Variational Wavefunctions in the era of AI

Virtually in the ether

Bryan Clark (bkclark@illinois.edu)
University of Illinois at Urbana Champaign

Dmitrii Kochkov

Di Luo

The Variational Approach

The variational approach is one of the key approaches to solving the quantum many-body problem.

Given a Hamiltonian H, find a good approximation Ψ to the ground state…

$$R \Longrightarrow \boxed{\Psi} \Longrightarrow \#$$

Choose a set of wave-functions you're going to consider…

$$\{\Psi_1, \Psi_2, \Psi_3, \ldots \Psi_N\}$$

$$\{\Psi(\vec{\theta})\}$$

Pick the best one from that list.

Outline

Choose a set of wave-functions you're going to consider…

$\{\Psi_1, \Psi_2, \Psi_3, \ldots \Psi_N\}$

$\{\Psi(\overrightarrow{\theta})\}$

Part 1

Wavefunctions

Neural Network Backflow

Pick the best one from that list.

Part 2

Optimization

How is the age of AI changing these two steps?

Wave-Functions through History

The age of (dressed) mean field.

Slater Determinants (i.e. Hartree Fock)

The age of tensor networks

The age of AI

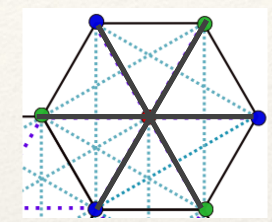
The Age of Mean Field

- Slater Determinants
- BDG States
- Pfaffians

Jastrow

$\Psi = \det M \exp[-U(R)]$

$\Psi = P[\det M]$ *(spin liquids)*



J1-J2 triangular spin liquid: 93% overlap with projected SD

Fractional Chern Insulators

$\Psi(r_1, r_2, ..r_n) = \det M;$

$M_{ij} = \phi_i(r_j)$

Single particle orbitals

The Age of Mean Field

- Slater Determinants
- BDG States
- Pfaffians

Jastrow

$\Psi = \det M \exp[-U(R)]$

$\Psi = P[\det M]$ *(spin liquids)*

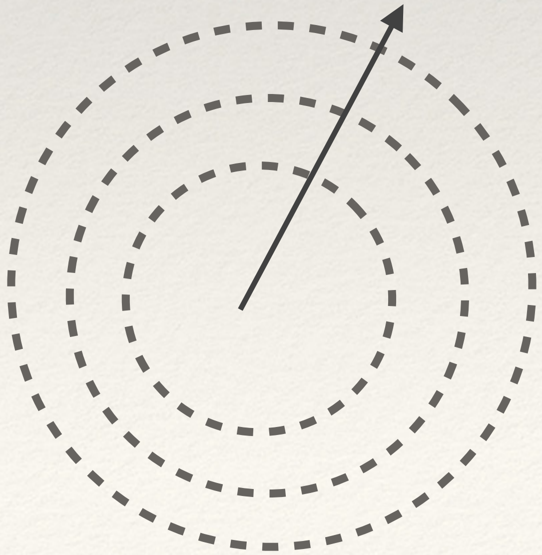
$\Psi(r_1, r_2, ,.. r_n) = \det M;$

$M_{ij} = \phi_i(r_j)$

Multi-Determinants

More determinants

Enough determinants give you everything

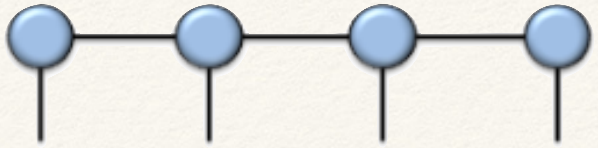


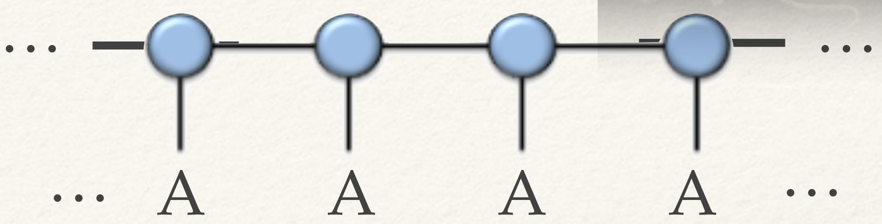
The Age of Tensor Networks

Low entanglement ansatz

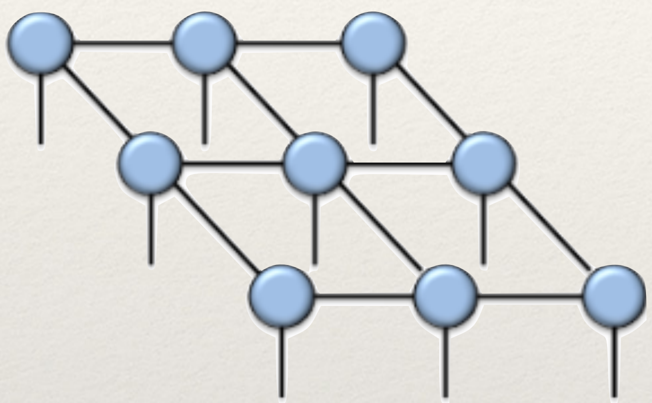
