nature machine intelligence

ARTICLES

https://doi.org/10.1038/s42256-021-00327-w

Check for updates

Machine-learning-based dynamic-importance sampling for adaptive multiscale simulations

Harsh Bhatia[1]✉, Timothy S. Carpenter[2], Helgi I. Ingólfsson[2], Gautham Dharuman[2], Piyush Karande[3], Shusen Liu[1], Tomas Oppelstrup[2], Chris Neale[4], Felice C. Lightstone[2], Brian Van Essen[1], James N. Glosli[2] and Peer-Timo Bremer[1]

Multiscale simulations are a well-accepted way to bridge the length and time scales required for scientific studies with the solution accuracy achievable through available computational resources. Traditional approaches either solve a coarse model with selective refinement or coerce a detailed model into faster sampling, both of which have limitations. Here, we present a paradigm of adaptive, multiscale simulations that couple different scales using a dynamic-importance sampling approach. Our method uses machine learning to dynamically and exhaustively sample the phase space explored by a macro model using microscale simulations and enables an automatic feedback from the micro to the macro scale, leading to a self-healing multiscale simulation. As a result, our approach delivers macro length and time scales, but with the effective precision of the micro scale. Our approach is arbitrarily scalable as well as transferable to many different types of simulations. Our method made possible a multiscale scientific campaign of unprecedented scale to understand the interactions of RAS proteins with a plasma membrane in the context of cancer research running over several days on Sierra, which is currently the second-most-powerful supercomputer in the world.

Understanding the nature of many scientific phenomena, natural or artificial, requires exploration across a wide range of spatial and temporal scales. Even with ever-increasing computing power, computational models and simulations struggle to cover all scales of interest at sufficient resolution. Therefore, multiscale modelling is often used to gain scientific insights by investing computational resources appropriately across scales. However, the same properties that make multiscale simulations desirable (for example, the different levels of detail and sizes of simulated systems) also pose considerable challenges. Foremost among these challenges is how to couple the different scales together[1–4].

Here, we present a new style of multiscale simulation that uses a macroscale (such as dynamic density functional theory, DDFT) model to create hundreds of thousands of microscale (such as molecular dynamics, MD) simulations through a dynamic-importance (DynIm) sampling approach based on machine learning (ML). Our framework offers two notable contributions to the field of large multiscale simulations. First, our sampling framework is designed to maximize multiscale interrogation by connecting each macro configuration to a micro simulation that is sufficiently similar to serve as its statistical proxy. As a result, the multiscale simulation enables the exploration of macro length and time scales, but with the effective precision of the microscale model. Second, the dynamic nature of our sampling automates a feedback process in which microscale data is used to improve the parameterization of the macro model while the simulation is running. As a result, our framework represents a self-healing paradigm in multiscale simulations.

Our DynIm sampling approach is designed to explore the phase space of possible local configurations and compute statistical measures on the distribution of this phase space. A key distinguishing characteristic of our approach is the use of ML to dynamically sample new configurations. Our framework rapidly explores macro configurations as they are generated and uses ML to identify those for which more-detailed exposition via microscale simulation is most important, with respect to a predefined scientific hypothesis. The specific usage put forward here focuses on enriching the sampling of rare macromolecular compositions by directing microscale simulations to explore observed macro configurations as uniformly as possible. To this end, we consider the 'novelty' of a configuration to be indicative of its importance and preferentially select novel macro configurations for microscale interrogation. This approach results in an importance distribution (the distribution of important samples) that is wider (with better exploration of the phase space of macro configurations) and flatter (preventing similar configurations) than an unbiased, random selection of macro configurations.

The idea of importance sampling[5] is often used in statistics and for Monte Carlo sampling approaches[6] to estimate the properties of a given 'true' distribution from a set of samples taken from from a different 'importance' distribution through appropriate weighting. However, there are several key differences between DynIm and the standard importance sampling techniques. Most importantly, (standard) importance sampling is designed to minimize the variance in the statistics estimated from the importance distribution. In contrast, the goal of our importance sampling is the opposite: we aim to explore data points as dissimilar as possible. Furthermore, standard approaches typically need to know the importance distribution and/or the size of the sample set a priori. Dynamically weighted sampling[7] differs from traditional importance sampling in that the importance weights come from a known distribution. Such ideas have been explored to sample spatial models[8,9], to improve the deep learning training process[10,11], and so on. Compared to such techniques, our framework removes the need to predefine

[1]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA. [2]Physical and Life Sciences, Lawrence Livermore National Laboratory, Livermore, CA, USA. [3]Computational Engineering, Lawrence Livermore National Laboratory, Livermore, CA, USA. [4]Theoretical Biology and Biophysics, Los Alamos National Laboratory, Los Alamos, NM, USA. ✉e-mail: hbhatia@llnl.gov

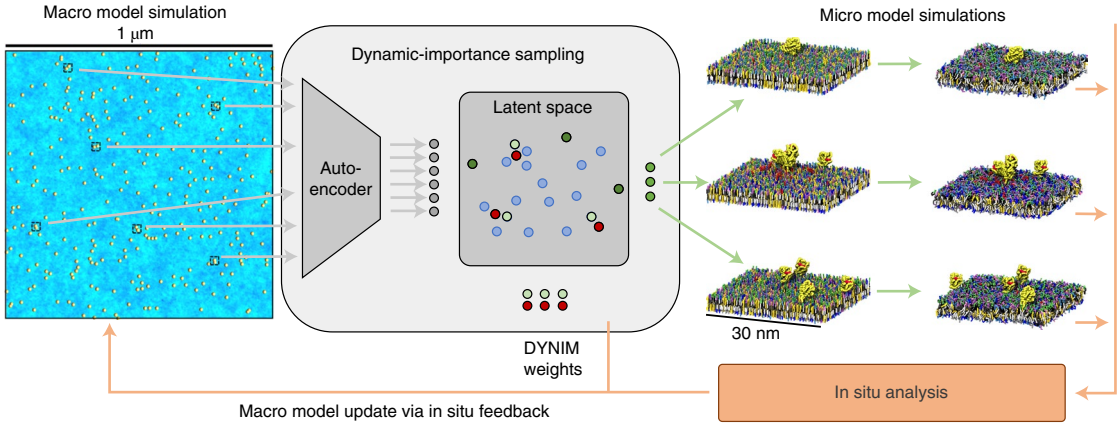

Fig. 1 | ML-based DynIm sampling framework. Our ML-based DynIm sampling framework enables a new paradigm of multiscale simulations that explore the phase space of a macro model (such as DDFT) as uniformly as possible using micro scale (such as MD) simulations. Using a variational autoencoder (VAE) to project the data onto a latent space, DynIm provides a mathematically relevant importance metric and a computationally feasible and scalable solution to massive multiscale simulations. DynIm is designed to allow reconstruction of the true data distribution using a small number of MD samples and can be used to create a self-healing feedback loop to improve the parameters of the macro model using insights from microscale simulation data.

importance weights or the sizes and types of distributions, lending more flexibility to the sampling process. Instead, we use ML to dynamically evaluate arbitrary data distributions based only on previously sampled data, correspondingly draw new samples on demand, and create appropriate importance weights.

