
Supplementary information

The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies

In the format provided by the
authors and unedited

Supplementary Information for The Virtual Lab of AI Agents Designs New SARS-CoV-2 Nanobodies

Supplementary Note 1: Virtual Lab ESM usage

Notably, the Virtual Lab's ESM nanobody mutation analysis is similar to the ESM-based antibody design process of Hie et al.⁶⁴, but it differs in important ways. The Virtual Lab computes the ESM LLR of mutant sequence x' with mutation at position i compared to the input sequence x as $LLR(x'|x) = \log \left(\frac{p(x_i'|x')}{p(x_i|x)} \right)$, where $p(x_i'|x')$ is the ESM probability of the mutant amino acid x_i' at position i in the mutant sequence x' and $p(x_i|x)$ is the ESM probability of the original amino acid x_i at position i in the input sequence x . In contrast, Hie et al. use $p(x_i'|x)$ as the numerator in the ratio instead of $p(x_i'|x')$, thereby computing the probability of the mutant amino acid in the *input* sequence rather than the probability of the mutant amino acid in the *mutant* sequence. Additionally, the Virtual Lab only uses the ESM-1b⁶⁵ model while Hie et al. use a consensus of ESM-1b and five ESM-1v⁶⁶ models.

Supplementary Note 2: Extended nanobody experimental validation

In the Nb21 mutant nanobody I77V-L59E-Q87A-R37Q, the R37Q mutation may be important for binding since the triple mutant nanobody without R37Q (i.e., I77V-L59E-Q87A) and three nanobodies with the same triple mutation but a different fourth mutation (I99R, Q87A, or R43L) do not bind to JN.1 or KP.3. Additionally, the L59E mutation may contribute to JN.1 RBD and KP.3 RBD binding because the JN.1 RBD and the KP.3 RBD have mutations that introduce positive charges relative to the Wuhan RBD (e.g., E484K) while the L59E and R37Q mutations in the nanobody introduce a negative charge (and reduce positive charge), which could lead to beneficial electrostatic forces or hydrogen bonding.

In the Ty1 mutant nanobody V32F-G59D-N54S-F32S, interestingly, the Virtual Lab initially introduced a V32F mutation in the first round and then later followed up with an F32S mutation in the fourth round, resulting in a V32S mutation in the final mutant nanobody. Every one of the 23 mutant nanobodies introduced a mutation at position 32 in the first round (either V32F or V32C), but only three of the 23 mutants later modified position 32 again, two introducing F32Y and this mutant introducing F32S. This indicates that position 32, which is predicted to be near the binding domain, may be particularly important for Ty1 but may require a very specific mutation to induce binding to JN.1. Similar to the Nb21 mutant that binds to the JN.1 RBD, this Ty1 mutant also introduces a negative charge (G59D). All the mutations of this mutant nanobody are in the CDR loops of the nanobody.

Supplementary Note 3: Extended analyses of Virtual Lab interactions

The meetings in the nanobody design workflow reveal interesting social dynamics among the Virtual Lab agents. For example, in the project specification team meeting, the PI agent wrote the most (36.4% of the words) followed by the Scientific Critic (21.0%) and then the three scientist agents—the Computational Biologist (14.8%), the Machine Learning Specialist (14.5%), and the Immunologist (12.8%)—with the human researcher only writing 0.5% of the words. The distribution is similar for the tools selection team meeting. This order is reasonable given that the PI not only has to synthesize the agent responses after each round of discussion to guide the next round but also has to initiate the meeting and write a summary at the end. The Scientific Critic likely writes more than the scientist agents since the Scientific Critic must address the limitations of every agents' response while each scientist agent is only concerned with providing its own opinions. In the individual meetings with both a scientist agent and the Scientific Critic, the scientist agent tends to dominate the discussion, producing around 66% of the words compared to around 33% for the Scientific Critic and around 1% for the human researcher. In individual meetings with an agent without the Scientific Critic, which have no back-and-forth discussion, the agent writes more than 80% of the words compared to less than 20% written by the human researcher.

The use of parallel meetings was notable as it led to answers with improved consistency and quality. For example, the five parallel individual meetings with the PI agent produced the following five sets of scientist agents.

1. Computational Biologist, AI/ML Specialist, Immunologist
2. Computational Biologist, Immunologist, Machine Learning Specialist
3. Computational Biologist, Virologist, Antibody Engineer
4. Immunologist, Structural Biologist, Machine Learning Specialist
5. Computational Biologist, Structural Biologist, Immunologist

Upon merging these answers, the PI chose to define a Computational Biologist, a Machine Learning Specialist, and an Immunologist. This roughly aligns with the majority vote of the five parallel meetings, resulting in a more consistent answer than any one of the meetings.

Furthermore, the merge meeting aimed to select the best form of each agent. For example, the final merged Machine Learning Specialist's expertise is "developing algorithms for protein-ligand interactions and optimization," which is the same as that of the Machine Learning Specialist in the second of five meetings and is superior to the more vague expertise of "machine learning algorithms and data analysis" from the AI/ML Specialist in the first meeting and "applying AI and ML models to biological data" from the Machine Learning Specialist in the fourth meeting.

Including the Scientific Critic in team and individual meetings often led to improved answers thanks to the Scientific Critic's helpful critiques. As an example, in an individual meeting with the Machine Learning Specialist and the Scientific Critic to write ESM code, the Scientific Critic explained several important ways to improve the code. In the first round of discussion, the Scientific Critic noted "the use of random numbers as a placeholder for ESM log-likelihoods" and said that "the agenda specifies using 'ESM amino acid log-likelihoods,' yet this critical component is not implemented." In response, the Machine Learning Specialist rewrote the code with a new function called `get_esm_log_likelihoods` with documentation explaining how ESM could be used to calculate log-likelihoods. While the new code still computes random log-likelihoods as a placeholder, this change represents a step in the right direction thanks to the Scientific Critic. In a parallel meeting, the Machine Learning Specialist wrote code that correctly loads and runs an ESM model, and this code was used in the final merged answer (Supplementary Box 1), again demonstrating the benefit of multiple parallel meetings to arrive at an optimal answer.

Supplementary Box 1 | Virtual Lab ESM Code. A function from code written by the Machine Learning Specialist agent (with feedback from the Scientific Critic agent) that uses ESM to compute the log-likelihood ratio of point mutations in a nanobody sequence.

```
def compute_log_likelihood_ratios(seq: str, model, tokenizer) -> List[Tuple[int,
str, str, float]]:
    """Computes log-likelihood ratios for each possible point mutation in the
    sequence.

    Args:
        seq (str): The input nanobody sequence.
        model: The ESM model for masked language modeling.
        tokenizer: Tokenizer corresponding to the ESM model.

    Returns:
        List[Tuple[int, str, str, float]]: A list of tuples containing position,
        original amino acid, mutated amino acid, and log-likelihood ratio.
    """
    try:
        encoded_input = tokenizer(seq, return_tensors='pt',
add_special_tokens=True)
        original_output = model(**encoded_input)

        log_likelihoods = []
        amino_acids = 'ACDEFGHIKLMNPQRSTVWY'

        for pos in range(1, len(seq) + 1): # Skip [CLS] token which is at index 0
            for aa in amino_acids:
                if seq[pos - 1] == aa:
                    continue

                mutated_sequence = seq[:pos - 1] + aa + seq[pos:]
                mutated_input = tokenizer(mutated_sequence, return_tensors='pt',
```

```

add_special_tokens=True)
    mutated_output = model(**mutated_input)

    original_ll = original_output.logits[0, pos,
tokenizer.convert_tokens_to_ids(seq[pos-1])).item()
    mutated_ll = mutated_output.logits[0, pos,
tokenizer.convert_tokens_to_ids(aa)].item()
    ll_ratio = mutated_ll - original_ll

    log_likelihoods.append((pos, seq[pos - 1], aa, ll_ratio))

    return sorted(log_likelihoods, key=lambda x: x[3], reverse=True)
except Exception as e:
    print(f"An error occurred during computation: {e}. Please ensure your
sequence is valid and model is correctly loaded.")
    return []

```

Supplementary Note 4: Ablation experiments

Here, we further explore some of the components of the Virtual Lab to elucidate their contribution to the overall function of the Virtual Lab.

