py", line 31,
     in create_api_error_from_http_exception
    raise cls(e, response=response, explanation=explanation)
docker.errors.APIError: 500 Server Error for
     http+docker://localhost/v1.41/containers/879bc867a01bf2f011e997b81
2e650fd8c893d1dc493bf69a544e9c38066734f/start: Internal Server Error
     ("could not select device driver "nvidia" with capabilities:
     [[gpu]]")‘
```

This is a known issue (`https://github.com/deepmind/alphafold/issues/479`) but without any known methods to resolve, and the error message only suggests that there were some issues with NVIDIA docker installation. After some search, we uninstall docker with snap, and then reinstalled with apt, which resolved the issue.

3.2. **Results.** The steps to run AlphaFold from command line are relatively simple: supposed we are at /home/alphafold.

1. Save the .fasta file containing the sequence(s) at /home/alphafold/alphafold/data.
2. In command line,

```
    alphafold@iss:~$ source alphafold/bin/activate
```

3. Run

```
    (alphafold) alphafold@iss:~$ python3
        alphafold/docker/run_docker.py
        --fasta_paths=alphafold/data/xxx.fasta
        --model_preset=monomer
        --data_dir=/media/disk1/alphafold/dbs/
```

   where xxx is the ID of protein. Or, if one wishes to let AlphaFold run in the background, run instead

```
    (alphafold) alphafold@iss:~$ nohup python3
        alphafold/docker/run_docker.py
        --fasta_paths=alphafold/data/xxx.fasta
        --model_preset=monomer
        --data_dir=/media/disk1/alphafold/dbs/ &
```

   and the output will be saved in file /home/alphafold/nohup.out, which can be inspected with

```
    alphafold@iss:~$ cat nohup.out
```

   To run AlphaFold-Multimer, one can change --model_preset=monomer to --model_preset=multimer. Further information can be found at https://github.com/deepmind/alphafold.
4. After the program finishes, the result can be found at /tmp/alphafold/xxx.

After the installation, we have tested with various protein and protein complexes, amongst which are ones with RCSB Protein Data Bank ID given by 5ZNG, 6A6I, 6GS2, 6H4B, 6IF2, as well as protein T0543 from CASP9, T0146s1 (sub-unit 1 of protein 6PX4 in RCSB database) and T1050 from CASP14. Figures of comparisons are given in Appendix.

4. Creation of a user interface

Due to the inherently convoluted nature of the whole process, we additionally set out to create an interface to interact with AlphaFold. This is the first project of this nature to ever be conceived, since until very recently, recent modifications of the program did not make it suitable for production usage by non-computer scientists. Apart from providing a convenient way to submit FASTA files for folding, we set out to include the following improvements over the raw AlphaFold pipeline:

1. A **queuing system** to submit jobs in advance to be sequentially processed one after the other.
2. An embedded **protein structure visualiser** to quickly analyse the folding results without specialised software.
3. Tools to **visualise task folding progress** in real time while the program is running.
4. A persistent **backlog** of completed tasks with their running time, sequence data and results.
5. Robust **result export** features to easily retrieve all PDB files from any task.

With these goals in mind, a first step to such an endeavour was to gain a deeper understanding of AlphaFold's inner workings. This would allow us to understand the cryptic output of the program, and it is only thanks to this understanding of the steps involved in the folding process that accurate feedback can be provided to the user in real time.

4.1. **The folding process.** A frequent criticism of deep learning models such as AlphaFold is that they essentially constitute a black box approach. Indeed, one inputs a protein sequence and, some time later, the three-dimensional protein structure is "magically" produced. Here for the sake of our understanding of the process, we aim to shed some light regarding what goes behind the scenes after a folding task is submitted. We will not aim for exhaustiveness or technicalities, as for those one can delve into the following document by DeepMind.