Our framework is highly scalable and adaptable to the size and availability of computational resources. As a result, it can provide arbitrary adaptivity and refinement to the resulting multiscale simulation and can be used effectively on machines with limited capacity and/or systems with limited data. Furthermore, the dynamic (re)assignment of importance allows for the reconstruction of the true distribution of the phase space of all macro configurations and any statistics thereof at any given time during the simulation. The latter provides a means of aggregating the insights from ongoing microscale simulations to facilitate an in situ feedback that updates the parameters of the macro model, continuously driving the multiscale simulation towards ever-increasing accuracy. We note that the importance of each microscale simulation may also be recomputed with respect to an extended and/or refined macro model in a post hoc manner, which can be useful for even larger exploration.

We apply DynIm sampling to enable a large multiscale simulation that explores the interactions between RAS proteins and the compositionally dynamic plasma membrane. RAS is part of the signalling chain for cell growth, and mutations in RAS are implicated in nearly a third of all cancers diagnosed in the USA[12,13]. The ultimate goal of this case study is to understand how RAS–lipid interactions and RAS multimerization moderate cancer initiation pathways[14] (used here as a case study). To fully sample the vast array of potential lipid environments that RAS can experience, a macroscale simulation is required. However, the specific interactions between RAS and the lipids, and the behaviour of RAS within those lipid environments necessitate microscale simulation detail. Thus, this system represents an ideal use case. Though beyond the scope of this paper, the characterization of the RAS–lipid interactions and RAS multimerization mechanisms may reveal potential drug targets.

Given a macro (DDFT) model that explores large spatial and temporal scales (microseconds and micrometres) to explore plasma membrane compositions, we use DynIm with unsupervised ML to encode local neighbourhoods of RAS into a reduced-dimensional latent space, which captures the complex response of the different types of lipids present in the plasma membrane bilayer to the presence of RAS molecule(s) as well as the lipid–lipid dynamics. DynIm evaluates the importance of different lipid configurations, interpreted based on their dissimilarity to previous selections. When computational resources are made available, DynIm selects the most important candidates, which are then simulated at the micro scale (near-atomistic resolution using coarse-grained (CG) Martini[15] MD).

This scientific campaign was made possible by integrating our DynIm sampling framework as part of the Multiscale Machine-Learned Modeling Infrastructure (MuMMI), which provides a scalable hardware/software infrastructure[16] and appropriate scientific modelling at the macro and micro scales[14]. This paper describes our innovations within a sampling framework using the unsupervised ML that enabled the MuMMI technology. The resulting multiscale simulation was run on the Sierra supercomputer[17], where MuMMI and DynIm efficiently utilized the full machine, with thousands of central processing units and graphics processing units (CPUs and GPUs) to conduct 119,686 MD simulations (see Fig. 1 in Ingólfsson et al.[14]), aggregating over 200 ms of RAS–lipid interactions at the micro scale. Utilizing the DynIm framework described here, a diverse sampling of local lipid environments around RAS has been archived; this could be used to characterize lipid compositions that drive increased/decreased RAS multimerization, RAS lipid fingerprints for the different RAS membrane states, and the main lipid driving forces for changing state dynamics[14].

Results

We demonstrate our DynIm sampling (see Fig. 1) and its key characteristics using the abovementioned case study of the multiscale simulation of RAS–lipid biology as well as additional experiments on synthetic data.

Dynamic-importance sampling. This paper introduces a dynamic-importance sampling approach that enables a new genre of multiscale simulations to explore the phase space as uniformly as possible. Since the true distribution is high-dimensional and, very likely, multimodal, sampling configurations from the macro model randomly will lead to redundant sampling of common configurations and generally ignore infrequent ones, which may yet be functionally important. Furthermore, even using the macro model, only relatively short timescales (as compared to biological timescales, such as the lifetime of a cell) may be sampled; therefore, rare configurations in the simulation could be crucial in answering critical scientific questions. Our ML-driven approach is designed to avoid

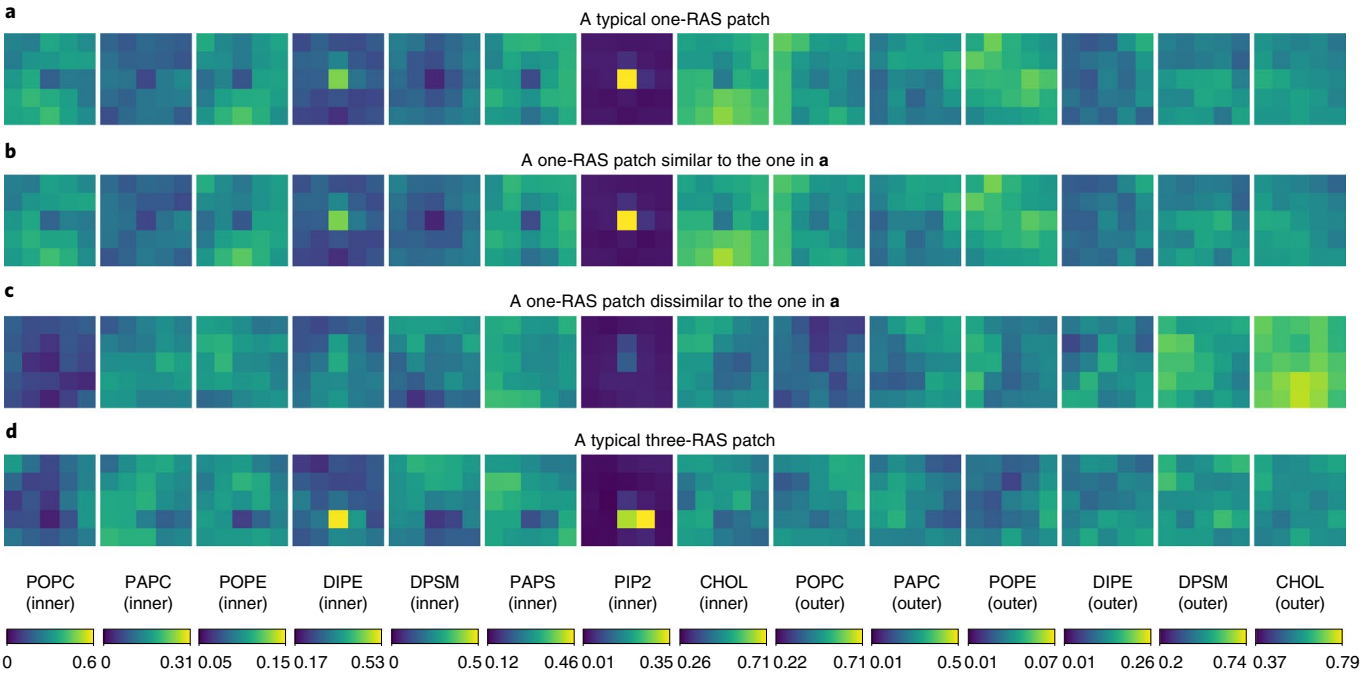

Fig. 2 | Patches represent concentrations of lipids on 30×30 nm^2 local neighbourhoods of RAS discretized into 5×5 grids. a–c, The figure shows a typical patch containing one RAS (**a**) and highlights the lipids that respond strongly to the presence of RAS (the centre pixel). Using the distance metric defined in the latent space, the figure identifies patches similar to **b** and dissimilar to **c** one RAS. **d**, A different configuration where the patch contains three RAS (one at the centre pixel and two below that). The plasma membrane mimic uses 8 different lipid types[33], with all eight on the inner leaflet and six on the outer leaflet. These eight lipid types include two phosphocholines (POPC and PAPC), two phosphoethanolamines (PE), one sphingomyelin (DPSM), one phosphatidylserine (PAPS), one phosphatidylinositol (PIP2), and cholesterol (CHOL).

redundant sampling of the phase space, and therefore, favours new and previously unseen types of configurations.