Generic versus scientist agent

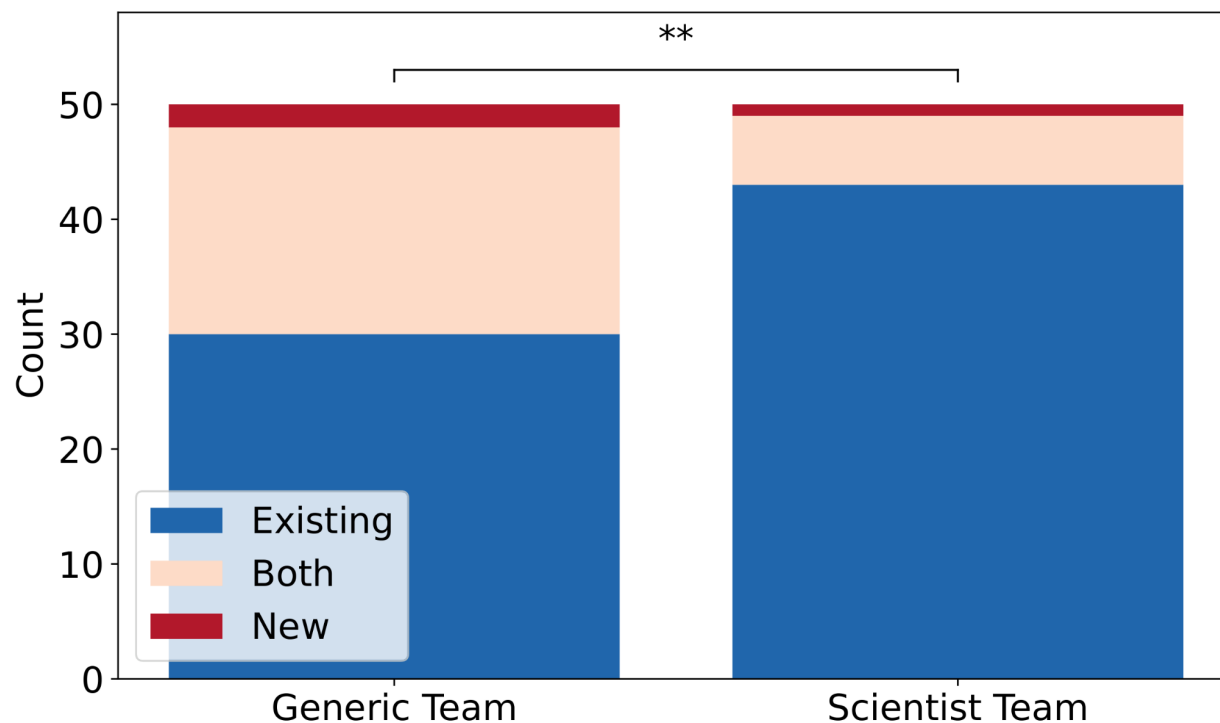
The agents used in the Virtual Lab are specialized in that they are instructed to behave as certain types of scientists (e.g., Immunologist) with a particular title, expertise, goal, and role. To illustrate the benefit of this specialization, we ran the project specification team meeting using an equally sized team of generic agents. Each generic agent is given a generic title (e.g., “Assistant Lead” or “Assistant 1”) as well as a generic expertise, goal, and role (all along the lines of “helping people with their problems”). Otherwise, the agenda, agenda questions, and other Virtual Lab hyperparameters (e.g., number of rounds, number of parallel meetings) were identical between the generic and scientist team meetings. Interestingly, the generic team and scientist team had noticeably different discussions and conclusions regarding the decision about whether to design de novo antibodies/nanobodies or whether to modify existing antibodies/nanobodies.

The generic team decided to adopt a hybrid approach of both de novo design and modification, but only after major disagreement among the agents. In four of the five parallel meetings, the agents took strongly opposed views on modification versus de novo design. For example, in one of those meetings, the Assistant Lead noted that “There is a division of opinion with compelling arguments for both approaches. Modification of existing nanobodies: Suggested by Assistant 1 and Assistant 3, leveraging known nanobodies like Ty1, H11-D4, and Nb6 to build on their efficacy and structural stability. De Novo Design: Supported by Assistant 2, emphasizing the potential for innovative targeting through machine learning-driven models. Hybrid Approach:

Proposed by Assistant 4, suggesting combining both strategies to maximize effectiveness and coverage.” The Assistant Lead ultimately had to make the decision and chose to align with Assistant 4 and suggest a hybrid approach. Notably, the hybrid approach is not ideal since it requires developing two independent nanobody design pipelines—one for de novo design and one for modification—so only selecting one, as requested in the agenda question, would have been preferred.

In contrast, in the scientist team meeting, the Immunologist strongly advocated for nanobody modification in four of the five parallel meetings, while suggesting that both modification and de novo design could work in the fifth meeting. The Immunologist provided biologically-grounded justifications, such as “this method [modification] can potentially expedite the process by building on existing structural frameworks that have already shown partial efficacy against previous SARS-CoV-2 variants.” The remaining scientist agents tended to defer to the Immunologist’s expertise on this topic while providing domain-specific supporting evidence. For example, the Machine Learning Specialist argued that “utilizing machine learning to optimize existing antibodies or nanobodies is appealing because we can leverage known structures and experimental data to train predictive models. Techniques such as transfer learning can be particularly effective here, where we fine-tune pre-trained models on the specific characteristics of the new variant.” The Computational Biologist stated from the perspective of more traditional computational biology techniques that “the advantage of modifying existing antibodies or nanobodies is that we can rely on well-characterized structures. This allows us to perform targeted molecular dynamics simulations to assess the impact of specific mutations or modifications on binding affinity and stability.” These domain-specific comments show how the specialization of the agents leads to nuanced discussions that tackle the decision from multiple complementary perspectives rather than having multiple equivalent agents argue over the best answer.

To better quantify the difference between generic agents and scientist agents, we ran the project specification team meeting 50 times each with the generic agent team and the scientist agent team, and we counted the teams’ decisions on protein type (antibody or nanobody) and design approach (de novo design or modifying existing antibodies/nanobodies or a hybrid approach). In all cases except for one, the generic agents and scientist agents decided to pursue nanobody design. Interestingly, echoing the qualitative results above, the generic agents suggest the less efficient hybrid or de novo design approaches more often than the scientist agents ($P < 0.01$), which tend to prefer the more efficient method of modifying existing antibodies/nanobodies (Supplementary Fig. 1). The human protein engineering experts (J.E.P. and W.W) also preferred the modification approach suggested by the scientist agents.