Infinite Ansatz

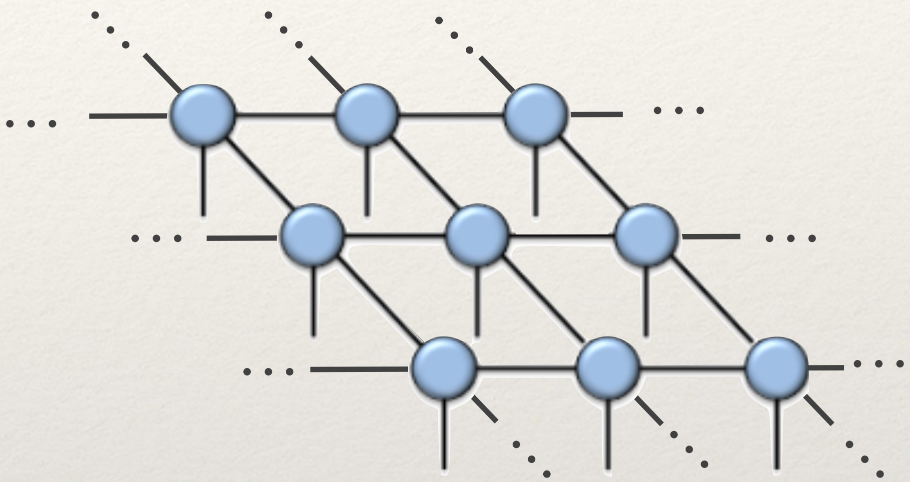
Matrix Product States



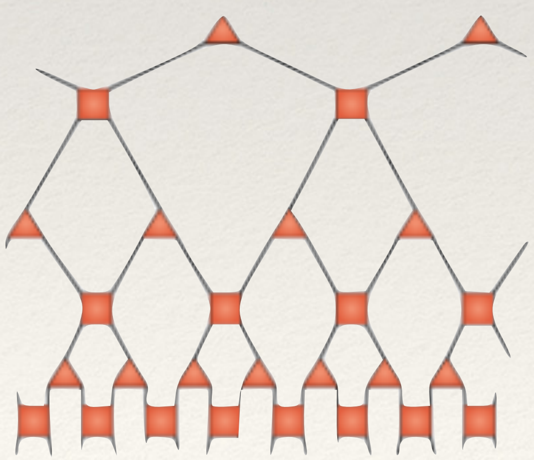


PEPS



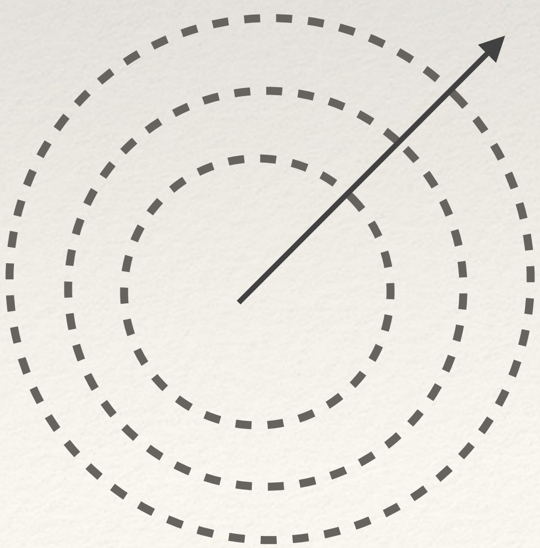


MERA



Larger Bond-dimension

Enough bond-dimension gives you everything

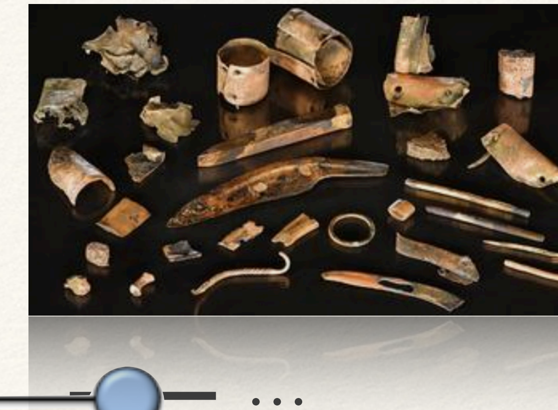


White, PRL

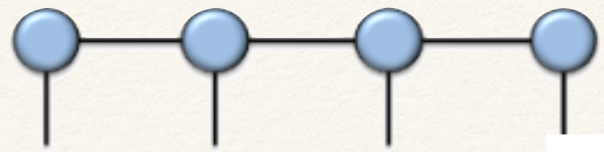
The Age of Tensor Networks

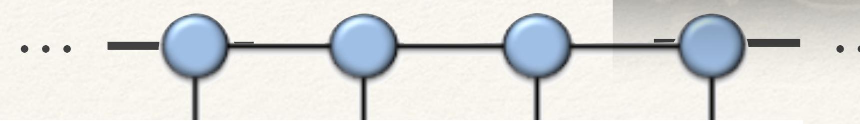
Low entanglement ansatz

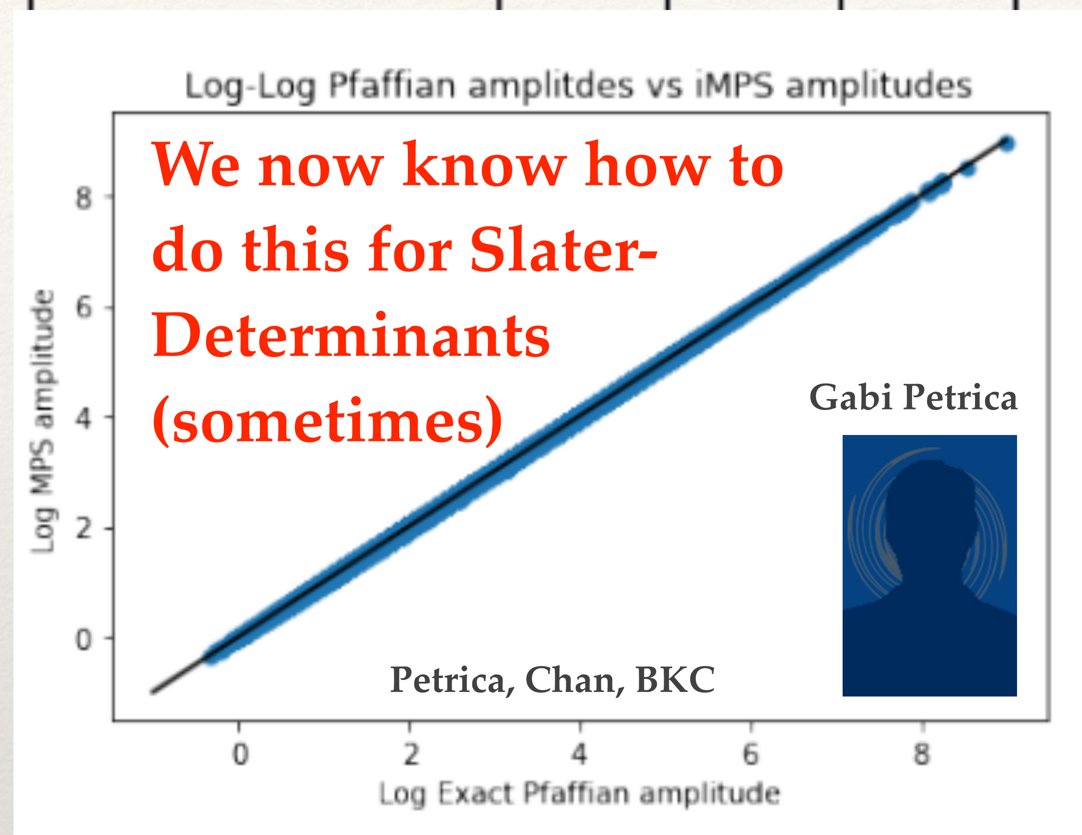
Infinite Ansatz



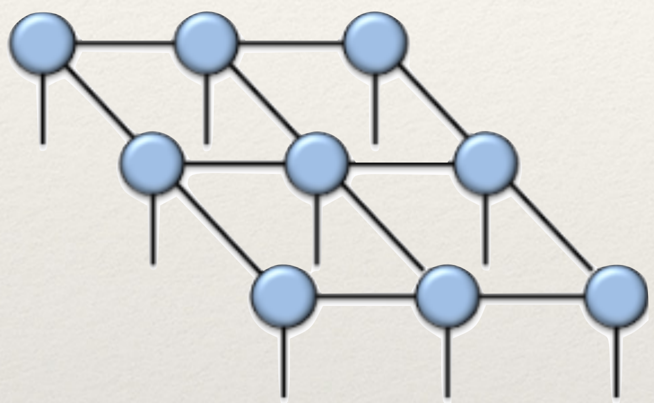
Matrix Product States



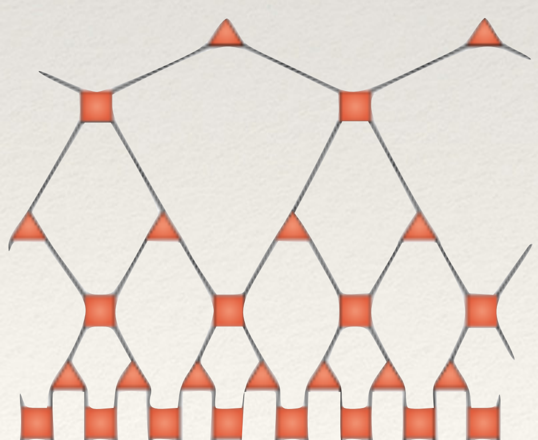




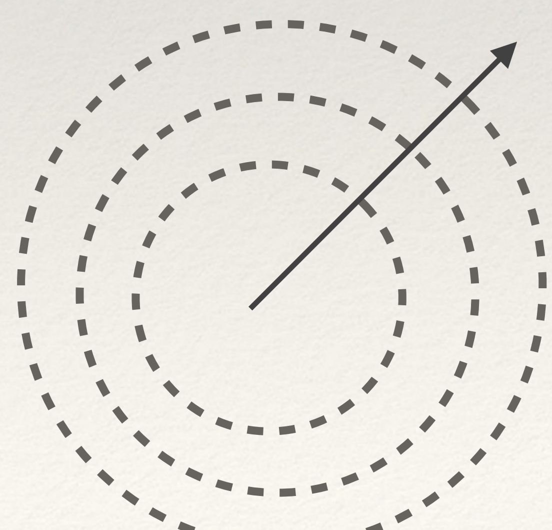
PEPS



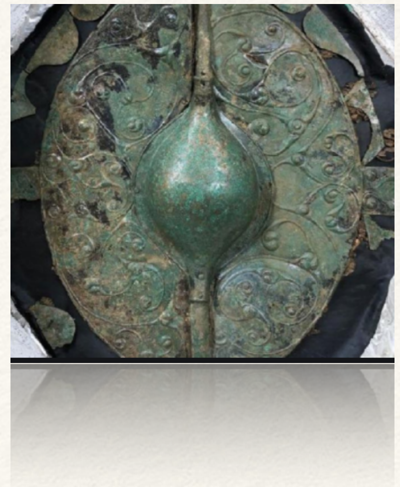
MERA



Enough bond-dimension gives you everything

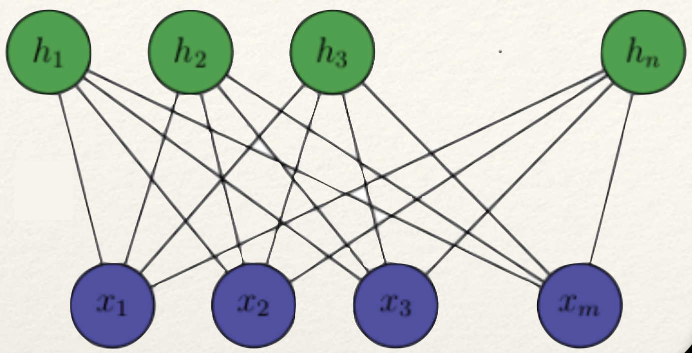


The Age of AI

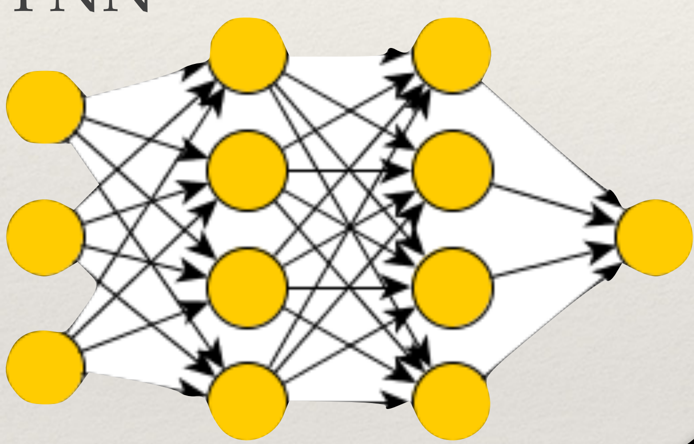


RBM

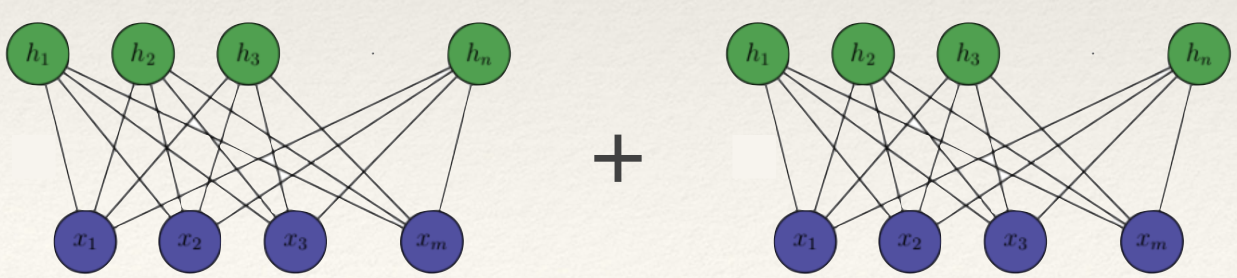
Carleo, Troyer, Nature….