In particular, our framework keeps a record of all previously explored configurations (previously selected samples) and selects the next configuration (the next sample) as the most dissimilar one. Alternatively, this can be seen as a 'novelty' importance metric that ranks all samples according to how similar they are to those already sampled.

Encoding of macro configurations. Local macro configurations are described as 'patches', with each patch representing 14 lipid concentrations (8 lipids on the inner and 6 on the outer leaflet) on a 30×30 nm^2 region in the plasma membrane plane, discretized as a 5×5 subgrid (see Fig. 2). This 350-dimensional pixel space ($5\times5\times14$) is challenging to sample because comparing two patches directly in the pixel space, such as by using an ℓ_2 metric, is neither mathematically desirable nor computationally viable. First, such per-pixel distance metrics are not meaningful because they treat each pixel uniformly, and fail to account for any correlations across space and different lipid types. Second, even more-sophisticated distance metrics that directly compare patches are confronted with major challenges because high dimensionality leads to sparseness in the distribution. Furthermore, different lipid types exhibit different concentration ranges. Therefore, disallowing the lipid species with large concentrations from dominating the overall distance metric would require appropriate weighting. Finally, the high computational cost of computing millions of distances in 350-dimensional space prevents real-time selection by direct approaches.

To address these limitations, we use a VAE[18] to encode patches into a reduced, 15-dimensional latent representation that captures the intrinsic dimensionality of the data and focuses on the complex, nonlinear relationships within lipid configurations, while discarding inherent correlations. This encoding provides two key advantages: a meaningful distance metric that captures key correlations and variations in data, and greatly reduced dimensionality. By design, the VAE maps patches with similar lipid configurations to nearby regions in the latent space. Therefore, the Euclidean distance between two patches in the latent space is used as an indicator to quantify similarity (see Fig. 2).

Sampling of macro configurations. Given millions of potentially interesting candidate patches, that is, the configurations that have been explored by the macro model but not yet selected to simulate at the micro scale, the next step is to sample the most important ones. To this end, DynIm uses a farthest-sampling approach in the space of all previously sampled configurations.

Using an efficient data structure that supports fast, almost-real-time evaluations[19,20], we track the DynIm score of all candidate patches. The DynIm score identifies how similar a candidate is to previously sampled ones; it is defined as the mean distance of the candidate to its k-nearest neighbours (we use $k=10$) in the set of previously sampled data in the latent space. Any time the application demands a new sample (for example, when a new computational resource becomes available), the top-ranked candidate is selected. All tracking information and data structures are updated correspondingly.

If the macro model simulation creates a configuration that is consequentially different from previously selected ones, we expect its DynIm score to be high. The selection of a patch, on the other hand, reduces the DynIm scores of all its similar patches. In this way, DynIm scores continuously guide the sampling process towards new configurations, reducing redundancy. We note that DynIm, by design, mitigates temporal correlations in the data arising from fine temporal sampling and, therefore, alleviates the user's burden

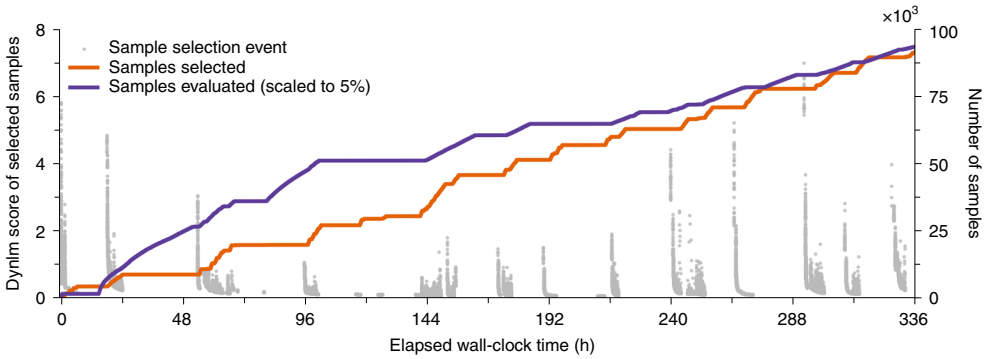

Fig. 3 | Event history of our ML-based DynIm sampling for a period of 14 days of wall-clock time. Each grey dot is a selection event with the score of the selected candidate mapped to the left axis. The line plots show the cumulative number of patches evaluated and selected (mapped to the right axis). Our framework can support 'bursts' of selection requests (such as when computational resources become available) and is designed to balance two competing goals: to uniformly sample the available phase space (indicated by the reduction of scores within each burst) and to steer the simulation to explore the phase space (indicated by sharp peaks for each selection burst).

to identify a suitable temporal resolution, such as the output rate of macro model frames.

To enable the multiscale simulation under discussion, DynIm dynamically selected 119,686 patches out of a total of 2,061,900 evaluated candidates of potential interest. Figure 3 visualizes a partial event history of our multiscale simulation with a total wall-clock time of about 14 days, which saw the selection of 92,242 patches out of 1,880,400 evaluated ones. Each dot in the plot represents a selection event, with the DynIm score of the selected patch mapped to the left axis. Selection events usually occur in chunks because MD simulations that are started together often conclude within short intervals of each other (MuMMI's MD simulations perform consistently with little variability, $1.02 \pm 0.002\,\mu s$ per day[16,21]), making the resources available for the next batch of MD simulations. Within each chunk of selection, the score of patches reduces progressively, consistent with the design intent of our selection algorithm. We note, however, that between peaks, the score of selected patches can increase. For about the first half of the simulation (about 168 h in the plot), an overall reduction in the height of such peaks is observed, indicating the exploration of new data evaluated between consecutive peaks. However, the second half of the simulation, where the self-healing, feedback mechanism (discussed in section 'Self-healing through in situ feedback') was used, sees a rapid increase in the heights of the peaks, reflecting the increased novelty of the forthcoming configurations. Figure 3 demonstrates the balancing of the two competing goals in our framework: the DynIm sampling is designed to exhaustively search the available phase space of macro lipid configurations, whereas increased sampling of the macro model coupled with the self-healing mechanism may create patches in previously unseen portions of the phase space.

Figure 3 also shows the time history of the number of patches evaluated as well as the number of patches selected by our framework mapped to the right axis of the plot. The plots also highlight the dynamic nature of our selection process: at any instance in time, our framework picks the most important patches among the ones seen thus far and the rate of selection is customizable and scalable. This dynamic nature distinguishes our work from post hoc importance sampling techniques, which may consider all data a priori to evaluate importance.

