nature portfolio

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

Supplementary information

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

In the format provided by the authors and unedited

Machine Learning Based Dynamic-Importance Sampling for Adaptive Multiscale Simulations

Harsh Bhatia,[*,†] Timothy S. Carpenter,[‡] Helgi I. Ingólfsson,[‡] Gautham Dharuman,[‡] Piyush Karande,[**P**] Shusen Liu,[†] Tomas Oppelstrup,[‡] Chris Neale,[§] Felice C. Lightstone,[‡] Brian Van Essen,[†] James N. Glosli,[‡] and Peer-Timo Bremer[†]

†*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, USA*
‡*Physical and Life Sciences, Lawrence Livermore National Laboratory, Livermore, USA*
P *Computational Engineering, Lawrence Livermore National Laboratory, Livermore, USA*
§ *Theoretical Biology and Biophysics, Los Alamos National Laboratory, Los Alamos, USA*

E-mail: hbhatia@llnl.gov

1 Supplementary Information

1.1 DynIm Algorithm

Algorithm 1 outlines the functionality of DynIm, to further facilitate understanding and use of our framework. The code and supporting examples and data are released under MIT license and can be found on github: https://github.com/LLNL/dynim.

Algorithm 1 DynIm sampling algorithm.

```
procedure DYNIM.INITIALIZE(encoder)
    dynim.encoder ← encoder          ▷ an ML encoder that satisfies the importance hypothesis
    dynim.candidates ← [ ]                                  ▷ empty list for encoded candidates
    dynim.selections ← [ ]                                  ▷ empty list for selections

procedure DYNIM.ADD_CANDIDATES(list_of_candidates)
    encoded_candidates ← dynim.encoder.encode(list_of_candidates)
    dynim.candidates.append_all(encoded_candidates)

procedure DYNIM.SELECT_IMPORTANT_CANDIDATES(k)
    ranks ← dynim.compute_importance(dynim.candidates)
    dynim.candidates ← sort(dynim.candidates, key=ranks, reverse=True)
    important_candidates ← dynim.candidates[:k]                    ▷ top k candidates
    dynim.candidates ← dynim.candidates[k:]                        ▷ remaining candidates
    dynim.selections.append_all(important_candidates)
    return important_candidates

procedure DYNIM.COMPUTE_IMPORTANCE(list_of_candidates)
    ranks = [ ]
    for each c ∈ list_of_candidates do
        knn = get_knn(c, target=dynim.selections, k=10)
        ranks.append(mean(‖c − knn‖))
    return ranks
```

1.2 Supplementary Figures

Figure 1 highlights a comparison between several autoencoder models developed and tested for this work in order to choose the most suitable one, with respect to the dimensionality of the latent space and the quality of reconstruction.

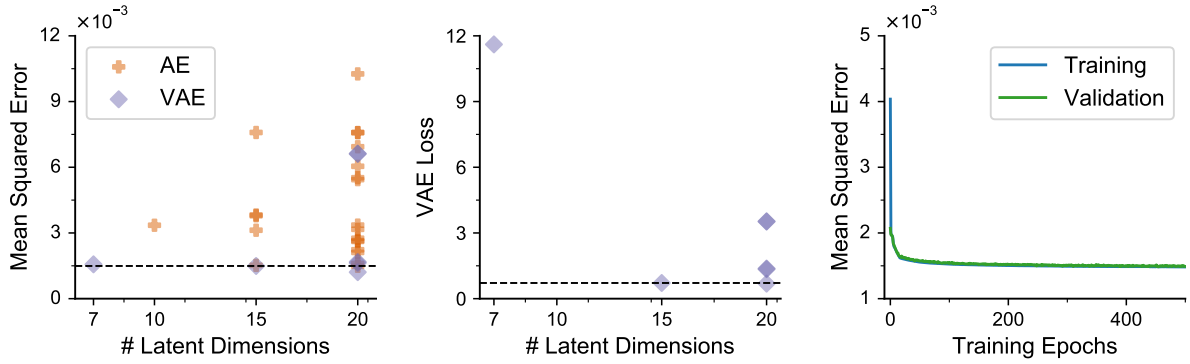

Figure 1: Identification of intrinsic dimension of data and search for a suitable model was performed by exploring several AE and VAE models. The selected VAE model was chosen as a balance between low dimensionality and reconstruction quality. Left: The mean squared error was used to assess the reconstruction quality of all models – horizontal line indicates the chosen 15-D VAE model. Middle: The VAE models were also evaluated on the VAE loss, which additionally accounts for similarity to a normal distribution – horizontal line indicates the chosen model. Right: The training history of the selected model indicates converged errors with no over-fitting.