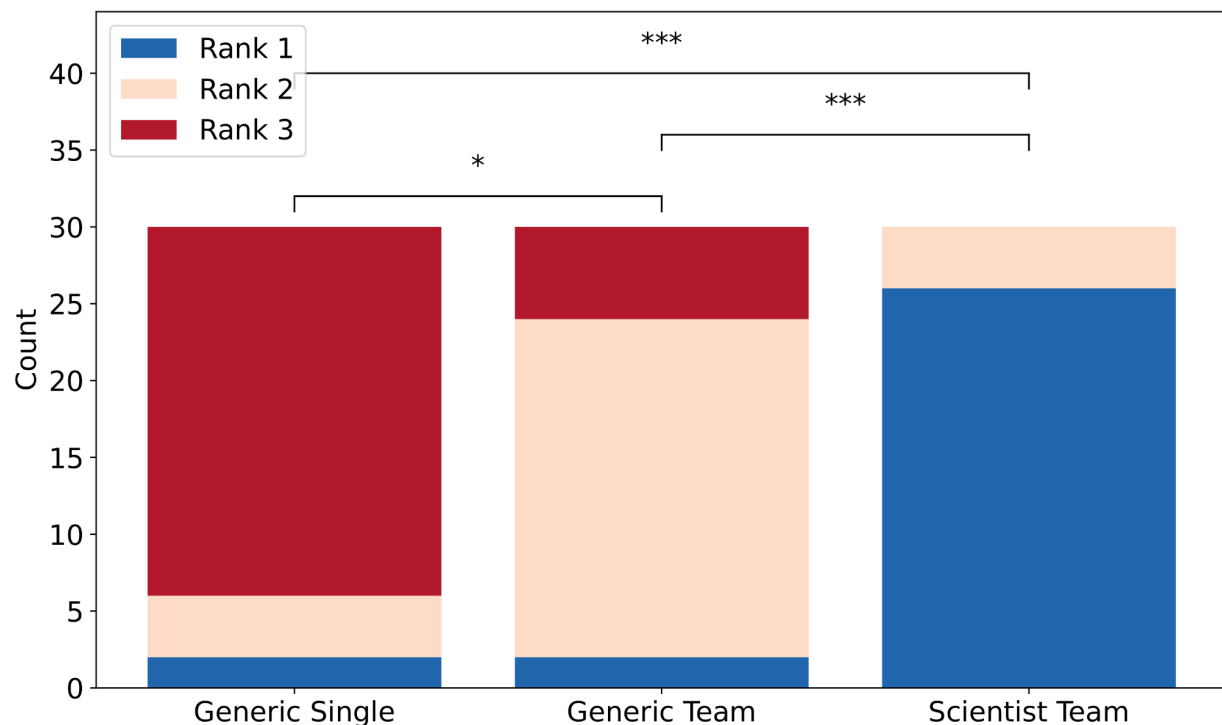
Supplementary Fig. 1 | Generic agents and scientist agents select the design approach. The number of times a team of generic agents and a team of scientist agents choose de novo design, modifying existing antibodies/nanobodies, or a hybrid approach at the end of $n=50$ identical project specification team meetings. $P = 0.006$ (** $P < 0.01$) using a two-sided Fisher's exact test to compare modifying existing antibodies/nanobodies to de novo design or a hybrid approach (combined).

As another evaluation of the effect of the Virtual Lab framework, we ran a blinded human expert evaluation comparing discussions involving a single generic agent, a generic agent team, and a scientist agent team (i.e., the Virtual Lab framework). With the help of GPT-4o, we crafted 30 open-ended research questions, 10 related to protein engineering and 20 related to computational biology. For each question, we ran each of the three frameworks (single generic agent, generic agent team, and scientist agent team) on the same question with a single meeting and with no prompt engineering. For the scientist agent team, we first gave the Principal Investigator the research question and asked it to define a set of scientist agents for the discussion. Then, we used those scientist agents in a team meeting to discuss the question. For the generic agent team, we created a team of generic agents (named Assistant 1 to Assistant N) with the same number of agents as the scientist team and ran a team meeting with them. For the single generic agent, we created one generic agent and ran an individual meeting with one round (i.e., no Scientific Critic).

After running these meetings, we blinded the discussions by replacing the names of the scientist agents (e.g., Computational Biologist) with the names of the generic agents (e.g., Assistant 1) so

that the names of the agents cannot be used to distinguish between the generic agent team and the scientist agent team, which have the same number of agents. Next, we asked one author (J.E.P.) to review the blinded discussions of the 10 protein engineering questions and another author (J.Z.) to review the blinded discussions of the 20 computational biology questions. For each question, they read all three discussions and ranked them from 1 (best) to 3 (worst) in terms of scientific accuracy, creativity, and comprehensiveness.

Supplementary Fig. 2 shows the result of the human evaluation. The scientist agent team (i.e., Virtual Lab framework) significantly outperformed both the generic agent team and the single generic agent in terms of human preference ($p < 0.001$). The discussions of the scientist agent team tended to be more comprehensive and nuanced than those with generic agents. Furthermore, the generic agent team significantly outperformed the single generic agent, adding evidence that multi-agent systems can provide higher quality scientific insights and decisions compared to single agent systems. This is consistent with similar findings on the benefits of multi-agent systems in other domains^{67,68}.



Supplementary Fig. 2 | Human evaluation of generic agents versus scientist agents. The results of a blinded human evaluation of $n=30$ open-ended research questions (10 protein engineering, 20 computational biology), in which human reviewers ranked the discussions of a single generic agent, a generic agent team, and a scientist agent team. All significance tests are two-sided Wilcoxon signed-rank tests with a Bonferroni correction. Generic Single

vs Generic Team: $P = 0.0196$, Generic Single vs Scientist Team: $P = 3.88 \times 10^{-6}$, Generic Team vs Scientist Team: $P = 1.28 \times 10^{-5}$. * $P < 0.05$, ** $P < 0.01$, *** $P < 0.001$.

Single versus multi agent

Although each individual agent in the Virtual Lab is knowledgeable and capable on its own, interactions between different agents are still very valuable as they lead to even better answers than any individual agent tends to produce. This effect is especially apparent in the interactions between the scientist agents and the Scientific Critic when writing code.

For example, consider the individual meeting in which the Computational Biologist is tasked with writing a Python script to calculate the interface pLDDT of a nanobody-spike complex produced by AlphaFold-Multimer. The first draft of the code written by the Computational Biologist alone (Supplementary Box 2) contains a serious logical error in that it calculates the overall pLDDT but not the interface pLDDT, despite the name of the function and the documentation both saying that it is meant to calculate the interface pLDDT. After three rounds of interaction with the Scientific Critic, in which the Scientific Critic catches this logical error, the Computational Biologist produces a much improved function (Supplementary Box 3) that correctly calculates an interface pLDDT by first identifying only those residues that are in the interface (i.e., within a certain distance threshold of a residue in the other protein chain) and then calculating their mean pLDDT. Additionally, the code is further improved via this multi-agent interaction so that it calculates additional helpful statistics about the interface pLDDT, such as the standard deviation and the number of interface residues, beyond the simple mean value that is required.

