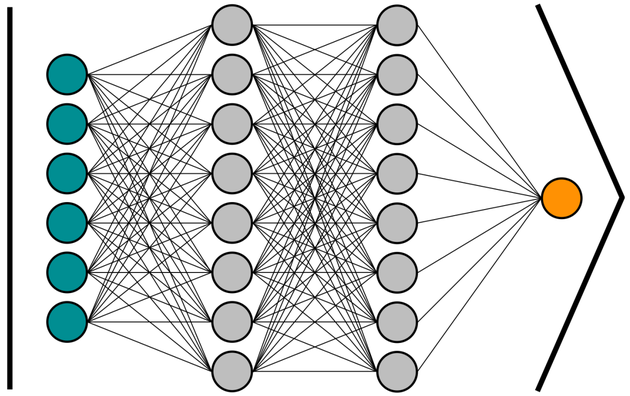


Solving the Quantum Many-Body Problem with Neural-Network Quantum States

Jane Kim
Michigan State University
Facility for Rare Isotope Beams
PhD Dissertation Defense
20 July 2023

MICHIGAN STATE UNIVERSITY

FRIB

Acknowledgements

- **Research Advisor:** Morten Hjorth-Jensen

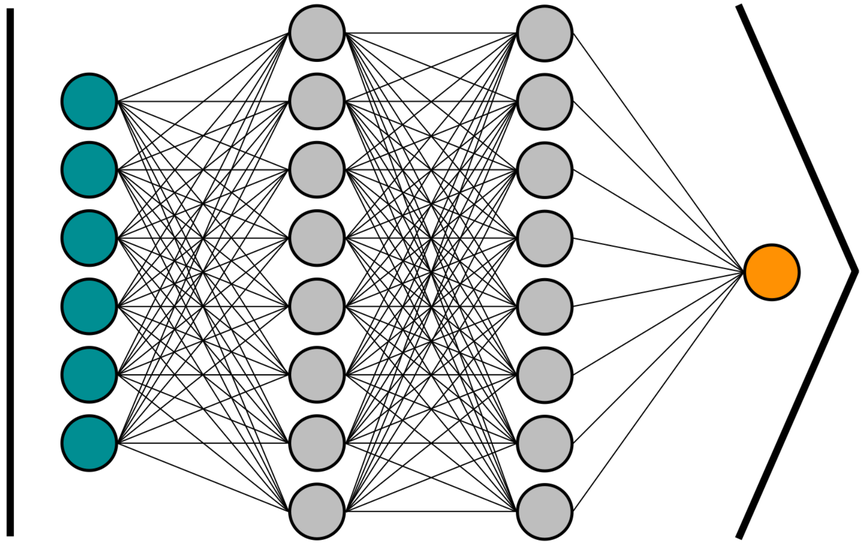
- **Guidance Committee:** Filomena Nunes, Artemis Spyrou, Stuart Tessmer, Michael Murillo, Huey-Wen Lin, (Matthew Hirn)

- **Collaborators:** Alessandro Lovato (ANL), Bryce Fore (ANL), Stefano Gandolfi (LANL), Giuseppe Carleo (EPFL), Gabriel Pescia (EPFL), Jannes Nys (EPFL), Even Nordhagen (UiO)

- **Funding:** US NSF Grants No. PHY-1404159 and PHY-2013047

Overview

- The quantum many-body problem
- Quantum Monte Carlo
- Machine learning and artificial neural networks
- Implementations in C++/MPI and Python/JAX
- Applications: Calogero-Sutherland model
  Dilute neutron matter
  Homogeneous electron gas
  Ultra-cold Fermi gases



The Quantum Many-Body Problem

Assumptions

- Time-independent Hamiltonian $\implies$ Solve time-independent Schrödinger equation

$$\hat{H}\,|\Psi\rangle = E\,|\Psi\rangle$$

- Indistinguishable particles $\implies$ $|\Psi\rangle$ must be symmetric or antisymmetric

- Work in continuous position (discrete spin, isospin) basis $\implies$ $|\Psi\rangle = \int d\mathbf{X}\; \Psi(\mathbf{X})\; |\mathbf{X}\rangle$

- Simulate N particles in d dimensions $\implies$ $|\mathbf{X}\rangle = |\mathbf{x}_1\rangle \otimes |\mathbf{x}_2\rangle \otimes \cdots \otimes |\mathbf{x}_N\rangle$

$$|\mathbf{x}_i\rangle = |\mathbf{r}_i\rangle \otimes |\mathbf{s}_i\rangle, \quad \mathbf{r}_i \in \mathbb{R}^d$$

Goals

- Find the ground state wave function $\Psi_0(\mathbf{X})$ of a system of interacting particles
- Develop a flexible, compact, and low-bias ansatz represented by neural networks (neural-network quantum states (NQS) first applied by Carleo and Troyer, Science, 2017)
- Achieve state-of-the-art results (benchmark with diffusion Monte Carlo)
- **Unique nuclear theory perspective:** spin and isospin exchange, strong and short-range interaction, and a possible three-body term motivates ansätze with even more generality and flexibility than those developed from a condensed matter or quantum chemistry perspective

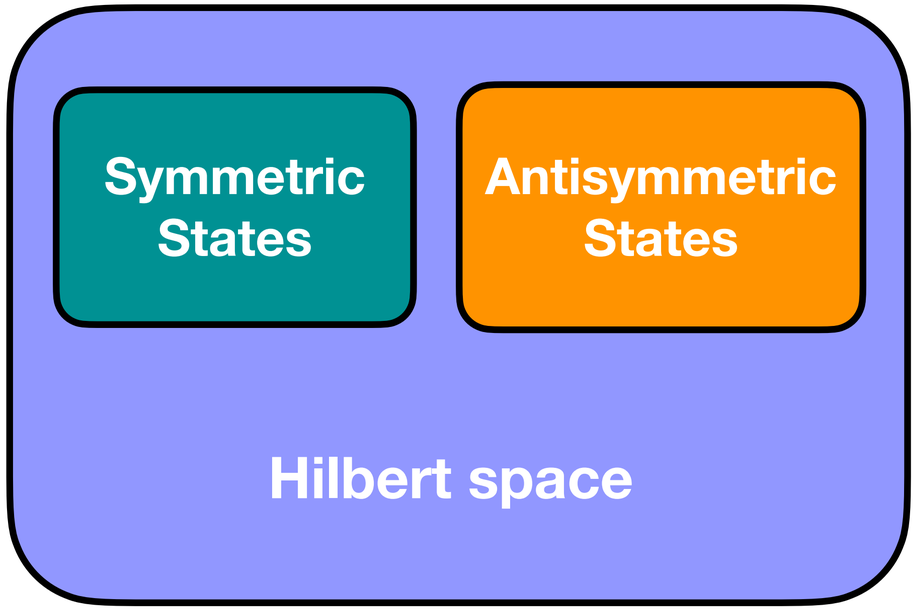
Computational Challenges

- Storing particle degrees of freedom in memory treats particles as distinguishable
- Hilbert space dimension grows exponentially with system size N

$$\mathscr{H} = \mathscr{H}_1 \otimes \mathscr{H}_2 \otimes \cdots \otimes \mathscr{H}_N$$

- But not all states within Hilbert space are physical!
- We need to enforce particle indistinguishability ourselves to restrict our search to the relevant areas of Hilbert space

$$\mathscr{H}_S \oplus \mathscr{H}_A \subseteq \mathscr{H}$$



Computational Challenges

- Enforcing other symmetries can further help reduce the volume of Hilbert space we need to search

- Continuous-space problems $\Longrightarrow$ infinite basis

- Quantum many-body methods aim to mitigate the problem of exponential scaling of the Hilbert space dimension, for example:

  - Configuration Interaction
  - Hartree-Fock theory
  - Many-Body Perturbation Theory
  - Coupled Cluster
  - In-Medium Similarity Renormalization Group
  - And many more…

Quantum Monte Carlo Methods

Variational Principle

- The expectation value of a Hamiltonian $\hat{H}$ computed with any trial wave function $|\Psi\rangle$ is an upper bound to the true ground state energy E_0

$$E[\Psi] \equiv \frac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle} \geq E_0$$

- The variance of the energy is zero only if $|\Psi\rangle$ is an eigenstate of $\hat{H}$

- Caveat: $|\Psi\rangle$ must satisfy the fundamental symmetries of the system described by $\hat{H}$

Variational Monte Carlo

- Direct application of the variational principle:

$$\min_{\theta} E[\Psi_\theta] \geq E_0$$

Variational parameters of trial wave function

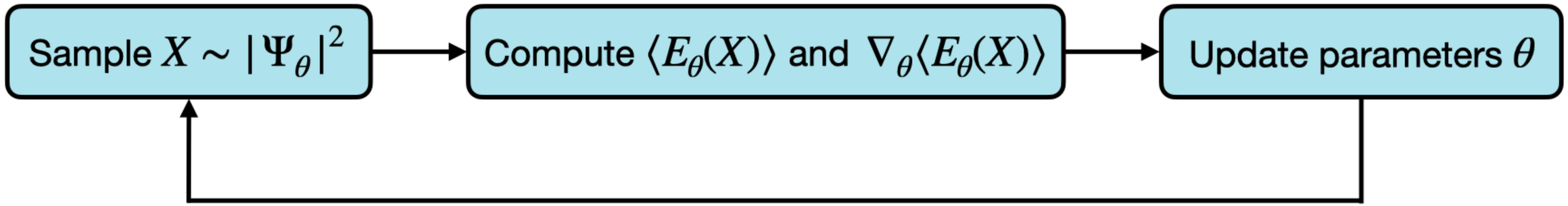
- Efficiently compute the variational energy via Monte Carlo integration:

$$E[\Psi_\theta] \equiv \frac{\langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle} = \frac{\int d\mathbf{X} \, |\Psi_\theta(\mathbf{X})|^2 E_\theta(\mathbf{X})}{\int d\mathbf{X} \, |\Psi_\theta(\mathbf{X})|^2} \approx \frac{1}{N_s} \sum_{i=1}^{N_s} E_\theta(\mathbf{X}_i) \equiv \langle E_\theta(\mathbf{X}) \rangle$$

Local energy $E_\theta(\mathbf{X}) \equiv \frac{\langle \mathbf{X} | \hat{H} | \Psi_\theta \rangle}{\langle \mathbf{X} | \Psi_\theta \rangle} = \frac{\hat{H}\Psi_\theta(\mathbf{X})}{\Psi_\theta(\mathbf{X})}$

Configurations sampled from $|\Psi_\theta(\mathbf{X})|^2$

Optimization



- Variational parameters θ are optimized using gradient descent methods

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \langle E_\theta(\mathbf{X})\rangle$$

- Compute gradient stochastically:

$$\nabla_\theta \langle E_\theta(\mathbf{X})\rangle = 2\Big(\langle E_\theta(\mathbf{X})\mathbf{O}_\theta(\mathbf{X})\rangle - \langle E_\theta(\mathbf{X})\rangle\langle \mathbf{O}_\theta(\mathbf{X})\rangle\Big) \text{ with } \mathbf{O}_\theta(\mathbf{X}) \equiv \frac{\nabla_\theta \Psi_\theta(\mathbf{X})}{\Psi_\theta(\mathbf{X})}$$

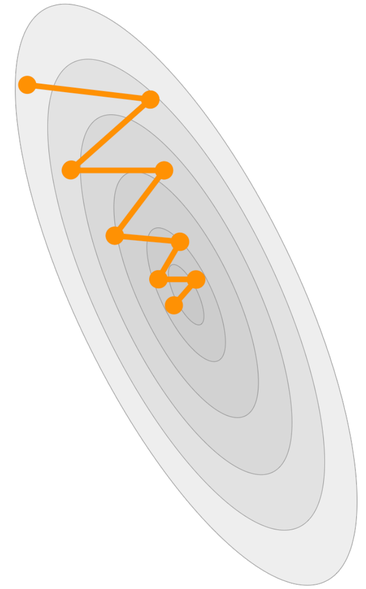
Stochastic Reconfiguration

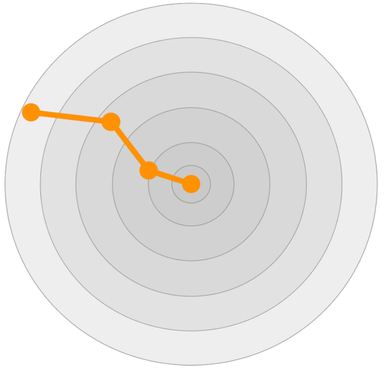
- Second-order optimization method that reshapes the energy landscape
- Derived by matching the variational state to a state obtained by applying the imaginary-time propagator
- Parameters are updated as

$$\theta_{t+1} = \theta_t - \eta S_\theta^{-1} \nabla_\theta \langle E_\theta(\mathbf{X}) \rangle$$

where S_θ is the quantum geometric tensor / Fisher information matrix

$$[S_\theta]_{ij} = \frac{\langle \partial_i \Psi_\theta | \partial_j \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle} - \frac{\langle \partial_i \Psi_\theta | \Psi_\theta \rangle \langle \Psi_\theta | \partial_j \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle}$$





Stochastic Reconfiguration

- It is possible for S_θ to be singular if some variational parameters are redundant
- Typically apply a small diagonal shift to stabilize the inverse $S_\theta^{-1} \mapsto [S_\theta + \varepsilon I]^{-1}$
- But constant diagonal shifts ignore the relative magnitudes of the parameters
- Better stabilization developed by Lovato et al. in Phys. Rev. Research **4**, 043178 (Dec 2022), inspired by the RMSprop algorithm

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1-\beta)\Big(\nabla_\theta \langle E_\theta(\mathbf{X})\rangle\Big)^2$$
$$\theta_{t+1} = \theta_t - \eta[S_\theta + \mathrm{diag}(\sqrt{\mathbf{v}_{t+1}} + \varepsilon)]^{-1} \nabla_\theta \langle E_\theta(\mathbf{X})\rangle$$

Trial Wave Functions

- In standard variational Monte Carlo calculations, the trial wave function is carefully designed for the specific system of interest

- Generally, the trial wave function has the form

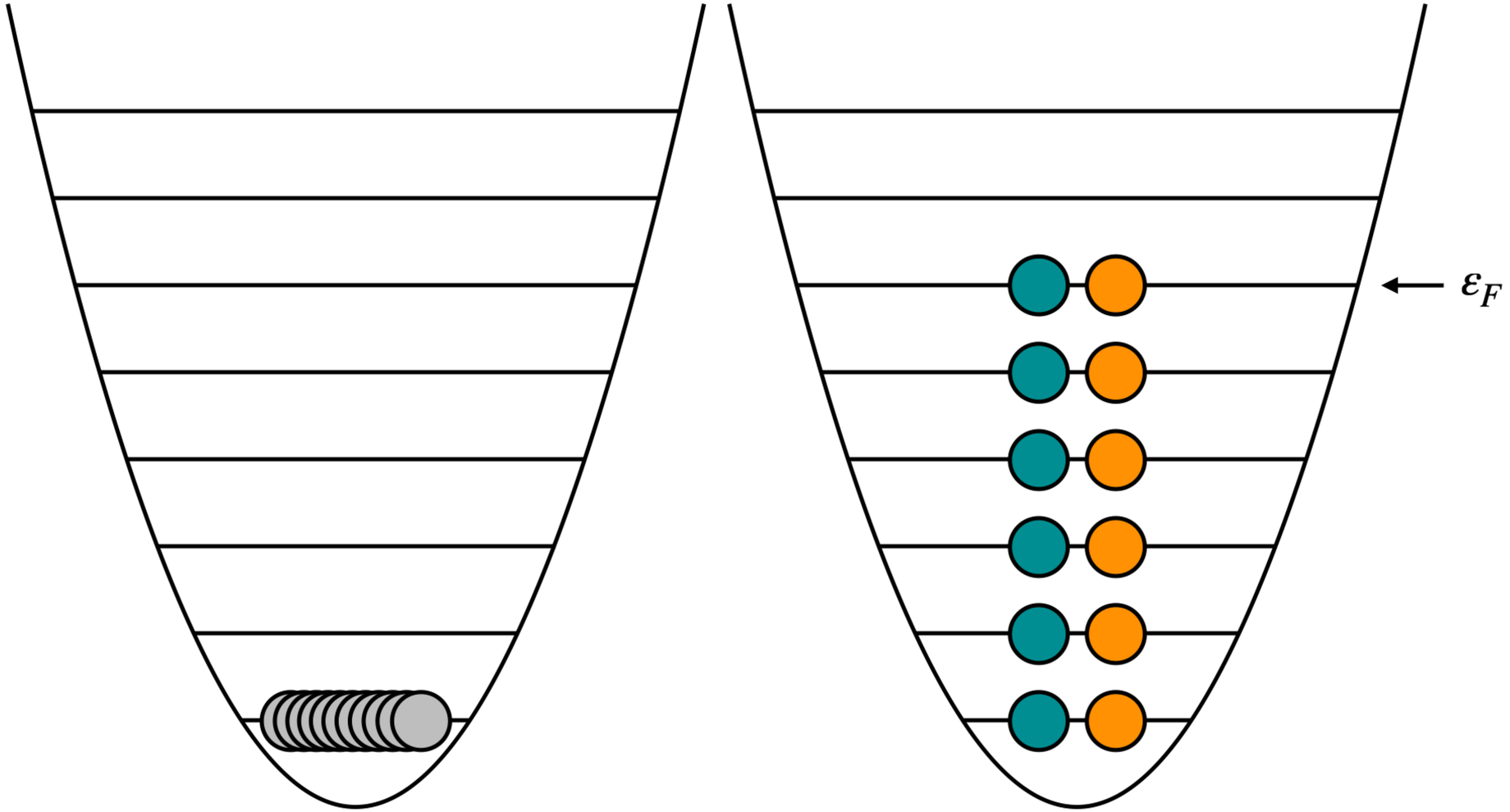
$$\Psi_\theta(\mathbf{X}) = \Phi_0(\mathbf{X})e^{J(\mathbf{X})}$$

- The noninteracting ground state constrains permutation symmetry/antisymmetry

- Jastrow correlator is designed to include correlations and satisfy Kato's cusp condition:

$$\lim_{r_{ij}\to 0} E[\Psi_0] < \infty \text{ for all pairs } i, j$$

Bosons vs. Fermions



Slater Determinants

- The simplest way to write down the antisymmetrized product of single-particle states:

$$\Phi_0(\mathbf{X}) = \det \begin{bmatrix} \varphi_1(\mathbf{x}_1) & \varphi_1(\mathbf{x}_2) & \cdots & \varphi_1(\mathbf{x}_N) \\ \varphi_2(\mathbf{x}_1) & \varphi_2(\mathbf{x}_2) & \cdots & \varphi_2(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_N(\mathbf{x}_1) & \varphi_N(\mathbf{x}_2) & \cdots & \varphi_N(\mathbf{x}_N) \end{bmatrix}$$

$\varphi_\alpha =$ single-particle spin-orbitals

$\mathbf{x}_i =$ single-particle degrees of freedom

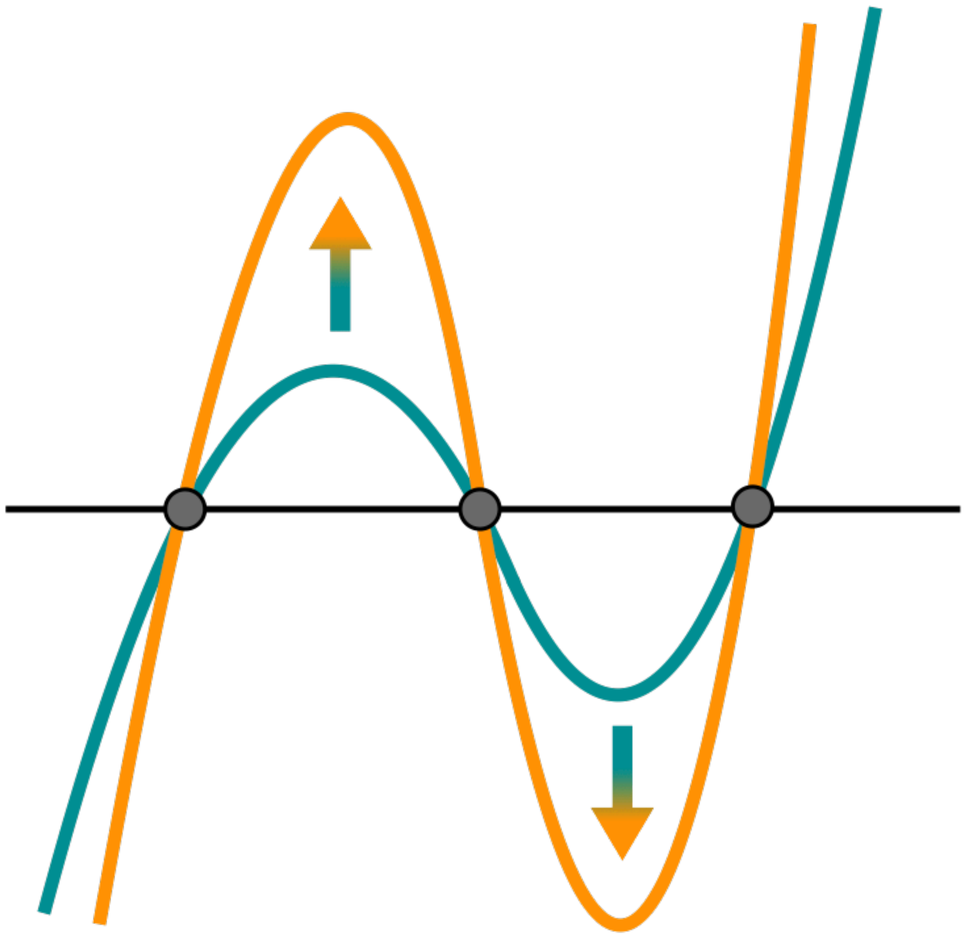
Backflow Transformations

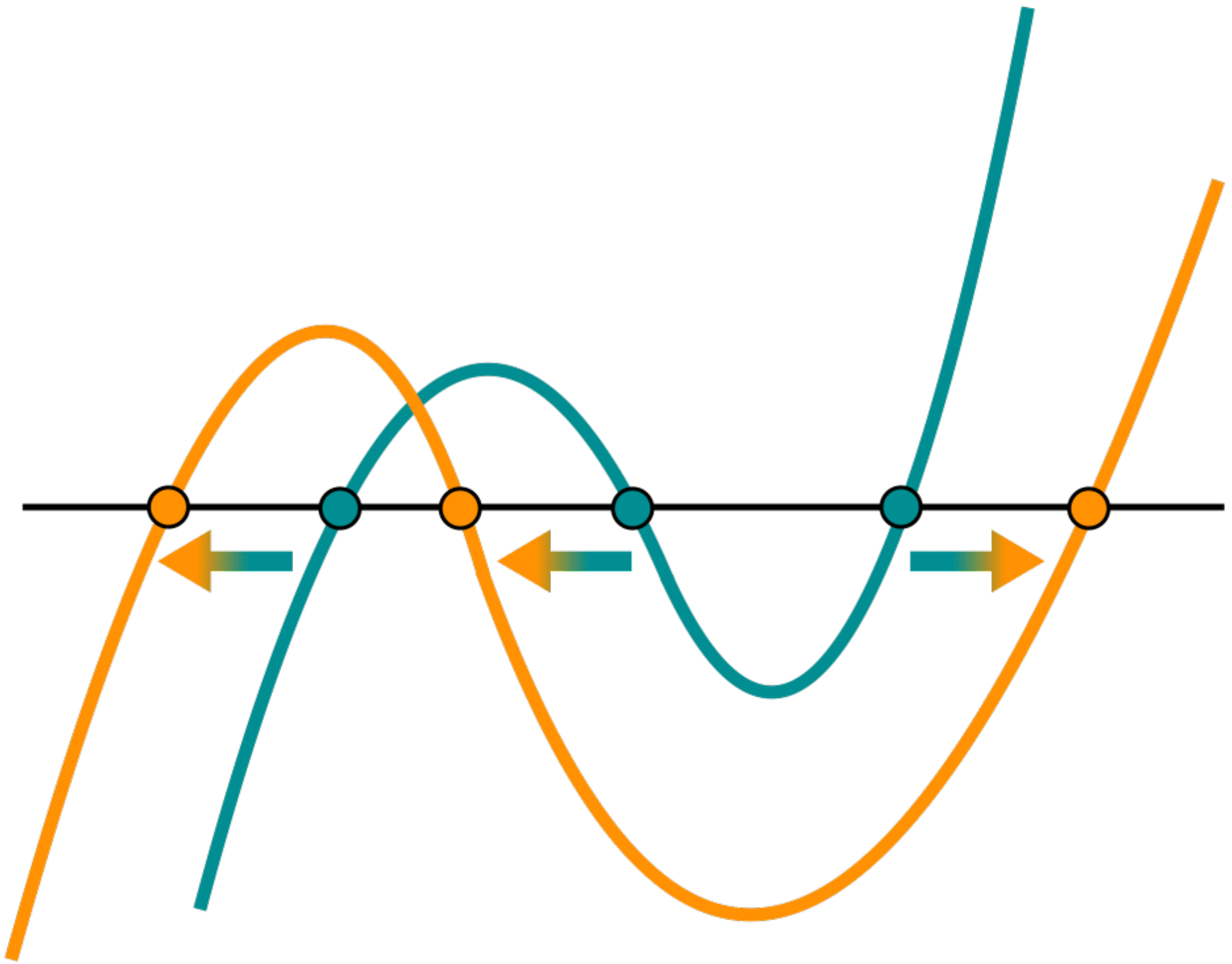
- Backflow correlations refer to the influence of all surrounding particles on the state of a given particle
- Jastrow factors encode some backflow correlations *but* they do not change the nodes of the fermionic wave function
- Backflow transformations typically alter the single-particle positions as

$$\mathbf{r}_i \mapsto \mathbf{r}_i + \sum_{j \neq i}^{N} \eta(r_{ij})(\mathbf{r}_i - \mathbf{r}_j)$$