Broadly explained, AlphaFold figures out the complex interrelationships of the protein's residues that dictate what structure that protein sequence adopts. Then, it iterates to improve the local structural details until a sufficient certainty is achieved. To achieve this, the following steps are performed:

1. MSA: Several databases of known proteins (uniprot, uniref, etc...) are queried with the aim of identifying sequences that are similar, but not identical to the input sequence. This process, referred to as JackHMMER, determines the parts of the sequence that have the strongest correlation

with existing sequences (Multiple Sequence Alignment). Next, AlphaFold also tries to identify proteins that may have a similar structure to the input - so called “templates” -, and constructs an initial representation of the structure, which it calls the “pair representation” (HHSearch). This is, in essence, a model of which amino acids are likely to be in contact with each other.

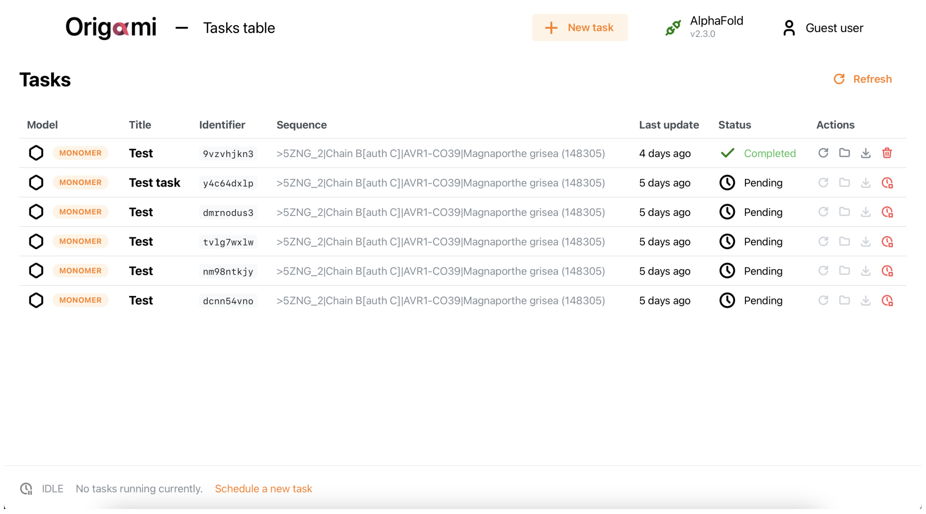
2. Transformation: In this second part, multiple sequence alignments and templates are fed through a so-called ”transformer” - a kind of oracle that quickly identifies which pieces of information are more telling and hence should be given more weight. This refinement process takes place iteratively, around 48 times in total.
3. Structure model: this sophisticated piece of the pipeline takes the refined ”msa” and “pair representation”, leveraging them to construct a three-dimensional model of the protein structure. Unlike the previous state-of-the-art models, this network does not use any optimisation algorithm: it generates a static, final structure, all in a single step. The end result is a long list of Cartesian coordinates representing the position of each atom of the protein, including side chains.

Following the completion of those steps, the output of the pipeline is fed back into the second stage to refine results further.

This allowed us to understand a range of outputs from AlphaFold, from the `timings.json` file that summarises the time taken for each pipeline step, to the different `ranked_n.pdb` files that produce the progressively more refined structure predictions. Here is the typical collection of outputted files for a given folding task:

```
features.pkl                 relax_metrics.json
msas                         result_model_1_pred_0.pkl
ranked_0.pdb                 result_model_2_pred_0.pkl
ranked_1.pdb                 result_model_3_pred_0.pkl
ranked_2.pdb                 result_model_4_pred_0.pkl
ranked_3.pdb                 result_model_5_pred_0.pkl
ranked_4.pdb                 timings.json
ranking_debug.json           unrelaxed_model_1_pred_0.pdb
relaxed_model_1_pred_0.pdb   unrelaxed_model_2_pred_0.pdb
relaxed_model_2_pred_0.pdb   unrelaxed_model_3_pred_0.pdb
relaxed_model_3_pred_0.pdb   unrelaxed_model_4_pred_0.pdb
relaxed_model_4_pred_0.pdb   unrelaxed_model_5_pred_0.pdb
relaxed_model_5_pred_0.pdb
```

4.2. **Origami - User Interface.** We hypothesise that the current lack of convenient interfaces to interact with AlphaFold can be attributed to its very recent and rapid development. Nonetheless, we set out to facilitate the points listed above with our fresh understanding of the pipeline. The result can be seen below: a modern web application that interacts with our own Application Programming Interface (API) to display relevant information to the user and accept their input.