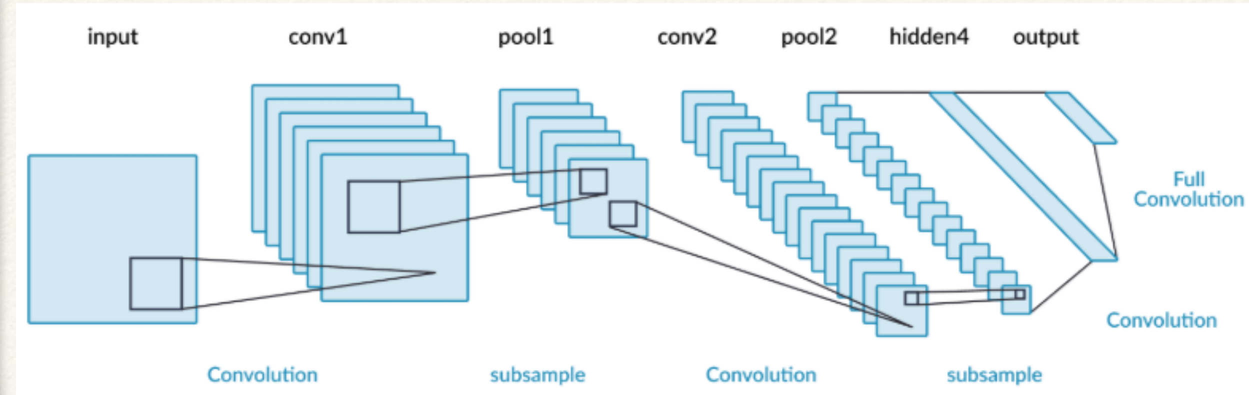
FFNN



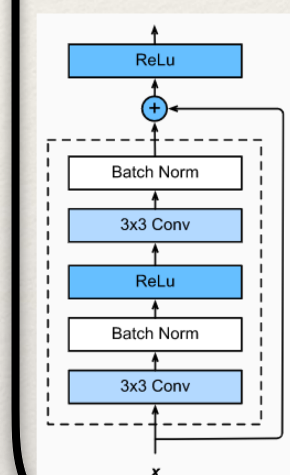
Multi-RBM



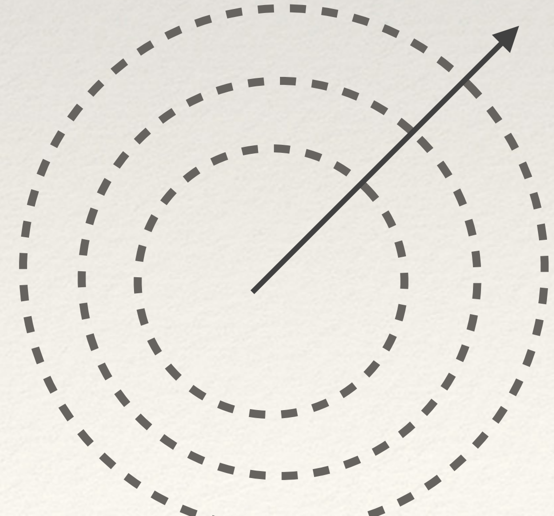
Convolution neural network



Resnet

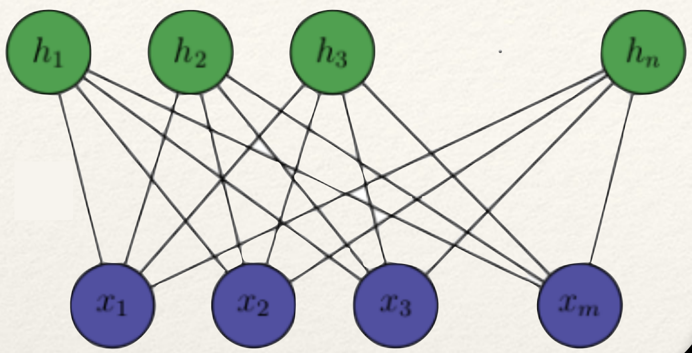


Larger NN,
Enough `neurons' gives you everything

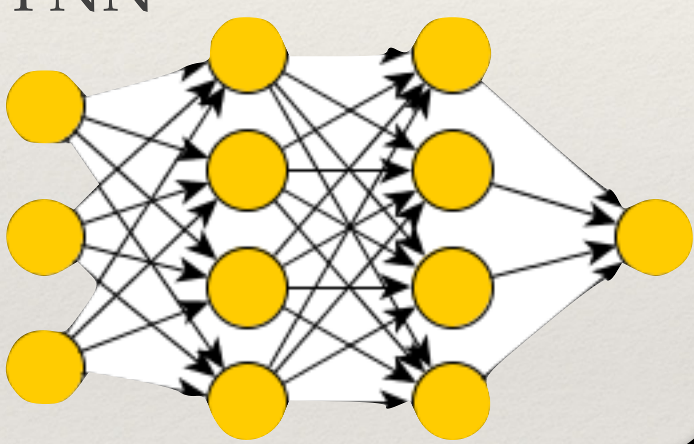


The Age of AI

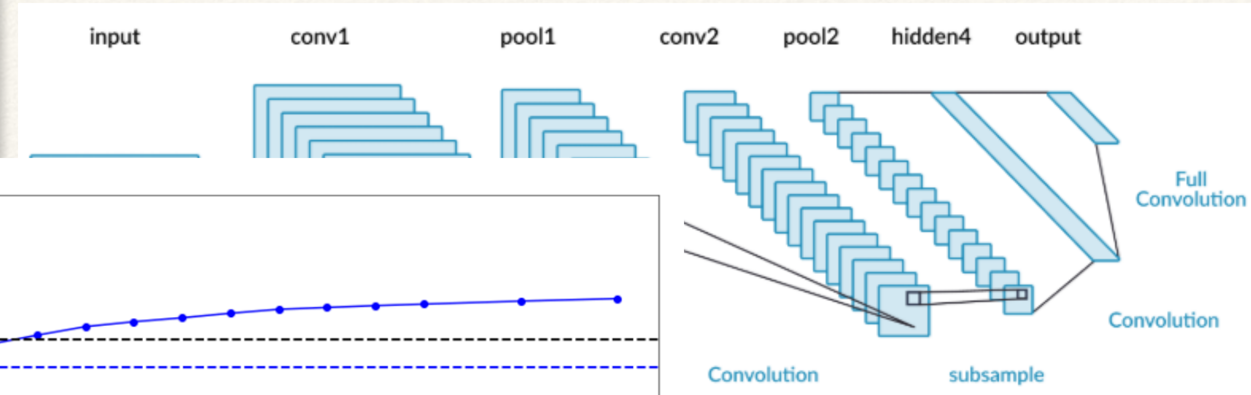
RBM