- The new positions are used as input to the Slater determinant, allowing the nodes to change

Jastrow Factors vs. Backflow Transformations





VMC vs. DMC

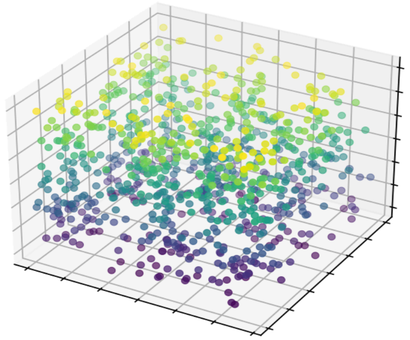
- Variational Monte Carlo is a direct application of the variational principle and Monte Carlo integration
- Relies on a “good” functional form for the trial wave function
- Higher bias than DMC, but intuitive meaning of the variational parameters
- No fermion sign problem
- Lower computational cost than DMC in standard applications

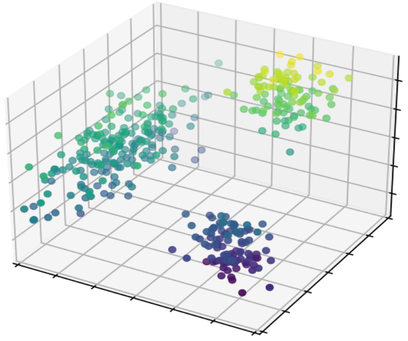
- Diffusion Monte Carlo (DMC) is a state-of-the-art method based on imaginary-time propagation
- Wave function is determined by the distribution of Monte Carlo walkers (no functional form)
- Relies on a good initial state, usually obtained through a VMC calculation
- Suffers from the sign problem, nodes must be fixed → residual dependence on VMC calculation

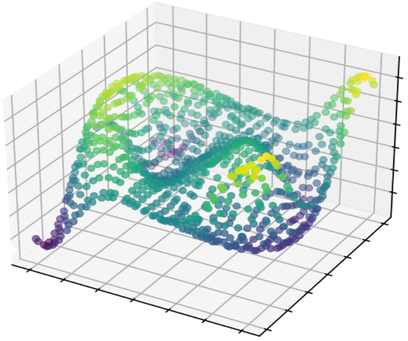
Machine Learning

Machine Learning

- Machine learning is a subset of artificial intelligence, a field that aims to endow computers with human-like cognition
- Encompasses a wide range of computational models, including neural networks
- Proven to be successful despite the **curse of dimensionality:** the density of a fixed number of data points decreases exponentially with the number of dimensions
- Why? The information content in a data set implies inherent structure

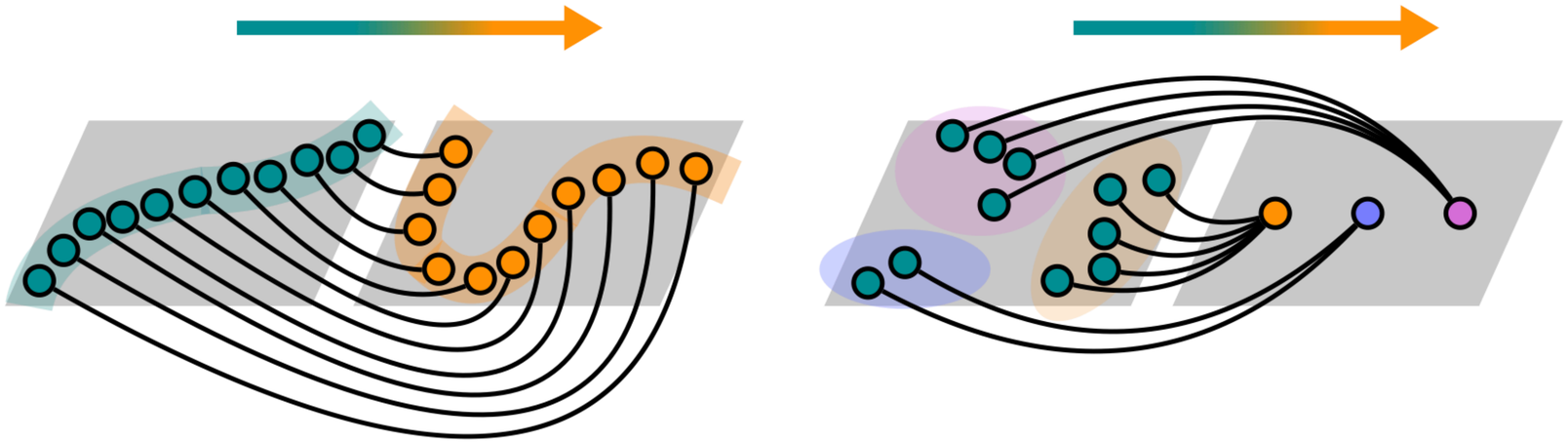






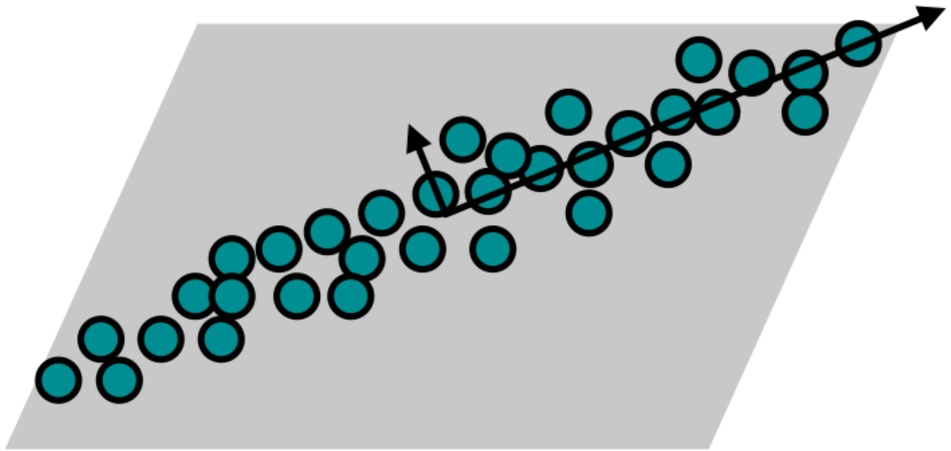
Supervised Learning

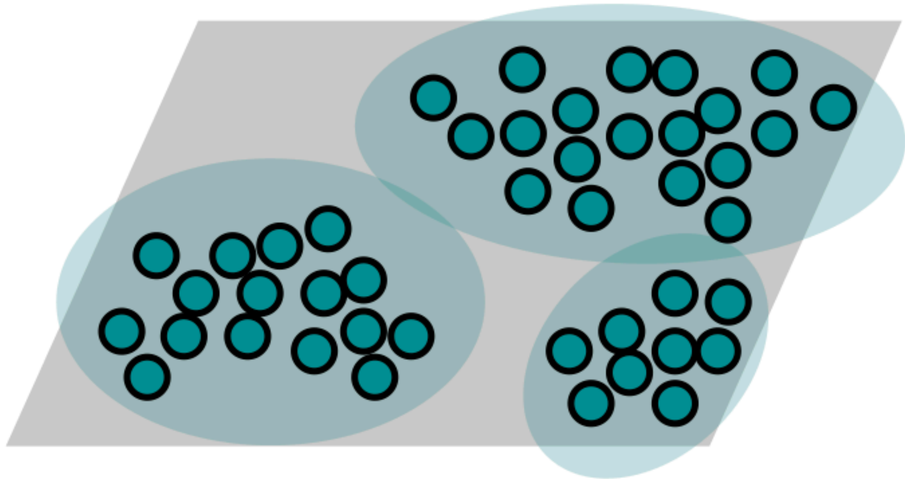
- Aims to find a mapping from the input data $\mathbf{x}$ to corresponding output values $\mathbf{y}$
- The cost function has the generic form $C(\mathbf{y}(\mathbf{x}), \hat{\mathbf{y}}(\mathbf{x}))$, where $\hat{\mathbf{y}}$ is the output of the supervised learning model
- Main categories: regression and classification



Unsupervised Learning

- Aims to learn the underlying structure of the data set $\mathbf{x}$
- Cost function only depends on inputs $C(\mathbf{x})$
- Main categories: clustering, dimensionality reduction, and anomaly detection



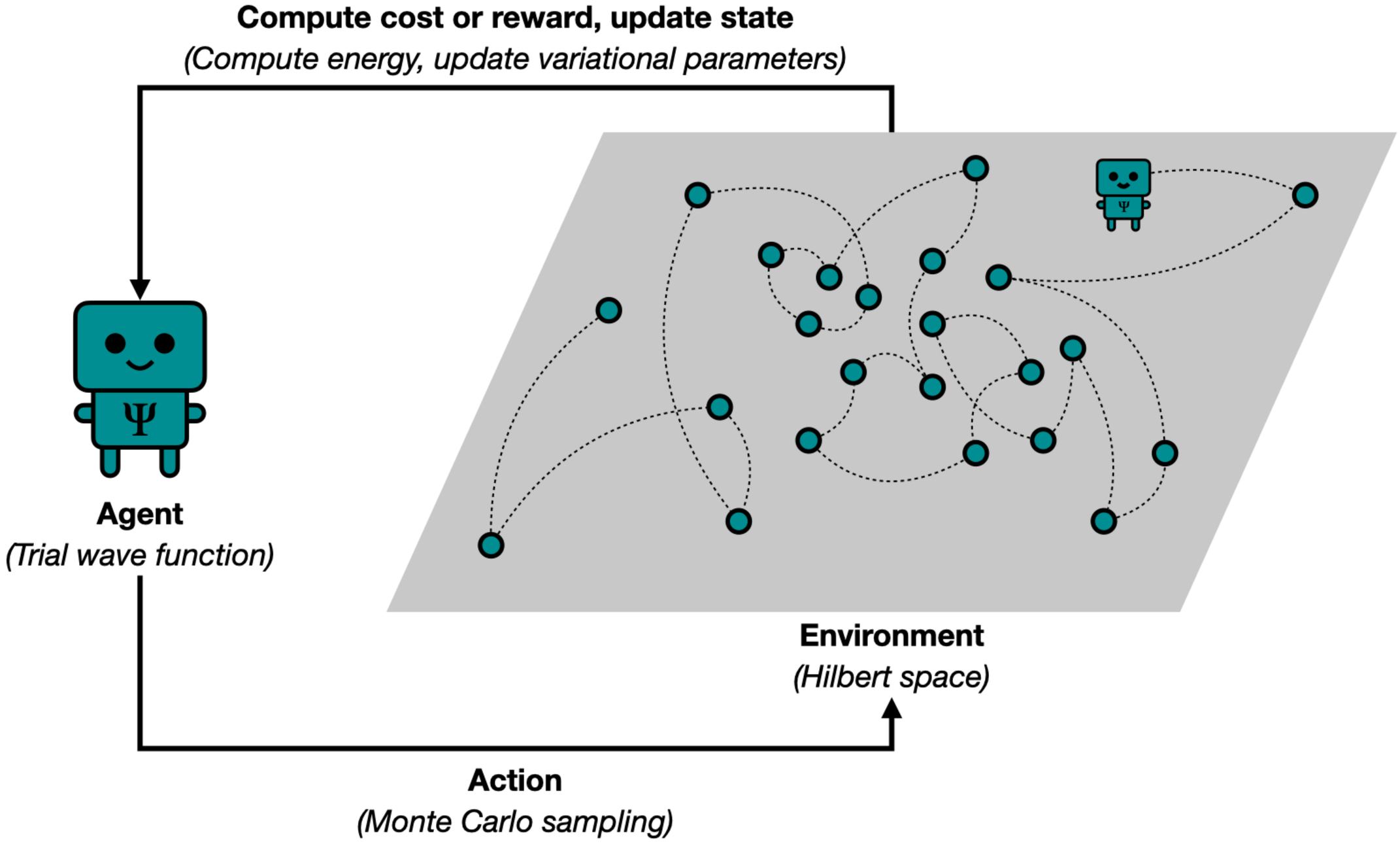


Reinforcement Learning

- Characterized by a data set that is dependent on the model itself
- Closest to how humans learn day-to-day (outside of the classroom)
- Cost function is recursive; it depends on all previous states of the model $\hat{\mathbf{y}}_i$ and data collected using that model $\mathbf{x}_i(\hat{\mathbf{y}}_i)$

$$C(\mathbf{x}_t(\hat{\mathbf{y}}_t(\mathbf{x}_{t-1}(\hat{\mathbf{y}}_{t-1}(\cdots(\mathbf{x}_0(\hat{\mathbf{y}}_0))\cdots)))))$$

- Commonly used to train robots in game-playing and navigation tasks


Compute cost or reward, update state
(Compute energy, update variational parameters)
Agent
(Trial wave function)
Environment
(Hilbert space)
Action
(Monte Carlo sampling)

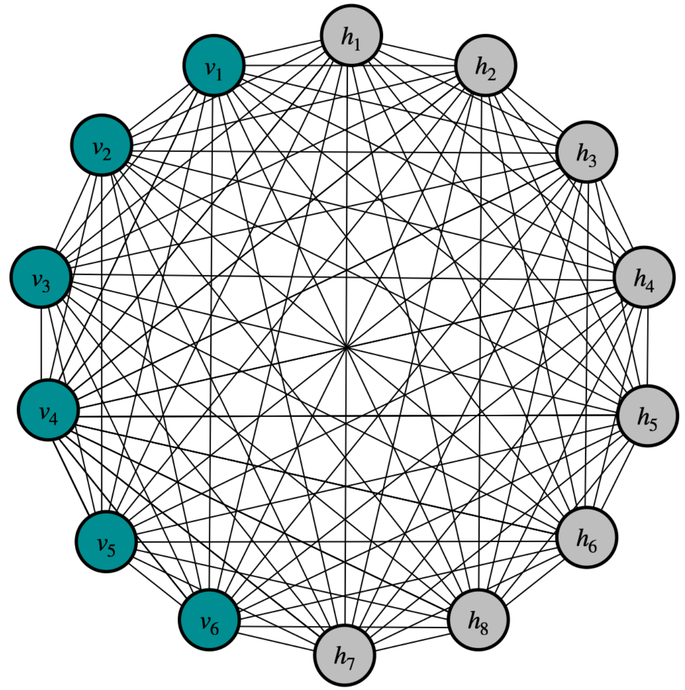
Artificial Neural Networks

Artificial Neural Networks

- Artificial neural networks (ANNs) are machine learning models inspired by the structure of the human brain, consisting of interconnected nodes (neurons) and nonlinear connections between them (synapses)
- The term is often used interchangeably with feedforward neural networks, the most common type of artificial neural network
- ANNs typically require a large amount of training data, only use when necessary
- Generally have low bias compared to other machine learning models, but there is still wide variation between different ANNs

Boltzmann Machines

- Generative stochastic artificial neural networks
- Typically used to learn the probability distribution of input data
- Two types of nodes that are fully connected:
  - **Visible nodes:** represents the inputs to the network
  - **Hidden nodes:** latent variables that capture higher-level features or patterns in the data
- General Boltzmann machines have limited applicability due to inefficient training

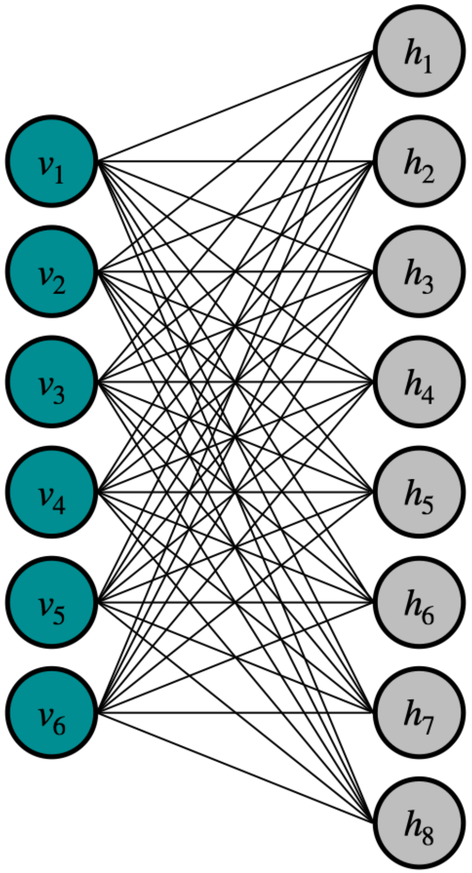


Restricted Boltzmann Machines

- A restricted Boltzmann machine (RBM) has no connections between nodes of the same type
- For continuous problems, the visible nodes are Gaussian and the hidden nodes are binary
- RBMs are constructed by defining an energy-like quantity

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{V} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{H} b_j h_j - \sum_{i=1}^{V} \sum_{j=1}^{H} \frac{v_i}{\sigma_i^2} W_{ij} h_j$$

- Reparameterize the inverse variances as $1/\sigma_i^2 = \exp(s_i)$



Restricted Boltzmann Machines

- Taking the Boltzmann distribution yields a corresponding joint probability distribution

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}$$

- Construct the marginal probability distribution of the visible nodes by summing over the possible values of the hidden nodes

$$P(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h})$$

- Since the wave function is a probability amplitude, RBMs are a natural choice for a neural-network quantum state $\Psi_\theta(\mathbf{X}) = P(\mathbf{v}(\mathbf{X}))$

Multivariate Gaussian-Binary RBMs

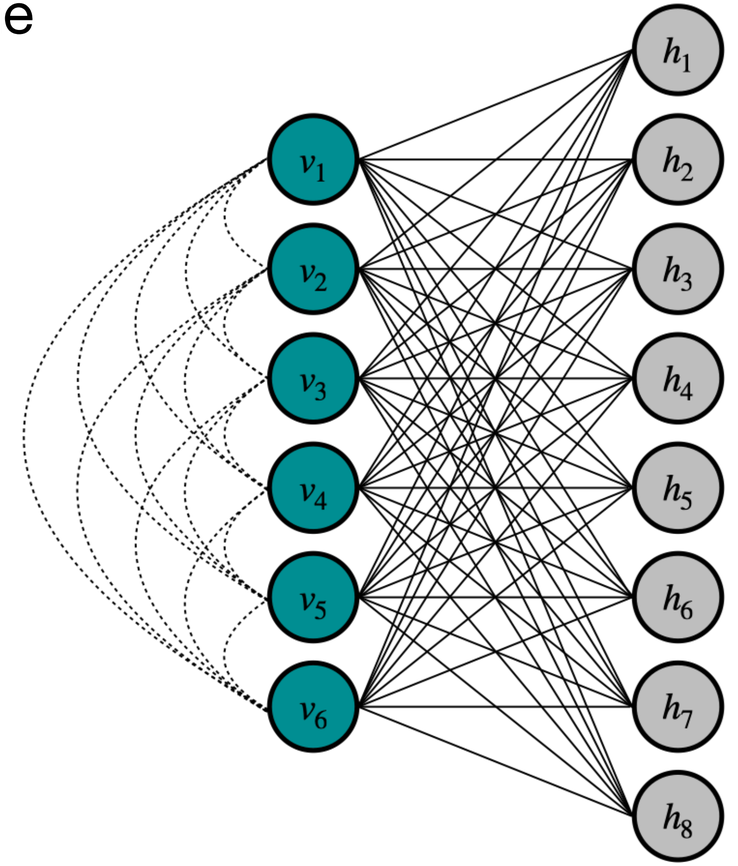
- A multivariate Gaussian-binary restricted Boltzmann Machine (mGB-RBM) is halfway a standard Gaussian-Binary restricted Boltzmann Machine and a general Boltzmann machine

- Math becomes easier to write when connections between visible nodes are allowed

- Energy of a node configuration:

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2}(\mathbf{v} - \mathbf{a})^T \Sigma^{-1} (\mathbf{v} - \mathbf{a}) - \mathbf{b}^T \mathbf{h} - \mathbf{v}^T \Sigma^{-1} W \mathbf{h},$$

where $\Sigma \equiv \mathrm{Cov}(\mathbf{v}, \mathbf{v})$



Multivariate Gaussian-Binary RBMs

- The covariance matrix is reparameterized as

$$\Sigma_{ij}^{-1} = \begin{cases} \exp(S_{ij}), & \text{for } i = j, \\ S_{ij}, & \text{for } i \neq j\,. \end{cases}$$

- The log-likelihood becomes

$$\log P(\mathbf{v}) = -\frac{1}{2}(\mathbf{v}-\mathbf{a})^T\Sigma^{-1}(\mathbf{v}-\mathbf{a}) + \sum_{j=1}^{H} f(z_j(\mathbf{v})),$$

where $\mathbf{z}(\mathbf{v}) = \mathbf{b} + W^T\Sigma^{-1}\mathbf{v}$ and

$$f(x) \equiv \log(1+\exp(x))$$ (softplus function)

Other Variations of Continuous RBMs

- If hidden binary nodes are allowed to take values of -1 and 1 (instead of 0 and 1), then

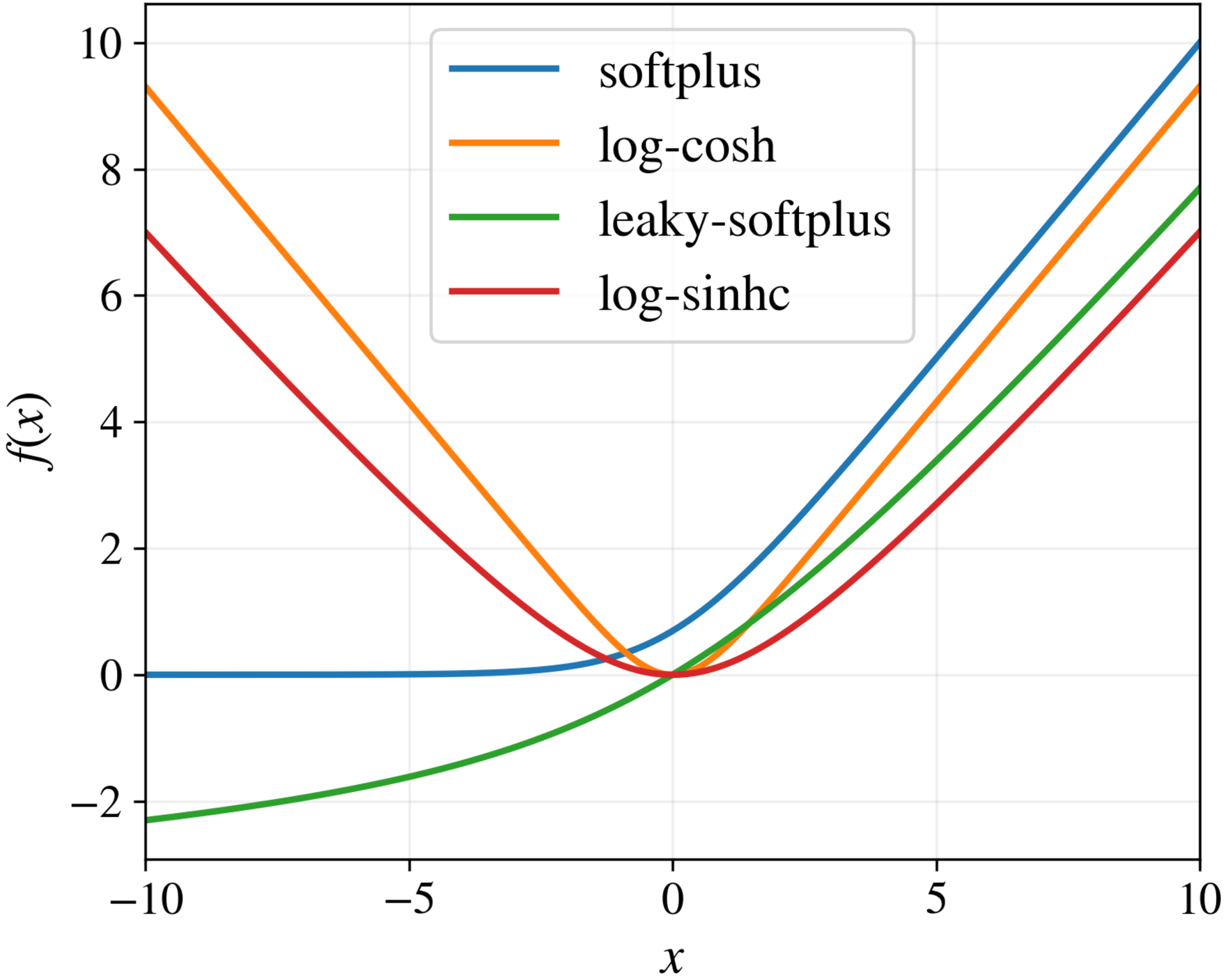
$$f(x) \equiv \log(\cosh(x))$$ (log-cosh function)