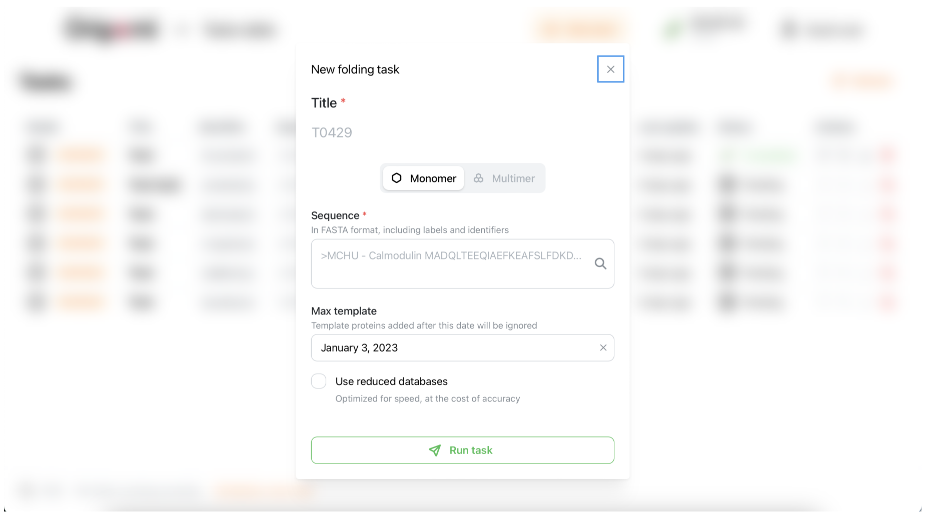


Tasks

Model	Title	Identifier	Sequence	Last update	Status	Actions
MONOMER	Test	9vzvhjkn3	>5ZNG_2|Chain B[auth C]|AVR1-CO39|Magnaporthe grisea (148305)	4 days ago	✓ Completed	
MONOMER	Test task	y4c64dxlp	>5ZNG_2|Chain B[auth C]|AVR1-CO39|Magnaporthe grisea (148305)	5 days ago	Pending	
MONOMER	Test	dmrnodus3	>5ZNG_2|Chain B[auth C]|AVR1-CO39|Magnaporthe grisea (148305)	5 days ago	Pending	
MONOMER	Test	tvlg7wxlw	>5ZNG_2|Chain B[auth C]|AVR1-CO39|Magnaporthe grisea (148305)	5 days ago	Pending	
MONOMER	Test	nm98ntkjy	>5ZNG_2|Chain B[auth C]|AVR1-CO39|Magnaporthe grisea (148305)	5 days ago	Pending	
MONOMER	Test	dcnn54vno	>5ZNG_2|Chain B[auth C]|AVR1-CO39|Magnaporthe grisea (148305)	5 days ago	Pending	

IDLE No tasks running currently. Schedule a new task

Figure 1: Folding tasks summary page in Origami.



New folding task

Title *

T0429

Monomer | Multimer

Sequence *
In FASTA format, including labels and identifiers

>MCHU - Calmodulin MADQLTEEQIAEFKEAFSLFDKD...

Max template
Template proteins added after this date will be ignored

January 3, 2023

Use reduced databases
Optimized for speed, at the cost of accuracy

Run task

Figure 2: Task creation workflow in Origami.

5. Installation of GraphSite

5.1. **Notes on installation.** Same as AlphaFold, the guide to be followed is mostly straightforward and available at `https://github.com/biomed-AI/GraphSite`. We installed C++ library `libcif++` which is required for HH-Suite and DSSP - the built files of which are contained in `/home/alphafold/hh-suite/build` and `/home/alphafold/dssp/build/`.