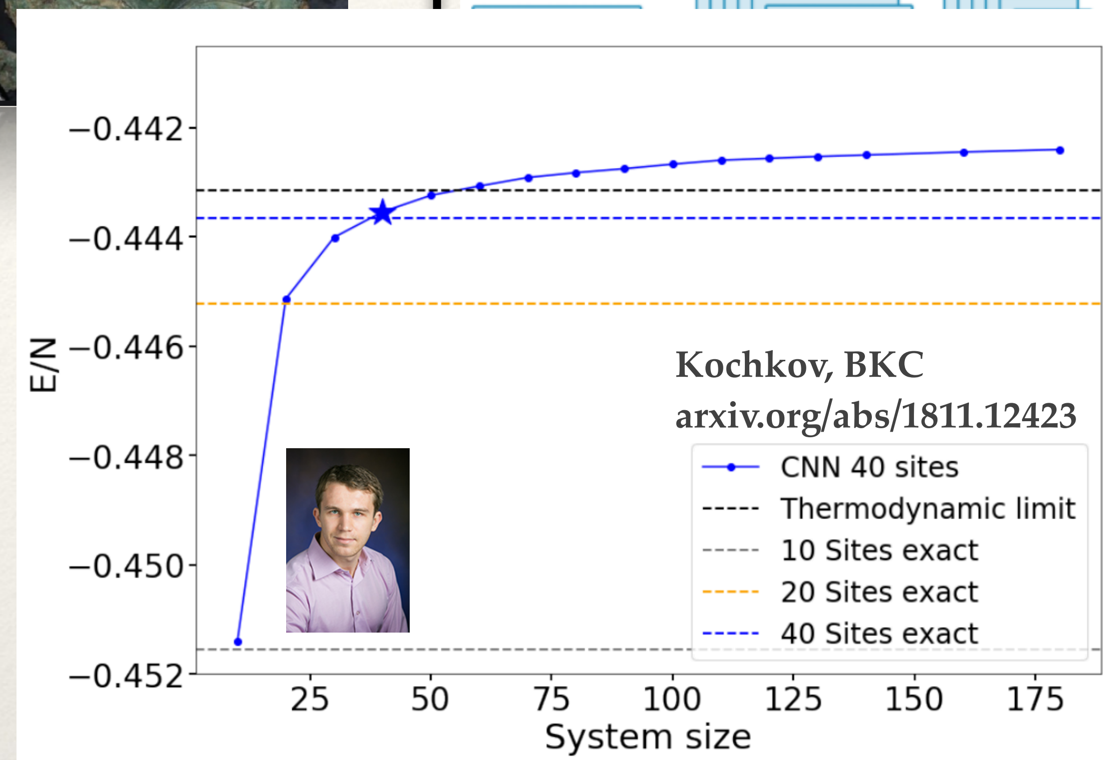


FFNN



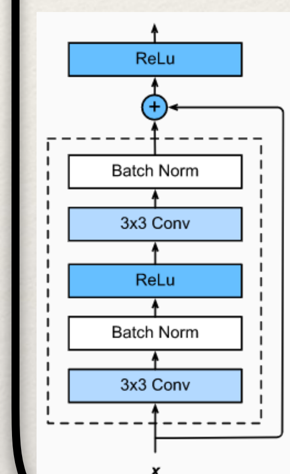
Convolution neural network



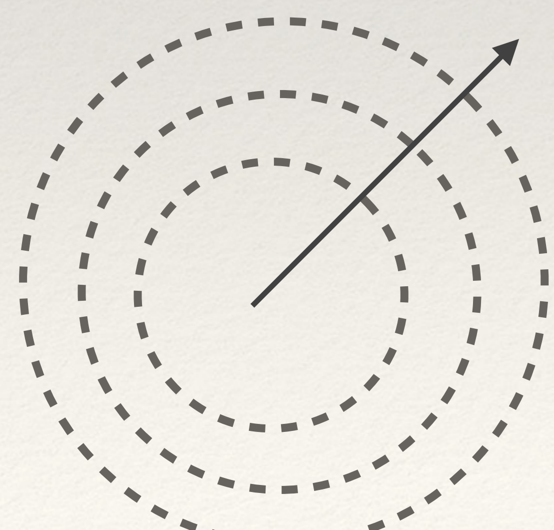


A New Infinite Ansatz

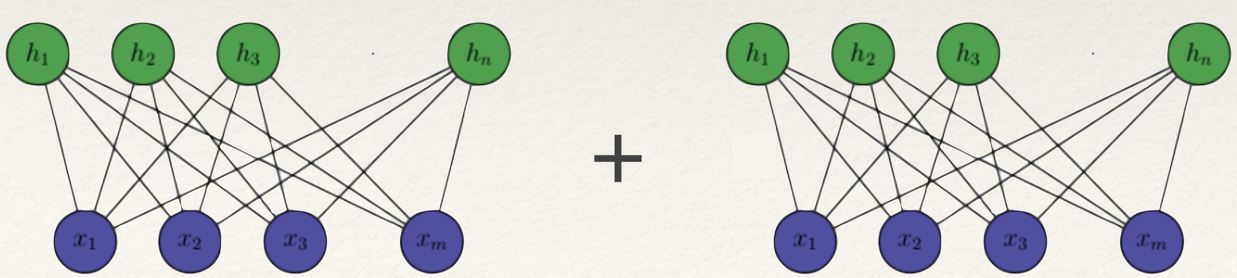
Resnet



Larger NN,
Enough `neurons' gives you everything

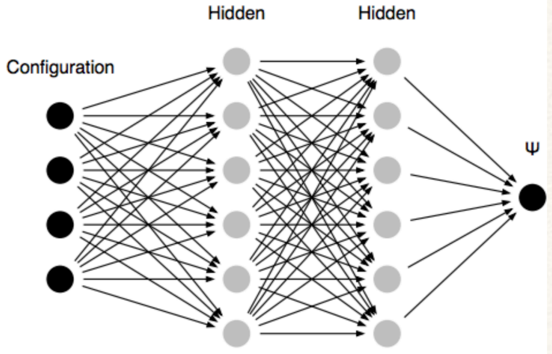


Multi-RBM

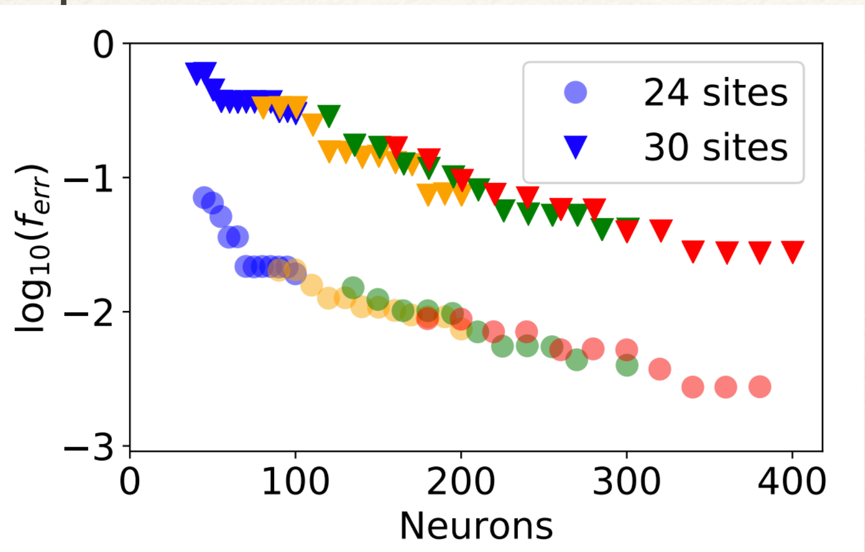


How good are neural networks..