- If hidden nodes are continuous-valued between 0 and 1, then

$$f(x) \equiv \log \frac{\exp(x) - 1}{x}$$ (leaky-softplus function)

- If hidden nodes are continuous-valued between -1 and 1, then

$$f(x) \equiv \log \frac{\sinh(x)}{x}$$ (log-sinhc function)

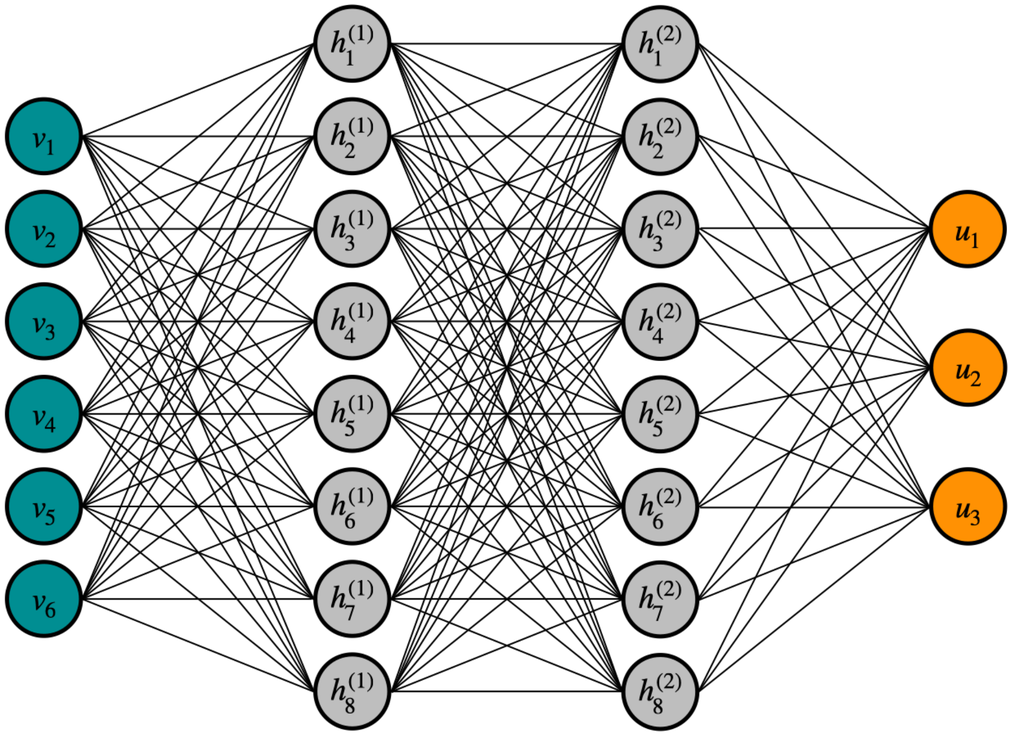

softplus
log-cosh
leaky-softplus
log-sinhc
10
8
6
4
2
0
−2
f(x)
−10
−5
0
5
10
x

Feedforward Neural Networks

- Feedforward neural networks (FNNs) are the most widely used type of artificial neural network

- **Universal approximation theorem:** a shallow FNN can approximate any continuous function on a compact subset of Euclidean space, given there are a sufficient number of hidden nodes and the activation function satisfies certain conditions.

- Unlike RBMs, FNNs have explicit outputs, connections are directed, but trainable parameters are difficult to interpret



Feedforward Neural Networks

- Alternating compositions of affine transformations and simple, nonlinear transformations

$$\mathbf{h}^{(0)} = \mathbf{v},$$
$$\mathbf{h}^{(\ell)} = \mathbf{f}_\ell\left(W^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \text{ for } \ell = 1, 2, \ldots, L$$

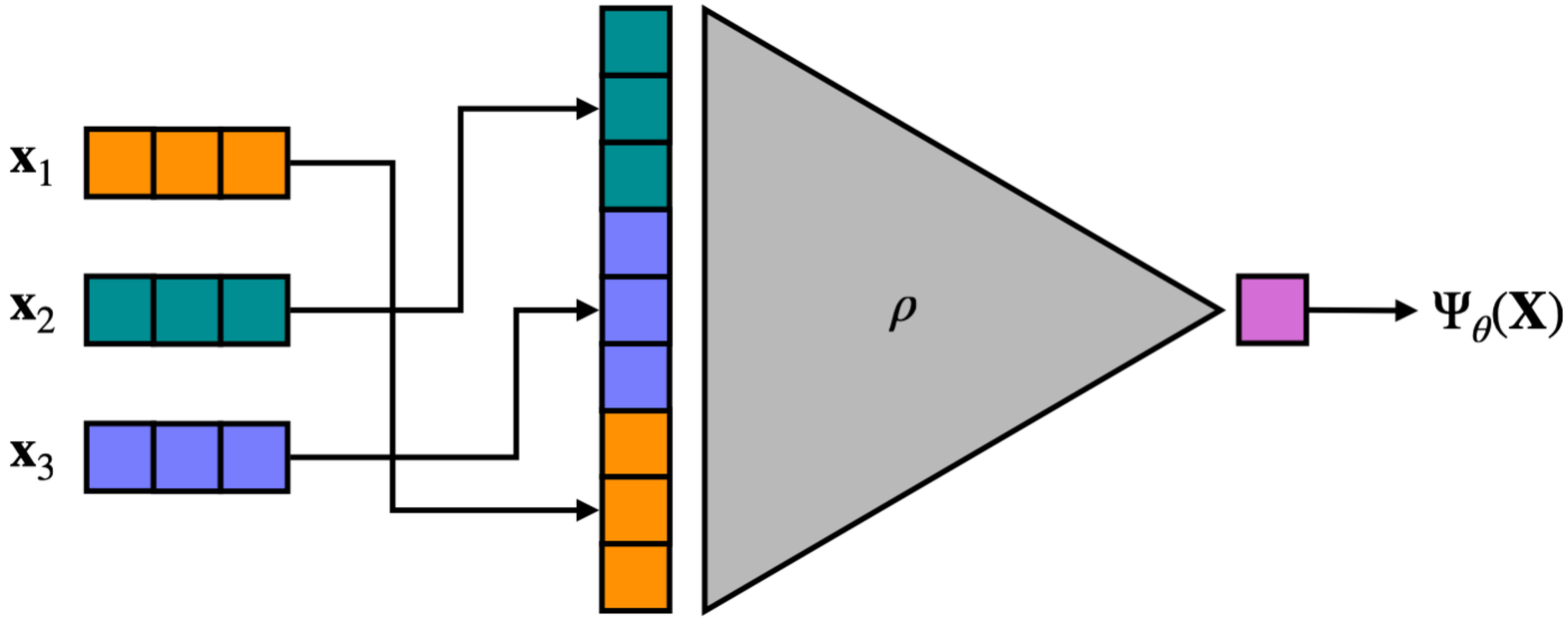
Activation function

- Straightforward to write down the derivatives of the outputs w.r.t. the weights and biases using the chain rule, e.g.

$$\frac{\partial \mathbf{h}^{(L)}}{\partial W^{(\ell)}} = \frac{\partial \mathbf{h}^{(L)}}{\partial \mathbf{h}^{(L-1)}}\frac{\partial \mathbf{h}^{(L-1)}}{\partial \mathbf{h}^{(L-2)}}\cdots\frac{\partial \mathbf{h}^{(\ell+1)}}{\partial \mathbf{h}^{(\ell)}}\frac{\partial \mathbf{h}^{(\ell)}}{\partial W^{(\ell)}}$$

Permutation Symmetry

- Permutation symmetry can be enforced in two ways:
  - Data preprocessing: sort the inputs according to a fixed rule
  - Architecture design: set structure is ingrained into the network itself



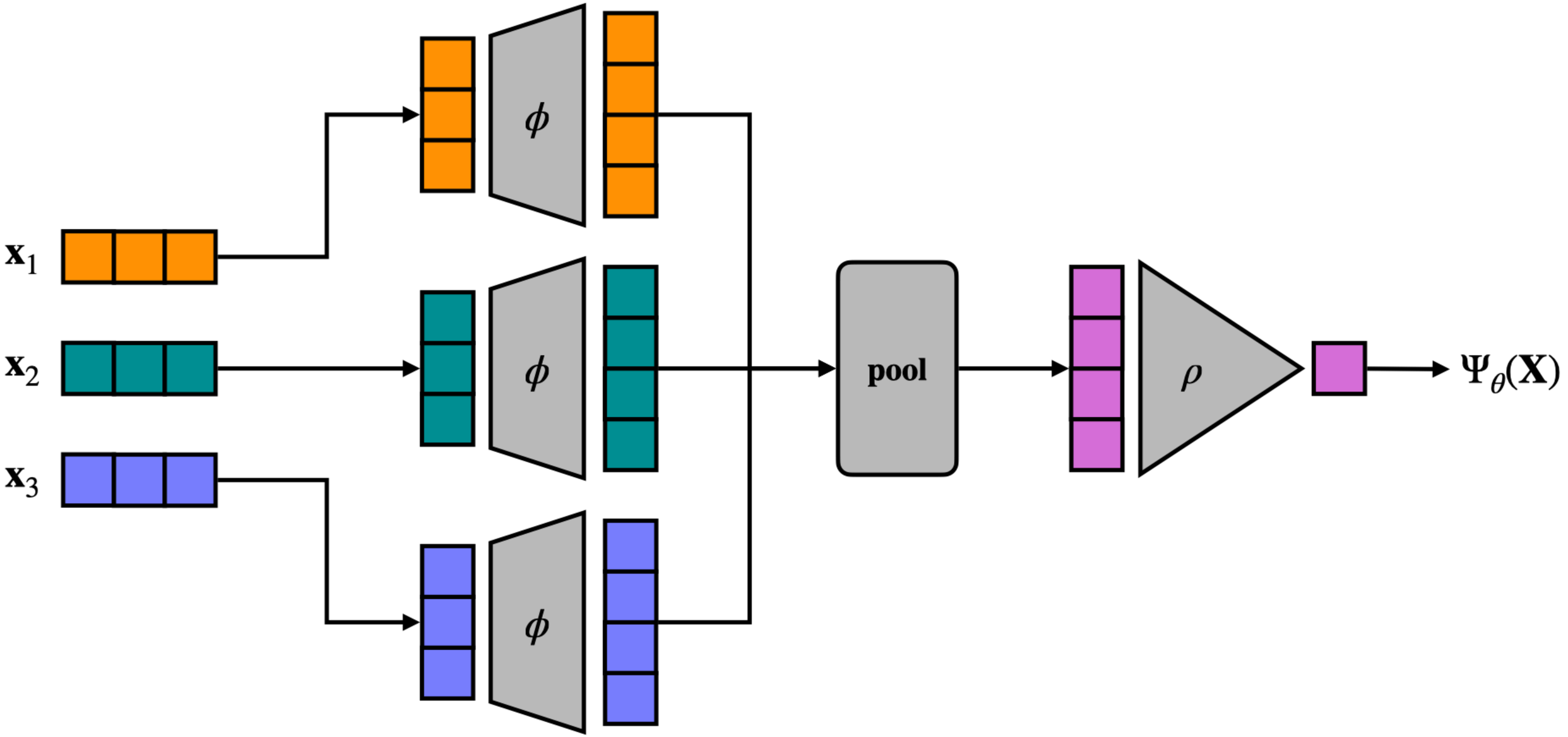
Deep Sets

- Introduced by Zaheer et al. in 2017 (arXiv:1703.06114)
- Suppose we want to learn on sets of N elements $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\}$
- Any permutation-invariant function can be decomposed as

$$f(\{\mathbf{v}_i\}) = \rho\left(\text{pool}(\{\phi(\mathbf{v}_i)\})\right)$$

Pooling operation destroys ordering

- If ρ and ϕ are feedforward neural networks, then $f(\{\mathbf{v}_i\})$ is a universal approximator for permutation-invariant functions

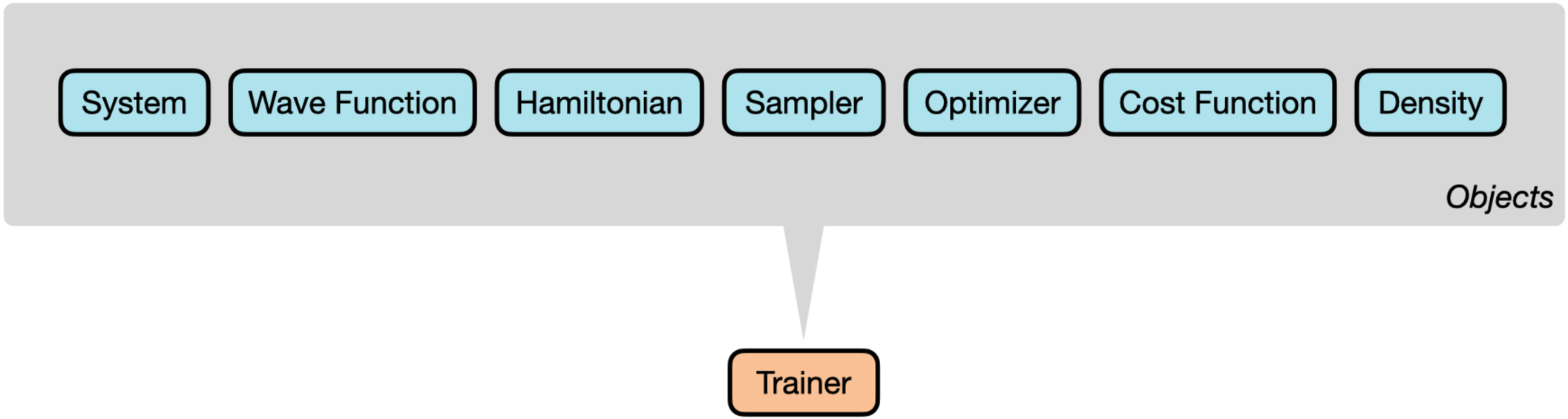

$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
ϕ
ϕ
ϕ
pool
ρ
$\Psi_\theta(\mathbf{X})$

Implementations

NeuralAnsatz

- Object-oriented C++ software written from scratch on top of a lightweight, header-only linear algebra library called Eigen, parallelized with Open MPI

- Prioritizes modularity, reproducibility, low memory costs, and exact gradient calculations



NeuralAnsatz

- Can handle localized systems of bosons and fermions (periodic systems and nucleons are implemented but not yet validated)
- Available input formats: sorted and unsorted one- and two-body inputs
- Available wave functions: all variations of continuous RBMs, deep feedforward neural networks, deep sets (need validation), Slater determinants, Gaussian envelope, and product wrapper
- Local kinetic energy and gradients w.r.t. parameters computed exactly in terms of the trainable parameters
- For numerical stability, always use long double precision numbers, all computations are in terms of $\log \Psi_\theta$, and singularities in Hamiltonian are regularized.

NeuralAnsatz

- Brute force Metropolis sampler and Importance sampler both perturb all particles in one Monte Carlo step, and exchange spin to preserve total spin projection S_z
- Various optimizers are included, but simple stochastic gradient descent works best with stochastic reconfiguration
- Cost functions include the energy with stochastic reconfiguration (for reinforcement learning) and the overlap integral (for supervised learning)
- Density class computes one- or two-body distribution snapshots during training
- **Trainer can work for both supervised and reinforcement learning**

NeuralAnsatz

```
// main.cpp
mpi::initialize();
rng::initialize();


std::shared_ptr<Object> pObject = std::make_shared<Object>(…);


Trainer T(pSystem, pWaveFunc, pSampler, pOptimizer,
          pCostFunc, pDensity, filename);
T.train();
```

Initializes MPI and automatically finalizes MPI when program is finished

Initializes random number generator with a unique seed on each parallel process

Construct pointers to all required objects like this

Place arguments here

Trainer takes all pointers to required objects

Python/JAX Implementation

- Software initially developed by Alessandro Lovato for nuclei, and extended by Bryce Fore and myself for periodic systems
- JAX is an open-source software that combines the flexibility and ease-of-use of numpy with the efficiency of compiled languages like C++
  - Compute gradients with **grad**
  - Just-in-time compilation with **jit**
  - Vectorization with **vmap**
  - Parallelization with **pmap**
- Numerical differentiation provides far more freedom to construct more complex neural networks

Python/JAX Implementation

- Constructing deep feedforward neural networks with JAX is easy:

```
init_func, apply_func = stax.serial(*layers)
```

List of layer objects (e.g. stax.Dense, stax.Gelu, etc…)

- The “init” functions initialize the weight and biases of the network, given some input dimension

- The “apply” functions compute the output of the network given a set of parameters and input data

Python/JAX Implementation

- To create Deep Sets, just need to create two feedforward neural networks:

```
lat_init, lat_apply = stax.serial(stax.Dense(n), stax.Gelu, stax.Dense(n))
out_init, out_apply = stax.serial(stax.Dense(n), stax.Gelu, 1)
```

- Numpy-like syntax makes implementation of Deep Set easy:

```
phi = self.lat_apply(phi_params, X)
phi = jnp.sum(phi, axis=0)
logpsi = self.out_apply(rho_params, phi)
```

- Almost all artificial neural networks can be constructed by appropriately combining dense feedforward neural networks with the appropriate pooling and concatenation operations

Applications

Calogero-Sutherland Model

- Exactly-solvable model of strongly-interacting bosons in a one-dimensional harmonic oscillator trap

- Excellent basis to assess neural-network quantum states:

  - Learn how to handle the singularity without enforcing the cusp condition by hand

  - One-dimensional systems are easy to visualize

  - Benchmark with exact solutions for the ground state energy *and* the ground state wave function

Calogero-Sutherland Model

- Hamiltonian ($\hbar = m = 1$):

$$\hat{H} = \sum_{i=1}^{N} \left(-\frac{1}{2}\frac{\partial^2}{\partial x_i^2} + \frac{1}{2}\omega^2 x_i^2 \right) + \sum_{i<j}^{N} \frac{\beta(\beta-1)}{x_{ij}^2}$$

- Exact solutions:

$$\Psi_0(\mathbf{X}) = \exp\left(-\sum_{i=1}^{N} \frac{1}{2}\omega x_i^2 \right) \prod_{i<j} x_{ij}^{\beta}$$

$$E_0 = \frac{1}{2}N\omega + \frac{1}{2}\beta N(N-1)\omega$$

Throughout this work,
$N = 6$ and $\beta = 2$

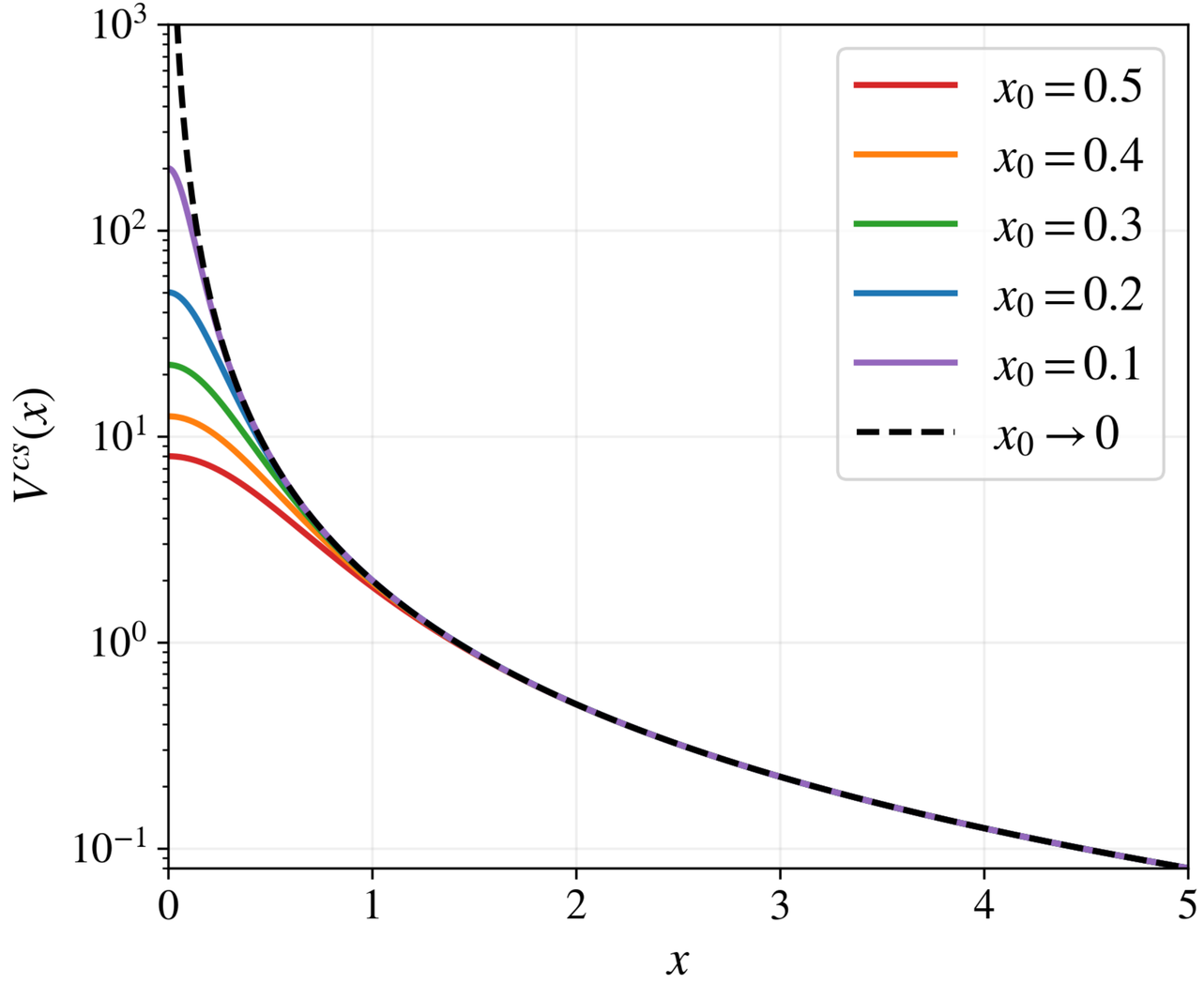
Calogero-Sutherland Model

- Regularize singularity:

$$\frac{1}{x^2} \mapsto \frac{\tanh^2(x/x_0)}{x^2}$$

- Transfer learning:

  Quickly train networks on very soft interaction ($x_0 = 0.5$) before moving on to harder interactions ($x_0 = 0.01$)



Calogero-Sutherland Model

- Continuous RBMs:

$$\Psi_\theta^{rbm}(\mathbf{X}) = P(\mathbf{v}(\mathbf{X}))$$

- FNNs with a very wide Gaussian envelope ($\sigma = 16$):

