AI in Drug Discovery and Development

Prof. Rajnish Kumar
Dept. of Pharmaceutical Engineering and Technology
IIT-(BHU), Varanasi

Week-05
Lecture-25

Welcome to the course on "AI in Drug Discovery and Development." So, earlier we discussed molecular docking and how it is being utilized for discovering novel hit molecules using structure-based drug design. So, in this session, we will explore how we can use G-NINA for docking the molecules, generating their poses, scoring them, and also identifying the top-scoring molecules. Yeah, so what you have to do is that I have created a Google Colab for that to make it easy for every one of us to use. So, what you have to do is you have to follow this link tinyurl.com slash aidd dash GNINA.

So once you open that link, you will end up here in this Colab, actually, GNINA tutorial. Since this tutorial is, you know, hosted in my Google Drive, you will not be able to run this tutorial. So, what you have to do instead is create a local copy of this tutorial file or this Colab file. So, you click on the file and then click on "Save a copy in Drive," okay? So, it will say "creating a copy," and it says "notebook copy complete," and then you have to open it in a new tab.

So, once you open it in a new tab, this is how you will see a copy of the GNINA tutorial dot ipynb. So, then you will be able to run the tutorial, and this is what we have to do to run this, actually. Ok, so Google Colab is a platform that lets you do all sorts of coding and even run those programs, install tools, and code. And the beauty is that you do it using Google's computational power, which means you are using the servers that belong to Google. And then this is kind of a limited free service that they are providing.

So, if you want to use high-performance GPUs, then of course you have to have a pro account; otherwise, you can do basic calculations without any problem, okay? So, we will be using the GNINA in the colab, and I will show you how we can do kind of basic docking, actually. You can go to this reference as well; this is the reference where they have reported the GNINA. The GitHub link is also provided, which you can go through. GNINA is a molecular docking software that builds upon AutoDock Vina with added deep learning capabilities. So, what it does is integrate convolutional neural networks to enhance the coding of protein-ligand interactions.

So, as I said, it is not doing the docking. So, docking is performed by Autodock Vina, and then it can additionally rescore those molecules, those poses, and those docking poses generated by Autodock Vina. So, it performs traditional docking to generate ligand binding poses within the receptor's active site, and then it uses a CNN-based scoring function that analyzes 3D molecular interaction patterns, offering improved pose prediction and binding affinity estimation. So, it also supports flexible ligand docking and it can be accelerated using the GPUs improving performance and scalability. So, it provides multiple scores per pose that we will see when we run it, such as the Vina score, which is the classical scoring coming from Autodock Vina.

and the CNN pose score, which is the likelihood of the correct pose, and the CNN affinity score, which is the predicted binding affinity. So, the advantage is which gnina has. So, it has a more accurate pose ranking compared to classical methods. And it is effective for virtual screening and structure-based drug design. And the most important thing is it can handle diverse ligand chemistries and binding site environments, and it can handle different file formats as well.

So, if you have used AutoDock Vina, you probably know that we have to convert those molecules into PDBQT files. Both the receptor and the ligands need to be converted into PDBQT files. But the beauty of Gnina is that it can handle, you know, the other file formats as well; it can take in the PDB format for the receptor, and we can use the SDF file format for the ligands as well. So, what we will do is first install the tools, Py3DMol as well as RDKit. So let us click on this small triangle.

Once you click on this small triangle, you will be able to install these tools. You will actually be able to run this cell. This is called a cell, and then you write some code into it, and then you run it. So now, once I clicked on this small triangle, and once I have run it, you can see that these tools are installed. So, installing the collected packages Py3Dmol and RDKit was successful.

And then we will also need Open Babel to convert the ligand files from one format to another, so let us install that as well. So, once you see a green tick on any cell while you are running it, you will see a green tick. And then you see the execution time as well as how much time it took to execute this: 10 seconds. So, if you see a green tick, it means that the cell has been successfully run and that the work is actually done once you do that. So, the next step is to let us see because GNINA is using, you know, deep learning, and deep learning is a little bit power-hungry.

So, those models are the same models that are a little power-hungry. So, let us see if we have access to the GPU from Google. Otherwise, we will ask Google to allocate some GPU power to us as well. So, once when we click on this NVIDIA SMI. So, if you see something like this, a table like this where you can see the Tesla T4 GPU, okay.