Our ML-based DynIm selection procedure is well suited to uniform sampling of the available phase space. We compare the sampling quality produced by our approach against that obtained through a random sampling of patches dynamically from 2,061,900 candidates (using a proxy resource allocation approach to simulate the dynamics of random selection). To visually compare the density of all patches and the densities of sampled patches, Fig. 4 shows five pairs of two-dimensional marginal densities computed in the 15-dimensional latent coordinates. As can be seen in Fig. 4, in all five cases, the random selection mimics the overall distribution of all patches, whereas DynIm sampling produces flatter and wider distributions, implying a more uniform coverage of the phase space.

To further demonstrate the superiority of the DynIm sampling, Fig. 4 also uses high-dimensional visualization techniques to compare the resulting distributions. Figure 4 shows parallel coordinate plots, which visualize the value distribution of all dimensions simultaneously. As noted from the density in the parallel coordinates, all patches and the random subset capture a similar overall distribution, whereas the DynIm sampling has a wider and more uniform density distribution compared to the random selection in every dimension (vertical lines in the plot). Although parallel coordinates are suitable to highlight the overall density pattern, other properties, such as the smoothness of the distribution, can be better demonstrated using topological data analysis. Through topological analysis, we segment the high-dimensional domain with respect to the local extrema of the density and present additional details on the distribution. Figure 4 uses topological spines visual encoding[22], which encodes the high-dimensional topology as a two-dimensional graph, with nodes representing local maxima. The visualization indicates that the random subset has a less complex (with less stable local extrema) density function compared to all patches, whereas the DynIm selection has fewer, flatter extrema. A detailed discussion on the high-dimensional visualization used in Fig. 4 was presented by Liu et al.[23].

Reconstruction of the true distribution. Since our DynIm framework is specifically designed to suppress redundant configurations in favour of less frequent ones, the resulting selection is biased against the modes in the original distribution. As such, recovering the true distribution or computing statistical measures on the sampled dataset requires debiasing through appropriate weighting.

The DynIm weight for each selected sample is defined using a reverse nearest neighbour (RNN)[24] approach in the space of all discarded samples. Samples with high weights are representative of the regions of high density in the latent space; the density in these regions was suppressed by selecting the representative patches and avoiding selection of similar ones. Correspondingly, the large number of patches with small weights indicates that our selection spans the 'flat regions' in the distribution. Finally, we refer back to Fig. 4, where the last row demonstrates that DynIm weights applied on the DynIm selection can recover the true distribution.

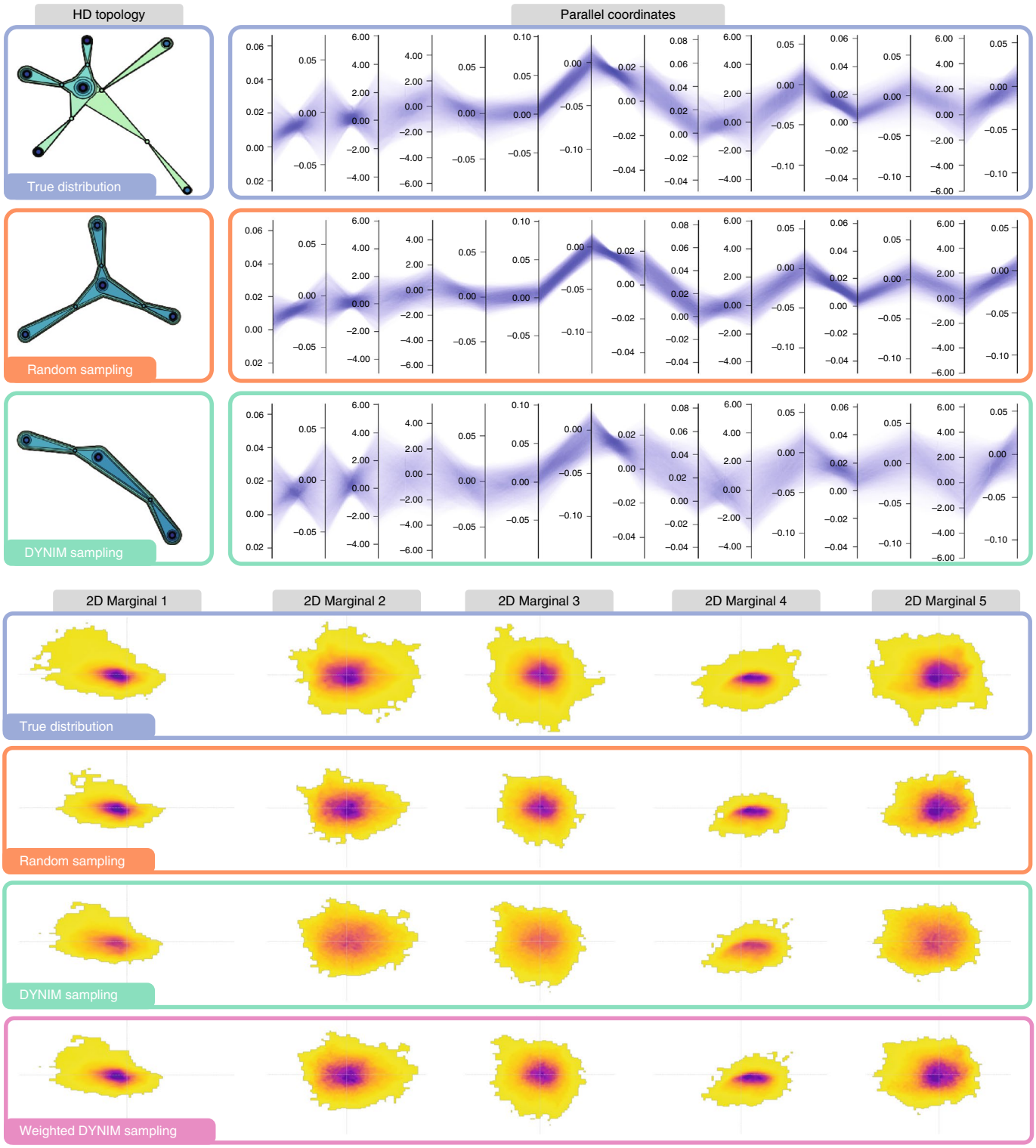

Fig. 4 | Evaluation and comparison of DynIm sampling. Our ML-based DynIm sampling is designed to explore the phase space of data as uniformly as possible. Three types of visualizations are shown to demonstrate the coverage and approximation provided by DynIm as compared to random sampling of the given true distribution. Top left panels: topological spines[22] showing the high-dimensional topology[23] of the data as a graph connecting the modes of the distribution. Top right panels: 15-dimensional parallel coordinates plot, where each vertical line is a dimension and each blue line spanning across dimensions is a single data point. Bottom panels: five pairs of two-dimensional marginal distributions of density with the zeros of the corresponding dimensions marked. The visualizations confirm that as compared to the 'true' distribution of all patches, an unbiased, random sampling closely replicates the input distribution, whereas the DynIm sampling produces a wider and flatter distribution, indicating the desired characteristics. Finally, we show that appropriately weighting the DynIm sampling allows the true distribution to be recovered.