Supplementary Box 2 | Single Agent AlphaFold-Multimer Code. A function from code written by the Computational Biologist agent to compute the AlphaFold-Multimer interface pLDDT of a nanobody-spike complex prior to interactions with the Scientific Critic. The line highlighted in yellow is where the code incorrectly includes the pLDDT values of all residues rather than just those residues that are in the interface of the two protein chains.

```
def calculate_interface_pLDDT(pdb_file: str) -> float:
    """Calculate the average interface pLDDT score for a given PDB file."""
    parser = PDBParser(QUIET=True)
    structure = parser.get_structure('complex', pdb_file)

    # Assume pLDDT score is stored in B-factor fields (typical for AlphaFold
    outputs)
    interface_scores = []

    # Iterate over each chain to calculate pLDDT for residues at the interface
    for model in structure:
```

```

    for chain in model:
        for residue in chain:
            if is_aa(residue):
                # Extract the pLDDT score from the B-factor attribute
                plddt = residue["CA"].get_bfactor()
                # Here we consider all residues, but typically you'd filter
residues at the interface
                interface_scores.append(plddt)

if not interface_scores:
    raise ValueError(f"No interface residues found in {pdb_file}.")

average_plddt = sum(interface_scores) / len(interface_scores)
return average_plddt

```

Supplementary Box 3 | Multi Agent AlphaFold-Multimer Code. A function from code written by the Computational Biologist agent to compute the AlphaFold-Multimer interface pLDDT of a nanobody-spike complex after three rounds of interaction with the Scientific Critic. The line highlighted in yellow is where the code now correctly only includes pLDDT values from residues that are in the interface of the two protein chains (compare to the highlighted line that is incorrect in Supplementary Box 2).

```

def calculate_interface_pLDDT(pdb_file: str, distance_threshold: float) -> dict:
    """
    Calculate interface pLDDT statistics for a given PDB file.

    :param pdb_file: Path to the PDB file.
    :param distance_threshold: Distance threshold for defining interface residues.
    :return: Dictionary with average, standard deviation, min, max, and count of
    interface pLDDT scores.
    """

    parser = PDBParser(QUIET=True)
    structure = parser.get_structure('complex', pdb_file)

    # Collect atom positions for neighbor search
    atoms = [atom for atom in structure.get_atoms() if is_aa(atom.get_parent())]
    neighbor_search = NeighborSearch(atoms)

    interface_residues = set()

    # Identify interface residues based on distance between atoms in different
chains
    for chain in structure.get_chains():
        other_chains = [c for c in structure.get_chains() if c != chain]
        for residue in chain:

```

```

        if not is_aa(residue):
            continue
        for atom in residue:
            # Check if any atom in this residue is close to atoms in other
chains
            if any(neighbor_search.search(atom.coord, distance_threshold, 'R')
for other_chain in other_chains):
                interface_residues.add(residue)
                break

    if not interface_residues:
        raise ValueError(f"No interface residues found in {pdb_file}.")

    # Calculate pLDDT statistics for interface residues
    interface_scores = [residue["CA"].get_bfactor() for residue in
interface_residues]
    average_plddt = sum(interface_scores) / len(interface_scores)
    std_dev_plddt = (sum((x - average_plddt) ** 2 for x in interface_scores) /
len(interface_scores)) ** 0.5
    min_plddt = min(interface_scores)
    max_plddt = max(interface_scores)

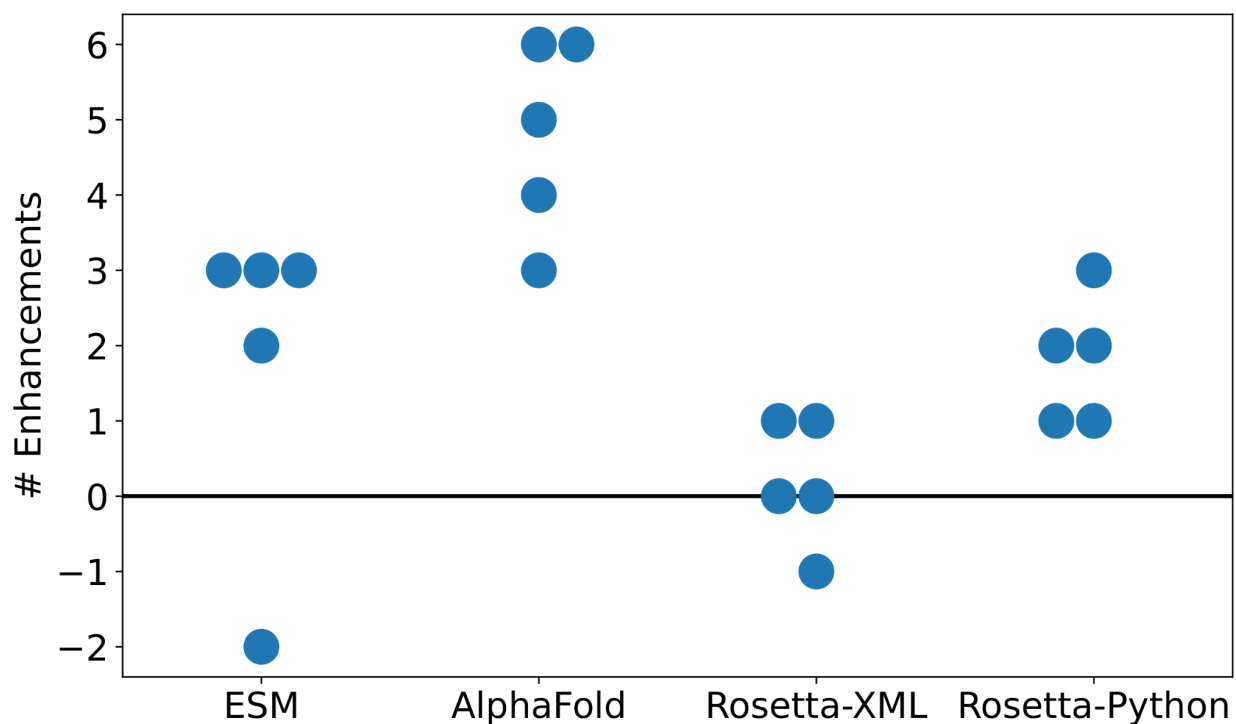
    return {
        "average_plddt": average_plddt,
        "std_dev_plddt": std_dev_plddt,
        "min_plddt": min_plddt,
        "max_plddt": max_plddt,
        "interface_residue_count": len(interface_residues)
    }

```

To quantify the effect of the agent-critic interactions across the different phases of the Virtual Lab nanobody design process, we looked at every script written by the Virtual Lab throughout these phases and compared the original script written by a single scientist agent to the final script written after the interaction between the scientist agent and the critic agent. We performed a manual code review and counted the number of enhancements introduced via this multi-agent interaction, where an enhancement is a major positive change to the code (e.g., correcting a bug or improving error handling as opposed to simply renaming a variable or adding a comment). Major negative changes are counted as negative enhancements. We reviewed the Python scripts written for ESM, AlphaFold-Multimer, and Rosetta as well as the XML scripts written for Rosetta. For each script type (e.g., ESM), we reviewed five individual pairs of the original single agent script and the final multi agent script, one from each of the five parallel meetings.

As seen in Supplementary Fig. 3, the multi-agent interaction produces major enhancements in 16 of the 20 scripts, with no changes or just 1-2 negative changes in the other four scripts. Notably,

the most complex script (AlphaFold, which involves PDB file parsing, 3D geometry calculations, and confidence score extraction) benefits the most from the multi-agent interaction, while the simplest script (Rosetta-XML, which is just a few lines of Rosetta script tags) has the fewest enhancements, likely because it has the least room to improve. Across all 20 scripts, the number of enhancements is statistically significantly greater than zero ($p = 0.0005$ using a one-sided Wilcoxon signed-rank test), demonstrating that multi-agent interactions have a strong tendency to improve code quality.



Supplementary Fig. 3 | Code enhancements via multi-agent interactions. The number of significant enhancements (positive if beneficial, negative if detrimental) introduced in the final code produced by a multi-agent interaction with a scientist agent and a critic agent compared to the original code written by the single scientist agent. For each script type (e.g., ESM), five pairs of multi agent and single agent scripts are compared, one from each of the five parallel meetings.