So, it means that we have a GPU which has been allocated. If it is not what you can do, you can always go to runtime; you click on the runtime. And then you can click on the change runtime type, and then you can select the GPU from here. So, you can select a T4 GPU if you are already using a CPU. So, you can select the T4 GPU, okay.

When you click on the T4 GPU, you can save it. So, then you will see that a GPU has been allocated to you by Google, and you can see it here as well. You can see the connection to the Python 3 Google Compute Engine backend and then T4 here. OK, so now we have made sure that we have got our T4 GPU. So, the next step is to let us create a folder using the Unix command mkdir.

So, we just create a folder named ACHE, and you can rename it as well; you can put any other name, but for the sake of simplicity, I have just used ACHE. Okay, so once we have this ache, we click on this, and now we can see that it has created a folder named "ache directory." And then you can cd into that directory, so now we can see that you are in content/ache. Okay, so now we are in the ache directory. So now what we have done is install the required tools, such as Py3Dmol, Open Babel, and RDKit as well.

Okay. So, the next thing is that we now need the structure of the receptor as well as the structure of the ligands. So, for this example, we will be using the structure of acetylcholinesterase where one of the drugs is co-crystallized into its binding pocket. Okay, and that has been deposited in the PDB with the name 1eve, so let us go to this 1eve. So let me give you an idea of the PDB. Actually, this is the Protein Data Bank, where people can upload new structures whenever they generate them.

Solving new 3D structures either using cryo-EM, structural NMR, or X-ray crystallography. So, they deposit their 3D structure coordinates in this database, and we can download those structures and use them directly from here. So, as I said, we will be using the structure named 1EVE, which is a three-dimensional structure of the anti-Alzheimer drug donepezil or E20 adicept complex with its target acetylcholinesterase. And then there are many, you know, things that we need to consider before selecting a structure as well. What we have to consider is that the first thing we need to take into account is the resolution, so we need to choose a structure with a high resolution, typically less than 2.5 angstroms, because Lower angstrom values mean better quality and more reliable atomic positions because many times we see structures that have a resolution of 3.5 or even 4 angstroms.

So, in that case, we will not be able to visualize those atoms correctly in the electron density, and in that case, we may miss some important information. And then another thing to consider when we choose the structure is the biological relevance. So, we need to make sure that the PDB structure it represents the biologically active form, which means it has the correct chain, the correct quaternary structure, or oligomerization state, etc.

And then we need to consider the presence of ligands or cofactors, where we need to prefer structures that already have a bound ligand if available. So, what it helps is it helps us identify the active site and the correct conformation. And then we need to think about the mutations and modifications as well. We need to check for engineered mutations, missing residues, or post-translational modifications that could affect the binding. And then we need to take care of the missing atoms and residues.

So we need to avoid the structures with major missing parts around the binding site unless we actually fix them. So ideally, what we have to do is, once we download the structure, we have to, you know, add all those missing side chains, missing residues, and missing atoms, and remove waters; all those things are considered a part of protein preparation. So, what we have to do is also important for chain selection. We have to think about the water molecules as well, so we need to decide whether crystallographic water molecules are needed. Some of them near the active sites can be critical, but mostly we do not need water molecules.

So, we actually remove most of them. Unless we have strong information about a water molecule that is responsible for creating a water bridge and for making the ligand bind to the active site. So, we have to keep those water molecules; otherwise, we just remove them. For chain selection, we need to choose the correct chain if multiple chains are present, and we have to avoid selecting unintended parts, like symmetry mates. And then, if multiple PDB files are available for the same target, like in the case of acetylcholinesterase, if you search for acetylcholinesterase in the PDB database.

So, we will see that there are, you know, plenty of structures available. So, if there are multiple structures available, what we have to do is select the ones with the best quality resolution and the binding site representation. And then we need to prefer the X-ray or cryo-EM structures over the low-quality NMR structures unless we do not have any other option, okay. And then relevance to the ligand, as well. So, ideally, we need to pick a receptor structure and co-crystallize it with a ligand similar to your docking molecule to ensure a realistic binding site confirmation.

That is also, you know, because the binding site can adapt to multiple molecules that are structurally diverse, okay. So, many of them will have a different orientation; not every molecule will bind in the same orientation, even physiologically. So, that is also one of the things that you need to consider. So, let us go back to this PDB. So, where you can see that the structure we have selected for this is coming from this animal, it is not actually from humans, and then the resolution is 2.