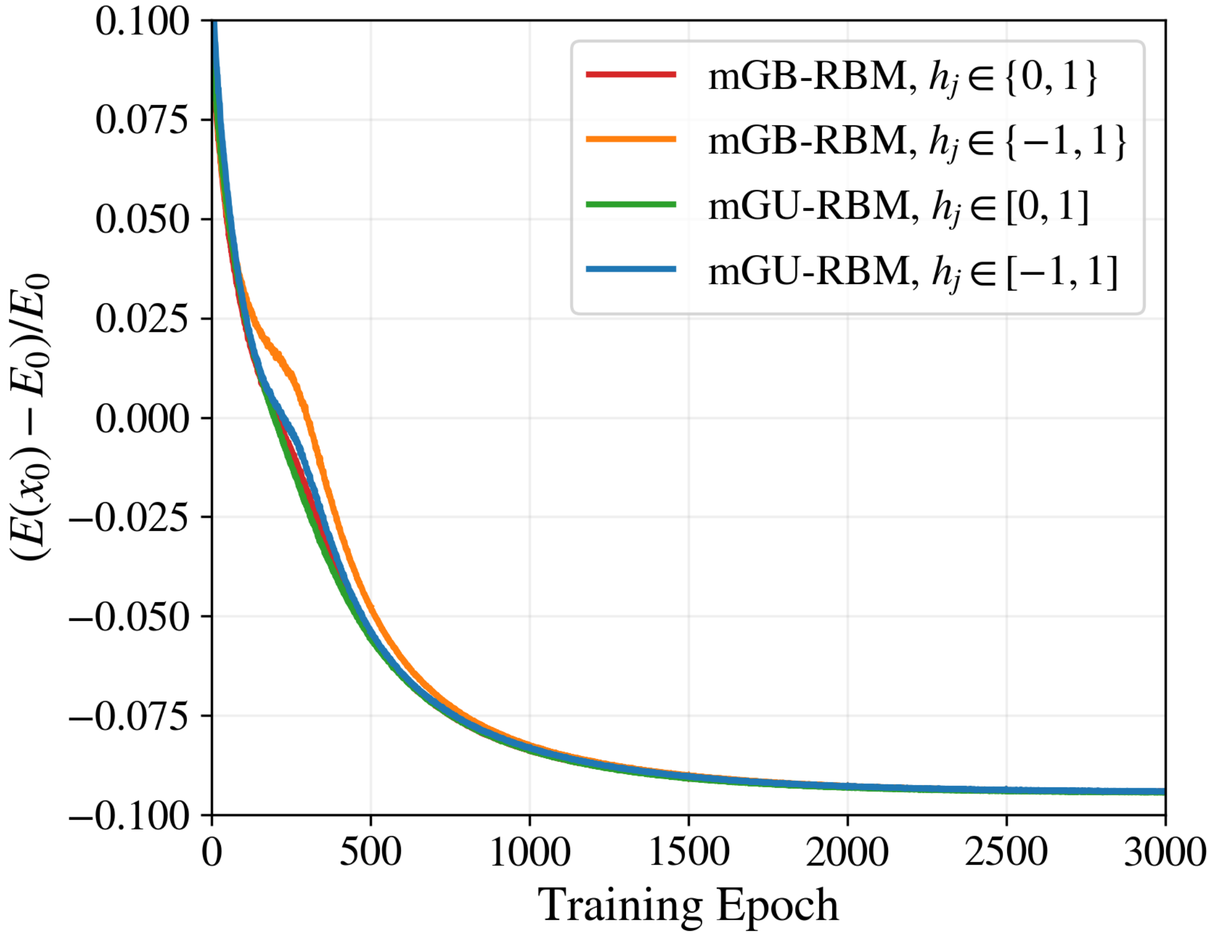
$$\Psi_\theta^{fnn}(\mathbf{X}) = \exp\Big(\rho(\mathbf{v}(\mathbf{X}))\Big)\exp\left(-\frac{\mathbf{X}^2}{2\sigma^2}\right)$$

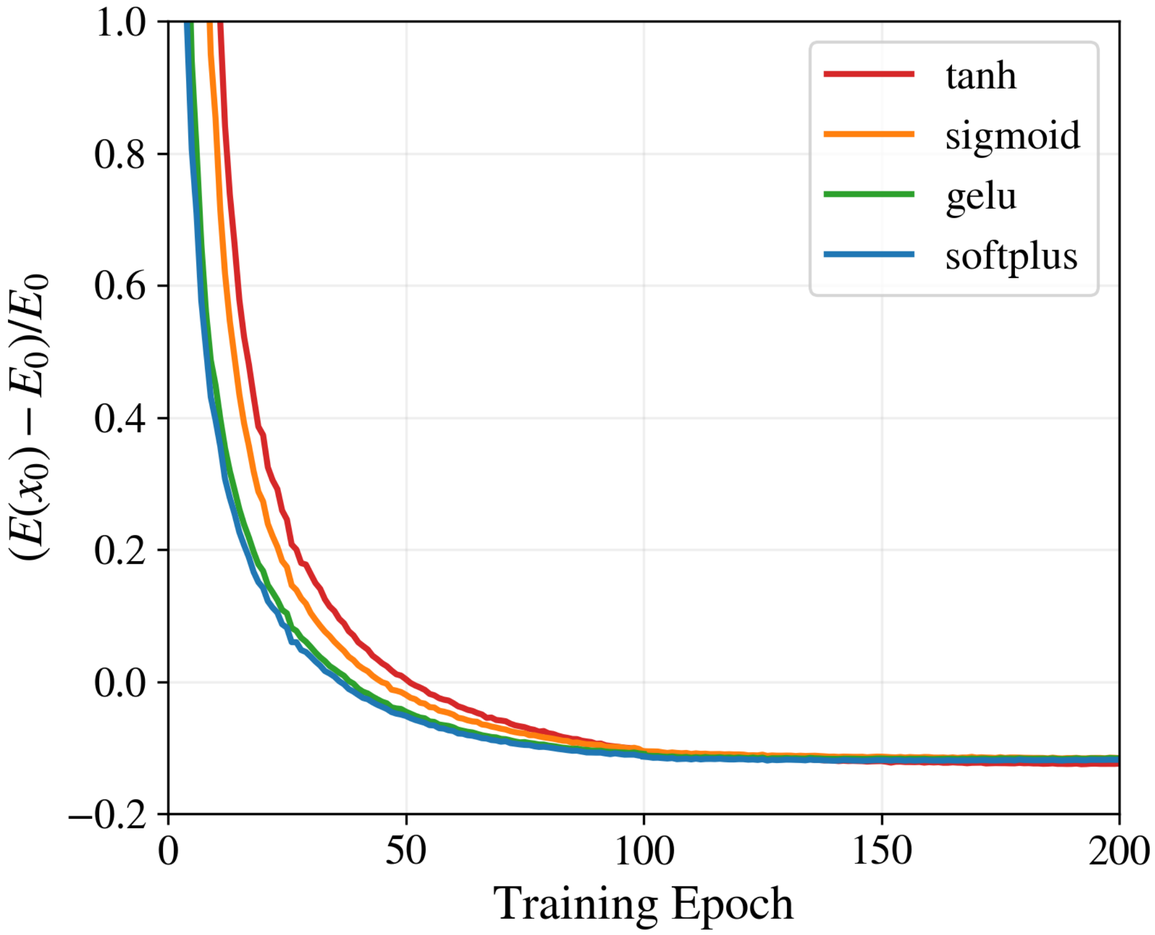
(Alternatively can pretrain the FNN to a Gaussian using supervised learning)

- $\mathbf{v}(\mathbf{X})$ denotes the visible nodes after sorting the input data

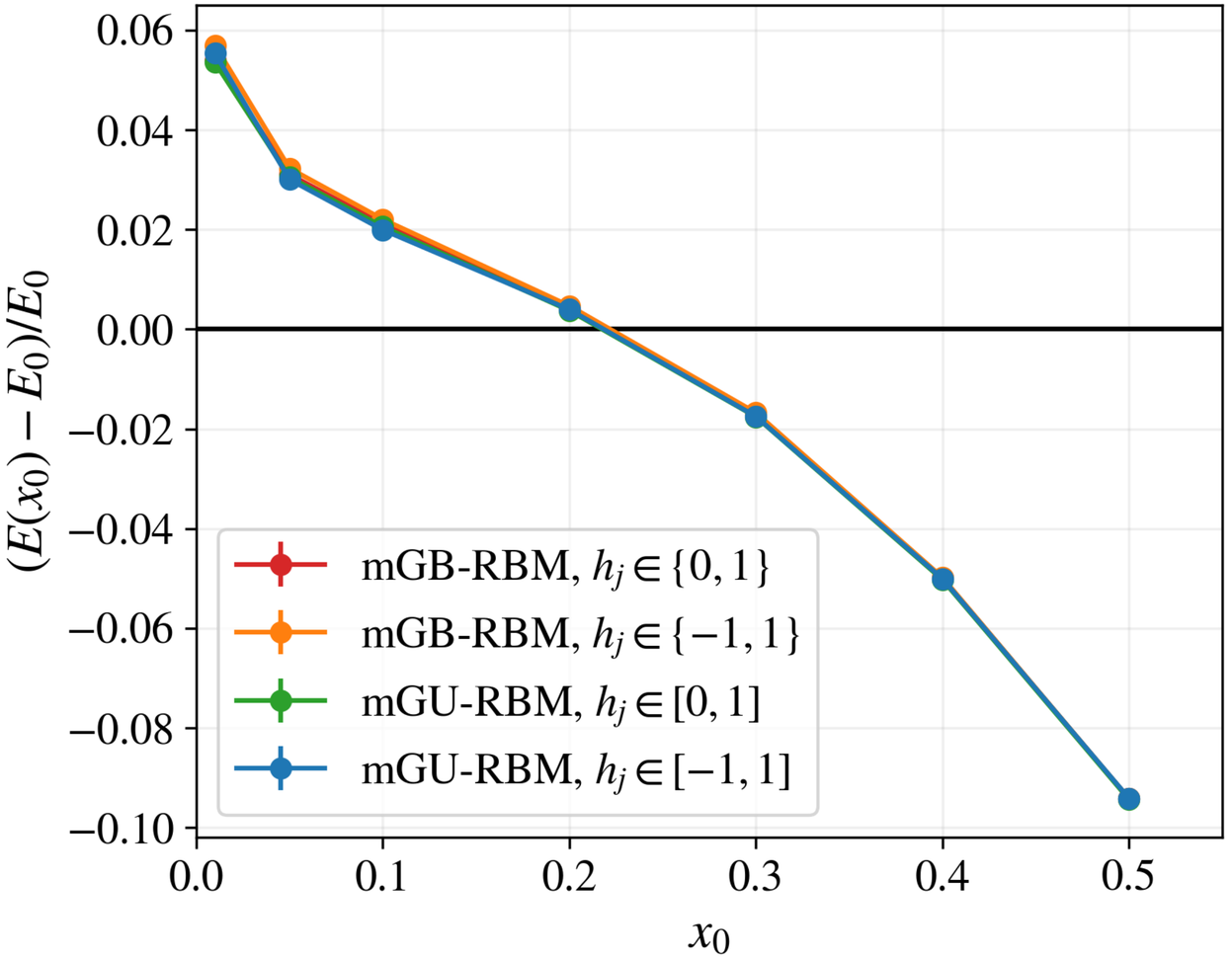
Calogero-Sutherland Model

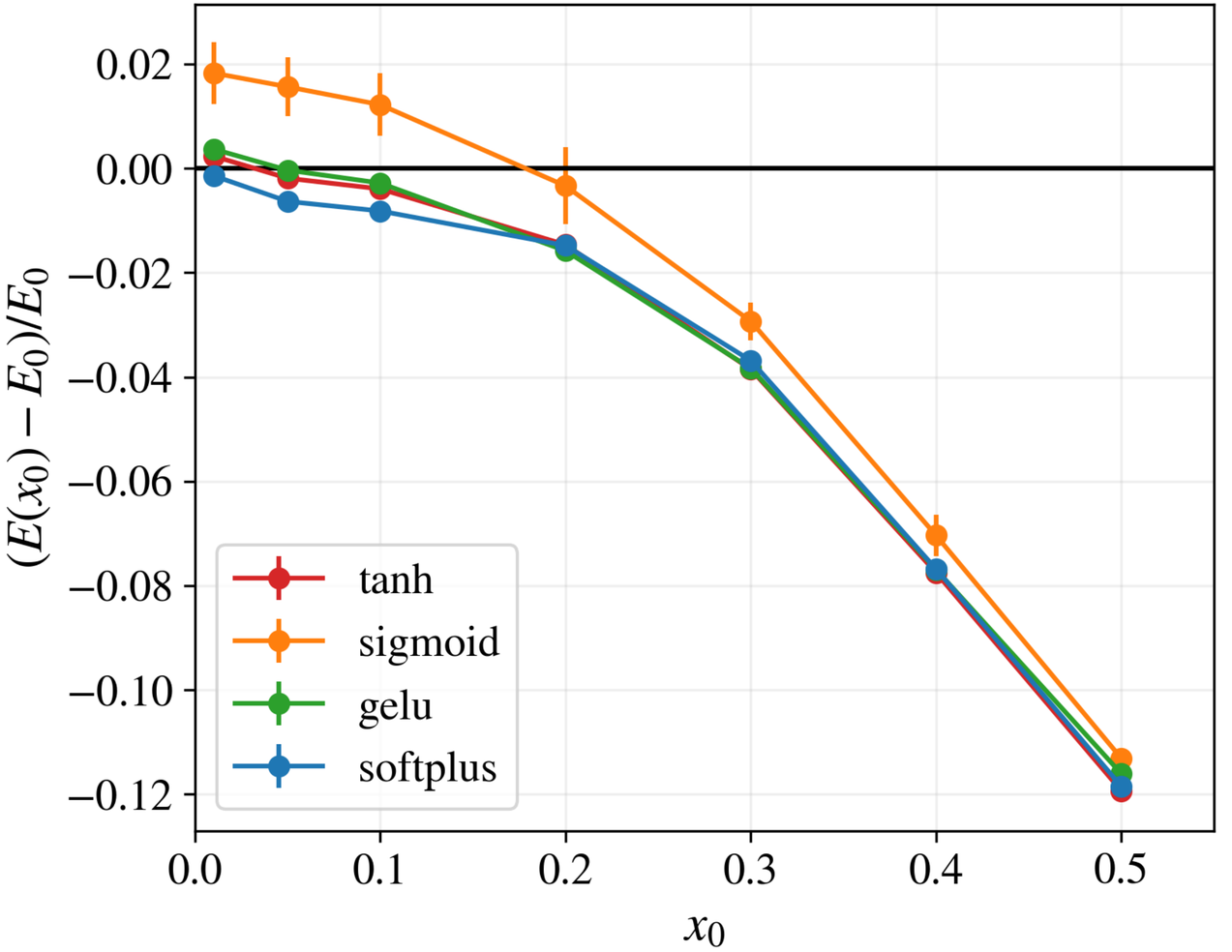
Initial pretraining stage with $x_0 = 0.5$

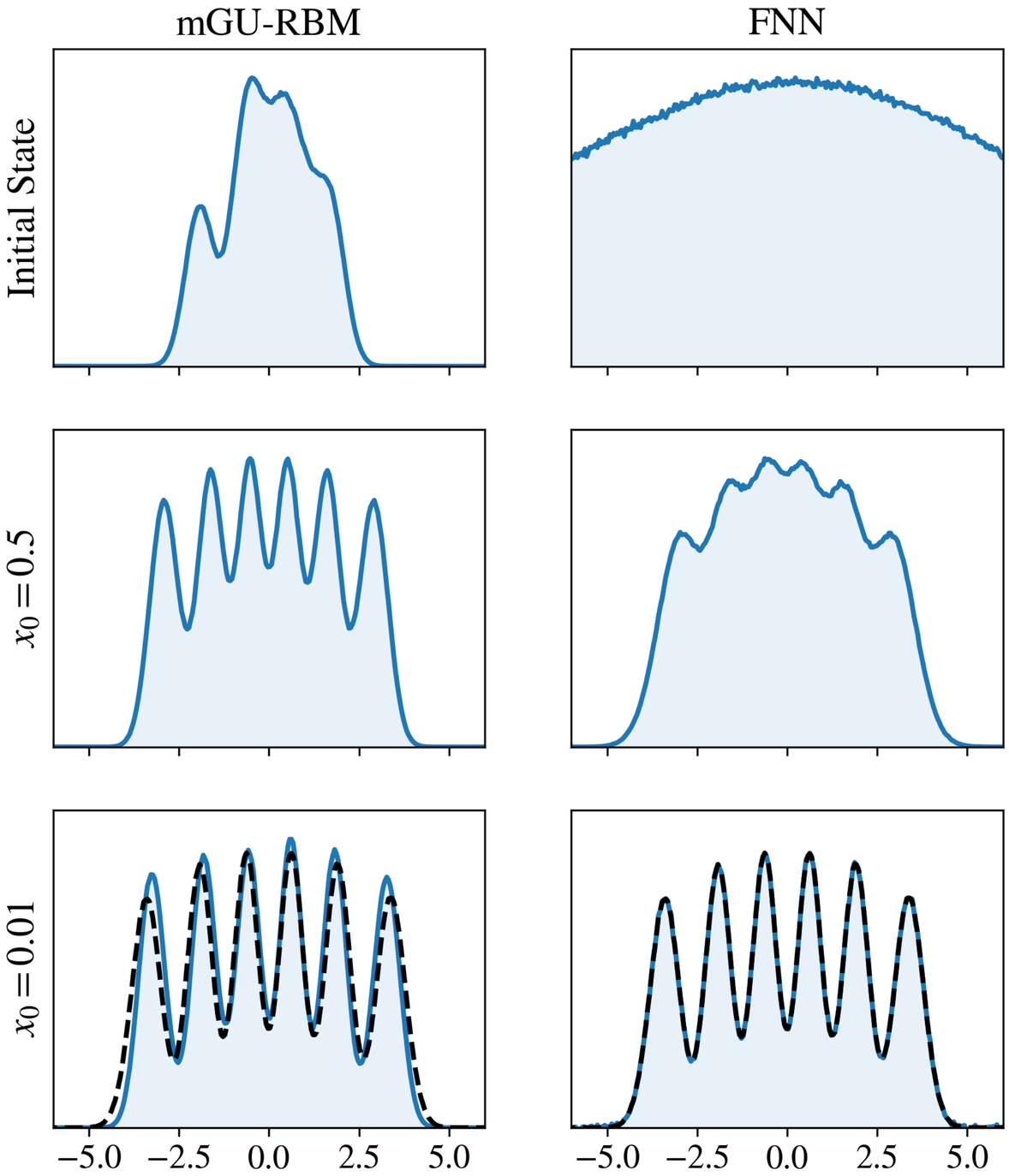




Calogero-Sutherland Model






mGU-RBM
FNN
Initial State
$x_0 = 0.5$
$x_0 = 0.01$
−5.0
−2.5
0.0
2.5
5.0

Calogero-Sutherland Model

- The different RBMs perform similarly, but the Gaussian-uniform variants perform slightly better at smaller values of x_0
- The assumption of Gaussian visible nodes biases results of RBMs
- FNNs provide more accurate results, but generally require more parameters to distinguish the individual peaks of the wave function (250 params vs. 70 params)
- Regularizing the singularity and using transfer learning is a controlled way of handling the singularity without enforcing the cusp condition

Dilute Neutron Matter

- Recently accepted for publication in Physical Review Research
- See our preprint titled "**Dilute neutron star matter from neural-network quantum states**" by B. Fore, J. Kim, G. Carleo, M. Hjorth-Jensen, and A. Lovato (arXiv:2212.04436, uploaded Dec 2022)
- We focus on lower densities $\rho \lesssim 0.08\ \mathrm{fm}^{-3}$ relevant to our understanding of the inner crust at the boundary of outer core in neutron stars
- In this density region, neutron star matter consists of a Coulomb lattice of neutron-rich nuclei, relativistic electrons, and a superfluid neutron gas

Dilute Neutron Matter

- Leading order pionless-EFT Hamiltonian ($\hbar = 1$):

$$\hat{H} = -\sum_{i=1}^{N} \frac{\nabla_i^2}{2m} + \sum_{i<j} v_{ij} + \sum_{i<j<k} V_{ijk}$$

Non-relativistic kinetic energy

NN contact potential

3N contact potential

$$v = v^{EM} + v_{LO}^{CI}$$

$$v_{LO}^{CI} = C_{01} C_1(r_{ij}) P_0^\sigma P_1^\tau + C_{10} C_0(r_{ij}) P_1^\sigma P_0^\tau$$

$$V_{ijk} = \frac{c_E}{f_\pi^4 \lambda_\chi} \frac{(\hbar c)^6}{\pi^3 R_3^6} \sum_{cyc} e^{-(r_{ij}^2 + r_{jk}^2)/R_3^2}$$

Dilute Neutron Matter

- Simulate 14 neutrons in a periodic box of side length L
- Enforce periodicity by transforming the coordinates as

$$\mathbf{r}_i \mapsto \left(\cos\left(\frac{2\pi \mathbf{r}_i}{L}\right), \ \sin\left(\frac{2\pi \mathbf{r}_i}{L}\right) \right)$$

- Mitigate finite-size effects:
  - Sum over contributions from neighboring cells
  - Subtract 14 particle kinetic energy, add kinetic energy in thermodynamic limit

Dilute Neutron Matter

- We use an ansatz belonging to the “hidden fermion” family of states introduced by J. R. Moreno et al. (arXiv:2111.10420)

- Introduce M fictitious “hidden” nucleons that are constructed using Deep Sets over the “visible” nucleons

$$\Psi_\theta^{HN}(\mathbf{X}) = \det \begin{bmatrix} \phi_1(\mathbf{x}_1) & \cdots & \phi_1(\mathbf{x}_N) & \phi_1(\mathbf{y}_1) & \cdots & \phi_1(\mathbf{y}_M) \\ \phi_2(\mathbf{x}_1) & \cdots & \phi_2(\mathbf{x}_N) & \phi_2(\mathbf{y}_1) & \cdots & \phi_2(\mathbf{y}_M) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \phi_N(\mathbf{x}_1) & \cdots & \phi_N(\mathbf{x}_N) & \phi_N(\mathbf{y}_1) & \cdots & \phi_N(\mathbf{y}_M) \\ \chi_1(\mathbf{x}_1) & \cdots & \chi_1(\mathbf{x}_N) & \chi_1(\mathbf{y}_1) & \cdots & \chi_1(\mathbf{y}_M) \\ \chi_2(\mathbf{x}_1) & \cdots & \chi_2(\mathbf{x}_N) & \chi_2(\mathbf{y}_1) & \cdots & \chi_2(\mathbf{y}_M) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \chi_N(\mathbf{x}_1) & \cdots & \chi_N(\mathbf{x}_N) & \chi_N(\mathbf{y}_1) & \cdots & \chi_N(\mathbf{y}_M) \end{bmatrix}$$