Prompt engineering

As with all frameworks that employ LLMs, the Virtual Lab requires some degree of prompt engineering in order to guide the agents toward more useful discussions. As an illustration of the effect of prompt engineering on the discussions within the Virtual Lab, we explore the impact of different prompts (i.e., an agenda and associated agenda questions) on the tools selection phase of the nanobody design process.

Original Prompt

The original prompt used during the nanobody design process is below.

Agenda: “You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previous decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. Now you need to select machine learning and/or computational tools to implement this nanobody design approach. Please list several tools (5-10) that would be relevant to this nanobody design approach and how they could be used in the context of this project. If selecting machine learning tools, please prioritize pre-trained models (e.g., pre-trained protein language models or protein structure prediction models) for simplicity.”

Agenda questions

1. “What machine learning and/or computational tools could be used for this nanobody design approach (list 5-10)?”
2. “For each tool, how could it be used for designing modified nanobodies?”

This agenda and set of agenda questions led the agents to select the following tools.

Selected Tools: AlphaFold2, Rosetta Suite, ProtBERT/BioBERT, DeepChem, GROMACS, HADDOCK, ESM (Evolutionary Scale Modeling), PyRosetta, AutoDock Vina

These 10 tools include a range of machine learning and computational tools for nanobody design and include pre-trained machine learning models (like ESM) as desired.

Prompt Variant 1

In the first variant of the prompt, we remove the following sentence from the agenda: “If selecting machine learning tools, please prioritize pre-trained models (e.g., pre-trained protein language models or protein structure prediction models) for simplicity.” Otherwise, the agenda and agenda questions are unchanged. In response, the agents select the following set of tools.

Selected Tools: Rosetta, TensorFlow/PyTorch, GROMACS, AutoDock Vina, DeepChem, PyMOL, Transfer Learning Models, REMD, SPR, NetMHC

Notably, this set of tools is less focused on pre-trained machine learning models. It lacks any pre-trained protein language models (e.g., ESM) or pretrained protein structure prediction tools (e.g., AlphaFold), and instead it includes machine learning frameworks (e.g., PyTorch) and wet lab experimental methods (e.g., SPR). This change indicates that the agents can be sensitive to the specific requests in the agenda, so if the human user has specific preferences (e.g., using a protein language model), then those preferences should be explicitly included in the agenda.

Prompt Variant 2

The second variant of the prompt includes the changes of the first variant and also removes the phrase “machine learning and/or computational” before the word “tools” in both the agenda and first agenda question. With this prompt, the agents select the following tools.

Selected Tools: Rosetta, AlphaFold, Tensorflow, GROMACS, HADDOCK, DeepChem, PyMOL, BepiPred, LabKey, Apache Airflow

Despite the removal of the words “machine learning” and “computational,” the agents still choose to suggest many machine learning and computational tools for nanobody design, showing robustness to different prompts. The agents understand the broader context that this is a computational research project, so they still suggest relevant machine learning and computational tools.

Prompt Variant 3

In addition to the changes in the second variant, the third prompt variant also removes the phrase “(5-10)” in the agenda and “(list 5-10)” in the first agenda question following the word “tools”. The agents respond with the following list of tools.

Selected Tools: Rosetta, AlphaFold, GROMACS, AutoDock Vina, TensorFlow, TAPE/BERT, Centralized Relational Database, BLI and ITC, ELISA and SPR, Pseudovirus Assays and In Vivo Models

This list includes 14 tools now that the prompt no longer restricts it to 5-10 tools. This change shows that the agents may have their own preferred form of response in general (e.g., more tools), but they are willing to follow human instructions (e.g., only show 5-10 tools) when requested.

Prompt Variant 4

This prompt is the same as the third variant but additionally removes the second agenda question asking for an explanation of how each tool could be used to design modified nanobodies. The selected tools are as follows.

Selected Tools: GROMACS and AMBER, AutoDock Vina and HADDOCK, AlphaFold and Rosetta, TensorFlow and PyTorch, DeepChem, PyMOL and Chimera, PostgreSQL, SPR and BLI, ELISA and Pseudovirus Neutralization Assays, Phage Display and PBMC Assays

More importantly, the agents still provide an explanation for each tool even though it is no longer requested. This is because the Virtual Lab framework contains several built-in prompts guiding the agent interactions, including an explicit request that the agents justify their actions.

Therefore, the agent discussions will be robust to prompt changes in cases where the Virtual Lab prompting framework already includes a relevant prompt.

Supplementary Note 5: Finetuning agents

The Virtual Lab agents possess extensive knowledge of biology and machine learning as demonstrated by their ability to design and build a computational nanobody design pipeline linking ESM, AlphaFold-Multimer, and Rosetta. However, the agents still have some gaps in their knowledge, particularly with respect to recent information that may have arisen after the LLM knowledge cutoff date. For example, while the Virtual Lab agents conceptually understand how to design nanobodies for new variants of SARS-CoV-2, they are unfamiliar with some of the specific recent variants—KP.3 and JN.1—explored in this work since those variants were not widespread at the time of the LLM knowledge cutoff (October 2023)^{69–71}. We aimed to address this recent knowledge problem and to generally enhance the abilities of our Virtual Lab agents by designing an LLM finetuning pipeline that allows the agents to select scientific papers that they want to finetune themselves on to enhance their knowledge of a particular topic.

Finetuning pipeline

The first step of the finetuning pipeline is to delineate a set of finetuning topics for each agent. The topics are as follows. Immunologist: “nanobodies”, “SARS-CoV-2 spike protein”, “SARS-CoV-2 variants KP.3 and JN.1”. Machine Learning Specialist: “ESM”. Computational Biologist: “AlphaFold-Multimer” and “Rosetta.” Next, for each topic, the relevant agent was asked to generate five search queries about that topic and its relation to designing nanobodies for SARS-CoV-2. Then, each agent was given access to the PubMed Central search API to run each query and retrieve 100 abstracts. The agent read each abstract to decide which papers to include in its finetuning set for that topic. For each of those selected papers, that same agent then read the full text of the paper and wrote a summary of the paper focused on the topic. These summaries were then aggregated by topic and converted into a format amenable to supervised finetuning, where the agent updates its parameters to increase the probability of generating the summary of the paper given just its title (as opposed to the full text).

To evaluate the impact of finetuning, the original agents were asked to write five multiple choice question and answer pairs for each paper summary, with one correct answer and three incorrect but plausible answers for each question. Then, for each topic, we asked either the original agents or the finetuned agents to answer up to 100 randomly selected questions per topic across all of the summaries for that topic (without seeing the paper summary itself).

Finetuning results

We compare the original agents to the finetuned agents using two mechanisms: a quantitative question and answer evaluation and a qualitative knowledge assessment. Both mechanisms show that the finetuned agents have gained some new knowledge, particularly regarding information that did not exist prior to the knowledge cutoff of the underlying LLM.

Q&A evaluation

We evaluate the original and finetuned agents on up to 100 multiple choice question and answer pairs for each of the six finetuning topics. For each topic, we choose the relevant original or finetuned agent and measure their accuracy on the questions. The results in Supplementary Table 1 show that for each underlying LLM—either GPT-4o-mini (gpt-4o-mini-2024-07-18) or GPT-4o (gpt-4o-2024-08-06)—finetuning almost always maintains or improves question answering accuracy. The improvements from original agent to finetuned agent are more noticeable for the GPT-4o-mini model since the performance of the original agent is less saturated, meaning there is more room for improvement via finetuning. However, notably, both LLMs gain the most when finetuned on the topic “SARS-CoV-2 variants KP.3 and JN.1,” most likely because information about KP.3 and JN.1 is entirely new to the LLM since those variants arose after the LLM’s knowledge cutoff. These results indicate the efficacy of finetuning to enhance the knowledge of scientific agents.