5 angstroms. So, the resolution is really good, and then you can see some more details about this as well, like how this structure has been sold if you go to the experiment details in this case. So, you can see here that the starting model they have used to solve the structure is a 2ACE, okay? And then, for crystallization, they crystallized the protein at a pH of 5.8. And then you can see the Matthew coefficient, which is one of the parameters that tells about the quality of the crystals. And then the solvent content you can see is 68 percent, and the crystal data unit lattice, diffraction data collected, shows how they collected the data.

and how they have done the refinement and how they have, you know, made the model and which software or tools they have used for data reduction, data scaling, model building, refinement, and the So all these details you can see here. So going back to the structure summary in this case, you can see it in the 3D as well. If you click on the structure, you can explore it in 3D. So, what you will see is the structure in 3D. Okay, so you can see here that this is the structure of donepezil bound to acetylcholinesterase.

If you click on this molecule, it will be zoomed out. And then you can see how it interacts with, you know, different binding site residues as well, so that you can see that too. Okay, so this is about the structure. Now that we have chosen the structure, let us go back to the Colab file where we will run this.

We will use the structure. So, once we have selected the structure, what we will do is download the structure. So, using wget, which is a command to download a file, and using this address for this level. So, we download the structure. Just run this cell. And once you have run this cell, you will see that you have downloaded the 1eve.

pdb. And now once you download it, so now we can visualize it. So, using Py3DMol, which we have installed in, you know, Colab, we can visualize the structure. So let us import Py3DMol, and then you can see here, for example, that you can open 1eve.pdb. You can create a viewer here so you can change the size of the viewer as well if you want to make it smaller or bigger.

So here I have made it like 1080, 720, but you can always make it, you know, like for example 600, 400 or whatever you want. And then you add a model view, add model PDB data, and then style the protein. You can add the style, like cartoon and then spectrum. And then you can highlight the ligand structure E20 in the stick style.

So, add style rest name E 20, stick radius 0.25, color magenta, and then we can zoom it to the rest name when we visualize it. So, once we run it, you can see that we can visualize this ligand in this viewer. So, now we have a nice visualizer where you can visualize the structure of any protein, actually. So, okay. So, this is now our donepezil, which is bound to the acetylcholinesterase.

So, this is just to visualize that, okay, we have downloaded the PDB file and everything looks fine. Okay, so the next step is that we can just issue a ls command to list all the files we have in this folder. We created the folder using mkdir, which is called ache, so now we have 1eve.pdb here. So, now you know the next step is to prepare the receptor.

So, what are the steps to prepare the receptors? As I said many times, when we are taking the structures from the crystallographic database, such as the Protein Data Bank. So, those structures are not well prepared, yeah. So, the steps to prepare the receptor: the first step is that we have to obtain this receptor structure, which we have already downloaded. So, the next thing is that we need to remove the water and known relevant molecules. Many times, when you are downloading those structures, they contain extra molecules as well, which can be, you know.

For example, coming from the surfactants or from the precipitant or buffers, actually. So, we usually remove all those molecules. So, we take out all these water molecules as well, which are, you know, not necessary or not important for the ligand to bind to the active site. So, we filter out water molecules or other known receptor molecules using a graph or similar tools. And then we add hydrogens because, in X-ray crystallography, we cannot see the hydrogens.

And that is why many times we cannot differentiate between an oxygen and whether this oxygen is a keto group or a hydroxy group, because we cannot see hydrogen. So, all these things we need to fix. So, we add hydrogen; we use tools like Open Babel or Reduce to add missing hydrogen to the receptor structure, and then we need to assign the protonation state. So, that is, you know, another important thing because, based on the pH, actually many amino acids in, you know, proteins can adapt different, you know, protonation states; like histidine is there, proline is there. So, they can all have different protonated states based on the pH.

So, at the pH that we are going to simulate, we will do the protein. So, we need to define the protonation states at that pH. So, there are multiple tools that can be used. So, we can use Open Babel or Reduce to automatically assign the protonation state, and then we can fix the missing or incorrect atom types. So, again, this can be done using Open Babel or Reduce, and then we have to prepare to convert the format as well.