Visible s.p. states
Visible nucleons

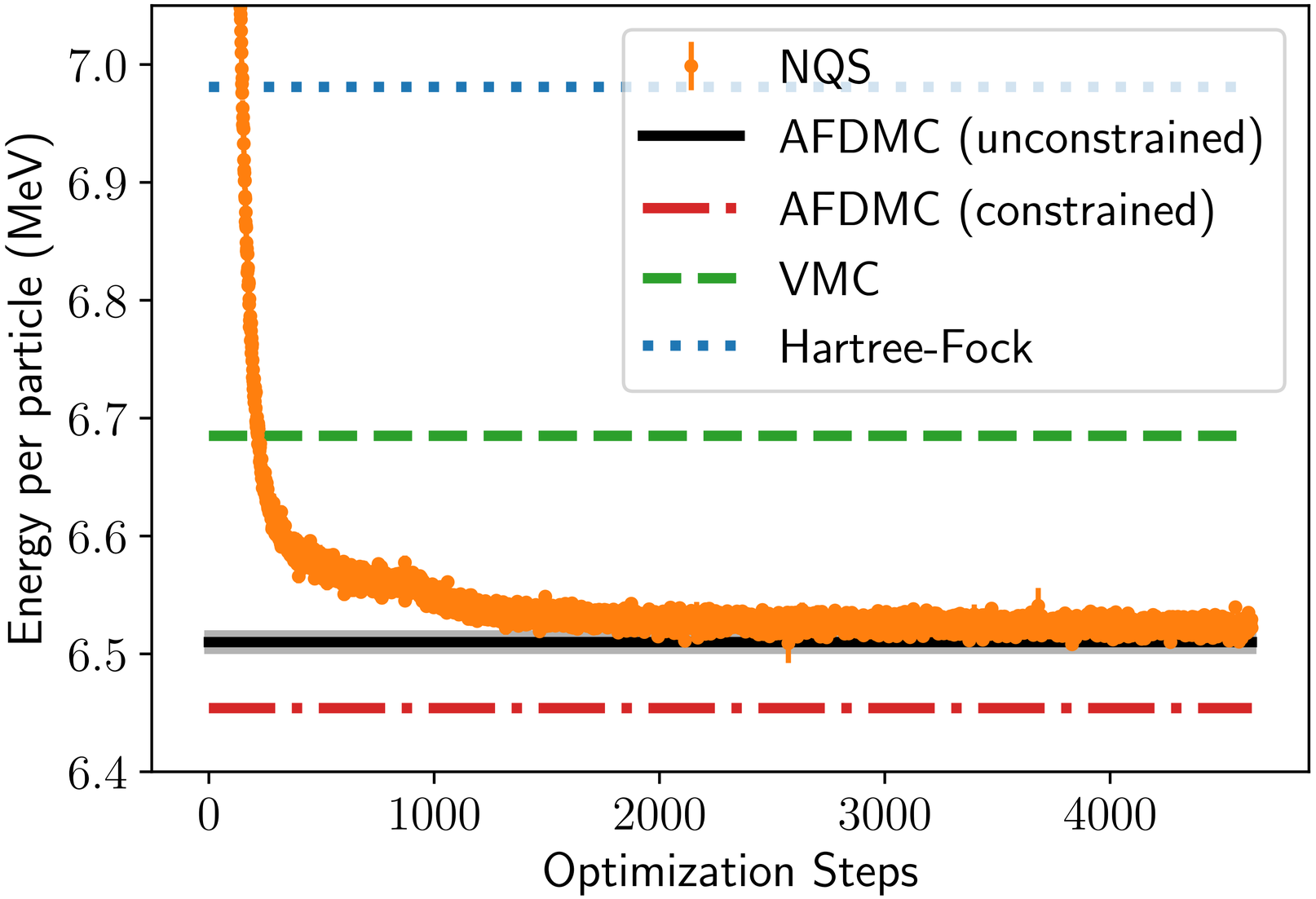
Visible s.p. states
Hidden nucleons

Hidden s.p. states
Visible nucleons

Hidden s.p. states
Hidden nucleons

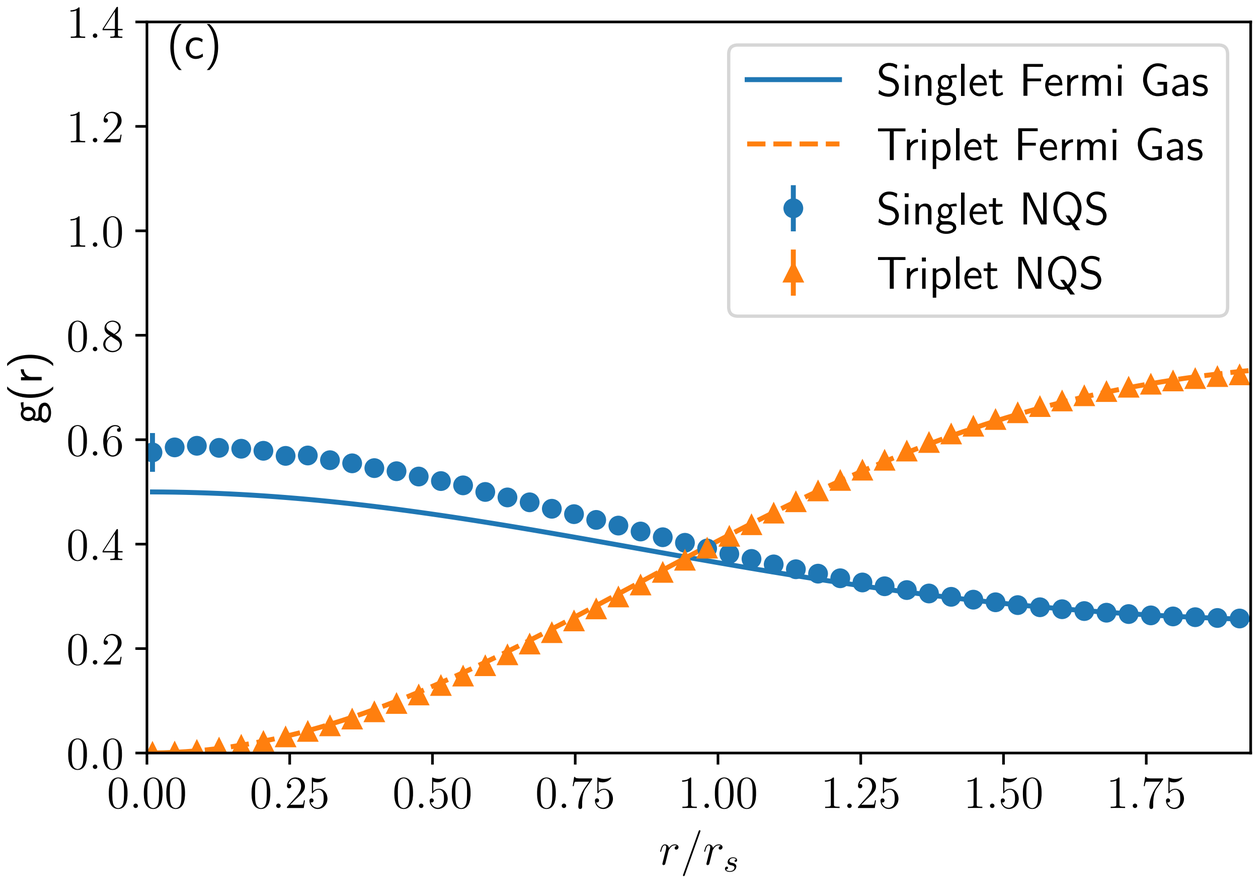
61

Dilute Neutron Matter

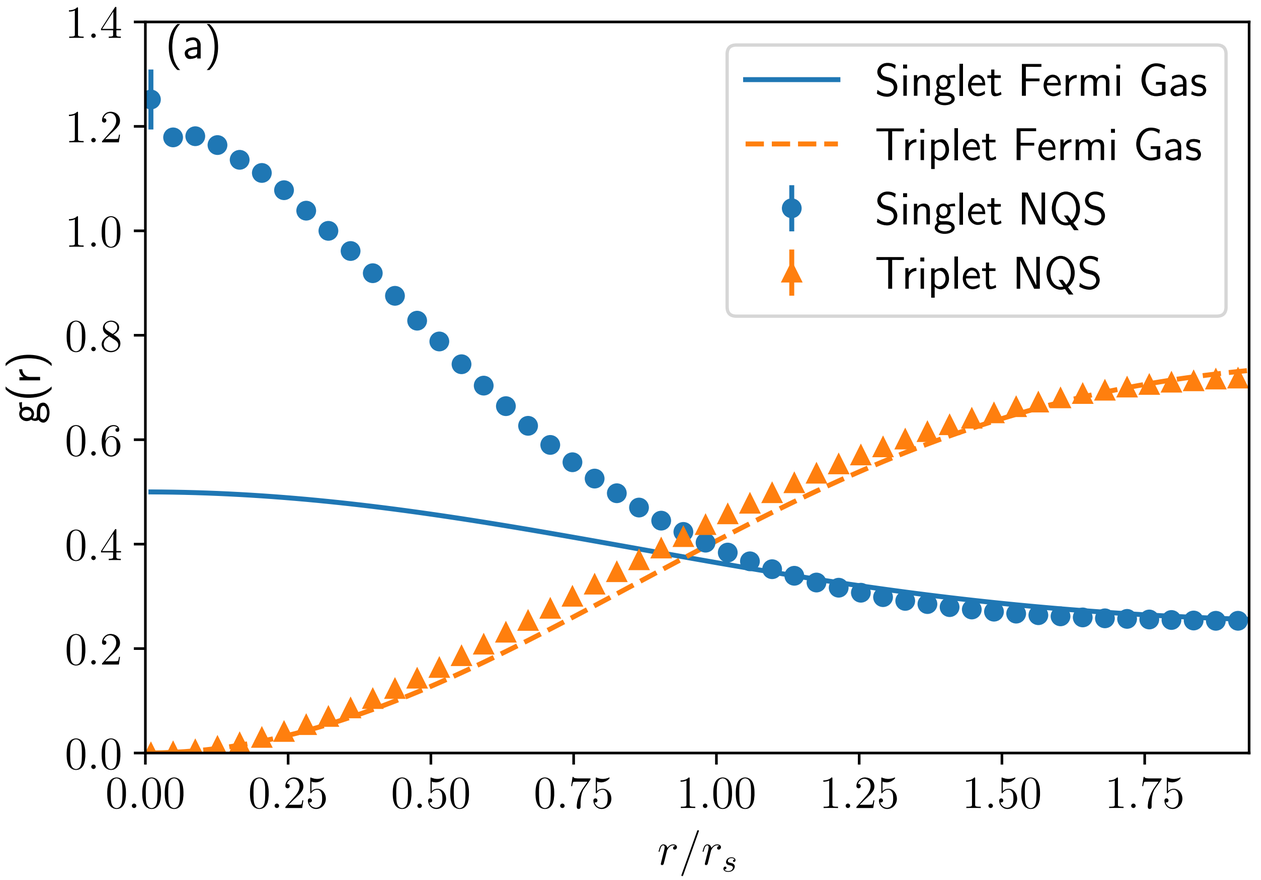


$\rho = 0.04\ \mathrm{fm}^{-3}$

Dilute Neutron Matter



$\rho = 0.08\ \mathrm{fm}^{-3}$



$\rho = 0.01\ \mathrm{fm}^{-3}$

Dilute Neutron Matter

- The hidden nucleon NQS agrees with unconstrained-path auxiliary field DMC
- Singlet pair distribution functions show evidence of pairing at lower densities
- A single ansatz can model both the normal and superfluid phases
- The formation of Cooper pairs is discovered by the NQS
- Future developments:
  - More realistic interactions (AV18+UIX and local Δ-full chiral-EFT)
  - Look for clustering in isospin-antisymmetric matter at low densities

Homogeneous Electron Gas

- See our preprint "**Message-passing neural quantum states for the homogeneous electron gas**" by G. Pescia, J. Nys, J. Kim, A. Lovato, G. Carleo (arXiv:2305.07240)

- Model for the electronic structure of solids

- Electrons move freely within a static, positive background, interacting with neighboring electrons

- Long-range nature of Coulomb interaction requires Ewald summation

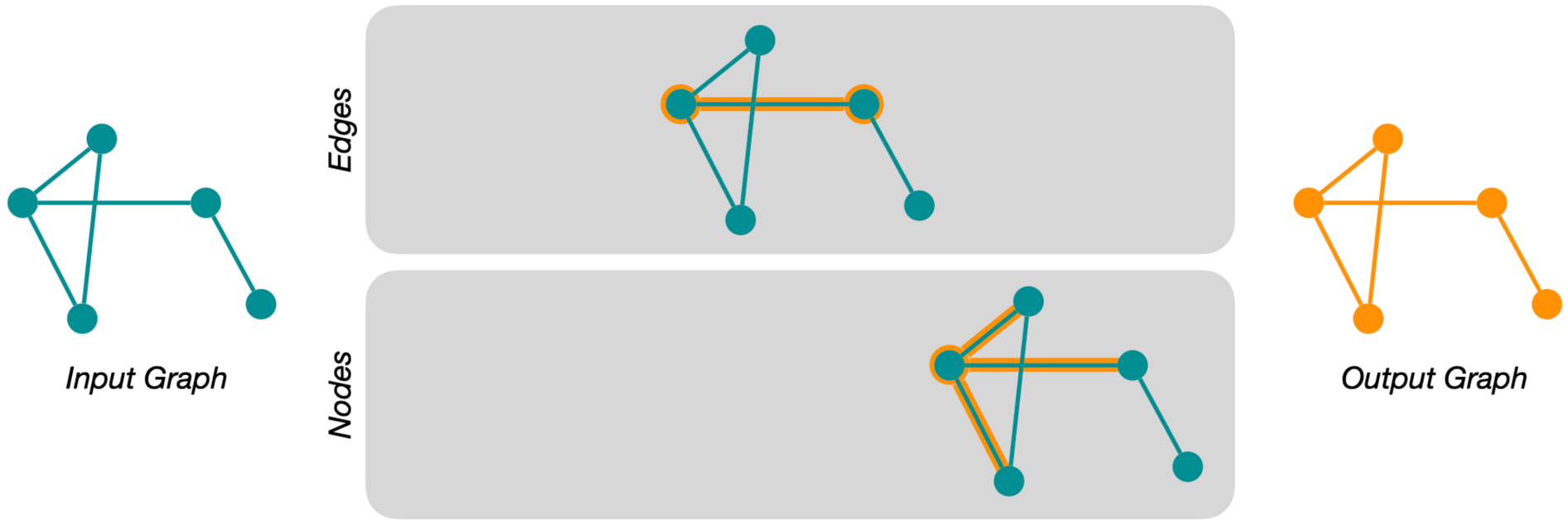
- Hamiltonian:

$$\hat{H} = -\frac{1}{2r_s^2}\sum_{i=1}^{N}\nabla_i^2 + \frac{1}{r_s}\sum_{i<j}^{N}\frac{1}{r_{ij}} + \text{const}\,.$$

Homogeneous Electron Gas

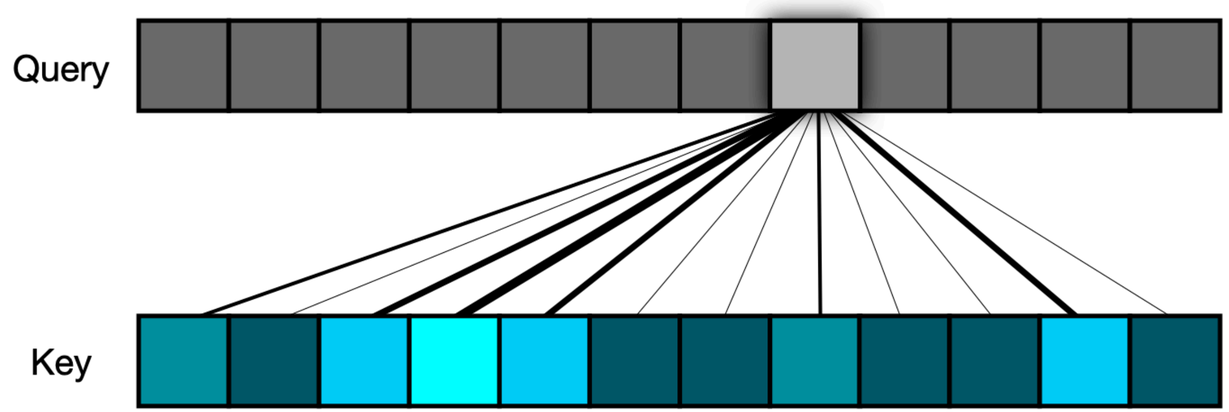
- Employ a permutation-equivariant message-passing neural network, a type of graph neural network, that iteratively encodes backflow correlations into new positions (nodes) and interactions (edges) of the system (graph)

- To enforce translational invariance,



Homogeneous Electron Gas

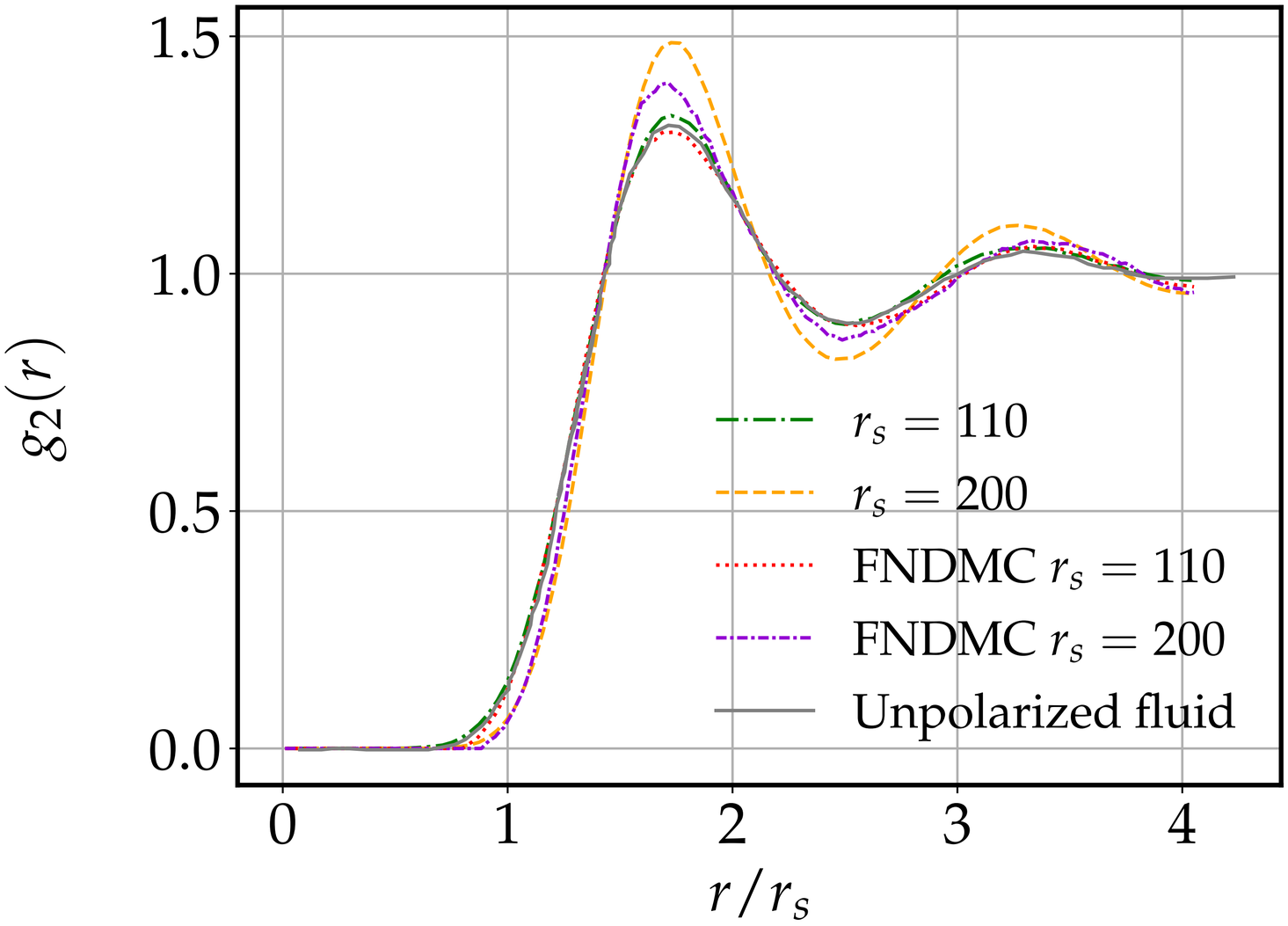
- Nodes and edges are updated using an attention mechanism
- Attention mechanisms are commonly used in natural language models to help neural networks focus on relevant data based on context clues
- Apply linear transformations to the edges, corresponding to queries and keys, then compute the overlap to determine the correlations between pairs of particles



Homogeneous Electron Gas

N	r_s	MP-NQS	LiNet [13]	FN-DMC	BF-DMC [14, 15]	FCI*/DCD** [12]
54	1	0.52973(1)	0.530019(1)	0.53094(2)	0.52989(4)	0.52973(3)*
	2	−0.014046(8)	−0.013840(1)	−0.01326(2)	−0.013966(2)	−0.01379**
	5	−0.079090(2)	−0.0788354(2)	−0.07867(1)	−0.079036(3)	−0.07837**
	10	−0.054448(1)	−0.0542785(1)	−0.054269(8)	−0.054443(2)	−0.05322**
	20	−0.0320524(5)	−0.0316886(1)	−0.031976(8)	−0.032047(2)	−0.03113**
	50	−0.0145015(1)	N/A	−0.01387(2)	−0.0144877(1)	−0.01281**
	100	−0.0076793(1)	N/A	−0.007674(3)	N/A	N/A

Homogeneous Electron Gas



Homogeneous Electron Gas

- Message-passing neural quantum state efficiently encodes correlations with orders of magnitude fewer parameters than other neural network quantum states (e.g. FermiNet, WAPNet)
- Can handle system sizes and Wigner-Seitz radii as large as $N = 128$ and $r_s = 200$
- We observe more pronounced density fluctuations at $r_s = 200$ compared to DMC, suggesting signs of a liquid-crystal phase transition
- Since translational symmetry is preserved in our NQS, the observation of a true floating Wigner crystal can be possible, unlike other studies in which the NQS breaks translational symmetry

Ultra-cold Fermi Gases

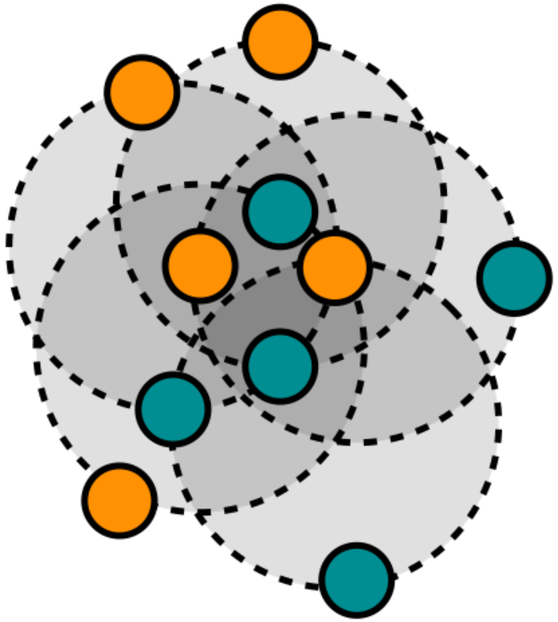
- See our preprint "**Neural-network quantum states for ultra-cold Fermi gases**" by J. Kim, G. Pescia, B. Fore, J. Nys, G. Carleo, S. Gandolfi, M. Hjorth-Jensen, and A. Lovato (arXiv:2305.08831)

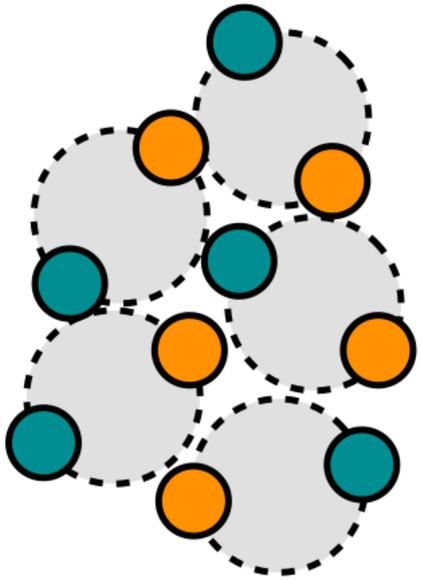
- Highly non-perturbative, short-range, attractive interaction between two-component fermions

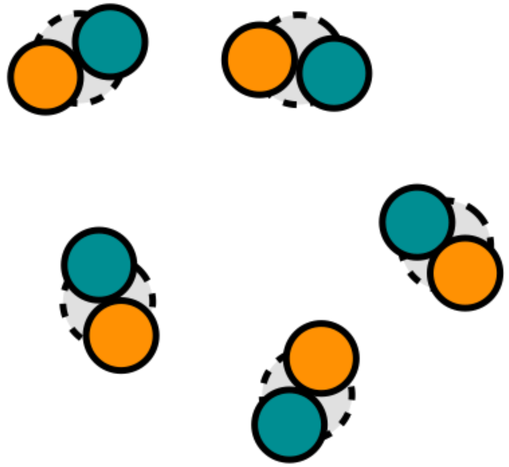
- Behavior mainly governed by s-wave scattering length a and effective range r_e

- Universal in the unitary limit $a \to \infty$

- Relevant for understanding superfluidity in fermionic systems, dilute neutron star matter, and development of many-body methods







BCS

Unitary

BEC

$1/k_F a$

-1 0 1

Crossover region

Weakly attractive

Strongly interacting

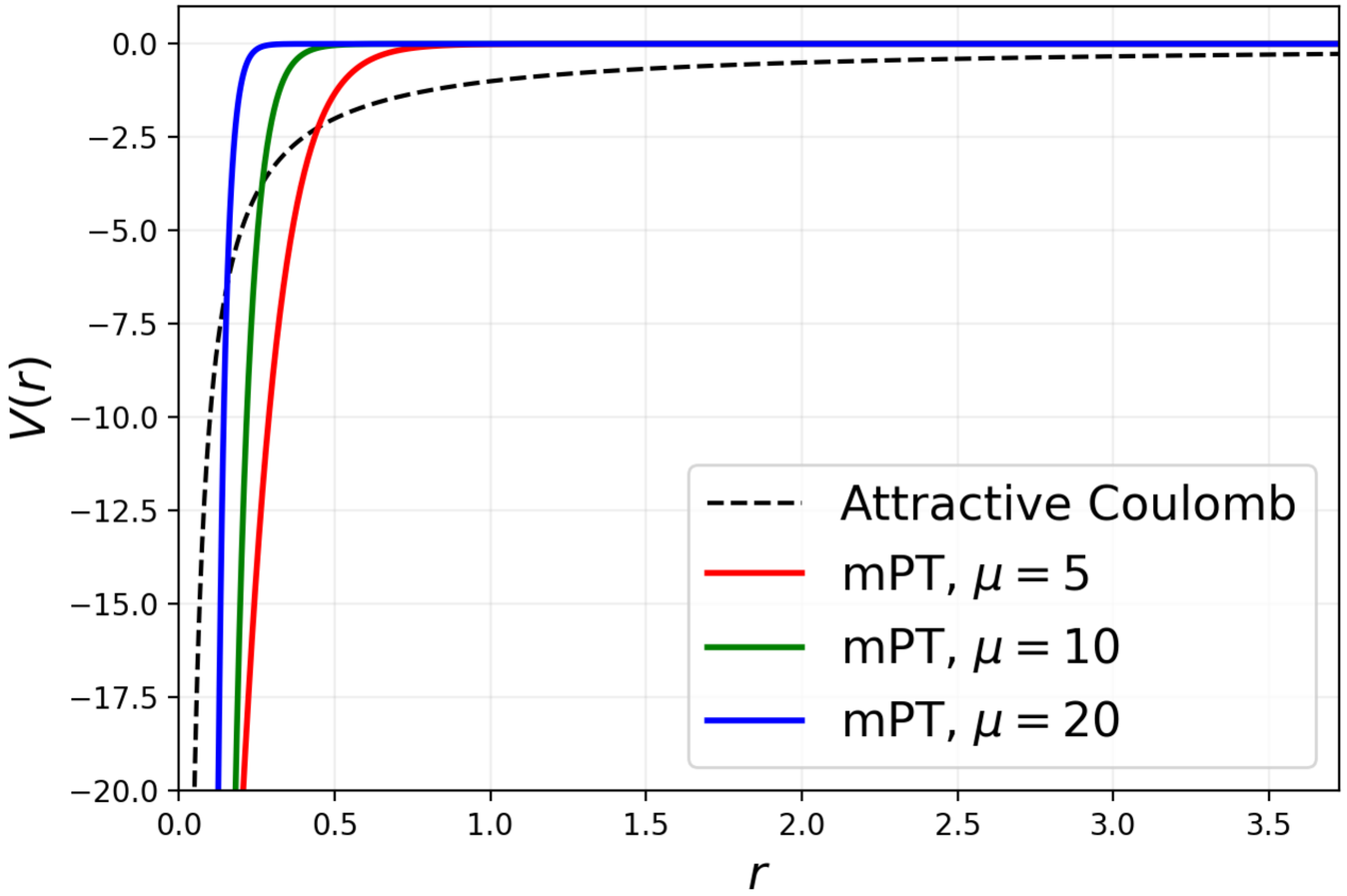
Weakly repulsive

Ultra-cold Fermi Gases

- Regularized, short-range attraction between opposite-spin pairs (Pöschl-Teller)

$$V_{ij} = (s_i^z s_j^z - 1)\; v_0 \frac{\hbar^2}{2m} \frac{\mu^2}{\cosh^2(\mu r_{ij})}$$

- Effective range $\approx 2/\mu$
- Keep v_0 fixed, pretrain with small μ
- Provides exact solution of two-body problem
- Other interaction potentials give similar results near unitarity as long as effective range is the same



Pfaffian Wave Function

- Most general way to construct an antisymmetric wave function from a pairing orbital

$$\Phi(X) = \mathrm{pf}\left[\phi(\mathbf{x}_i, \mathbf{x}_j)\right]$$

  where ϕ must be antisymmetric.