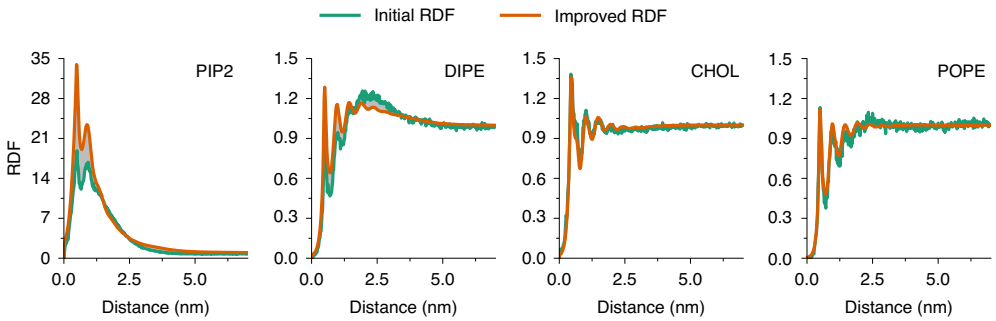

Fig. 5 | The self-healing mechanism enabled by the in situ computation of DynIm weights allows refinement of macro model parameters, such as RAS–lipid RDFs, by appropriately aggregating the results of micro scale simulations. The initial RDF is compared with the improved RDF after several rounds of feedback. Here feedback is shown for the inner leaflet PIP2, DIPE, CHOL and POPE, 4 of the 14 lipid channels in the patch (see Fig. 2).

Self-healing through in situ feedback. One of the unique characteristics of our DynIm framework is its support for an in situ feedback loop, which enables the self-healing of the resulting multiscale simulation. As discussed earlier, the framework can generate and analyse tens of thousands of MD simulations by sampling the space of lipid configurations driven by the macro simulation. This sampling, however, depends upon the initial parameterization of the macroscale model. The (initial) macro model used in the framework was parameterized using previously executed MD simulations. However, this preliminary data was rapidly dwarfed by the output of the current campaign, both in the number of MD simulations and the variations in sampled local environments. Although the fidelity of a static macro model is limited by the size of the initial training dataset, our framework provides a unique opportunity to continuously improve model parameters and, in turn, to improve future sampling.

We use an online feedback mechanism, in which the in situ analysis of MD data is used to update the parameters of the macro model. In particular, the protein-lipid parameters—the radial distribution functions (RDFs) between the proteins and all the lipids—are computed through in site analysis in MuMMI[14]. These RDFs are weighted by the prevalence of each simulation (that is, using the DynIm weights) and the updated RDFs are used to construct free-energy functionals to use in the macro model. As shown in Fig. 5, this self-healing mechanism progressively refines the macro parameters based on increasingly larger amounts of MD data, resulting in improved accuracy of the macro model. Referring back to Fig. 3, we focus on the intermittent spikes in the scores of selected patches in the second half of the simulation. These spikes are created largely by the self-healing feedback mechanism, which updates the macro model, potentially generating macro configurations.

The DynIm framework is computationally efficient and allows for a frequent feedback mechanism: the (re)computation of weights with respect to hundreds of thousands of candidate patches takes only on the order of a few minutes. Nevertheless, there may exist computational and input/output limitations on capturing and aggregating the statistics of interest; computing the RDFs for several thousand simulations, reading/writing these data and aggregating them all are likely to take substantially longer. Regardless, DynIm can be configured to perform feedback at a user-desired frequency and/or specific instances of time. For example, owing to computational and scheduling concerns in our scientific campaign, the feedback was applied only in the second half of the simulation (Fig. 5).

One limitation arises when working with the feedback mechanism. In this work, we assume that the macro configurations generated after the reparameterization due to feedback can still be encoded meaningfully as described above. Although we found this assumption to be largely true for our experiments (that is, the distribution of patches did not markedly digress from the original data), asserting mathematical guarantees on the impact of self-healing requires more work. In particular, depending upon the application at hand, one may consider retraining the ML model after every feedback loop to update the encoding, which, however, raises more challenges: the computational cost of retraining ML is likely to be prohibitive and retraining may substantially modify the latent space, making future selections inconsistent with the past decisions. Further investigation is needed to address these concerns.

Convergence study using synthetic data. We next establish the flexibility, generalizability, and convergence properties of our sampling approach using a synthetic dataset. As mentioned earlier, our importance sampling approach is fundamentally different from the usual, well-understood importance sampling techniques in that our goal is to explore novel configurations rather than minimizing the variance, as is the focus of typical approaches. Therefore, we instead evaluate our technique against random sampling—a realistic and often-used alternative.

For simplicity and interpretability, we create a synthetic two-dimensional dataset by sampling 4,800 points from two Gaussian distributions each and 400 points from a uniform distribution (as a proxy to some 'rare' configurations). We then dynamically sample 1,000 data points using 500 sampling events of two samples each, using both the random sampling and DynIm sampling. Figure 6 shows the density of these distributions after 250 such selection events. This result (similar to Fig. 4) illustrates that DynIm sampling provides a substantially better coverage of the space while still capturing the true distribution through weights.

To quantify these results, we perform this experiment 25 times, and for each experiment, we compare the DynIm and random sampling to the true distribution and the uniform distribution using the KL divergence[25], which measures how similar two distributions are—a lower value implies more similarity. Figure 6 also shows the trends of the KL divergence values for four comparisons: (1) random versus uniform, (2) random versus true, (3) DynIm versus uniform, and (4) weighted DynIm versus true. As expected, plot (1) shows consistently high values because random sampling fails to provide good coverage and differs substantially from a uniform sampling. Plot (2) also confirms the expected behaviour that the randomly sampled distribution is very similar to the true distribution. Plots (3) and (4) demonstrate the flexibility that DynIm offers by providing a good coverage and reconstruction of the true distribution. In particular, the difference between plots (1) and (3) indicate much better coverage achieved by DynIm, whereas the narrow gap between plots (2) and (4) show that DynIm still offers a good way to estimate the true density. Finally, we note that plot (3)—the dashed green line—trends upwards and overtakes plot (4)—the

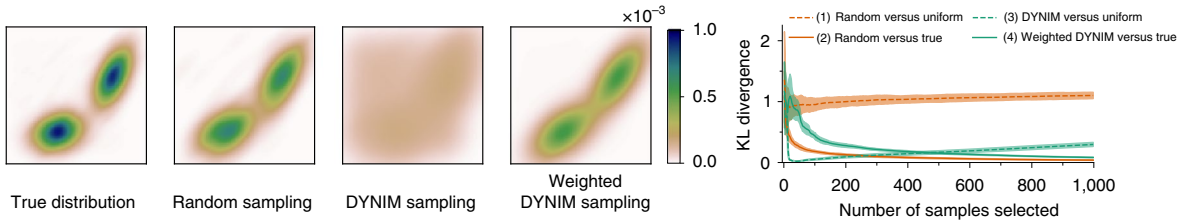

Fig. 6 | Convergence of DynIm sampling. Given a synthetic dataset generated from the known ('true') distribution, a random sampling approach simply recreates the true distribution and fails to sufficiently capture outliers to provide good coverage. The DynIm sampling instead specifically focuses on uniformly sampling the space and provides substantially better coverage and, using the importance weights, also allows the true distribution to be reconstructed. The left panels show the results of a sampling experiment after selecting 500 samples. Using the KL divergence as a metric to compare two distributions (a lower value means more similar), the right-most plot demonstrates that that DynIm sampling provides high-quality uniform sampling (hence good coverage), especially when compared to random sampling and that the weighted DynIm sampling is capable of capturing the true distribution. The right panel shows the results of 25 sampling experiments with plot lines and shaded regions representing the mean and standard deviation of the experiments. This result also identifies the crossover point (about 470 samples) where the DynIm distribution appears to have sufficiently expanded the coverage and now begins to over-refine the modes of the distribution.

solid green line—at about 470 samples. This behaviour indicates the nature of the dataset: since the given data is highly biased towards its two modes and has few outliers, DynIm is able to capture these rare occurrences rather quickly, and any further sampling simply provides an improved resolution into the modes. We assert that this crossover point can be seen as a sufficiency condition in the sampling.