So, in this case of using the AutoDock tool, we need to convert it into PDBQT; otherwise, using GNINA, we do not need to convert. So, we can directly use it; we can use the PDB file as well. So, this conversion can be done, and then you can add partial charges and atom types using Autodock tools or any other similar tools as well. And then we need to define the docking grid.

We have the protein, and now we have the ligand. So, now the possibility is that it can bind to anywhere in this you know in the protein. But if we know that our binding pocket or our active site is here in this location. So, what we can do is take the ligand and ask the tool to dock it into this area only. So, now that is actually called a docking grid. So, we need to define an area where our ligand will be explored, actually where the tool will explore its binding poses.

So, that docking grid can also be defined using multiple tools. So, there are plenty of methods as well, and we will discuss when we will run this docking. So, there are tools like Prody or Pymol that can also be used. And then we need to check finally whether it is correct or not. So, we need to verify the structure and docking parameters to ensure that it is ready for simulation. So, in this case, we are not going to prepare it in the way that we will step through all the steps.

But for proper preparation of the receptor file, one can use the MECO, which is a highly useful tool recently developed. So, this MECO can be used for, you know, automated protein preparation. So, in this case, what we will do is that we now have the PDB file 1eve, which we downloaded. So, we will grep is a command to take extract all these atoms. So, we will extract all the atoms and then put them into the files called receptor.

pdb and rec.pdb. Okay, and then we will take this rec.pdb using Open Babel; we will convert it into prepare.pdb. So, what Open Babel is going to do is correct all those, you know, atoms, like if there is any problem with, you know, the bond order. And then it will assign the charges as well, and then it will save it into a prepare.

pdb. Okay, so let us run it. You can see one molecule converted. And then, in the next step, what we will do is extract the co-crystal ligand, which is E20, a donepezil, from the PDB 1EVE, and then we will save it as lig.pdb, okay. So, let us extract the ligand. So, now we have extracted the ligand as well as lig.pdb, and in this case, as I said, what we are going to do is, It is just a very simple docking exercise where we have a protein with a co-crystal ligand, and we will take out this ligand, and then we will prepare this ligand.

We will prepare this receptor, and then we will re-dock this ligand into the receptor pocket by using GNINA. So, this is a very simple exercise, to be honest. So, there are multiple steps to prepare the ligands as well. The first thing is that you have to obtain the structure. So, in this case, we have already extracted it from the PDB file, which was co-crystallized, and then we have to remove the non-ligand components.

So, if there are, for example, water or heteroatoms and other non-ligand entities from the structure. So, many times, those ligands are in the salt form. So, we need to, you know, remove those counterions, okay. And then we need to add hydrogen. So, we can use tools like Open Babel to add missing hydrogen atoms to the ligand structure and assign partial charges.

So, partial charges like Gastegier are assigned to atoms to represent the ligands' electronic distribution, because that is very important; only on that basis will the scoring function work properly. So, it is very important to calculate the proper charges on those atoms. And then we need to minimize the structure as well because it is considered that the minimal structure conformation is the biologically active conformation. So, we minimize the structure; we perform the energy minimization to relax the ligand geometry and remove the steric clashes. And then we check the tautomers and ionization states; we ensure that the ligand is in the correct tautomeric form and ionization state, generating possible alternatives if needed.

And then we convert it to the docking format. So, we convert the ligand to the required input format for docking. It could be, for example, if you are using the, you know, AutoDock. So, we need to convert it to PDBQT. But in this case, since we are going to use G-NINA, we do not care about it because G-NINA can handle the SDF file as well.

And then we visualize and clean the structure. So, we inspect the ligand for issues such as steric clashes or correct geometry using molecular visualization tools. And again, we can use MECO for proper ligand preparation. As I said, MECO is a very good tool for automated ligand and protein preparation, even for virtual screening as well. Okay, so now in this case, we are going to use Open Babel. So, we will use Open Babel, call Open Babel, and then give the input structure of the ligands as lig.

pdb. We will ask it to save the output structure as lig_optimized.sdf. So, what we are asking it to do is to add hydrogens, add hydrogens, and minimize the structure using the universal force field. Okay, and also add the Gastegier charges as partial charges from the Gastegier, so it will add. Now, let us click on this to run this cell. And then you can see one molecule converted, so we have got one molecule converted.