To run GraphSite, it is also preferable to provide ”single representation” of the protein, i.e. the first row of the MSA representation. Although GraphSite can run without single representation via mode `evo`, the authors also warned that ”Set `--msa evo` to use only evolutionary features (PSSM + HMM) as MSA information (might causes large performance drop)”, and we wish to install the most accurate version of GraphSite. Unfortunately, neither AlphaFold 2 nor AlphaFold-Multimer returns single representation (or any representations derived from MSA, for that matter), and there is no conceivable way to construct it from the output. This is a known issue (`https://github.com/deepmind/alphafold/issues/471`) which has no solution at the moment of writing, but to rerun the whole prediction based on ColabFold (`https://github.com/yuxin212/intfold`), which introduces new issues, such as

- redundancy in computational work,
- poor maintainability, as upgrading to new models would require updating two separated programs,
- unnecessary complexity.

AlphaFold must use such representation in its pipeline, for there exist mentions of such a representation in the original paper [5]. And indeed, further inspection into the code base shows that there is the option to return such representation (`/alphafold/model/modules.py` at line 145 for AlphaFold 2 and `/alphafold/model/modules_multimer.py` at line 424), which are by default turned off to save memory. Thus, we edited the code base and turned such feature on. Future maintainers of the code base are thus asked to take this into consideration.

5.2. **Result.** With the modifications to the code base made, the single representation is now given in Pickle files, and we tested with some proteins, as well as the test protein given by authors of [17]. The limited length of the report does not allow a more detailed description, thus here we give the result for protein with RSCB Protein Data Bank ID 2L09, of 62 nucleotides (which was chosen for its small size).

Nevertheless, it appears that running GraphSite in its full mode takes

- considerably more time due to the querying of UniRef90 BLAST+ database and making of `.pssm` file, and
- considerably more RAM due to calling of HHblits, which is a known issue,

with only marginal benefit as described in Table 2 of [17]. Users are thus advised to take into consideration trade-off between resource and accuracy. Alternatives to HHblits exist, but without any comparisons to the published methods, we cannot provide any guarantee about accuracy. It is certainly resolvable by providing more RAM, but the limited conditions did not allow such a solution.

References

[1] A. Fossati, C. Li, F. Uliana, F. Wendt, F. Frommelt, P. Sykacek, M. Heusel, M. Hallal, I. Bludau, T. Capraz, P. Xue, J. Song, B. Wollscheid, A. W. Purcell, M. Gstaiger, and R. Aebersold, "PCprophet: a framework for protein complex prediction and differential analysis using proteomic data," *Nature Methods*, vol. 18, pp. 520–527, may 2021.

[2] J. Wilce, J. Vivian, and M. Wilce, "Oligonucleotide Binding Proteins," in *Protein Dimerization and Oligomerization in Biology* (J. M. Matthews, ed.), ch. 6, pp. 91–104, Springer International Publishing, 2012 ed., 2012.

[3] D. A. Kretov, P. A. Curmi, L. Hamon, S. Abrakhi, B. Desforges, L. P. Ovchinnikov, and D. Pastré, "mRNA and DNA selection via protein multimerization: YB-1 as a case study," *Nucleic Acids Research*, vol. 43, pp. 9457–9473, oct 2015.

[4] J.-C. Bourdon, S. Surget, and M. P. Khoury, "Uncovering the role of p53 splice variants in human malignancy: a clinical perspective," *OncoTargets and Therapy*, p. 57, dec 2013.

[5] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, aug 2021.

[6] W. Zheng, Y. Li, C. Zhang, X. Zhou, R. Pearce, E. W. Bell, X. Huang, and Y. Zhang, "Protein structure prediction using deep learning distance and hydrogen-bonding restraints in ¡scp¿CASP14¡/scp¿," *Proteins: Structure, Function, and Bioinformatics*, vol. 89, pp. 1734–1751, dec 2021.

[7] J. Xu, "Distance-based protein folding powered by deep learning," *Proceedings of the National Academy of Sciences*, vol. 116, pp. 16856–16865, aug 2019.

[8] M. Mirdita, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov, and M. Steinegger, "ColabFold: making protein folding accessible to all," *Nature Methods*, vol. 19, pp. 679–682, jun 2022.