Discussion

We leverage the capabilities of unsupervised ML to develop a dynamic-importance sampling framework that draws samples dynamically and adaptively from a partially seen data distribution. Our framework consists of two components, an ML-based encoder and a farthest-point sampler. In this work, these components are designed to steer the sampling towards 'novel' samples (macro configurations). The framework presented in this paper answers a call to identify rare events, in this case, in a biological system. Nevertheless, the sampling of rare events is often a requirement that goes beyond the application at hand. We believe DynIm is a powerful tool that can be used in a wide variety of applications.

The ideas presented here are generalizable in several ways. First, our ML model (VAE) was designed to serve a specific hypothesis of interest[14]; it is straightforward to use a different ML model for different hypotheses, different type of data, or even a different type of scientific study. As an example, we are currently applying our DynIm framework to connect molecular and atomistic scales (in a similar context to RAS–membrane simulations) to enable a three-scale simulation. This extension simply requires a way to encode molecular configurations as dictated by the investigation of interest. Furthermore, the farthest-point sampling in our current work uses Euclidean distances in the latent space. However, distance metrics—for example, based on certain scientific observables—could be designed to support alternative scientific goals. We are currently investigating other ML techniques related to manifold learning and metric learning that may require non-Euclidean metrics, but will still be easily supported by DynIm.

The dynamic nature of our sampling lends noteworthy advantages to large multiscale simulations. Given sufficient computational resources, the sampling can create a large ensemble of micro simulations, such that there exists at least one microscale simulation that is sufficiently similar to any given macroscale configuration. As a result, the resulting multiscale simulation can provide microscale precision at macro length and time scales. The dynamic weights provided by our sampling approach allows the true distribution of macro configurations to be reconstructed, including during the run time, which enables an automatic feedback. DynIm is highly scalable and can be used on small personal computers or the largest computers on the planet.

Methods

Multiscale simulation of RAS–lipid dynamics. Here, we describe the different scales of the target scientific study that are coupled by MuMMI using the DynIm approach. In particular, the MuMMI multiscale simulation[14] consists of a single macroscale simulation, and hundreds of thousands of microscale simulations of important macro configurations selected by DynIm. Each macro configuration of interest is represented as a patch, which is also a unit of conversion between the two scales.

The macro model. MuMMI uses a macro model to rapidly explore long time and length scale behaviour of RAS–membrane dynamics, capturing the continuum degrees of freedom. Specifically, the macro model uses the DDFT to describe the lipid–lipid behaviour[26]. The lipid bilayer is represented as a two-dimensional surface, whereas each protein and its conformational state on the bilayer is represented by a single particle, which interacts with the lipids through a potential of mean force. The input parameters to the macro model are lipid–lipid direct correlation functions and self-diffusion constants, RAS–lipid and RAS–RAS potentials, and RAS diffusion constants. These parameters were derived from several hundred coarse-grained plasma membrane and RAS–plasma membrane simulations.

MuMMI implements the macro model into the MOOSE finite-element framework[27]. This implementation was coupled together with MD simulations using ddcMD[28] to integrate the protein equations of motion. In this campaign, the macro model was used to simulate a $1\times1\,\mu m^2$ plasma membrane at a resolution of $1{,}200\times1{,}200$ cubic-order elements, with 300 RAS molecules. Using 100 computational nodes of Sierra, the macro model was able to simulate about 11.5 μs per day, where lipid concentrations were updated every 20 ns, and the RAS particles were integrated with a 25-ps time step.

Patches: macro to micro using patches. Although the macro model simulation can explore the phase space of local lipid configurations and the impact of RAS, it does not provide sufficient resolution into the underlying molecular biophysics. To deliver accurate insights into the molecular-level interactions of interest, MuMMI uses coarse-grained MD simulations to evolve the dynamics of the 'important' spatiotemporally local regions of the membrane.

In particular, since the goal is to explore RAS–membrane dynamics—that is, how RAS proteins interact with the lipids of the plasma membrane—MuMMI extracts $30\times30\,nm^2$ patches around each RAS for each frame of the macro simulation. A patch captures the complex dynamics of the lipids as affected by the protein. Although a patch is always centred around a corresponding RAS, there may be additional RAS proteins within the specified neighbourhood, in which case the lipid concentrations within the patch reflect the effects of more than one RAS.

A patch is a unit of conversion from the macro model that can be mapped to a molecular configuration. Although at the native resolution of the macro simulation (1,000 nm = 1,200 grid points), a patch forms a 37×37 grid, the mapping of a patch to a coarse-grained configuration assumes each patch to be a 5×5 grid. Therefore, the native resolution of a patch is aggregated into a $5\times5\times14$ image, with each of the 14 lipid concentrations represented as a separate channel. Figure 2 illustrates four different patches by visualizing all 14 channels individually and identifies the channels (lipids) that respond strongly to the presence of RAS.

Over the course of the multiscale simulation, MuMMI typically generates millions of 'candidate' patches, which are analysed dynamically for importance using our DynIm framework.

The micro model. Given a patch selected using the DynIm approach, MuMMI instantiates and equilibrates a Martini[15] coarse-grained simulation using a modified version of the *insane* membrane building tool[29] and the GROMACS MD package[30] (CPU-only version) for energy minimization, equilibration, and to pull the proteins to the bilayer. Each coarse-grained particle system contains $3{,}070 \pm 82$ lipids. The entire process takes approximately 90 min using 24 CPU cores of a single computational node.

Once equilibrated, the particle system is shipped to ddcMD[28,31,32] for performing the MD simulations. MuMMI highlights the GPU capabilities in ddcMD designed to accelerate the Martini coarse-grained force field[21]. Using one GPU per simulation, ddcMD can produce approximately 1.02 µs of trajectories per day (of wall-clock time) at a time resolution of 20 fs.

In situ analysis of MD simulations. To support the self-healing capabilities provided by DynIm and to scale up to hundreds of thousands of MD simulations, MuMMI performs in situ analysis of trajectories generated by ddcMD. The online analyses to be performed are designed and chosen based on parameters of interest from preliminary simulations and those needed for re-optimization of the macro model. Examples of features of interest are RAS–RAS contacts, RAS–lipid contacts, RAS orientation and lipid distributions.