Okay, now let us see how it is different from the co-crystallized structure. So then again, we will call Py3Dmol, and we will run Py3Dmol, and in this case, what we are doing here is We are adding the lig_optimized.sdf, which is the ligand structure prepared using obable, which we prepared in the previous step. And we will also add the lig.pdb, which was the original ligand that was the co-crystal ligand, okay.

So, in this case, we can see that the lig.pdb is in the green carbon; you can see here we have set its style to the green carbon for the lig.pdb, and our lig_optimized.sdf is in the cyan carbon, okay. So, when we look at this structure here, you can see the structure of both of these. So, here is the green one, which is the co-crystal structure, and this cyan one, which is, you know, the prepared.

So, the difference you can see is the addition of hydrogens because when we downloaded it from the PDB structure 1EVE. So, there were no hydrogens added to this structure, right? So, here we can see that we have added the hydrogens here. So, now you see the hydrogens, as well as a little bit of change in the conformation when we have prepared it, okay. So, yeah, there was a little bit of conformational change when we prepared it, but it is not a lot because, you know, we already started with the co-crystal structure, and it means that it possibly has the least energy, okay.

So, now we are satisfied with the ligand's conversion. Many times, what happens is you are using some tools, and if they cannot recognize that this is a benzene ring, they fail to provide accurate results. So, what they will do is make it a cyclohexane and add all these CH2 everywhere. So, they add multiple hydrogens, two hydrogen atoms on each carbon atom. So, that is why visualization is very important. So, whenever we are converting or preparing some structure, we always need to visualize it to make sure that it is working fine.

It has no problems. Okay. So now we are happy with the ligand preparation. We are happy with the ligand preparation. So, the next step is to download G-NINA. So, let us click on this cell where we will download the G-NINA execution file for GNINA. So, we have now downloaded. So, it is a kind of 536 MB file, and for running it, we need to give permission to the gnina to run as well.

So, using the chmod command, we give permission to GNINA, and then we can also check by running the next cell what we can do is check which version we are using. So, you can see the GNINA Dash version is where we are; you know, using version 1.0.1, which was released on March 23, 2021.

Okay, so the next step is that we can always get back to GNINA for help as well. If you issue the help command, dash help, it will show all the possibilities you can do with GNINA. So, let us quickly go through some of them. To use the GNINA, now what we have to do is supply a receptor structure with an argument, with a dash r followed by an argument. Like the dash dash, we can use dash dash receptor as well, or dash r, and then argument the name of the receptor.

And then dash l for the ligand again, followed by the name of the ligand file. And then we can also specify the possibility to do flexible docking, where we can define the flexible side chains that we can keep for the flexible docking. Usually, what happens is that whenever we are docking, it is called rigid docking, where we keep the receptor rigid and take the ligands as flexible. So, in this case, you can also specify that in this binding pocket, I have these two residues that are important for interaction, and I want them to be flexible. So, you can use them flexibly for flexible docking as well. And then you can specify the flex distance, flex limit, and flex max, which, you know, the details are given here.

And then the most important thing is the search space, which you can also refer to as the docking grid. So, this is the space where this molecule will be searched and explored; this is the space where the program will explore the different conformations of molecules and how they can bind. So, in this case, we have multiple ways, actually. So, one way is classical to AutoDock tools as well as AutoDock Vina, where you can specify it using the coordinates: XYZ coordinates center X, center Y, center Z, X coordinate, Y coordinate, and Z coordinate.

And a size of this box is actually a size in the xyz direction in angstroms. Okay, that is one way; another way is to use the autobox ligand argument where. What we can do is say that, okay, I have this receptor structure where a ligand is already bound that has been co-crystallized. I have maybe ten different molecules that I want to explore in terms of how they bind in place of this co-crystallized ligand. So, I am giving this co-crystal ligand as a reference; I am saying that, okay, this is my reference point, and then I will just explore how my new molecules are binding in the place where this co-crystal structure is bound, okay? So, this is where we use the autobox ligand. So, we need to specify the ligand that is already bound in that pocket, and in this specific tutorial, we will be using this method.