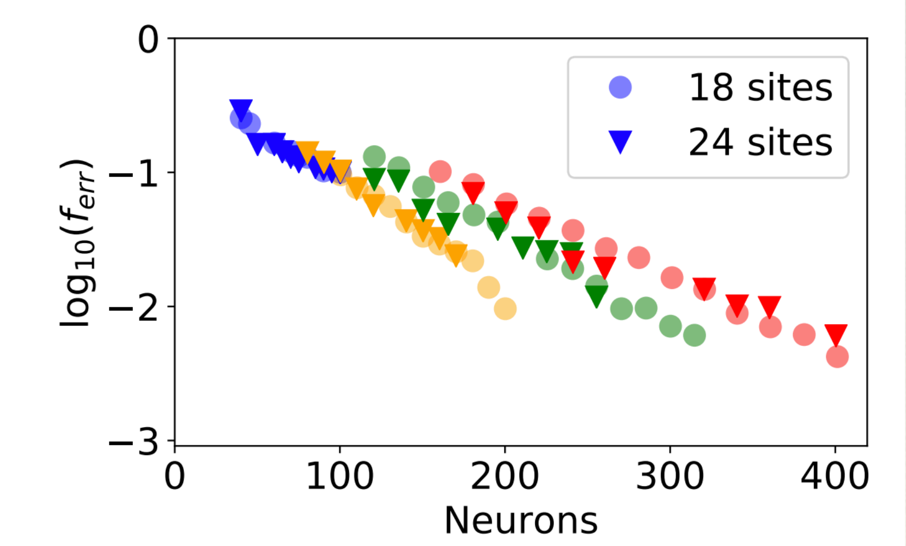
J1 Heisenberg Models



Square



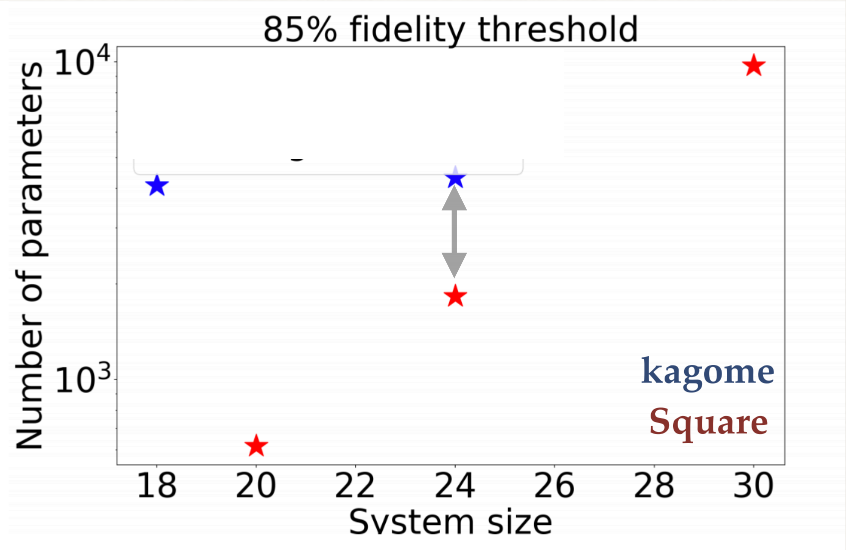
Kagome



Exponential improvement with neuron number

Depth doesn't help (hurts a little?)

24-30 site systems require a lot of parameters (~10^4)



Square: Much fewer parameters…

Wave-Functions with Signs

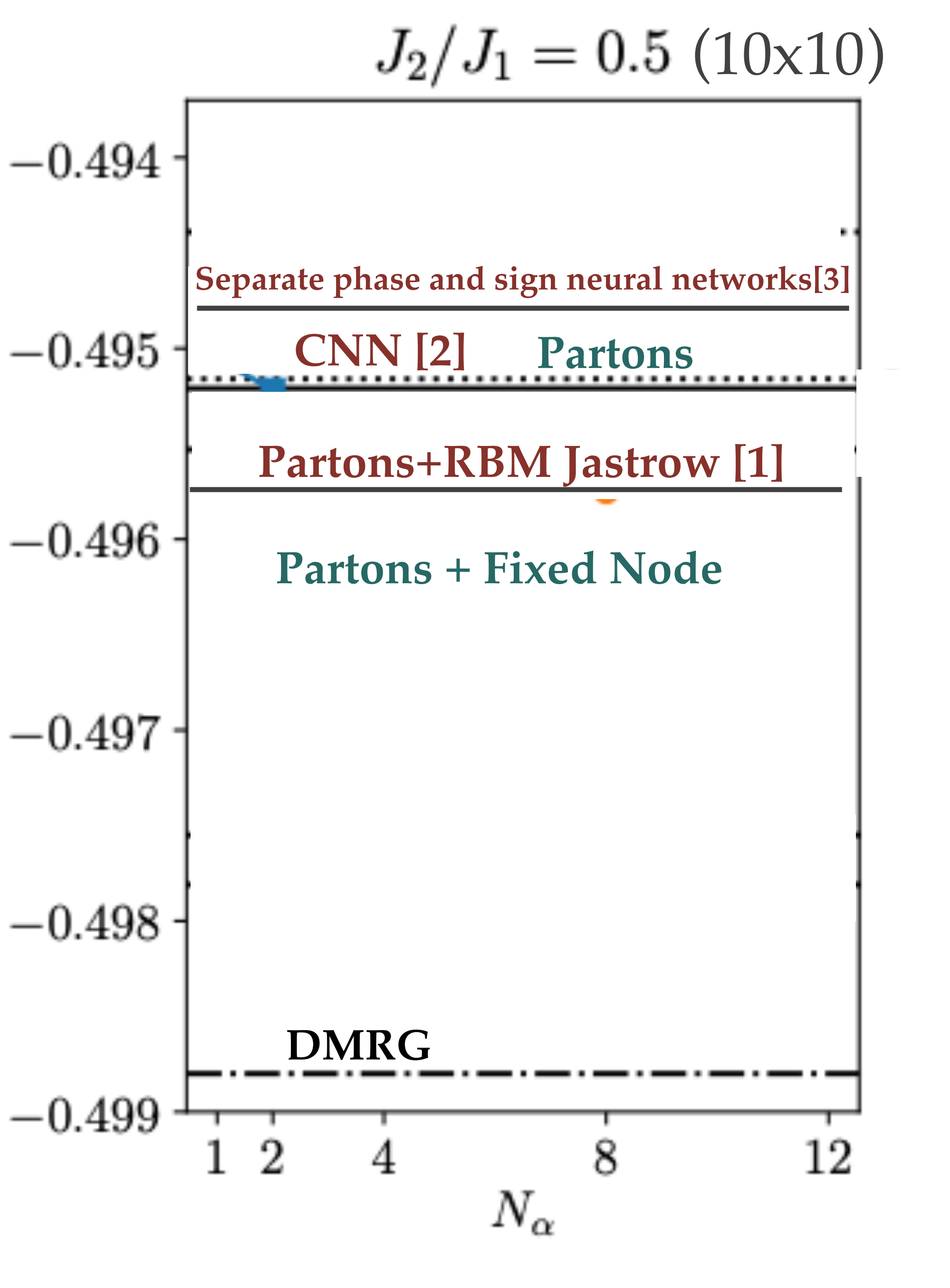
Neural Nets do well with wave-functions with simple sign structures (but so does QMC).
What about wave-functions with complicated sign structure?