DynIm sampling. *Encoding of macro configurations.* In this work, we use a VAE[18] implemented using a deep neural network that learns a reduced representation of the data in an unsupervised manner. An autoencoder progressively reduces the dimensionality of the input data until the so-called bottleneck layer is reached, which represents a reduced, latent encoding of the input data. Using this latent representation, which is typically a nonlinear combination of the input features, the autoencoder then progressively increases the dimensionality to reconstruct the input data. By minimizing the reconstruction error through the training process, the model learns the most important features in the data, which must be preserved for its faithful representation. When trained appropriately, suitable autoencoders are robust to noise in the input data. Among the types of autoencoder, we choose a VAE because it provides several favourable mathematical properties, such as a continuous distribution in the latent space, which are important for statistical analysis on the resulting sampling. Since each latent dimension represents an irreducible degree of variation in the data, the resulting latent space provides a meaningful way to compare patches by defining similarity between them, such as using Euclidean distances.

Identifying the implicit dimensionality in the data was a key consideration when choosing an encoder model. During the exploration, we developed several autoencoder and VAE models, varying the number, width, and type of layers in the model as well as different sizes of the output latent space, and evaluated them using reconstruction loss.

To compensate for the differences in the function ranges of different lipid concentrations, we normalized each channel independently with respect to the dynamic range of the corresponding concentration across all training data. The same normalization was used to transform the input data during inference. This normalization allowed us to use the mean-squared error to evaluate the reconstruction, along with KL-divergence of the resulting latent distribution from the normal distribution, as is typical for a VAE.

Owing to the limited spatial resolution of the input patches, we found that spatial convolutional filters were not suitable for our target models. However, unsurprisingly, we noticed improvement in the accuracy of the models when we applied one-dimensional convolutional filters across channel (lipid concentration) as the model could capture the corresponding correlations.

Our final VAE model used a single convolutional layer (across channels) followed by seven fully connected layers (number of nodes = 700, 350, 175, 80, 40, 20 and 15) interspersed by batch normalization layers and 20%-dropout layers, encoding the patches into a 15-dimensional latent space. Batch normalization layers normalize the output of previous layers and help to stabilize the network towards faster training. Dropout layers set certain randomly selected weights to zero and help to prevent over-fitting. The decoder used five fully connected layers (number of nodes = 20, 40, 80, 175 and 350) to reconstruct the data. As highlighted in Supplementary Fig. 1, we noted the quality of reconstruction to be notably worse for lower-dimensional latent spaces. On the other hand, larger latent spaces produced only marginally better reconstruction error, which we chose to discard in favour of smaller dimensionality.

Supplementary Fig. 1 also shows the training history of the chosen model, indicating that the error converged to a small value. All models were trained using 302,100 patches generated prior to our scientific campaign using a macro simulation that was parameterized consistently. The training was performed using the `adam` optimizer (with learning rate 0.001) for about 500 epochs, until the error converged. To account for possible rotational bias in the data, we augmented the data by rotating the input patches by 90°, 180° and 270°.

Sampling of macro configurations. Given an encoded patch, $\ell(\mathbf{p})$, that is a candidate for selection, we define its DynIm score, $r(\mathbf{p})$, as the average distance to its k-nearest neighbours in $\mathbf{S}$, the set of all previously selected patches. Through experimental analysis, we chose $k = 10$, which provides robustness against noise while maintaining computational viability. We store $\mathbf{S}$ using an efficient tree-based data structure[19,20] that provides fast, approximate-nearest-neighbour queries, allowing for almost-real-time evaluations of DynIm scores for millions of candidate patches.

A dynamic selection infrastructure is enabled using an in-memory priority queue, $\mathbf{C}$, that contains all candidate patches ordered by their DynIm score. There exist two types of update operation on $\mathbf{C}$. First, as the macro simulation progresses, new patches are created and encoded. In this case, the DynIm scores of new patches are computed and the patches are inserted in $\mathbf{C}$. Second, when computational resources are available, a patch is selected (and removed) from $\mathbf{C}$ and added to $\mathbf{S}$. In this scenario, the ranking of all remaining candidate patches in $\mathbf{C}$ must be re-evaluated as the corresponding DynIm distances in the latent space change.

If the macro simulation creates a new patch that is consequentially different from existing configurations in $\mathbf{S}$, we expect the DynIm score of the new patch to be high. However, once added to $\mathbf{C}$, by design, the DynIm score of existing candidates cannot increase, because as new patches are added to $\mathbf{S}$, the distance of k-nearest neighbours can only decrease. Therefore, we limit $\mathbf{C}$ to a computationally feasible and scientifically relevant length as the discarded patches (tail of $\mathbf{C}$) will never be selected. The choice of the the maximum length of $\mathbf{C}$ is guided by the size of the computational resource, the input rate of macro patches, and the anticipated rate of sampling. For our experiments on Sierra (about 16,000 GPUs), $\mathbf{C}$ was capped at a length of 35,000.

Reconstruction of the true distribution. Since we use (approximate) k-nearest neighbours to define the DynIm score of candidate patches, it is natural to use the RNN[24] to define the DynIm weights for the selected patches.

Given two sets $\mathbf{P}$ and $\mathbf{Q}$ defined on the same space, the RNN of a point $\mathbf{q} \in \mathbf{Q}$ is the set of all points $\mathbf{p} \in \mathbf{P}$ whose nearest neighbour in $\mathbf{Q}$ is $\mathbf{q}$. In this way, the RNN can be used to define the influence of $\mathbf{q}$ on the set $\mathbf{P}$. Previously[24], RNN is defined for $\mathbf{P} = \mathbf{Q}$. Here, we require $\mathbf{P}$ and $\mathbf{Q}$ to be mutually exclusive and, therefore, redefine RNN as $\mathrm{RNN}(\mathbf{q}) = \{\mathbf{p} \in \mathbf{P} \mid \forall \mathbf{q}' \in \mathbf{Q}$ such that $\mathrm{dist}(\mathbf{p}, \mathbf{q}) < \mathrm{dist}(\mathbf{p}, \mathbf{q}')$ for $\mathbf{q}' \neq \mathbf{q}\}$.

Using the RNN with $\mathbf{P} = \mathbf{C}$ and $\mathbf{Q} = \mathbf{S}$, we define the DynIm weight of all selected patches as $w(\mathbf{q}) = 1 + |\mathrm{RNN}(\mathbf{q})|$, where the self-weight of 1 has been added.

Data availability

Sample data for DynIm is also made available along with the code repository. The data related to the multiscale simulation described in the paper will be made available upon reasonable request; the size of all raw data is hundreds of terabytes. For more information, please see details of the simulation[14]

Code availability

The framework for DynIm has been released open source under the MIT license: https://github.com/LLNL/dynim.