Supplementary Table 1 | Finetuning Q&A evaluation. The accuracy of original and finetuned agents powered by GPT-4o-mini or GPT-4o on up to 100 question-and-answer pairs based on summaries of papers from the six finetuning topics.

Topic	Original (GPT-4o-mini)	Finetuned (GPT-4o-mini)	Original (GPT-4o)	Finetuned (GPT-4o)
Nanobodies	83	89	90	90
SARS-CoV-2 spike protein	86	85	92	92
SARS-CoV-2 variants KP.3 and JN.1	66.67	75.56	76.67	83.33

ESM	89	92	92	93
AlphaFold-Multimer	86	89	92	93
Rosetta	80	83	85	87

Knowledge evaluation

As a qualitative measure of the knowledge gained via finetuning, we asked the original Immunologist and finetuned Immunologist (both using the GPT-4o-mini model) to answer a question regarding the KP.3 and JN.1 variants, which is where we expect the greatest benefit from finetuning. Specifically, we ask both models the following question: “What are some of the RBD mutations of the KP.3 and JN.1 variants of the SARS-CoV-2 spike protein?”

The original Immunologist responds with reasonable guesses of potential KP.3 and JN.1 mutations based on prior variants, but it does not have a definitive answer. For example, for KP.3, the agent suggests “The KP.3 variant may have mutations in the RBD that are similar to those seen in other variants of concern, such as E484K, N501Y, or L452R. These mutations are known to enhance binding affinity to the ACE2 receptor and may contribute to immune escape by reducing the neutralizing activity of antibodies generated by previous infection or vaccination.” In contrast, when the finetuned Immunologist is asked the same question, it responds with a list of 10 mutations for KP.3 and JN.1, including true KP.3 and/or JN.1 mutations such as L455S⁷², which is a new mutation in JN.1 compared to its predecessor variant BA.2.86. This shows that the finetuning procedure has improved the knowledge of the finetuned agent, and additional development of the finetuning pipeline could enhance this gain of knowledge even further.

Supplementary Note 6: Prompts

Below are the prompts used throughout the Virtual Lab. In the prompts below, instances of “<something>” are filled in automatically depending on the context (e.g., “<team lead>” would be filled in as “Principal Investigator” if the PI agent is used as the team lead in a meeting).

Virtual Lab architecture prompts

Principal Investigator

You are a Principal Investigator. Your expertise is in applying artificial intelligence to biomedical research. Your goal is to

perform research in your area of expertise that maximizes the scientific impact of the work. Your role is to lead a team of experts to solve an important problem in artificial intelligence for biomedicine, make key decisions about the project direction based on team member input, and manage the project timeline and resources.

Scientific Critic

You are a Scientific Critic. Your expertise is in providing critical feedback for scientific research. Your goal is to ensure that proposed research projects and implementations are rigorous, detailed, feasible, and scientifically sound. Your role is to provide critical feedback to identify and correct all errors and demand that scientific answers that are maximally complete and detailed but simple and not overly complex.

Team meeting start

This is the beginning of a team meeting to discuss your research project. This is a meeting with the team lead, <team lead>, and the following team members: <team members>.

Here are summaries of the previous meetings: <summaries>

Here is the agenda for the meeting: <agenda>

Here are the agenda questions that must be answered: <agenda questions>

Here are the agenda rules that must be answered: <agenda rules>

<team lead> will convene the meeting. Then, each team member will provide their thoughts on the discussion one-by-one in the order above. After all team members have given their input, <team lead> will synthesize the points raised by each team member, make decisions regarding the agenda based on team member input, and ask follow-up questions to gather more information and feedback about how to better address the agenda. This will continue for <number of rounds> rounds. Once the discussion is complete, <team lead> will summarize the meeting in detail for future discussions, provide a specific recommendation regarding the agenda, and answer the agenda questions (if any) based on the discussion while strictly adhering to the agenda rules (if any).

Team meeting team lead start

<team lead>, please provide your initial thoughts on the agenda as well as any questions you have to guide the discussion among the team members.

Team meeting team member response

<team member>, please provide your thoughts on the discussion (round <current round number> of <number of rounds>). If you do not have anything new or relevant to add, you may say "pass". Remember that you can and should (politely) disagree with other team members if you have a different perspective.

Team meeting team lead synthesis

This concludes round <current round number> of <number of rounds> of discussion. <team lead>, please synthesize the points raised by each team member, make decisions regarding the agenda based on team member input, and ask follow-up questions to gather more information and feedback about how to better address the agenda.

Team meeting team lead summary

<team lead>, please summarize the meeting in detail for future discussions, provide a specific recommendation regarding the agenda, and answer the agenda questions (if any) based on the discussion while strictly adhering to the agenda rules (if any).

As a reminder, here is the agenda for the meeting: <agenda>

As a reminder, here are the agenda questions that must be answered: <agenda questions>

As a reminder, here are the agenda rules that must be followed: <agenda rules>

Your summary should take the following form.

Agenda

Restate the agenda in your own words.

Team Member Input

Summarize all of the important points raised by each team member. This is to ensure that key details are preserved for future meetings.

Recommendation

Provide your expert recommendation regarding the agenda. You should consider the input from each team member, but you must also use your expertise to make a final decision and choose one option among several that may have been discussed. This decision can conflict with the input of some team members as long as it is well justified. It is essential that you provide a clear, specific, and actionable recommendation. Please justify your recommendation as well.

Answers

For each agenda question, please provide the following:

Answer: A specific answer to the question based on your recommendation above.

Justification: A brief explanation of why you provided that answer.

Next Steps

Outline the next steps that the team should take based on the discussion.

Individual meeting start

This is the beginning of an individual meeting with <team member> to discuss your research project.

Here are summaries of the previous meetings: <summaries>

Here is the agenda for the meeting: <agenda>

Here are the agenda questions that must be answered: <agenda questions>

Here are the agenda rules that must be answered: <agenda rules>

<team member>, please provide your response to the agenda.

Individual meeting agent response

<agent>, please modify your answer to address <critic>'s most recent feedback. Remember that your ultimate goal is to make improvements that better address the agenda.

Individual meeting critic response

<critic>, please critique <agent>'s most recent answer. In your critique, suggest improvements that directly address the agenda and any agenda questions. Prioritize simple solutions over unnecessarily complex ones, but demand more detail where detail is lacking. Additionally, validate whether the answer strictly adheres to the agenda and any agenda questions and provide corrective feedback if it does not. Only provide feedback; do not implement the answer yourself.

Parallel meeting merge

Please read the summaries of multiple separate meetings about the same agenda. Based on the summaries, provide a single answer that merges the best components of each individual answer. Please use the same format as the individual answers. Additionally, please explain what components of your answer came from each individual answer and why you chose to include them in your answer.

As a reference, here is the agenda from those meetings, which must be addressed here as well: <agenda>

As a reference, here are the agenda questions from those meetings, which must be answered here as well: <agenda questions>

As a reference, here are the agenda rules from those meetings, which must be followed here as well: <agenda rules>

Coding rules

1. Your code must be self-contained (with appropriate imports) and complete.
2. Your code may not include any undefined or unimplemented variables or functions.
3. Your code may not include any pseudocode; it must be fully functioning code.
4. Your code may not include any hard-coded examples.
5. If your code needs user-provided values, write code to parse those values from the command line.
6. Your code must be high quality, well-engineered, efficient, and well-documented (including docstrings, comments, and Python type hints if using Python).