- For our NQS, we define

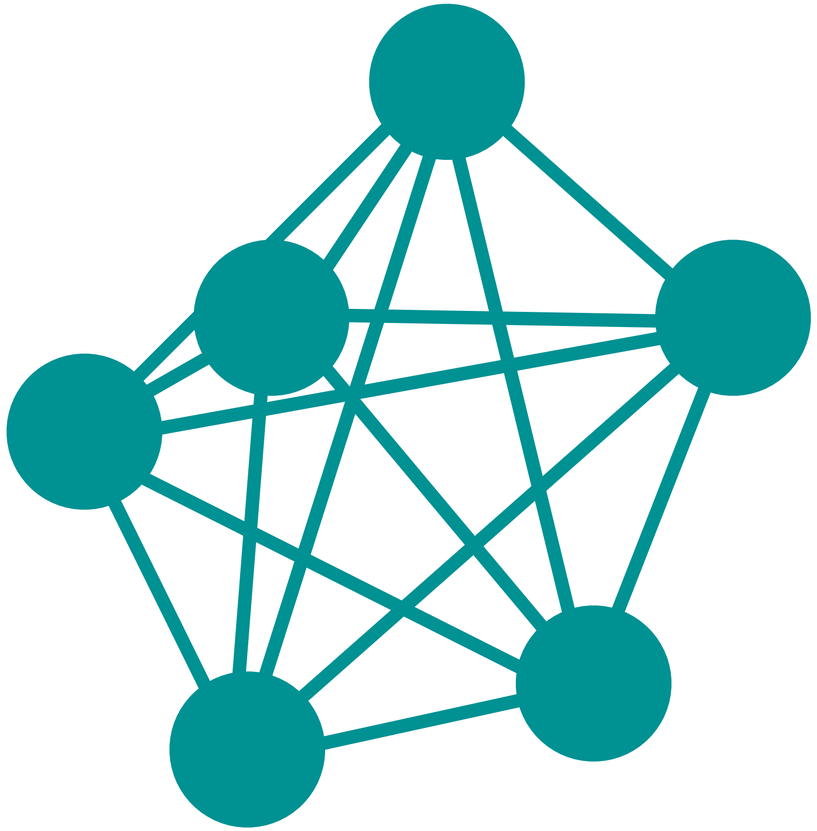
$$\phi(\mathbf{x}_i, \mathbf{x}_j) \equiv \nu(\mathbf{x}_i, \mathbf{x}_j) - \nu(\mathbf{x}_j, \mathbf{x}_i)$$

  where ν is a neural network.

- Systematically improvable (universal approximation theorem)

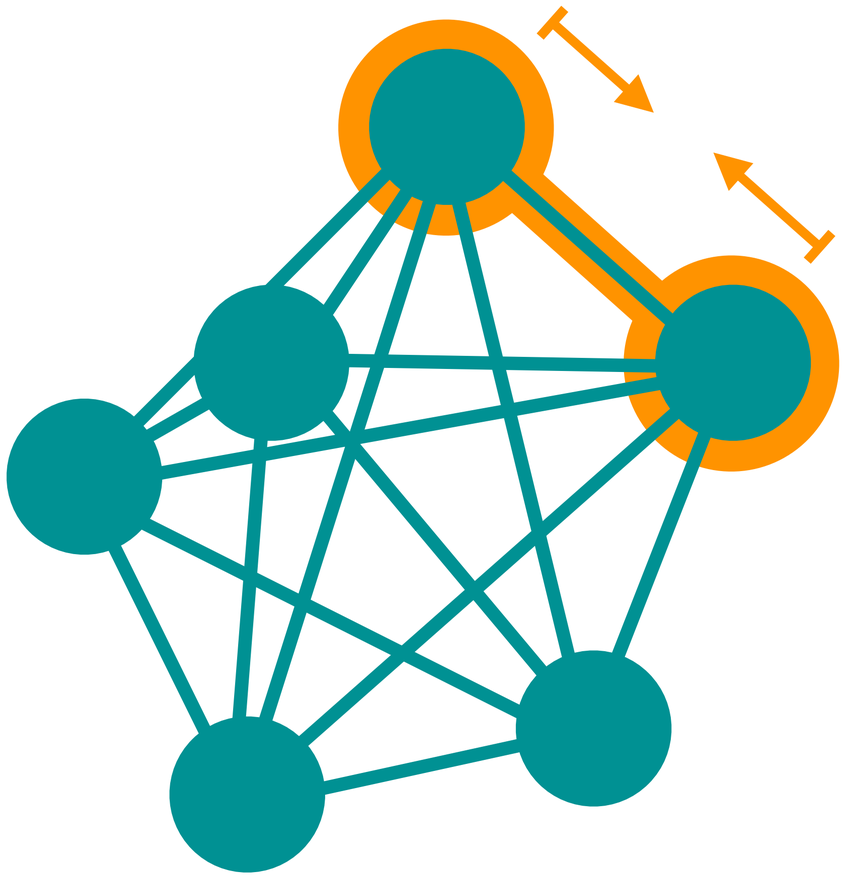
- Naturally encodes singlet and triplet pairing because ν takes spins as input

Message-Passing Neural Network



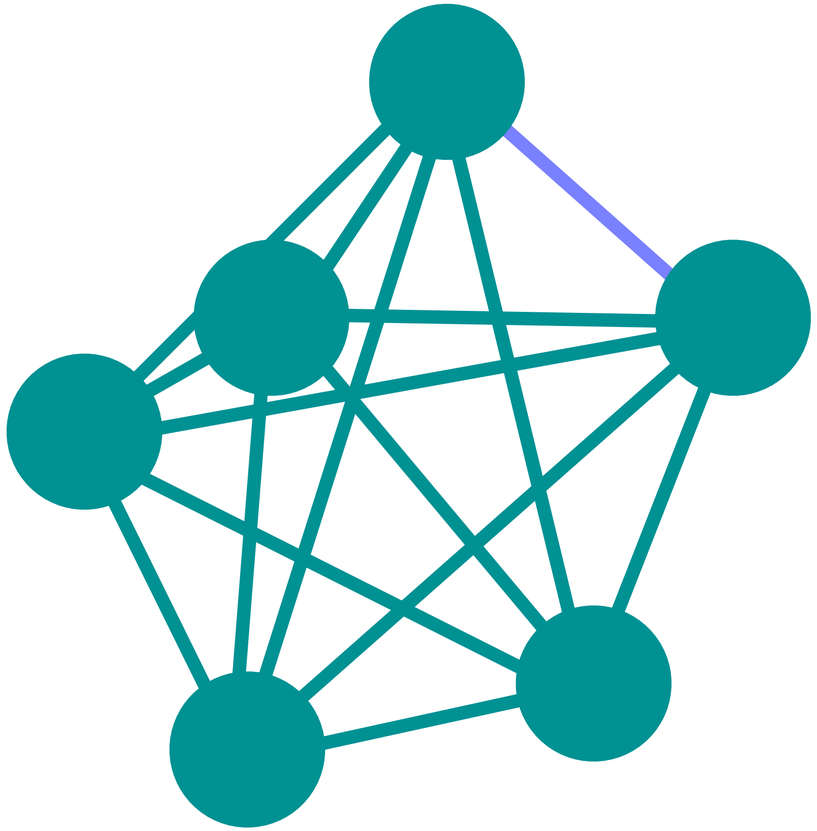
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



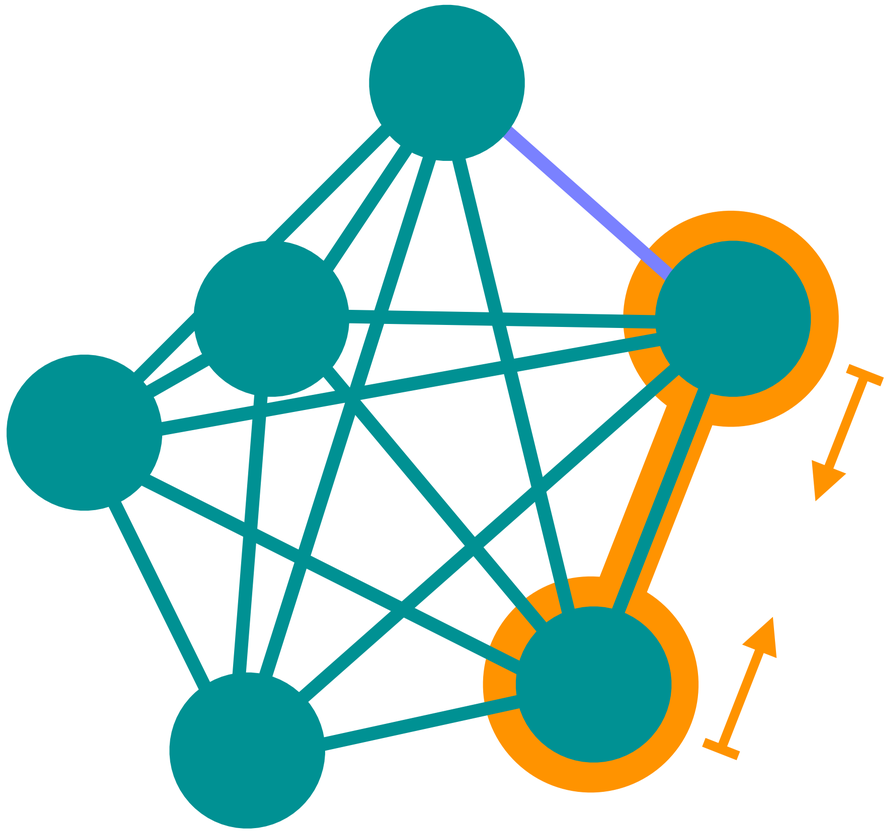
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



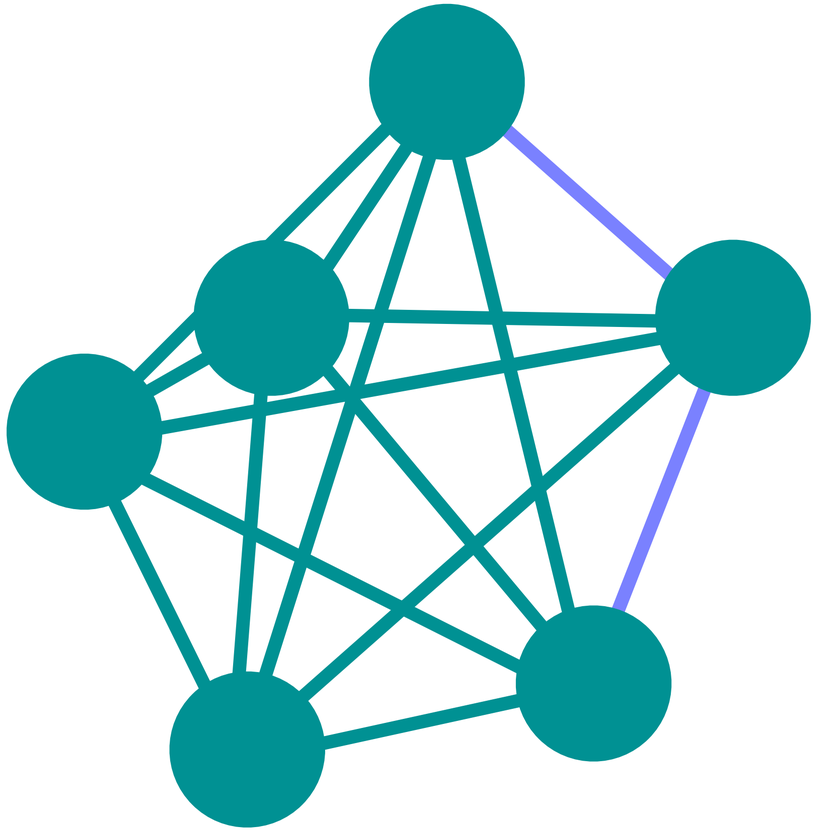
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



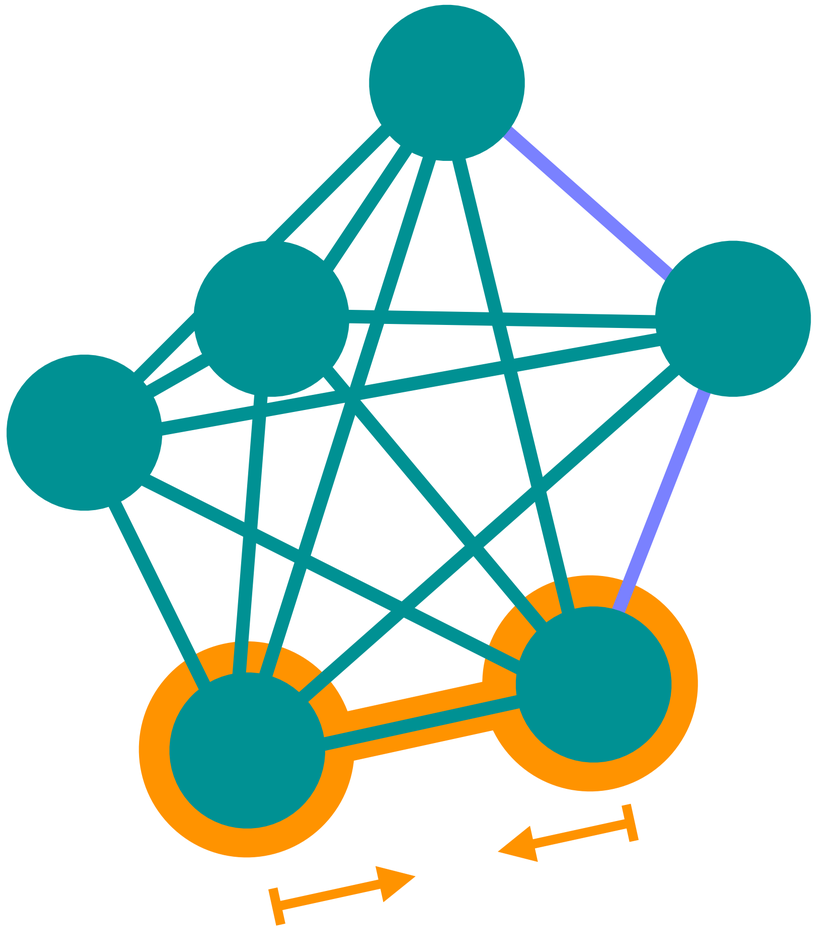
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



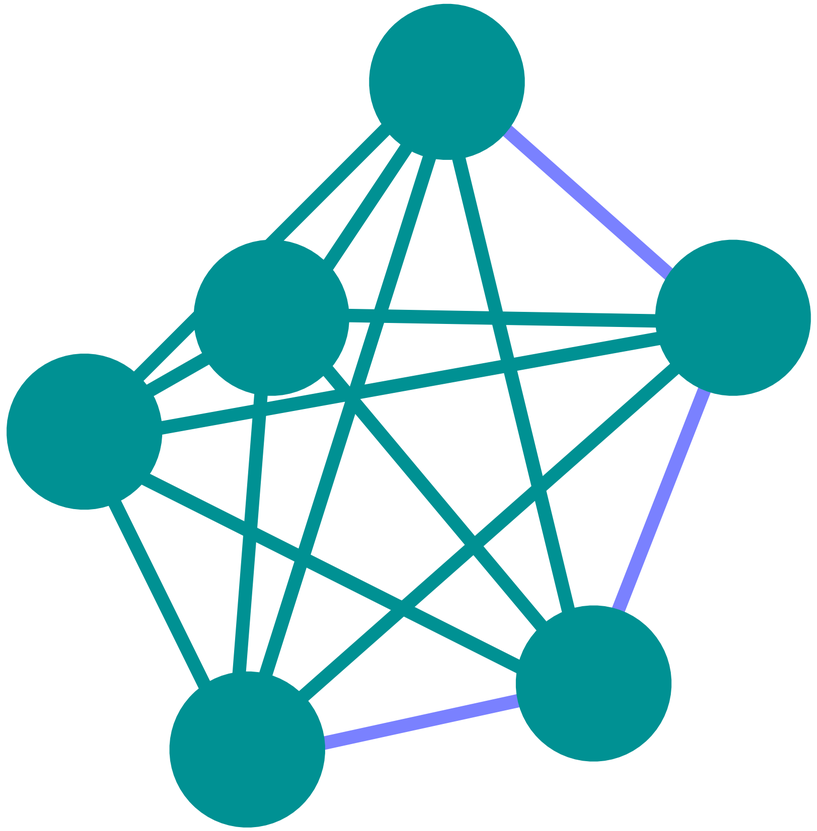
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



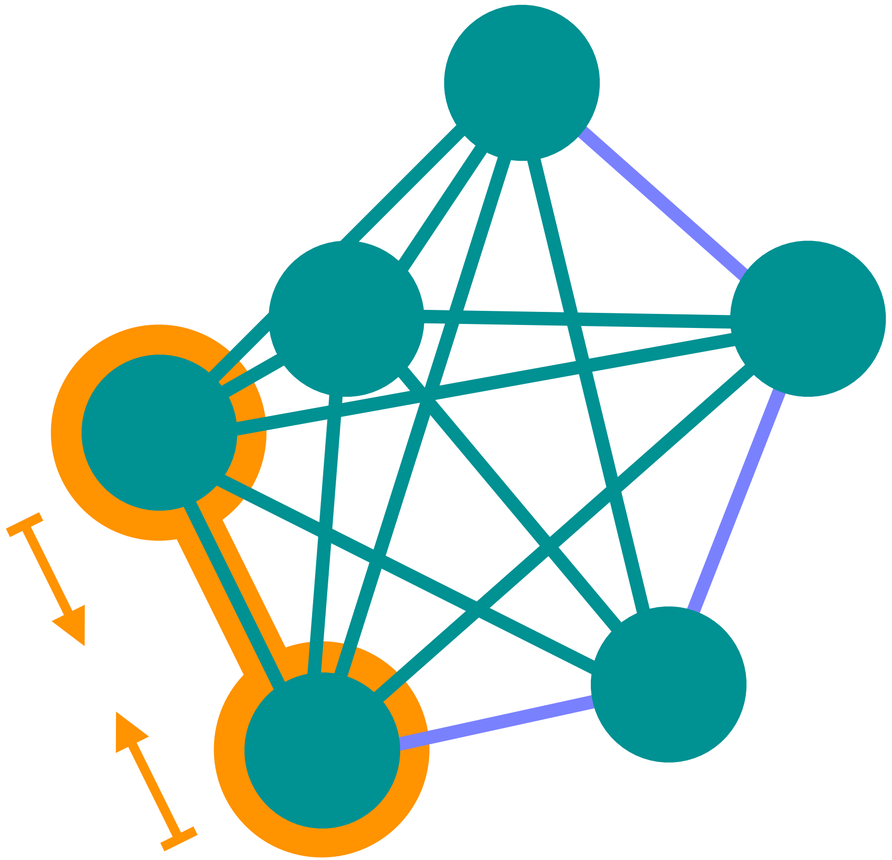
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



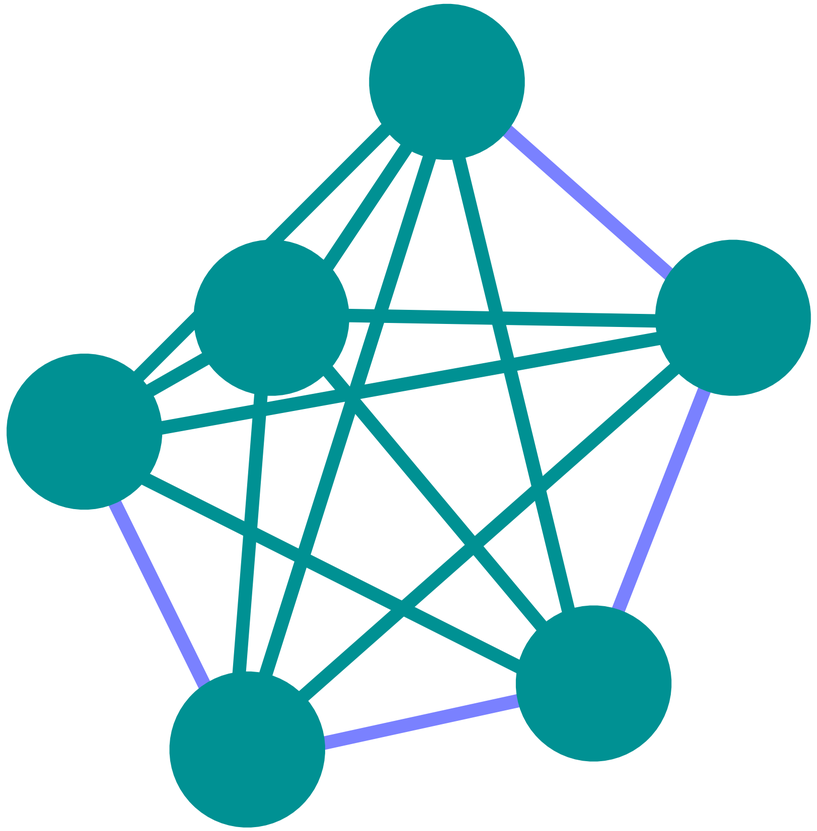
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



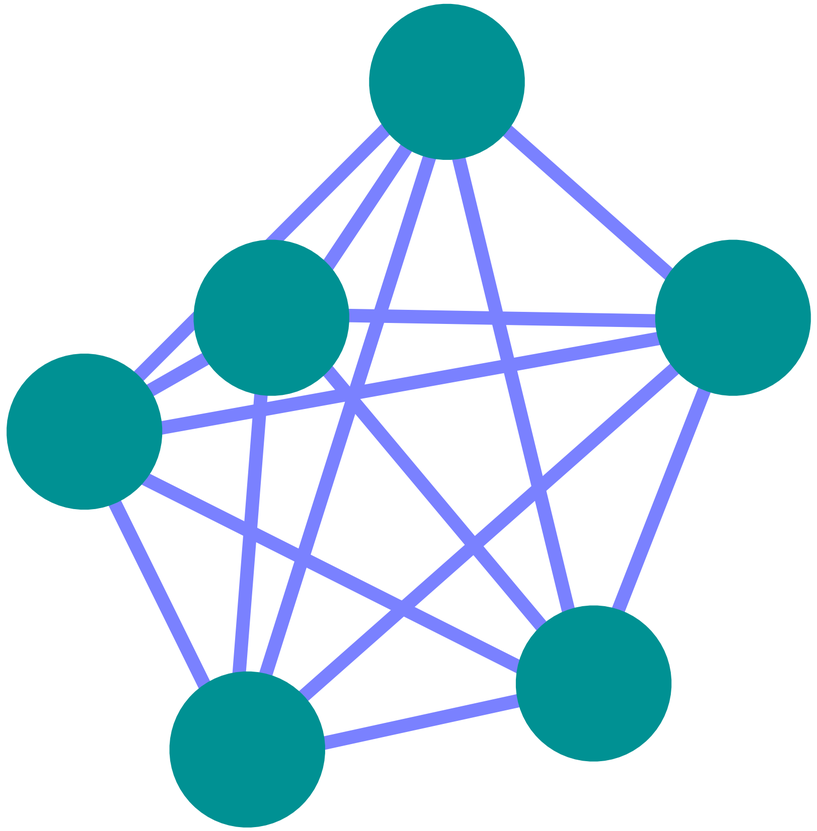
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