[9] M. Baek, F. DiMaio, I. Anishchenko, J. Dauparas, S. Ovchinnikov, G. R. Lee, J. Wang, Q. Cong, L. N. Kinch, R. D. Schaeffer, C. Millán, H. Park, C. Adams, C. R. Glassman, A. DeGiovanni, J. H. Pereira, A. V. Rodrigues, A. A. van Dijk, A. C. Ebrecht, D. J. Opperman, T. Sagmeister, C. Buhlheller, T. Pavkov-Keller, M. K. Rathinaswamy, U. Dalwadi, C. K. Yip, J. E. Burke, K. C. Garcia, N. V. Grishin, P. D. Adams, R. J. Read, and D. Baker, "Accurate prediction of protein structures and interactions using a three-track neural network," *Science*, vol. 373, pp. 871–876, aug 2021.

[10] Z. Du, H. Su, W. Wang, L. Ye, H. Wei, Z. Peng, I. Anishchenko, D. Baker, and J. Yang, "The trRosetta server for fast and accurate protein structure prediction," *Nature Protocols*, vol. 16, pp. 5634–5651, dec 2021.

[11] J. G. Greener, S. M. Kandathil, and D. T. Jones, "Deep learning extends de novo protein modelling coverage of genomes using iteratively predicted structural constraints," *Nature Communications*, vol. 10, p. 3977, sep 2019.

[12] R. Sheridan, R. J. Fieldhouse, S. Hayat, Y. Sun, Y. Antipin, L. Yang, T. Hopf, D. S. Marks, and C. Sander, "EVfold.org: Evolutionary Couplings and Protein 3D Structure Prediction," *biorxiv2*, 2015.

[13] S. H. P. de Oliveira, E. C. Law, J. Shi, and C. M. Deane, "Sequential search leads to faster, more efficient fragment-based de novo protein structure prediction," *Bioinformatics*, vol. 34, pp. 1132–1140, apr 2018.

[14] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, A. dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, and A. Rives, "Language models of protein sequences at the scale of evolution enable accurate structure prediction," *biorxiv*, 2022.

[15] R. Wu, F. Ding, R. Wang, R. Shen, X. Zhang, S. Luo, C. Su, Z. Wu, Q. Xie, B. Berger, J. Ma, and J. Peng, "High-resolution de novo structure prediction from primary sequence," *biorxiv*, 2022.

[16] C. Outeiral, D. A. Nissley, and C. M. Deane, "Current structure predictors are not learning the physics of protein folding," *Bioinformatics*, vol. 38, pp. 1881–1887, mar 2022.

[17] Q. Yuan, S. Chen, J. Rao, S. Zheng, H. Zhao, and Y. Yang, "AlphaFold2-aware protein–DNA binding site prediction using graph transformer," *Briefings in Bioinformatics*, vol. 23, mar 2022.

[18] J. Si, R. Zhao, and R. Wu, "An Overview of the Prediction of Protein DNA-Binding Sites," *International Journal of Molecular Sciences*, vol. 16, pp. 5194–5215, mar 2015.

[19] Y. Zhang, W. Bao, Y. Cao, H. Cong, B. Chen, and Y. Chen, "A survey on protein–DNA-binding sites in computational biology," *Briefings in Functional Genomics*, vol. 21, pp. 357–375, sep 2022.