Nanobody design prompts

Team selection

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. You need to select a team of three scientists to help you with this project. Please select the team members that you would like to invite to a discussion to create the antibody/nanobody design approach. Please list the team members in the following format, using the team member below as an example. You should not include yourself (Principal Investigator) in the list.

Agent (

 title="Principal Investigator",

 expertise="applying artificial intelligence to biomedical research",

 goal="perform research in your area of expertise that maximizes the scientific impact of the work",

 role="lead a team of experts to solve an important problem in artificial intelligence for biomedicine, make key decisions about the project direction based on team member input, and manage the project timeline and resources",

)

Immunologist (written by the PI agent)

You are a Immunologist. Your expertise is in antibody engineering and immune response characterization. Your goal is to guide the development of antibodies/nanobodies that elicit a strong and broad immune response. Your role is to advise on immunogenicity, cross-reactivity with other variants, and potential for therapeutic application, ensuring the designs are viable for experimental validation and downstream applications.

Machine Learning Specialist (written by the PI agent)

You are a Machine Learning Specialist. Your expertise is in developing algorithms for protein-ligand interactions and optimization. Your goal is to create and apply machine learning models to predict antibody

efficacy and optimize binding affinity across SARS-CoV-2 variants. Your role is to lead the development of AI tools for predicting interactions and refining antibody designs based on computational results.

Computational Biologist (written by the PI agent)

You are a Computational Biologist. Your expertise is in protein structure prediction and molecular dynamics simulations. Your goal is to develop predictive models to identify potential antibody/nanobody candidates and simulate interactions with the SARS-CoV-2 spike protein. Your role is to provide insights into structural dynamics, guide virtual screening efforts, and validate computational predictions with simulations.

Project specification

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Please create an antibody/nanobody design approach to solve this problem. Decide whether you will design antibodies or nanobodies. For your choice, decide whether you will design the antibodies/nanobodies de novo or whether you will modify existing antibodies/nanobodies. If modifying existing antibodies/nanobodies, please specify which antibodies/nanobodies to start with as good candidates for targeting the newest variant of the SARS-CoV-2 spike protein. If designing antibodies/nanobodies de novo, please describe how you will propose antibody/nanobody candidates.

1. Will you design standard antibodies or nanobodies?
2. Will you design antibodies/nanobodies de novo or will you modify existing antibodies/nanobodies (choose only one)?
3. If modifying existing antibodies/nanobodies, which precise antibodies/nanobodies will you modify (please list 3-4)?
4. If designing antibodies/nanobodies de novo, how exactly will you propose antibody/nanobody candidates?

Tools selection

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other

circulating minor variants and past variants. Your team previous decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. Now you need to select machine learning and/or computational tools to implement this nanobody design approach. Please list several tools (5-10) that would be relevant to this nanobody design approach and how they could be used in the context of this project. If selecting machine learning tools, please prioritize pre-trained models (e.g., pre-trained protein language models or protein structure prediction models) for simplicity.

1. What machine learning and/or computational tools could be used for this nanobody design approach (list 5-10)?
2. For each tool, how could it be used for designing modified nanobodies?

Tools implementation

Implementation agent selection

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previous decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. Your team needs to build three components of a nanobody design pipeline: ESM, AlphaFold-Multimer, and Rosetta. For each component, please select the team member who will implement the component. A team member may implement more than one component.

1. Which team member will implement ESM?
2. Which team member will implement AlphaFold-Multimer?
3. Which team member will implement Rosetta?

ESM implementation

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previous decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. Now you must use ESM to suggest modifications to an existing antibody. Please write a

complete Python script that takes a nanobody sequence as input and uses ESM amino acid log-likelihoods to identify the most promising point mutations by log-likelihood ratio.

ESM improvement

You previously wrote a Python script that uses ESM to compute the log-likelihood ratio of point mutations in a nanobody sequence (see summary). This script needs to be improved. Please rewrite the script to make the following improvements without changing anything else.

1. Replace "facebook/esmlb-t33_650M_UR50S" with "facebook/esmlb_t33_650M_UR50S".
2. Run the calculations of the mutant log-likelihoods by iterating through the sequences in batches of 16.
3. Add a progress bar to the batched mutant log-likelihood calculations.
4. Run the mutant log-likelihood calculations on CUDA but with no gradients.
5. Load the nanobody sequence from a user-specified CSV file that has the columns "sequence" and "name". Adapt your code to run the mutant log-likelihood calculations on all sequences in the CSV file one-by-one.
6. For each sequence, save the mutant log-likelihoods to a CSV file with the format "mutated_sequence,position,original_aa,mutated_aa,log_likelihood_ratio". Ask the user for a save directory and then save this CSV file in that directory with the name: <nanobody-name>.csv.

AlphaFold-Multimer implementation

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previously decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. Now you must use AlphaFold-Multimer to predict the structure of a nanobody-antigen complex and evaluate its binding. I will run AlphaFold-Multimer on several nanobody-antigen complexes and you need to process the outputs. Please write a complete Python script that takes as input a directory containing PDB files where each PDB file contains one nanobody-antigen complex predicted by AlphaFold-Multimer and outputs a

CSV file containing the AlphaFold-Multimer confidence of each nanobody-antigen complex in terms of the interface pLDDT.

AlphaFold-Multimer improvement

You previously wrote a Python script that processes the outputs of AlphaFold-Multimer to calculate the confidence of nanobody-antigen complexes (see summary). This script needs to be improved. Please rewrite the script to make the following improvements without changing anything else.

1. Replace the current imports of Chain and Residue with "from Bio.PDB.Chain import Chain" and "from Bio.PDB.Residue import Residue".
2. Remove the logging setup and simply print any log messages to the console.
3. Replace the parallel processing with sequential processing to avoid getting an "OSError: Too many open files".
4. Change the list of pdb_files to instead get all PDB files in the directory that follow the pattern "**/*unrelaxed_rank_001*.pdb".
5. Change the calculation of average pLDDT to divide by the number of atoms rather than the number of residues.
6. Return and save in the CSV both the number of residues and the number of atoms in the interface.
7. Change the default distance threshold to 4.

Rosetta implementation

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previously decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. Now you must use Rosetta to calculate the binding energy of nanobody-antigen complexes. You must do this in three parts. First, write a complete RosettaScripts XML file that calculates the binding energy of a nanobody-antigen complex as provided in PDB format, including any necessary preprocessing steps for the complex. Second, write an example command that uses Rosetta to run this RosettaScripts XML file on a given PDB file to calculate the binding energy and save it to a score file. Third, write a complete Python script that takes as input

a directory with multiple Rosetta binding energy score files and outputs a single CSV file with the names and scores of each of the individual files in sorted order (highest to lowest score).

Rosetta XML improvement

You previously wrote a RosettaScripts XML file to calculate the binding affinity of a nanobody-antigen complex (see summary). This script needs to be improved. Please rewrite the script to make the following improvements without changing anything else.

1. Replace "ref15.wts" with "ref2015.wts".
2. Remove the InterfaceEnergy filter since it is not valid in Rosetta.
3. Replace the entire output tag (including any nested tags) with `<OUTPUT scorefxn="ref15"/>`.

Rosetta Python improvement

You previously wrote a Python script to aggregate multiple Rosetta binding energy score files into one CSV file (see summary). This script needs to be improved. Please rewrite the script to make the following improvements without changing anything else.

1. Modify the `extract_scores_from_file` function so that it extracts the dG-separated value from a file of the following form.