So, we will define the grid using the co-crystal structure. Okay. And then you can also have this, you know, the autobox add argument as well, where you can mention the amount of buffer space to add to the auto-generated box. Okay. The default is plus four, but you can just have plus five or plus four plus six if you have large molecules that are comparatively larger than your co-crystal structure. So, what you can do is specify this, you know, otobox add plus 5 plus 6, which will, you know, increase the search space. And then you have the auto box extend as well, which expands the auto box if needed to ensure the input confirmation of the ligand being docked can freely rotate within the box.

And then you can do the no-ligand sampling as well, which is for minimizing the sampling of flexible residues. And then you can have the scoring and minimization options too. In Gnina, there are multiple scoring methods; at least they have these three scoring methods. So, what we need to do is specify dash dash scoring with an argument.

So, we need to specify alternative built-in scoring functions: AD4 scoring, default, or dikoes fast. So, dikoes scoring, dikoes scoring old or vina vinaldo. So, all of these are different scoring methods you can apply, and then you can specify them using dash-dash scoring. And then you can do the custom scoring as well; you can have a custom scoring function file, and you can do the custom atoms as well, where you can add the custom atom type parameter file. And then you can just do the score only where the score provided the ligand pose.

So you provide a ligand pose, and then it will only score it. It will not do the docking. And then you can do the local search only using the autobox. You probably want to use dash dash minimize before that, and then you can use minimize for energy minimization and randomization. So, all of these are, you know, different options you can use. Okay. So, the ones that were important I just explained to you, and then, yeah, coming to the CNN scoring function, that is important because this is where we are using deep learning, okay.

So, for CNN scoring, you can use the amount of CNN scoring. So, either you have to specify none, rescore, which is default, or refinement and all when we were discussing this gnina as well. So, we saw that it can have the three at three different levels of scoring as well. So, in this case, the default one is rescore. So, that means your ligand has been docked okay, the poses have been generated, and then GNINA will rescore those poses. Ok, and then you can also specify the CNN-CNN as well, where you can indicate which model to use, such as the built-in model, and specify the prefix ensemble to evaluate an ensemble of models starting with the prefix either cross doc default 2018, cross doc default 2018-1, or cross doc default 2018-2.

So, they have multiple, you know, models that they have developed. So, which CNN model did you want to use to rescore your ligands? That can also be specified. Okay, and then you can see the CNN model file. If not specified, a default model will be used. Even you can have a CNN model file. As well, you can have your own, you know, a custom build file, and then you can use that as well for rescoring using the CNN and then CNN weights, okay, and then CNN resolution.

Okay, yeah, so these are some of the important options for the model, and then coming to the output, that dash O will specify where you want to save those files. The output file name format is taken from the file extension, so "dash dash out flex" is for output files for flexible receptor residues. And then, the dash dash log is for, you know, if you wanted to write the log files; it's just optional, actually. And then if you don't have access to a GPU, you can use the CPU as well, specifying the number of CPUs to use.

So the default is to try to detect the number of CPUs or, you know, failing that, it uses one CPU. And then you can also specify the seed argument as well. You can also specify seed, which is the explicit random seed to perform the docking. And then you can specify the exhaustiveness as we do in Autodock Vina, and you can specify the num modes, where the default maximum number of binding modes to generate is 9. So, it will produce 9 poses, actually, and then you can specify -q, which suppresses the output message, -H, which automatically adds hydrogen to the ligand.

So, you know that is the advantage of this gnina, you know. So, it can take various input formats; it can add hydrogens while working with those ligands as well, and it can take them as SDF, PDB, or PDBQT files as well. Okay, and then optionally, you can provide a config file as well. So, where all this information can be put into a text file, and then you can supply that config file, and then it can use that information to perform the molecular docking.

Ok. So, now let us do the docking. So, before that, let us check whether we have all those files that we need. So, we got this pdb file that we downloaded. We installed the GNINA, extracted the lig.dot.pdb, and prepared it into lig_optimized.

And then we prepared the PDB receptor file, which we extracted from the PDB, and we prepared the receptor file, okay. So, now I think we have all the necessary files to run the docking. In the next step, what we will do is use this dot slash GNINA command, and then dash r will specify the receptor. Dash L, we specify the ligand that we want to dock, and dash dash autobox ligand, we specify the lig.dot pdb from which we have extracted the co-crystal ligand from the receptor.