RBM: Can't even get sign structures without complex weights (and this hasn't worked out yet)

Square Heisenberg

- Neural-networks competitive with partons…
- Worse then fixed-node+partons
- Much worse with DMRG

Partons+RBM-Jastrow also done in Hubbard by Imada



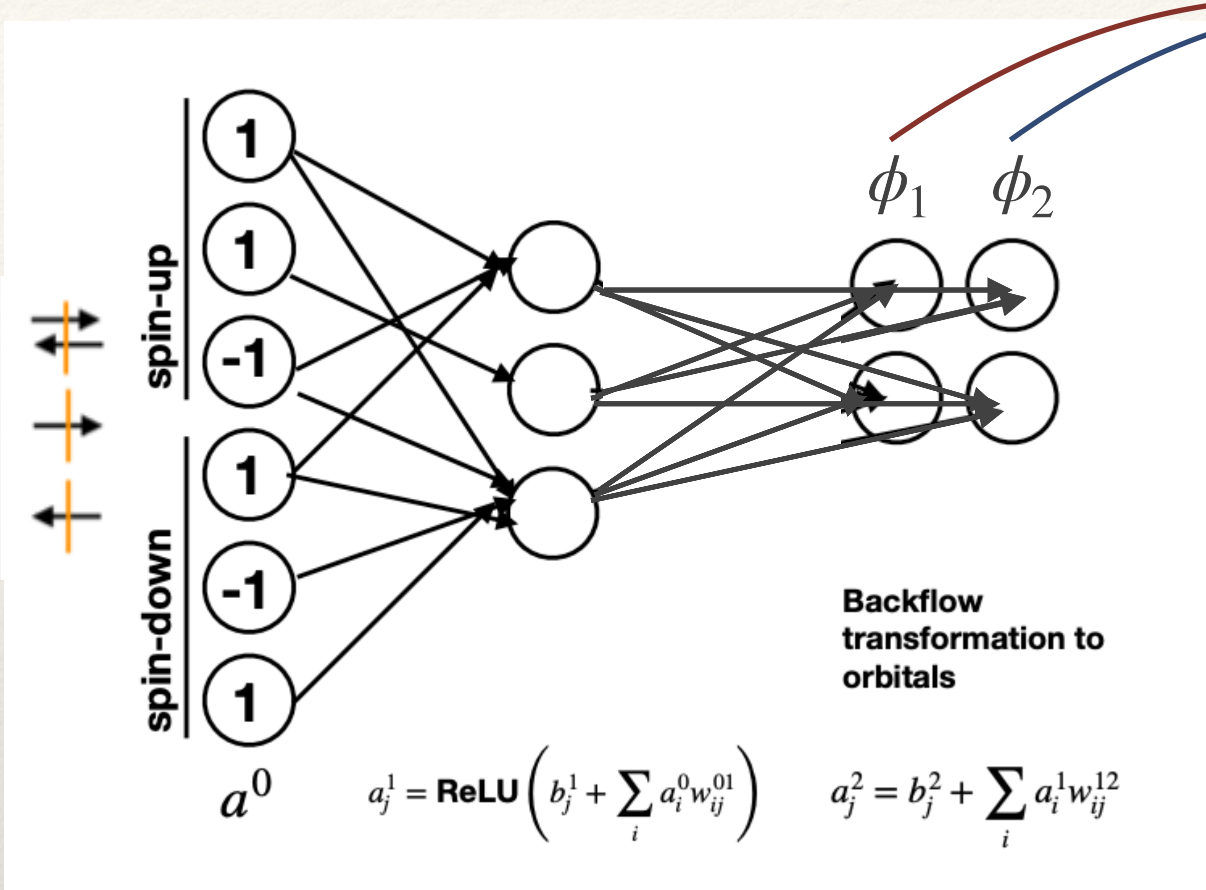
[1] Neural Gutzwiller-projected variational wave functions
Francesco Ferrari,[1,*] Federico Becca,[2] and Juan Carrasquilla[3,4]

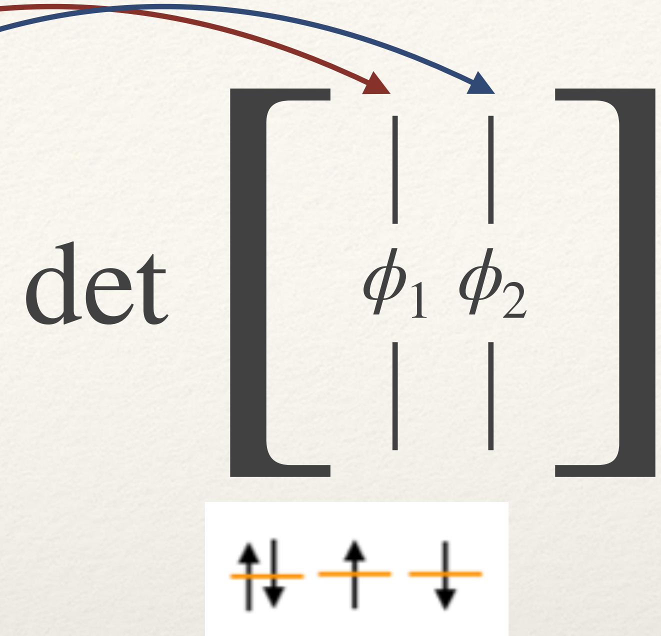
[2] Study of the Two-Dimensional Frustrated J1-J2 Model with Neural Network Quantum States
Kenny Choo,[1] Titus Neupert,[1] and Giuseppe Carleo[2]

[3] Neural network wave functions and the sign problem
Attila Szabó and Claudio Castelnovo

[4] Solving frustrated quantum many-particle models with convolutional neural networks
Xiao Liang,[1,2] Wen-Yuan Liu,[1,2,3] Pei-Ze Lin,[1,2] Guang-Can Guo,[1,2] Yong-Sheng Zhang,[1,2,*] and Lixin He[1,2,†]