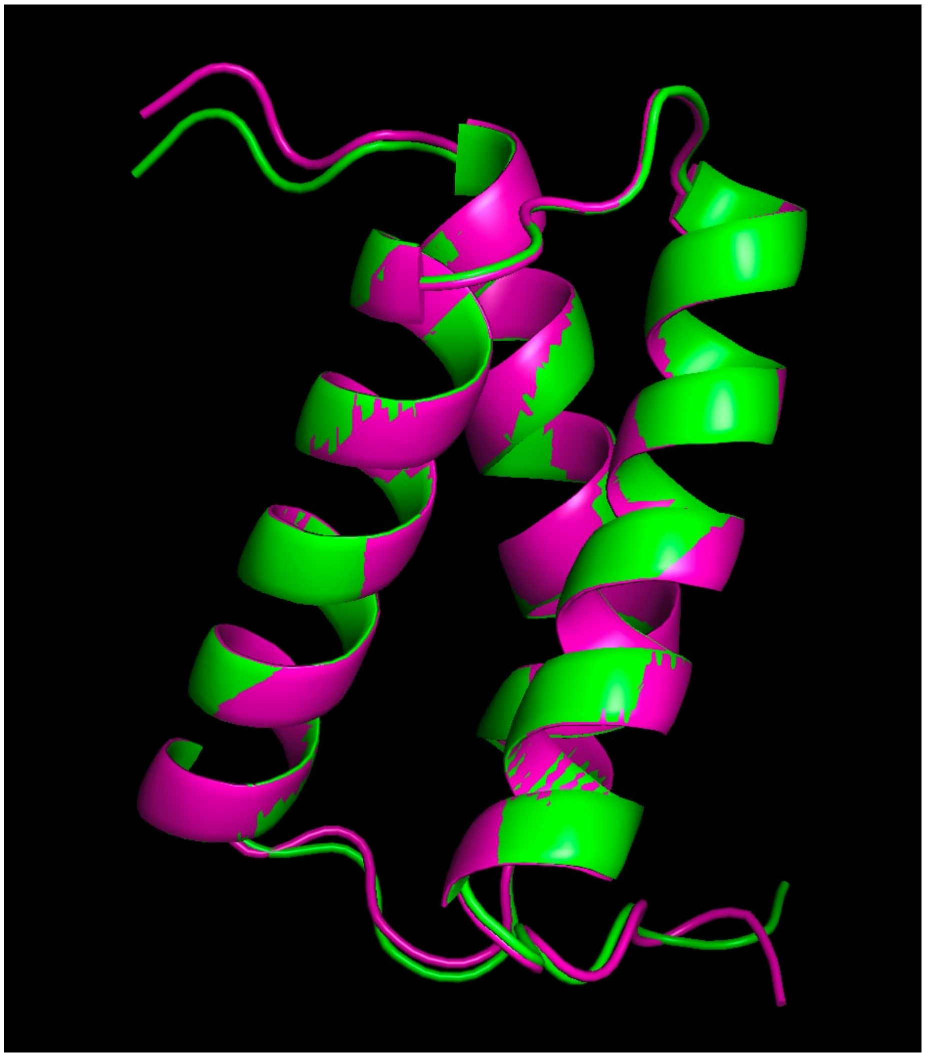


Figure 3: Comparison of structure of protein T0543 as experimentally determined (green) versus prediction made by AlphaFold 2 (pink).

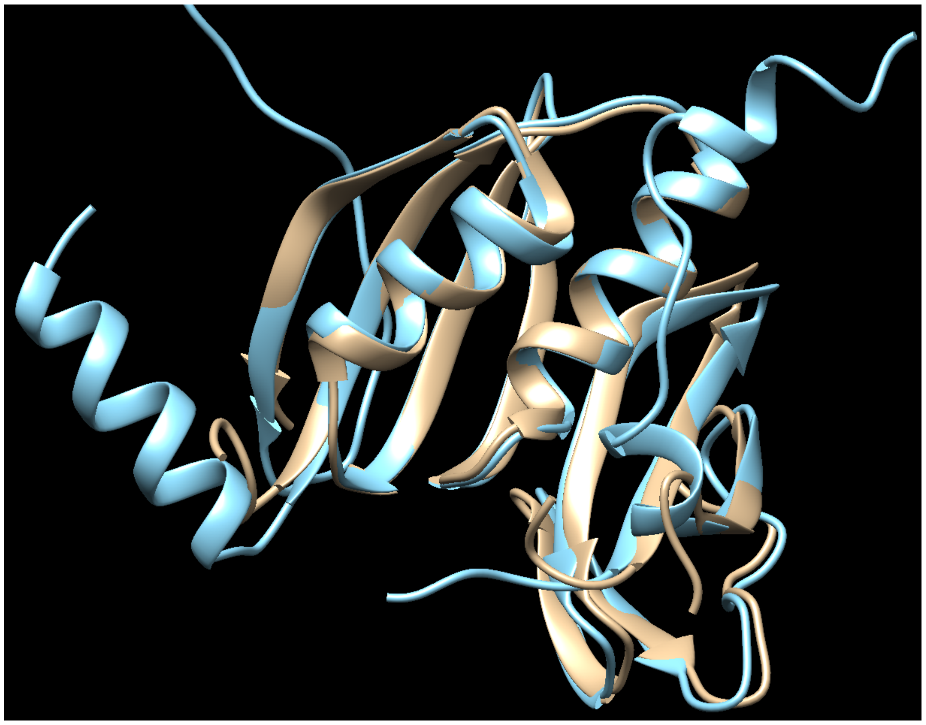


Figure 4: Comparison of structure of protein 5ZNG as experimentally determined (blue) versus prediction made by AlphaFold 2 (gold).