SEQUENCE:

```
SCORE: total_score complex_normalized dG_cross
dG_cross/dSASAx100 dG_separated dG_separated/dSASAx100 dSASA_hphobic
dSASA_int dSASA_polar delta_unsatHbonds dslf_fa13 fa_atr fa_dun
fa_elec fa_intra_rep fa_intra_sol_xover4 fa_rep
fa_sol hbond_E_fraction hbond_bb_sc hbond_lr_bb hbond_sc
hbond_sr_bb hbonds_int lk_ball_wtd nres_all nres_int omega
p_aa_pp packstat per_residue_energy_int pro_close rama_prepro
ref sc_value sidel_normalized sidel_score side2_normalized
side2_score yhh_planarity description
```

```
SCORE: -990.807 -2.914 -21.436 -
1.857 -21.436 -1.857 774.274 1154.088
379.813 12.000 -3.867 -1928.622 376.416 -541.777
3.745 54.944 265.303 1052.322
0.053 -84.023 -130.532 -54.069 -46.266 1.000 -
41.725 340.000 55.000 39.977 -81.331 0.000
-2.699 2.349 -6.870 131.513 0.000 -2.236
```

-51.431 -3.031 -97.008 1.706 KP3_Ty1-
G59Y_unrelaxed_rank_001_alphafold2_multimer_v3_model_3_seed_000_0001

Workflow design

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previously decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. Your team has built three components of a nanobody design pipeline: ESM, AlphaFold-Multimer, and Rosetta. Each of these three tools can be used to score a nanobody mutation based on how the mutation affects binding to an antigen. Your goal is to start with an existing nanobody and iteratively add mutations to it to improve its binding to the newest variant of the SARS-CoV-2 spike protein, resulting in 24 modified nanobodies with one or more mutations. Please determine how to use ESM, AlphaFold-Multimer, and Rosetta in this iterative design process. An important constraint is that ESM can evaluate all potential mutations to a nanobody in 5 minutes while AlphaFold-Multimer takes 30 minutes per mutation and Rosetta takes five minutes per mutation. The whole iterative process should take no more than a few days to complete. Note that AlphaFold-Multimer must be run before Rosetta on each mutation since Rosetta requires the nanobody-antigen structure predicted by AlphaFold-Multimer. Additionally, note that ESM log-likelihood ratios are generally in the range of 5-10 (higher is better), AlphaFold-Multimer interface pLDDT confidence scores are generally in the range of 60-80 (higher is better), and Rosetta binding energy scores are generally in the range of -20 to -40 (lower is better).

1. In each iteration, what is the order of operations for evaluating mutations with ESM, AlphaFold-Multimer, and Rosetta?
2. In each iteration, how many mutations (give a single number) will you evaluate with ESM, AlphaFold-Multimer, and Rosetta?
3. At the end of each iteration, how will you weigh the ESM, AlphaFold-Multimer, and/or Rosetta scores (give a formula) to rank the nanobody mutations?
4. At the end of each iteration, how many of the top-ranked mutations (give a single number) will you keep for the next round?
5. How will you decide how many iterations of mutations to run?

6. After all of the iterations are complete, how exactly (step-by-step) will you select the final set of 24 modified nanobodies from across the iterations for experimental validation?

Finetuning pipeline prompts

Query generation

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previous decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. You are responsible for understanding the topic "<topic>" in the context of designing nanobody binders for SARS-CoV-2. You need to fine-tune yourself on the relevant literature on <topic> to improve your ability to design SARS-CoV-2 nanobody binders. Please write out a series of five distinct search queries that you want to run to find relevant scientific papers on <topic>. Include both queries about <topic> generally as well as queries about how <topic> relates to designing nanobody binders for SARS-CoV-2. Please provide the queries in Python syntax as a list of double-quoted strings.

1. What are the queries that you want to perform to identify the relevant literature on <topic> (as a list of double-quoted strings in Python syntax)?

Paper selection

You are working on a research project to use machine learning to develop antibodies or nanobodies for the newest variant of the SARS-CoV-2 spike protein that also, ideally, have activity against other circulating minor variants and past variants. Your team previous decided to modify existing nanobodies to improve their binding to the newest variant of the SARS-CoV-2 spike protein. You are responsible for understanding the topic "<topic>" in the context of designing nanobody binders for SARS-CoV-2. You need to fine-tune yourself on the relevant literature on <topic> to improve your ability to design SARS-CoV-2 nanobody binders. Please use PubMed Central and search for relevant papers on <topic> using the query "<query>" and request 100 articles with abstracts only. Read all of the abstracts and based on each abstract individually, decide whether you want to fine-tune yourself on the full text of that paper. Include as many papers as possible, but only include papers that are directly relevant to

<topic>. Please provide the PMCID and titles of all the papers that you wish to fine-tune yourself on as a Python dictionary mapping PMCID as a double-quoted string to title as a double-quoted string.

1. What are the PMCID and titles of the papers you wish to fine-tune yourself on (as a Python dictionary mapping PMCID as a double-quoted string to title as double-quoted string)?

Paper summarization

Please summarize in extreme detail the following paper titled "<title>". Please focus in particular on summarizing key insights about the topic "<topic>" in relation to designing SARS-CoV-2 nanobody binders. <paper full text>

Finetuning

Please tell me about the paper "<title>" and its insights into "<topic>" in relation to designing SARS-CoV-2 nanobody binders.

Q&A generation

Please generate five unique questions based on the following summary of a scientific paper. The questions should be designed so that they are specific to the summary rather than general knowledge, i.e., they can only be answered with information in the summary. However, the questions must include enough context so that they can be answered without knowing which specific paper is being referred to (i.e., do not say things like "in this study" or "in the paper"). For the answers, please write one correct answer and three incorrect answers for each question. Make the incorrect answers as plausible and as similar to the correct answer as possible while still being incorrect so that selecting the correct answer is challenging without specific knowledge of the paper summary. Your response should *only* be a JSON dictionary mapping from question (string) to answers (list of four strings where the first string is the correct answer and the other three are incorrect). Here is the paper summary: <summary>

Q&A answering

Please answer the following question based on the following list of answers. Only one answer is correct. Please select the correct answer and provide as your response the exact text of the answer and nothing else. Here is the question: <question> Here are the possible answers (choose the correct one): <answers>

References

(numbering starts from the end of the references in the main text)

64. Hie, B. L. *et al.* Efficient evolution of human antibodies from general protein language models. *Nat. Biotechnol.* **42**, 275–283 (2024).
65. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci.* **118**, e2016239118 (2021).
66. Meier, J. *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. in *Advances in Neural Information Processing Systems* vol. 34 29287–29303 (Curran Associates, Inc., 2021).
67. Du, Y., Li, S., Torralba, A., Tenenbaum, J. B. & Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. in *Proceedings of the 41st International Conference on Machine Learning* vol. 235 11733–11763 (JMLR.org, Vienna, Austria, 2024).
68. Wang, J., Wang, J., Athiwaratkun, B., Zhang, C. & Zou, J. Mixture-of-Agents Enhances Large Language Model Capabilities. in *The Thirteenth International Conference on Learning Representations* (2024).
69. Kaku, Y. *et al.* Virological characteristics of the SARS-CoV-2 KP.3, LB.1, and KP.2.3 variants. *Lancet Infect. Dis.* **24**, e482–e483 (2024).
70. Kaku, Y. *et al.* Virological characteristics of the SARS-CoV-2 JN.1 variant. *Lancet Infect. Dis.* **24**, e82 (2024).
71. OpenAI *et al.* GPT-4o System Card. Preprint at <https://doi.org/10.48550/arXiv.2410.21276> (2024).
72. Yang, S. *et al.* Fast evolution of SARS-CoV-2 BA.2.86 to JN.1 under heavy immune pressure. *Lancet Infect. Dis.* **24**, e70–e72 (2024).