And in this case, we are using this extracted ligand file to specify the grid. We are now telling Gnina to just dock the lig_optimized.pdb ligands into the place where the lig.pdb is bound, okay? So, this is how we are specifying the docking grid: dash out is docked.

sdf, we are saving it into docked.sdf, and then dash dash seed is 0, okay. So, let us run this. So, when we run it, you can see that it is again the star will move from 0 to 100 percent since it is taking a little bit more time compared to AutoDock Vina. Because in this case it is using the CNN to rescore those molecules' poses, it will give us not only the AutoDock Vina score but also the CNN score as well. So, depending on the structure, another benefit of using the SDF file is that you can include multiple molecules in a single file because the

SDF file format can handle multiple molecules. So, if you wanted to dock maybe 20, 50, or 1000 molecules, you can keep all of them in a single file and then dock them.

Okay, so now the docking is done, so you can see the pose on the affinity kCal per mole calculated from the AutoDock Vina, minus 11.7 CNN score. So here, 0.8548, and then CNN affinity 6.

879. Okay, and I can see that these poses are ranked according to the CNN pose score, not according to the affinity. Because you can see here, like pose four is having minus ten point well, or even post nine is having minus 10.36, which ideally should come at number two if we are using this as a ranking, actually. So, here the poses are ranked according to the CNN pose score, which has been calculated using the CNN algorithm in Gnina.

So, now that the docking is done, let us just visualize it. So, the beauty of CoLab is that you can do everything in it. Now, for visualizing, when we click on it. So, what we are doing here is importing the receptor.

pdb, opening the receptor.pdb, and setting it up in the stick radius color spectrum. And then we are also adding the molecule lig.pdb, which is our co-crystal structure, in the color dim gray carbon. And then we are adding the docked output files dock.sdf in the green carbon, okay? And then we are animating it so that every 5 seconds all the poses are visualized. So, let me run it again.

The green one is the docked pose, okay? As you can see, this is posed 3, actually, which is nicely aligned with the co-crystal structure. So, the grey one is the co-crystal structure, and the green ones are the docked poses. So, now when I am doing it, when I am animating it, I can see all of them, you know, one by one. So, you can see that one by one I can see, and if I want to start again, I can quickly run this again. So, you just run this and then you will start from scratch. So, you can see this is pose 1, which is not as good; pose 2 is not as good as well because you can see that it is kind of a flipped orientation.

Pose 3 is really good; you can see a nice improvement if you calculate the RMSD of these two, then you can get a value closer to 1, actually. So, then let us again issue the ls command, and here you can see that we have the docked.sdf file, which contains the information about the docking pose that is generated. And then in the end, what we can do is zip all these files and then download them. And we can even visualize them in other tools like Discovery Studio Visualizer, Schrodinger Maestro, or PyMol.

So any tool you can use to visualize them, and then we can analyze them as well. We can plot, you know, we can do plotting in Colab as well. But I just kept it quite simple so that you can make some effort to analyze these structures. So, these docking poses. So, one thing could be how you can upload or use your own structure, your own molecules.

So, for that, let me first download it. So, if you click on this, you will get it downloaded, and then you can see now it has downloaded ACHE.zip, which can be used for visualizing or making graphics. So, one thing you wanted to know is maybe how to use your own molecules. So, there are multiple ways to do that.

So one way is to mount your Google Drive. So, you can mount your Google Drive in Colab and then access files from there. Other ways, you can just open this folder here. So, if you click on this folder, right, and then you can click on this ACHE folder, because here you see all these files. So, even you can upload files from here as well; like if I wanted to upload some file, if I just click on upload and then I have this furesmide.sdf file and then I open it, it will say some warning.

And then you can see that it has been uploaded in this directory since it is not in the ACHE directory that I created. So, I can simply move it by dragging as well if I drag it here and then put it there, so you can see that now I have this file here in the ache directory. And that I can confirm in the Colab as well, like if I go, if I issue this ls command again, so now you can see that furesamide.sdf is here in this directory as well. And now, since we have this file, we can prepare it, use it for docking, and download it.

And one more thing, as soon as we close this colab, all the data will be lost. So, the most important thing is that whenever you are using this for docking or for doing some simulation—I am talking about Colab, actually—if you are using it for something. So, you make sure that you download all the data which is generated before closing it down. Okay, so yeah, with that, we would like to wrap up this hands-on session. So, I hope you enjoyed it and that you can explore more on your own. Thank you so much.