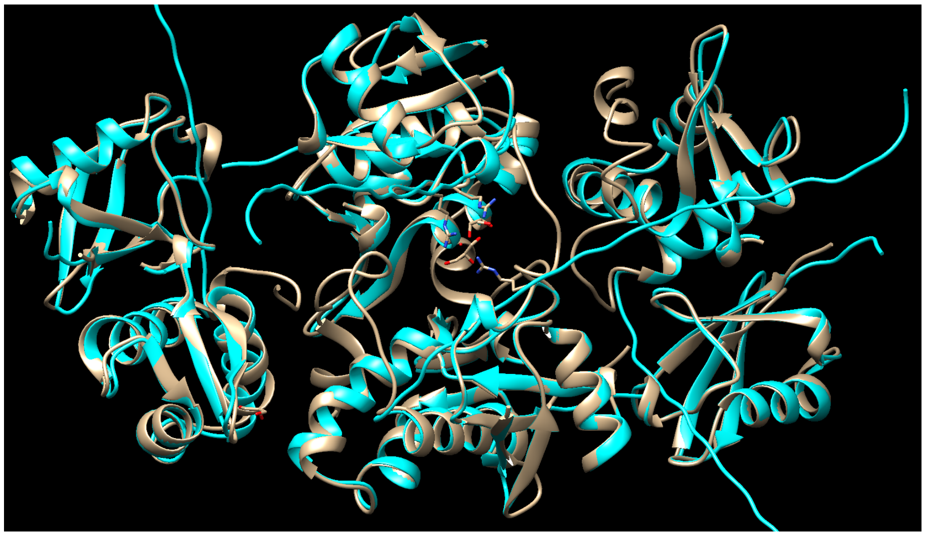


Figure 5: Comparison of structure of protein 6A6I as experimentally determined (cyan) versus prediction made by AlphaFold 2 (gold).

No.	Amino Acid	Probability	No.	Amino Acid	Probability
1	M	0.0843	32	A	0.0080
2	N	0.0657	33	R	0.0150
3	L	0.0514	34	Q	0.0252
4	R	0.0791	35	A	0.0165
5	W	0.0794	36	E	0.0126
6	T	0.0557	37	Q	0.0278
7	S	0.0645	38	D	0.0420
8	E	0.0392	39	I	0.0547
9	A	0.0094	40	V	0.0247
10	K	0.1917	41	T	0.0497
11	T	0.0176	42	P	0.0369
12	K	0.2078	43	E	0.0178
13	L	0.0603	44	L	0.0130
14	K	0.3017	45	V	0.0105
15	N	0.0870	46	E	0.0312
16	I	0.1144	47	Q	0.0419
17	P	0.4437	48	A	0.0117
18	F	0.5047	49	R	0.1777
19	F	0.5637	50	L	0.0108
20	A	0.0562	51	E	0.0153
21	R	0.7695	52	F	0.1376
22	S	0.2242	53	G	0.0219
23	Q	0.3386	54	Q	0.0759
24	A	0.0136	55	L	0.0350
25	K	0.3839	56	E	0.0360
26	A	0.2716	57	H	0.0350
27	R	0.2031	58	H	0.0091
28	I	0.0146	59	H	0.0263
29	E	0.0304	60	H	0.0226
30	Q	0.1644	61	H	0.0146
31	L	0.0117	62	H	0.0196

Table 1: Prediction of protein-DNA binding site for protein with ID 2L09. The threshold of the predictive score to determine is set to be 0.27. Amino acids with scores above the threshold are highlighted in green. Result is obtained with mode `--msa single`.