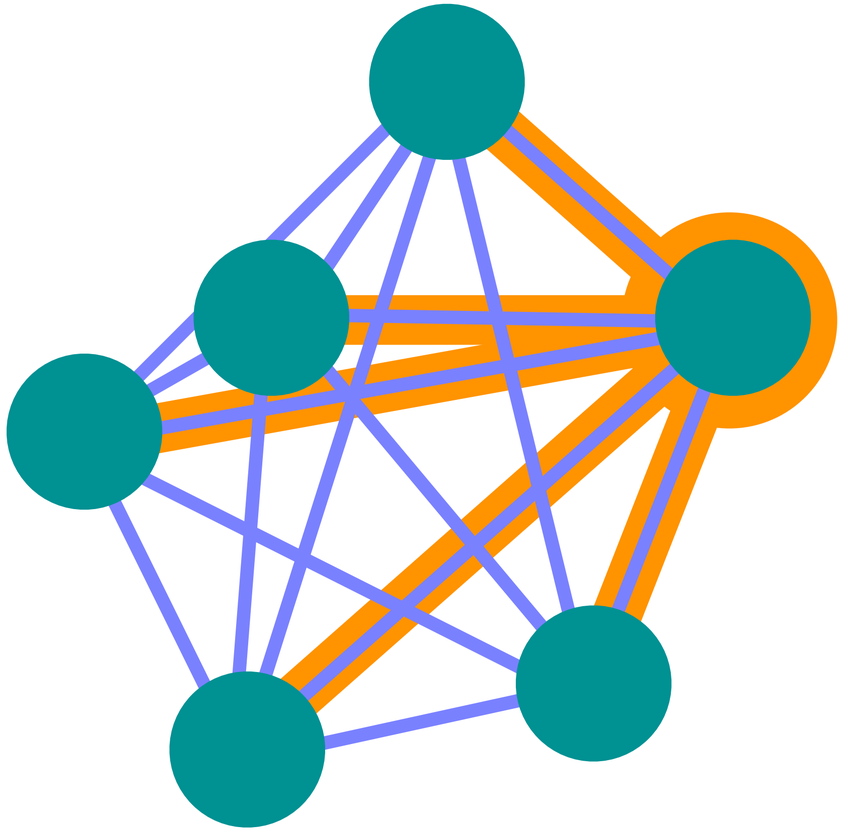
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



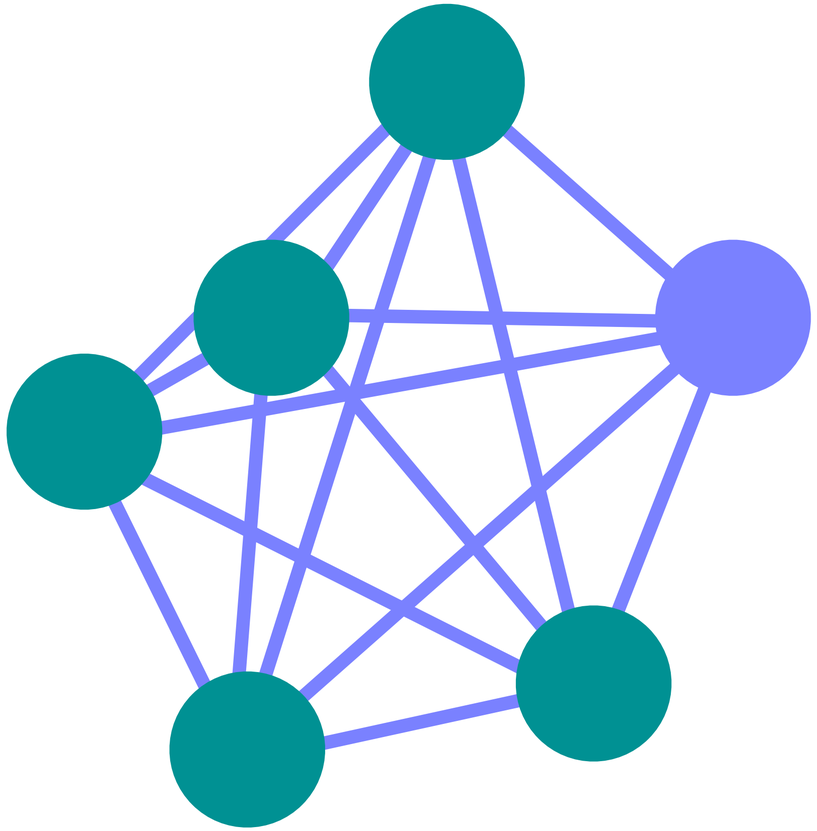
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



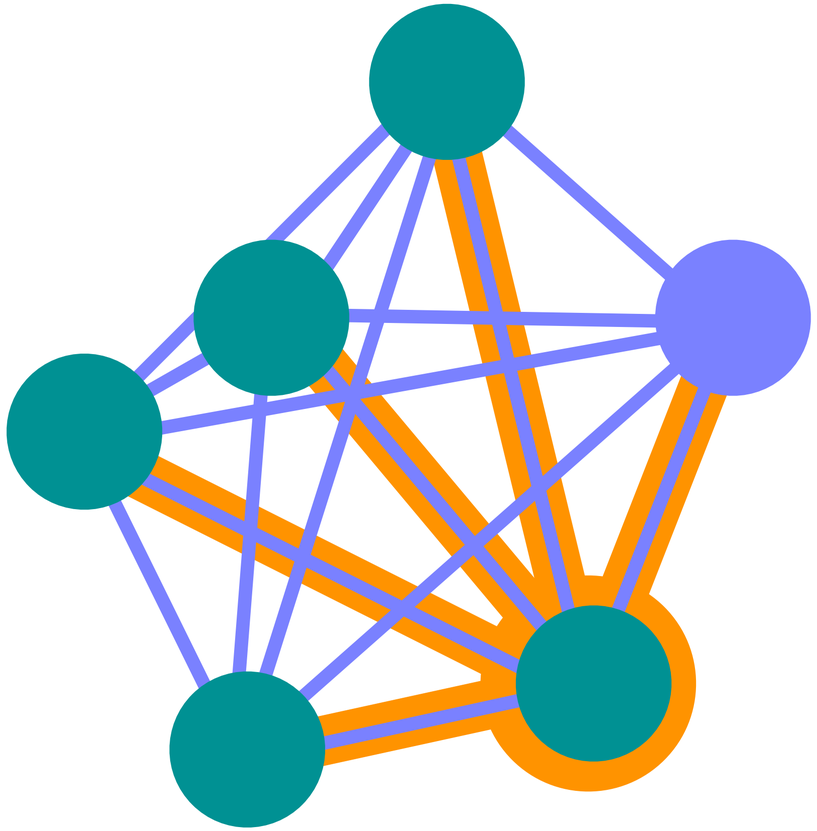
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



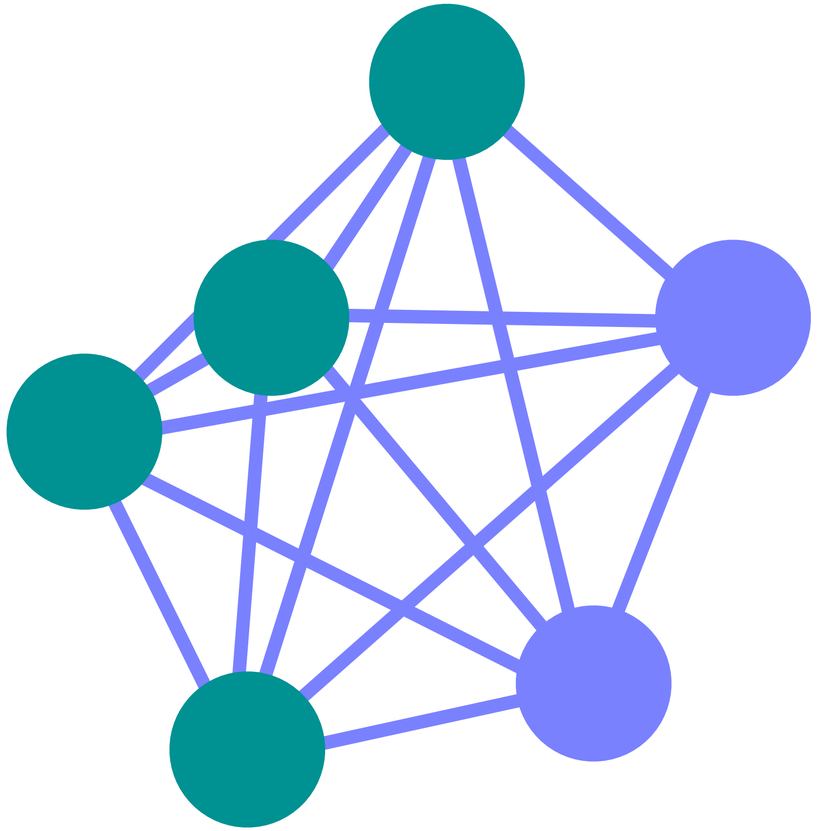
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



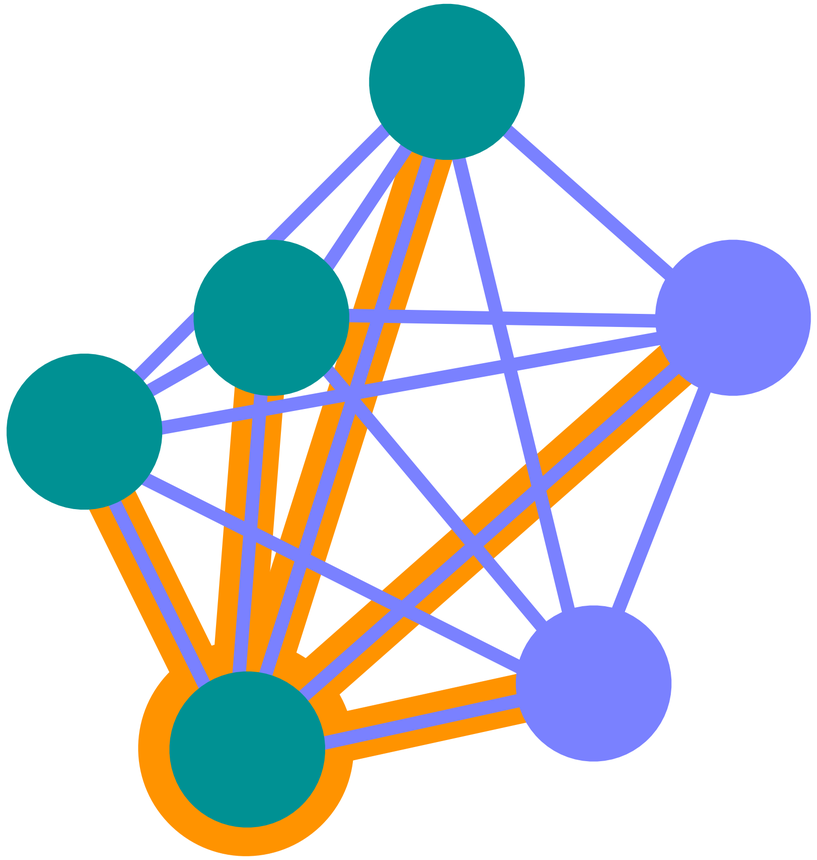
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



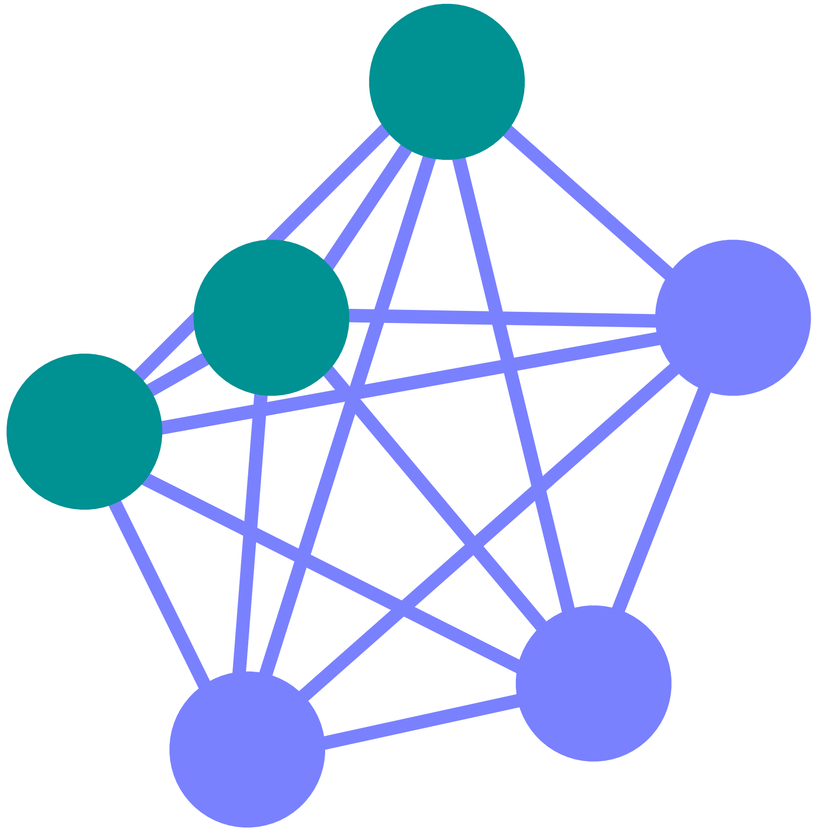
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



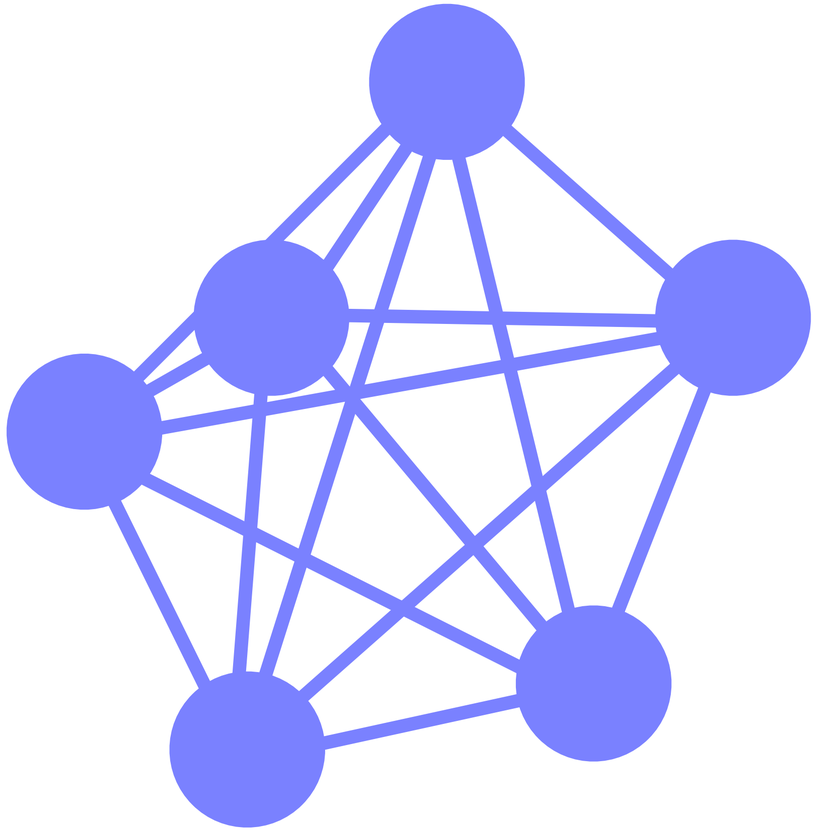
- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network

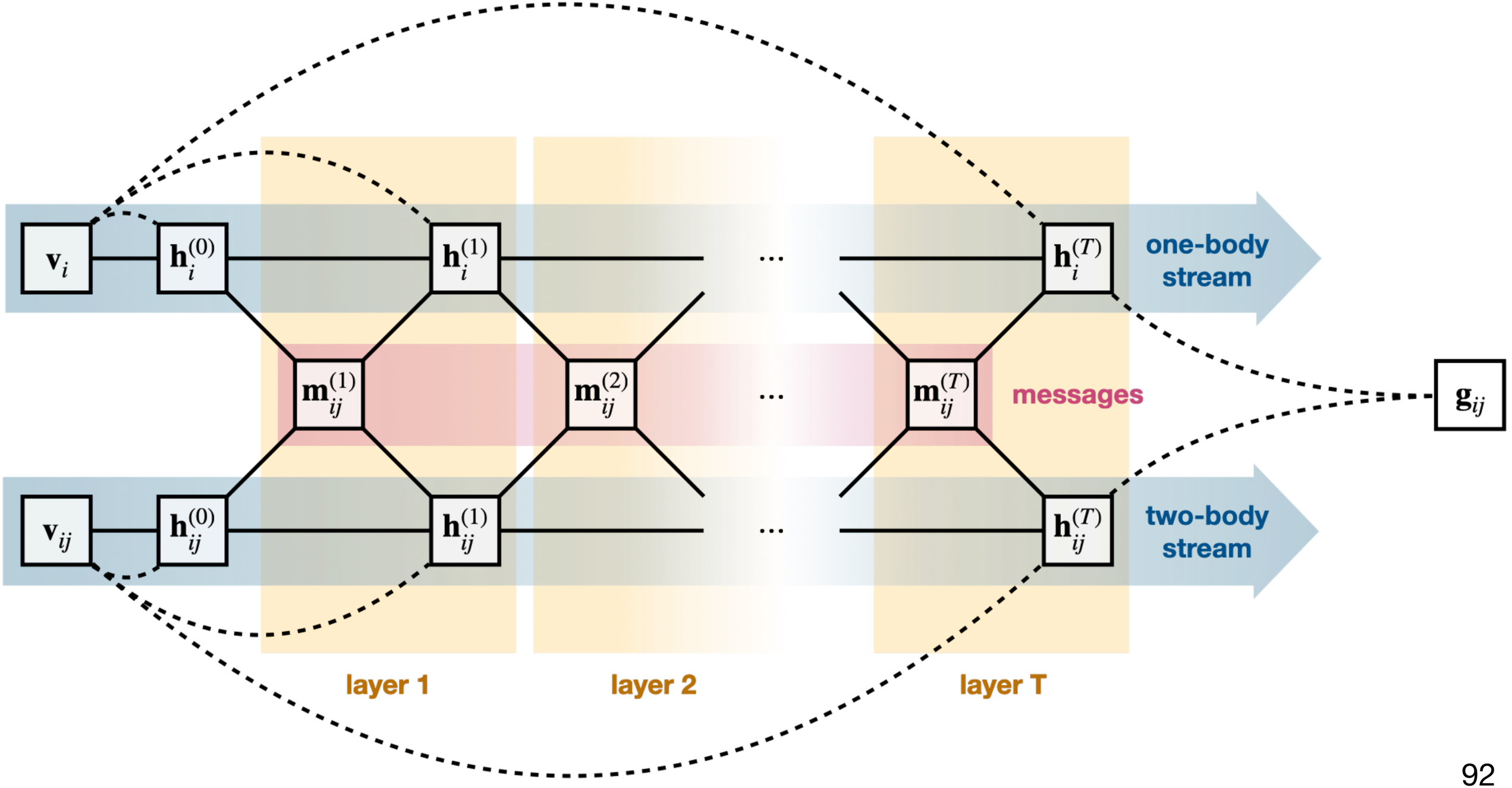


- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features

Message-Passing Neural Network



- Used in our study of the homogenous electron gas (Pescia et al. arXiv:2305.07240)
- Type of graph neural network
- Must be permutation equivariant to maintain antisymmetry
- Iteratively build correlations into new one-body and two-body features from original ones
- Nodes = one-body features
- Edges = two-body features


$\mathbf{v}_i$
$\mathbf{h}_i^{(0)}$
$\mathbf{h}_i^{(1)}$
$\mathbf{h}_i^{(T)}$
one-body stream
$\mathbf{m}_{ij}^{(1)}$
$\mathbf{m}_{ij}^{(2)}$
$\mathbf{m}_{ij}^{(T)}$
messages
$\mathbf{g}_{ij}$
$\mathbf{v}_{ij}$
$\mathbf{h}_{ij}^{(0)}$
$\mathbf{h}_{ij}^{(1)}$
$\mathbf{h}_{ij}^{(T)}$
two-body stream
layer 1
layer 2
layer T

Pfaffian Wave Function with Backflow

- Use output of MPNN as input to pairing orbital instead of raw coordinates

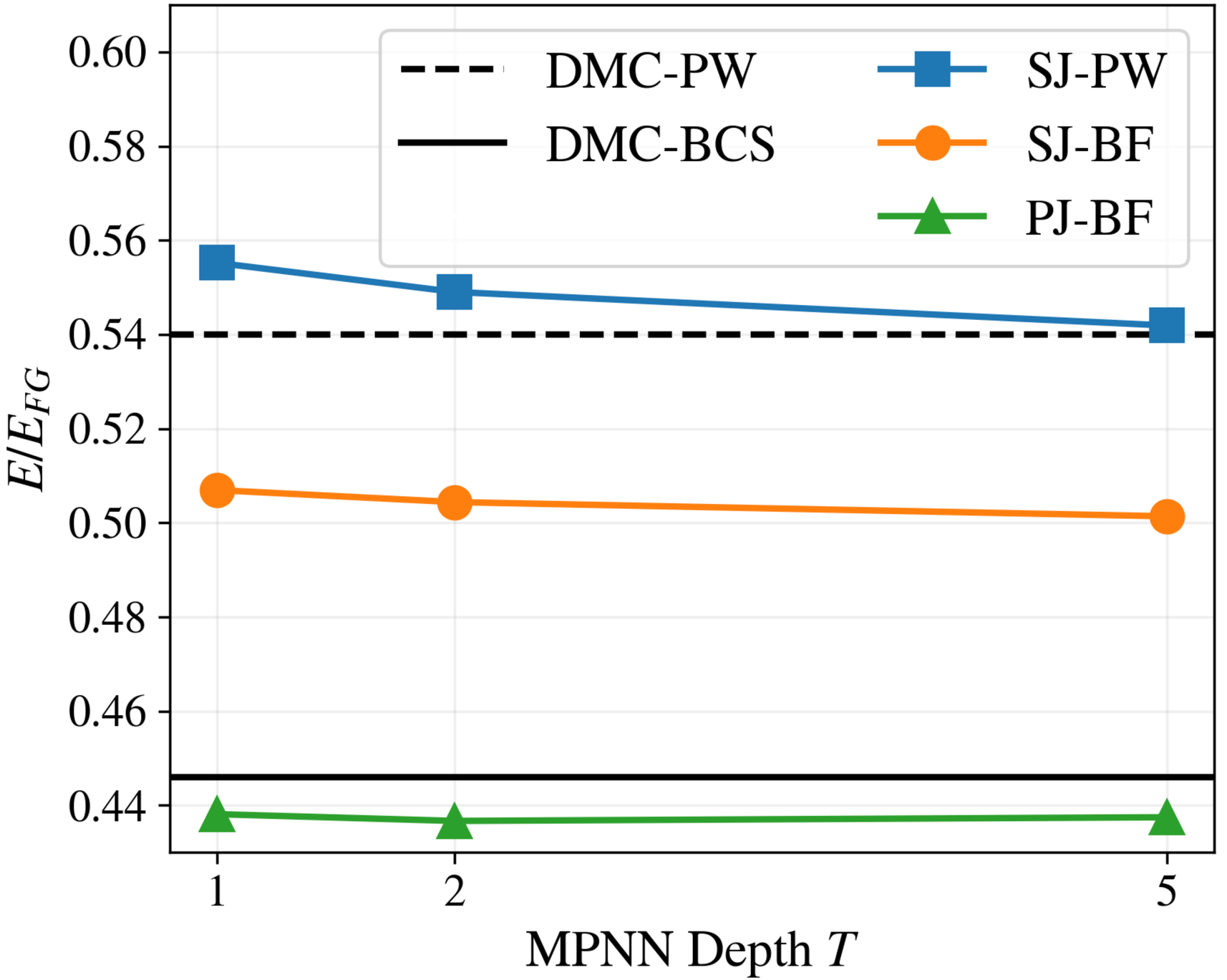
$$\Phi(X) = \mathrm{pf}\left[\phi(\mathbf{g}_{ij})\right]$$