Received: 1 June 2020; Accepted: 26 February 2021;
Published online: 22 April 2021

References

1. Ingram, G., Cameron, I. & Hangos, K. Classification and analysis of integrating frameworks in multiscale modelling. *Chem. Eng. Sci.* **59**, 2171–2187 (2004).
2. Weinan, E. *Principles of Multiscale Modeling* (Cambridge Univ. Press, 2011).
3. Hoekstra, A., Chopard, B. & Coveney, P. Multiscale modelling and simulation: a position paper. *Phil. Trans. R. Soc. A* **372**, 20130377 (2014).
4. Chopard, B., Borgdorff, J. & Hoekstra, A. A framework for multi-scale modelling. *Phil. Trans. R. Soc. A* **372**, 20130378 (2014).
5. Geweke, J. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica* **57**, 1317–39 (1989).
6. MacKay, D. J. C. *Information Theory, Inference, and Learning Algorithms* (Cambridge Univ. Press, 2003).
7. Liang, F. Dynamically weighted importance sampling in Monte Carlo computation. *J. Am. Stat. Assoc.* **97**, 807–821 (2002).
8. Liang, F. & Cheon, S. Monte Carlo dynamically weighted importance sampling for spatial models with intractable normalizing constants. *J. Phys. Conf. Ser.* **197**, 012004 (2009).
9. Joubert, D. J. & Marwala, T. Monte Carlo dynamically weighted importance sampling for finite element model updating. In *Topics in Modal Analysis and Testing* (ed. Mains, M.) Vol. 10, 303–312 (Springer, 2016).
10. Katharopoulos, A. & Fleuret, F. Not all samples are created equal: deep learning with importance sampling. In *Proc. 35th Int. Conf. on Machine Learning* (eds Dy, J. & Krause, A.) Vol. 80, 2525–2534 (Proceedings of Machine Learning Research, 2018). http://proceedings.mlr.press/v80/katharopoulos18a.html"

11. Johnson, T. B. & Guestrin, C. Training deep models faster with robust, approximate importance sampling. In *Advances in Neural Information Processing Systems* (eds Bengio, S. et al.) Vol. 31, 7265–7275 (Curran Associates, 2018). http://papers.nips.cc/paper/7957-training-deep-models-faster-with-robust-approximate-importance-sampling.pdf
12. Simanshu, D. K., Nissley, D. V. & McCormick, F. RAS proteins and their regulators in human disease. *Cell* **170**, 17–33 (2017).
13. Waters, A. M. & Der, C. J. KRAS: the critical driver and therapeutic target for pancreatic cancer. *Cold Spring Harbor Persp. Med.* **8**, a031435 (2018).
14. Ingólfsson, H. I. et al. Machine learning-driven multiscale modeling reveals lipid-dependent dynamics of RAS signaling proteins. Preprint at *Research Square* https://www.researchsquare.com/article/rs-50842/v1 (2020).
15. Marrink, S. J., Risselada, H. J., Yefimov, S., Tieleman, D. P. & de Vries, A. H. The MARTINI Force Field: coarse grained model for biomolecular simulations. *J. Phys. Chem. B* **111**, 7812–7824 (2007).
16. Di Natale, F. et al. A massively parallel infrastructure for adaptive multiscale simulations: modeling RAS initiation pathway for cancer. In *Supercomputing '19: Int. Conf. High Performance Computing, Networking, Storage, and Analysis* 57 (ACM, 2019).
17. *November 2019 TOP500 The List* https://www.top500.org/lists/2019/11/ (2019).
18. Doersch, C. Tutorial on variational autoencoders. Preprint at https://arxiv.org/abs/1606.05908 (2016).
19. Jégou, H., Douze, M., Johnson, J. & Hosseini, L. FAISS. *GitHub* https://github.com/facebookresearch/faiss (2021).
20. Johnson, J., Douze, M. & Jégou, H. Billion-scale similarity search with gpus. *IEEE Trans. Big Data* https://doi.org/10.1109/TBDATA.2019.2921572 (2019).
21. Zhang, X. et al. ddcMD: a fully GPU-accelerated molecular dynamics program for the Martini Force Field. *J. Chem. Phys.* **153**, 045103 (2020).
22. Correa, C., Lindstrom, P. & Bremer, P. T. Topological spines: a structure-preserving visual representation of scalar fields. *IEEE Trans. Visualiz. Comput. Graph.* **17**, 1842–1851 (2011).
23. Liu, S. et al. Scalable topological data analysis and visualization for evaluating data-driven models in scientific applications. *IEEE Trans. Visualiz. Comput. Graph.* **26**, 291–300 (2020).
24. Korn, F. & Muthukrishnan, S. Influence sets based on reverse nearest neighbor queries. *SIGMOD Rec.* **29**, 201–212 (2000).
25. Kullback, S. & Leibler, R. A. On information and sufficiency. *Ann. Math. Stat.* **22**, 79–86 (1951).
26. Marconi, U. M. B. & Tarazona, P. Dynamic density functional theory of fluids. *J. Chem. Phys.* **110**, 8032–8044 (1999).
27. Tonks, M. R., Gaston, D., Millett, P. C., Andrs, D. & Talbot, P. An object-oriented finite element framework for multiphysics phase field simulations. *Comput. Mater. Sci.* **51**, 20–29 (2012).
28. Streitz, F. H., Glosli, J. N. & Patel, M. V. Beyond finite-size scaling in solidification simulations. *Phys. Rev. Lett.* **96**, 225701 (2006).
29. Wassenaar, T. A., Ingólfsson, H. I., Böckmann, R. A., Tieleman, D. P. & Marrink, S. J. Computational lipidomics with *insane*: a versatile tool for generating custom membranes for molecular simulations. *J. Chem. Theor. Comput.* **11**, 2144–2155 (2015).
30. Abraham, M. J. et al. GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **1/2**, 19–25 (2015).
31. Streitz, F. H. et al. 100+ TFlop solidification simulations on BlueGene/L. In *Proc. 2005 ACM/IEEE Conf. Supercomputing (SC '05)* http://www.cresco.enea.it/SC05/schedule/pdf/pap307.pdf (ACM, 2005).
32. Glosli, J. N. et al. Extending stability beyond CPU millennium: a micron-scale atomistic simulation of Kelvin–Helmholtz instability. In *Proc. 2007 ACM/IEEE Conf. Supercomputing (SC '07)* 58:1–58:11, https://doi.org/10.1145/1362622.1362700 (ACM, 2007).
33. Ingólfsson, H. I. et al. Capturing biologically complex tissue-specific membranes at different levels of compositional complexity. *J. Phys. Chem. B* **124**, 7819–7829 (2020).

Acknowledgements

This work has been supported in part by the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) programme established by the US Department of Energy (DOE) and the National Cancer Institute (NCI) of the National Institutes of Health (NIH). For computing time, we thank Livermore Computing (LC) and Livermore Institutional Grand Challenge. This work was performed under the auspices of the US DOE by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 and Los Alamos National Laboratory under contract DEAC5206NA25396. Release number: LLNL-JRNL-806073.

Author contributions

The framework was designed by H.B., T.S.C., H.I.I., G.D., B.V.E., J.N.G., P.-T.B. and F.C.L.; the framework was implemented by H.B. Sampling analysis was done by H.B., P.K., S.L., T.O. and C.N. All authors contributed to the writing of the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-021-00327-w.

Correspondence and requests for materials should be addressed to H.B.

Peer review information *Nature Machine Intelligence* thanks Shangying Wang, and the other anonymous reviewer(s), for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.