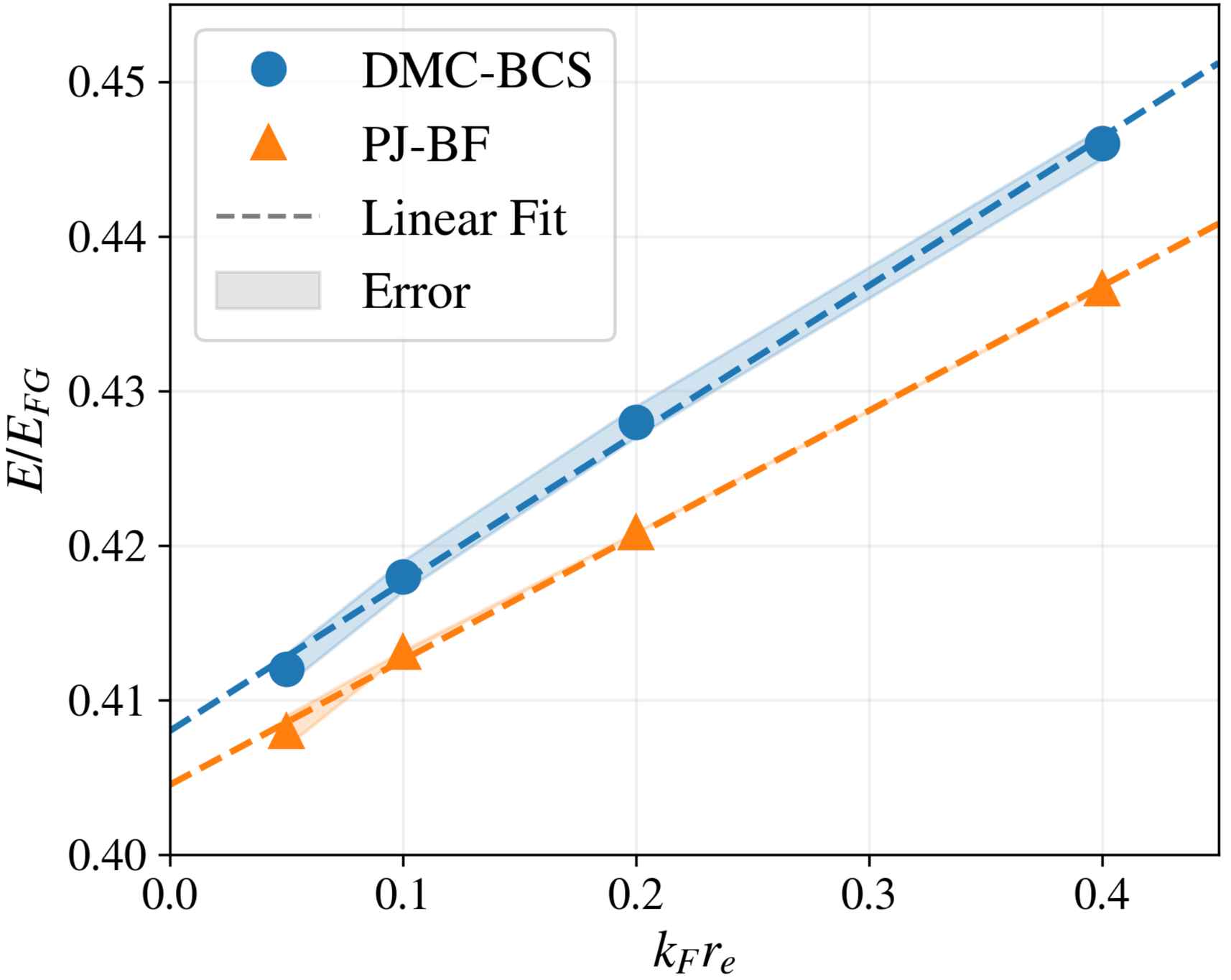
- Jastrow correlator based on a “Deep Set” (Zaheer et al. - arXiv:1703.06114)

$$J(X) = \rho\left(\sum_{i\neq j}\zeta(\mathbf{g}_{ij})\right)$$

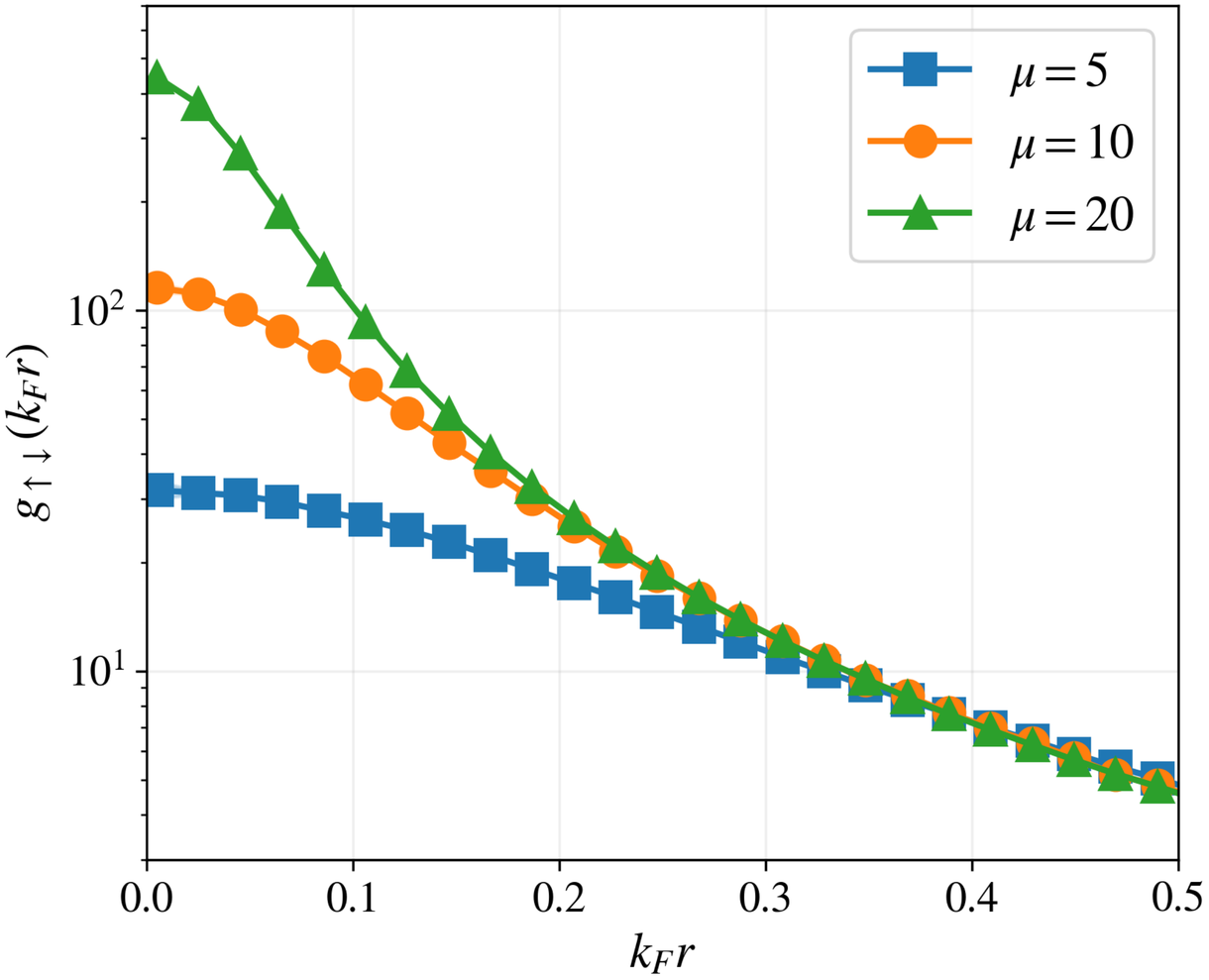
  where ρ and ζ are neural networks.

- Most results shown use an MPNN with 2 iterations (~8500 parameters total)
- We also enforce periodicity, translational invariance, parity and time-reversal symmetry
- This is the first time neural backflow transformations have been applied to the Pfaffian :)

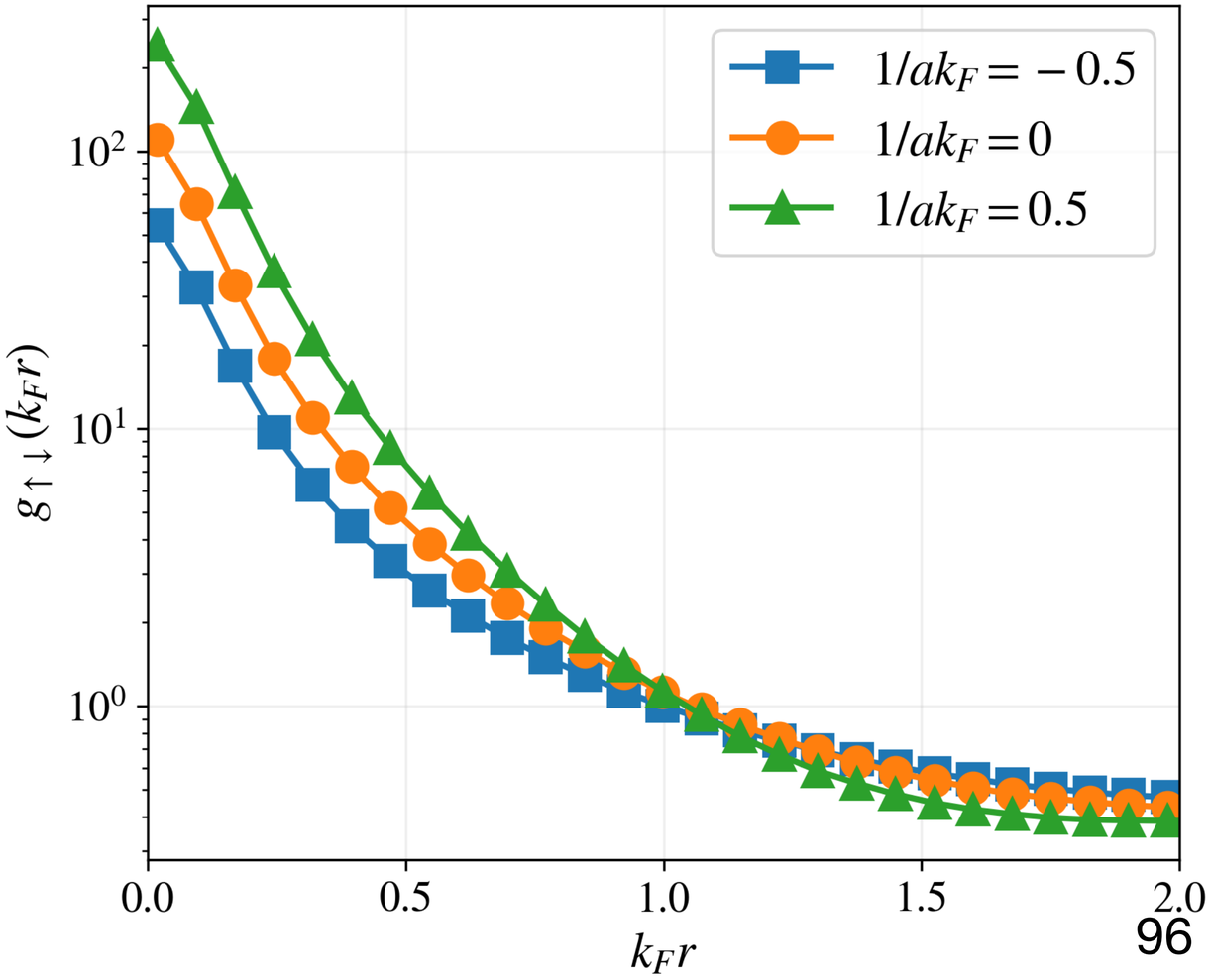

DMC-PW
DMC-BCS
SJ-PW
SJ-BF
PJ-BF
0.60
0.58
0.56
0.54
0.52
0.50
0.48
0.46
0.44
E/E_{FG}
1
2
5
MPNN Depth T

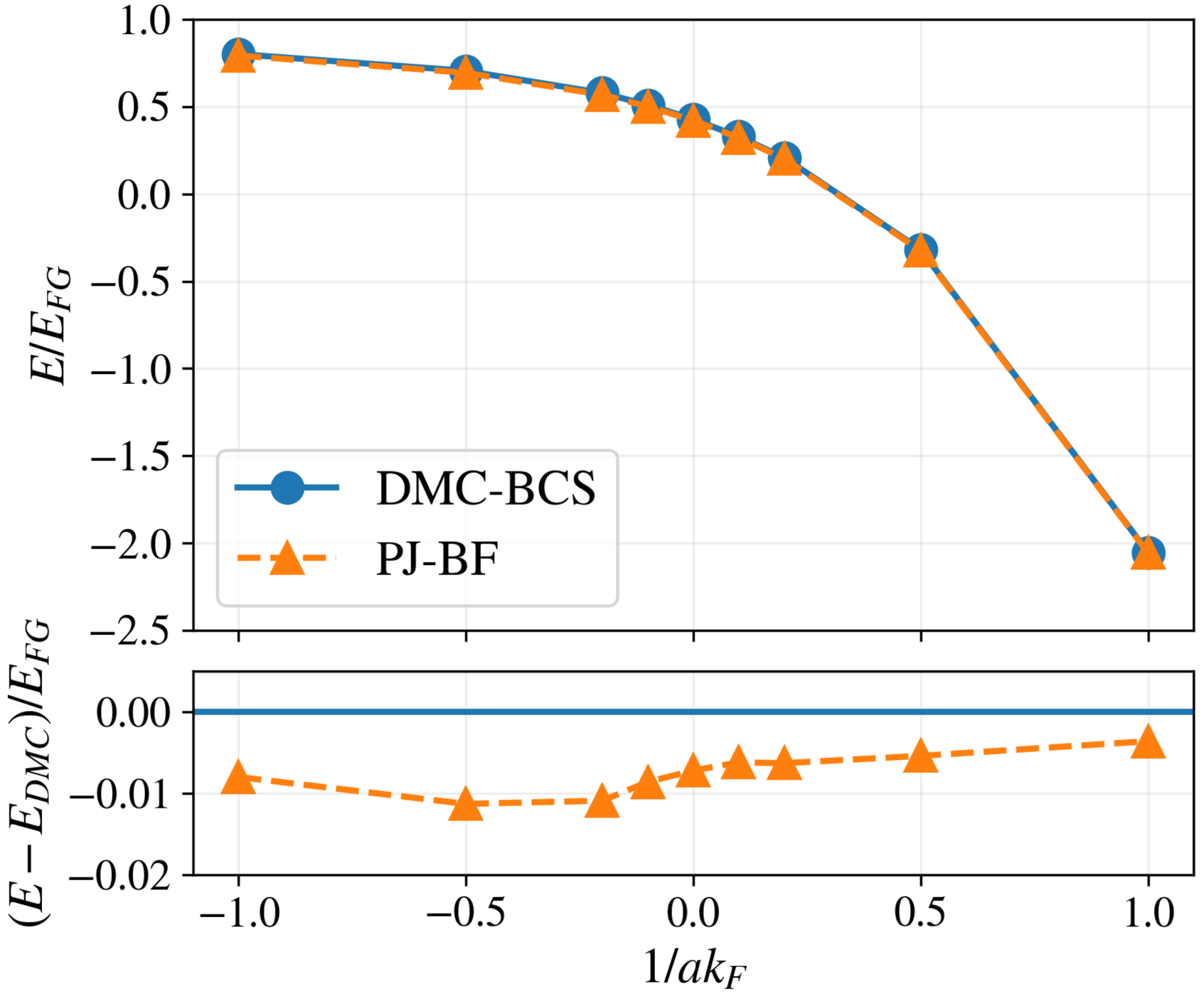

DMC-BCS
PJ-BF
Linear Fit
Error
0.45
0.44
0.43
0.42
0.41
0.40
E/E_{FG}
0.0
0.1
0.2
0.3
0.4
$k_F r_e$

Different effective ranges $r_e = 2/\mu$ at unitarity $(1/ak_F = 0)$



Different scattering lengths near unitarity (fixed $r_e = 0.2$)




1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
E/E_{FG}
DMC-BCS
PJ-BF
0.00
−0.01
−0.02
$(E-E_{DMC})/E_{FG}$
−1.0
−0.5
0.0
0.5
1.0
$1/ak_F$

Pairing Gap

- The pairing gap can be evaluated using

$$\Delta(N) = E(N) - \frac{1}{2}\Big(E(N+1) + E(N-1)\Big)$$

  for odd N.

- To accommodate odd-N cases in our Pfaffian NQS, we add one additional neural network to represent the unpaired single-particle spin-orbital

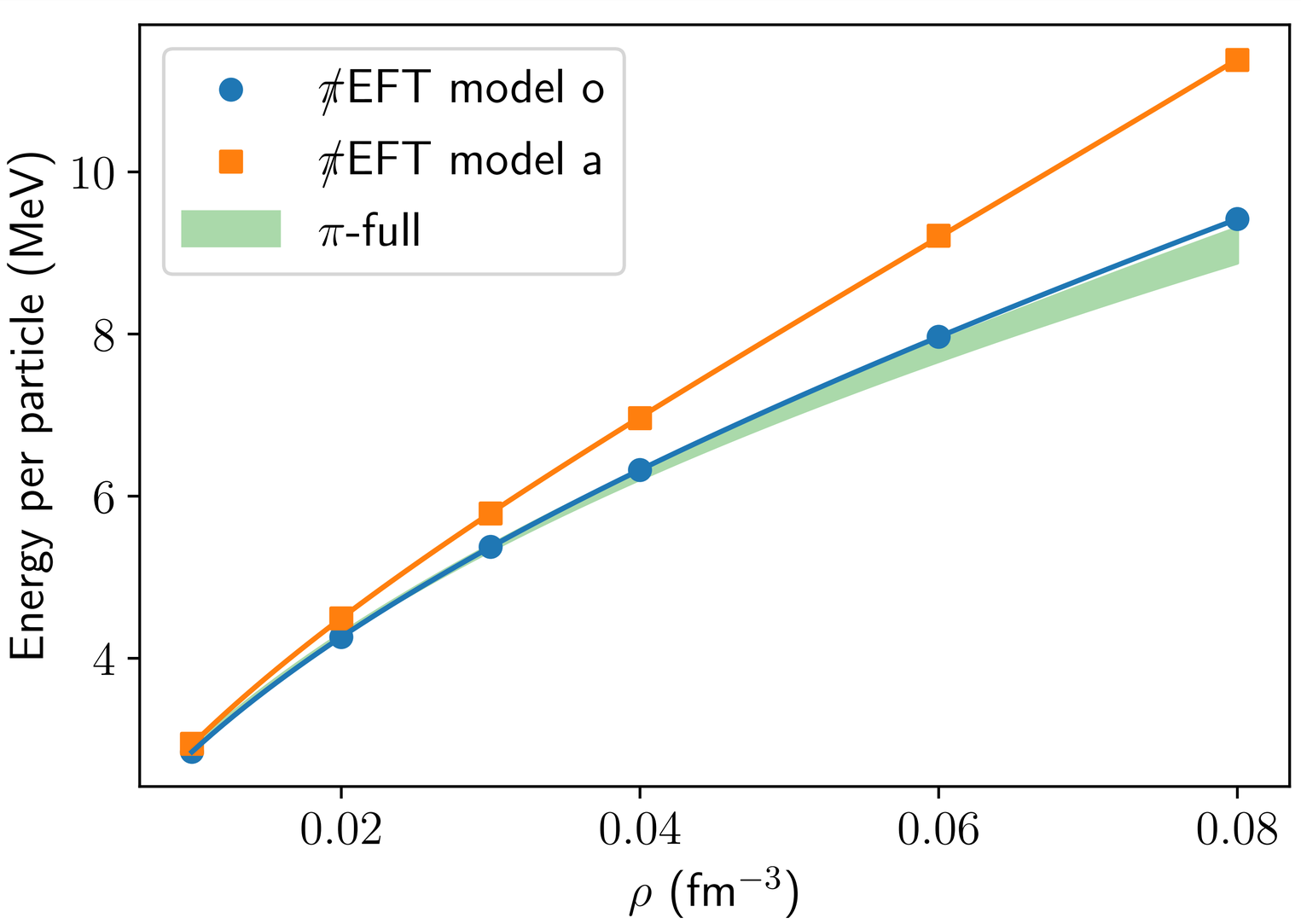
$\Delta(15)/E_{FG}$	DMC-BCS	PJ-BF
Non-translation invariant	0.9620	0.8618
Translation invariant	1.6475	1.5263

Ultra-cold Fermi Gases

- Our Pfaffian ansatz is very general — works for any Hamiltonian (even those that exchange spin!)
- Can obtain lower energies than state-of-the-art diffusion Monte Carlo methods
- We require far fewer parameters compared to other NQS applied to similar problems (~8500 vs millions)
- Future work:
  - Smaller r_e: better linear extrapolation to the $r_e \to 0$ limit
  - Larger N: As an initial test, we only used $N = 14$
  - Test if our Pfaffian NQS works just as well for partially-polarized systems

Thank you for your attention!

Dilute Neutron Matter



Number-Projected BCS Wave Function

$$\Phi(X) = \det\left[\phi(\mathbf{x}_{i\uparrow}, \mathbf{x}_{j\downarrow})\right] = \det \begin{bmatrix} \phi(\mathbf{x}_{1\uparrow}, \mathbf{x}_{1\downarrow}) & \phi(\mathbf{x}_{1\uparrow}, \mathbf{x}_{2\downarrow}) & \cdots & \phi(\mathbf{x}_{1\uparrow}, \mathbf{x}_{N/2\downarrow}) \\ \phi(\mathbf{x}_{2\uparrow}, \mathbf{x}_{1\downarrow}) & \phi(\mathbf{x}_{2\uparrow}, \mathbf{x}_{2\downarrow}) & \cdots & \phi(\mathbf{x}_{2\uparrow}, \mathbf{x}_{N/2\downarrow}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\mathbf{x}_{N/2\uparrow}, \mathbf{x}_{1\downarrow}) & \phi(\mathbf{x}_{N/2\uparrow}, \mathbf{x}_{2\downarrow}) & \cdots & \phi(\mathbf{x}_{N/2\uparrow}, \mathbf{x}_{N/2\downarrow}) \end{bmatrix}$$

Periodic Inputs

- Separation vector

$$\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j \longmapsto \Big(\cos(2\pi\mathbf{r}_{ij}/L),\ \sin(2\pi\mathbf{r}_{ij}/L)\Big)$$

- Distance

$$\|\mathbf{r}_{ij}\| \longmapsto \|\sin(2\pi\mathbf{r}_{ij}/L)\|$$

- Positions are ignored to enforce translational invariance

Parity and Time-Reversal Symmetry

- We carry out the VMC calculations for the unpolarized gas using $\Psi^{PT}(R,S)$ given by

$$\Psi^{P}(R,S) = \Psi(R,S) + \Psi(-R,S)$$

$$\Psi^{PT}(R,S) = \Psi^{P}(R,S) + (-1)^{N/2}\Psi^{P}(R,-S)$$

where R and S are the set of all positions and spins, respectively.

MPNN Equations

- Iteratively builds correlations into new one- and two-body features from old ones
- Skip connections help stabilize training and avoid vanishing gradients
- Has been effective for the electron gas, despite having orders of magnitude fewer parameters compared to FermiNet

$$\mathbf{v}_i = (s_i^z)$$
$$\mathbf{v}_{ij} = (r_{ij},\ \mathbf{r}_{ij},\ s_i^z \cdot s_j^z)$$

$$\mathbf{h}_i^{(0)} = (\mathbf{v}_i,\ A\mathbf{v}_i)$$
$$\mathbf{h}_{ij}^{(0)} = (\mathbf{v}_{ij},\ B\mathbf{v}_{ij})$$

for $t = 1, \ldots, T$:

$$\mathbf{m}_{ij}^{(t)} = \mathbf{M}_t\left(\mathbf{h}_i^{(t-1)},\ \mathbf{h}_j^{(t-1)},\ \mathbf{h}_{ij}^{(t-1)}\right)$$
$$\mathbf{h}_i^{(t)} = \left(\mathbf{v}_i,\ \mathbf{F}_t\left(\mathbf{h}_i^{(t-1)}, \mathbf{m}_i^{(t)}\right)\right),\ \mathbf{m}_i^{(t)} = \mathrm{Pool}\left(\{\mathbf{m}_{ij}^{(t)} \,|\, j \neq i\}\right)$$
$$\mathbf{h}_{ij}^{(t)} = \left(\mathbf{v}_{ij},\ \mathbf{G}_t\left(\mathbf{h}_{ij}^{(t-1)}, \mathbf{m}_{ij}^{(t)}\right)\right)$$

Determinants vs. Pfaffians

Defined for $n \times n$ matrices

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

$$\det(A^T) = \det(A)$$

$$\det(A)\det(B) = \det(AB)$$

Defined for $2n \times 2n$ skew-symmetric matrices

$$\operatorname{pf}(A) = \frac{1}{2^n n!} \sum_{\sigma \in S_{2n}} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{\sigma(2i-1),\sigma(2i)}$$

$$\operatorname{pf}(A^T) = (-1)^n \operatorname{pf}(A)$$

$$\operatorname{pf}(A)\operatorname{pf}(B) = \exp\left(\frac{1}{2}\operatorname{tr}\log(A^T B)\right)$$

$$\det(A) = \operatorname{pf}(A)^2$$