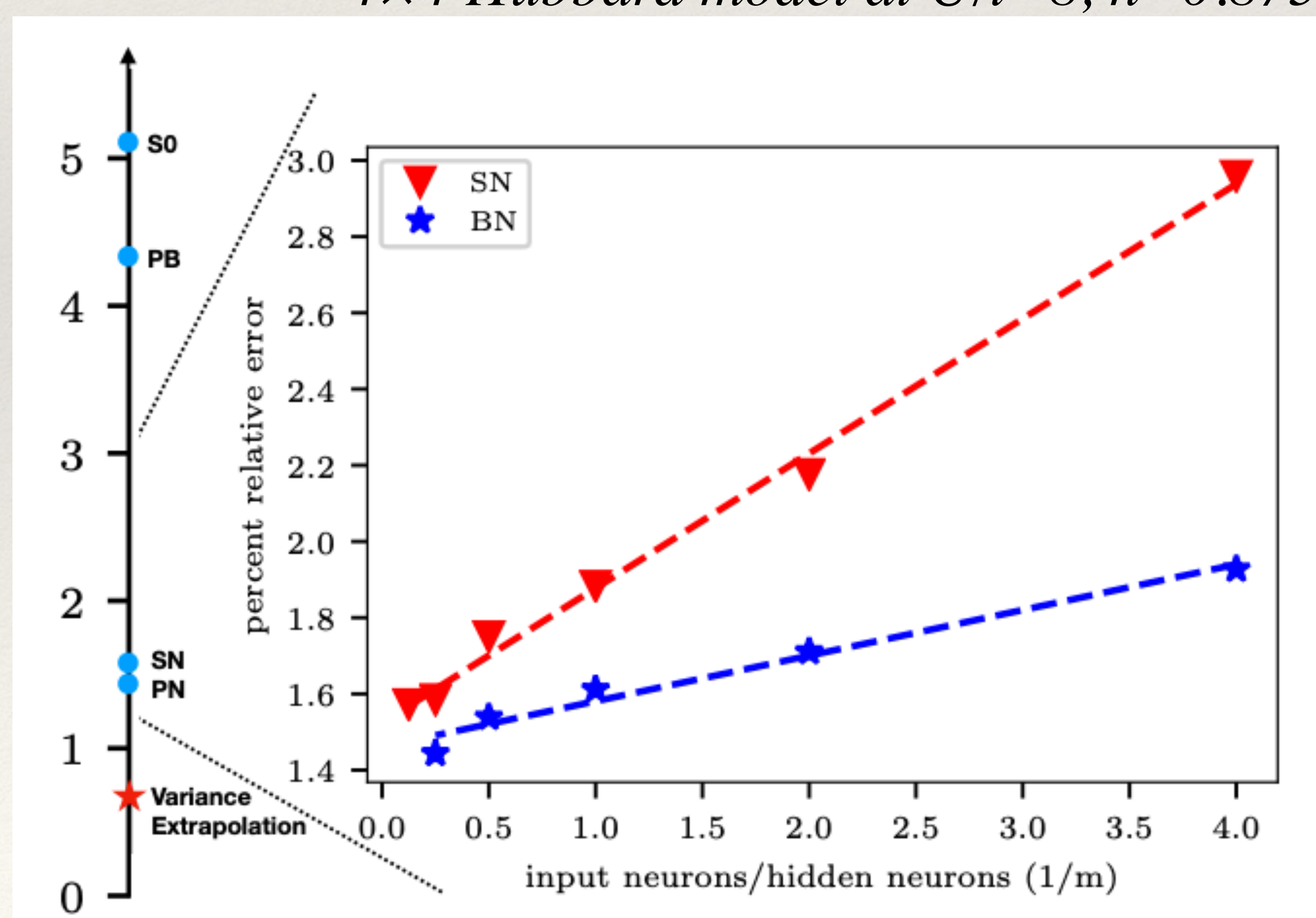
Neural Network Backflow





$$\psi_{SD}(\boldsymbol{r}) = \det\left[M^{SD,\uparrow}\right] \det\left[M^{SD,\downarrow}\right];$$
$$M_{ik}^{SD,\sigma} = \phi_{k\sigma}(r_{i\sigma})$$
$$\phi_{k\sigma}^{b}(r_{i,\sigma};\boldsymbol{r}) = \phi_{k\sigma}(r_{i,\sigma}) + a_{ki,\sigma}^{NN}(\boldsymbol{r})$$

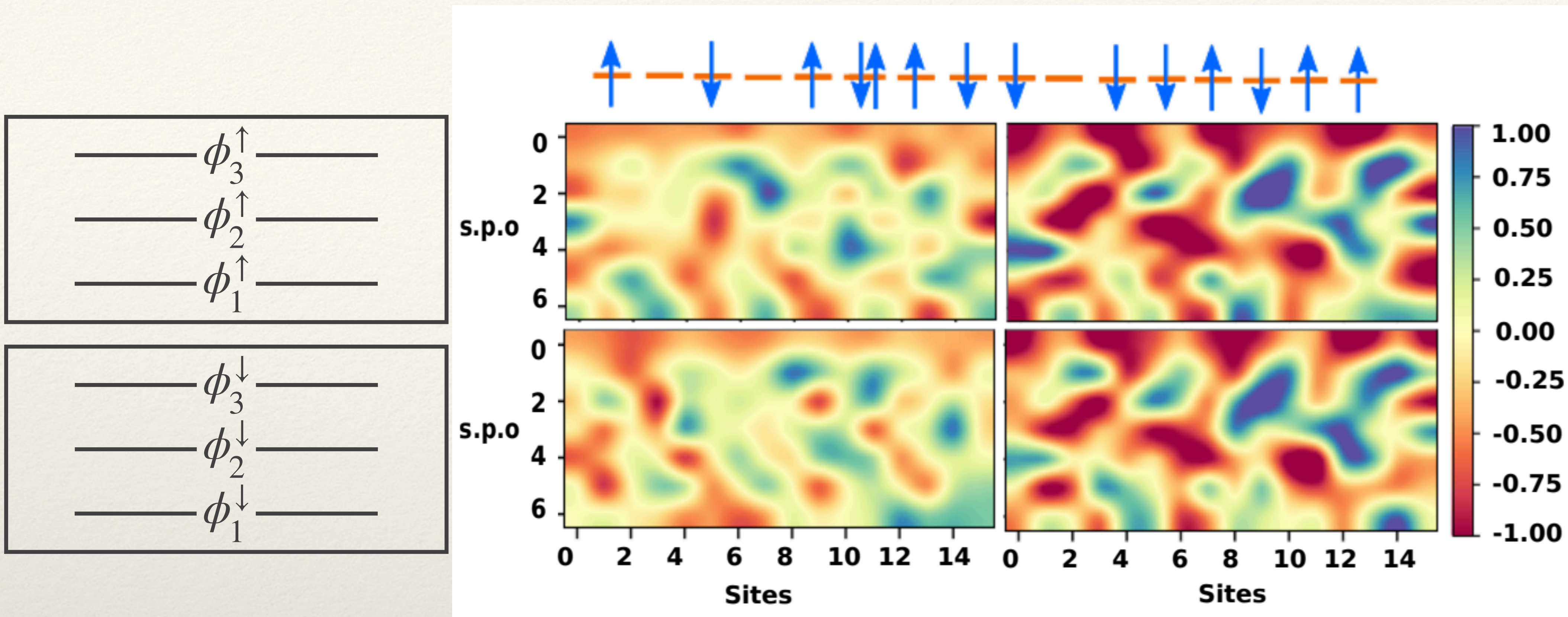
4×4 Hubbard model at U/t=8, n=0.875



Cost: $O(N^4)$ per sweep

Duo, BKC, PRL

We not only get better energies; we restore the symmetry.



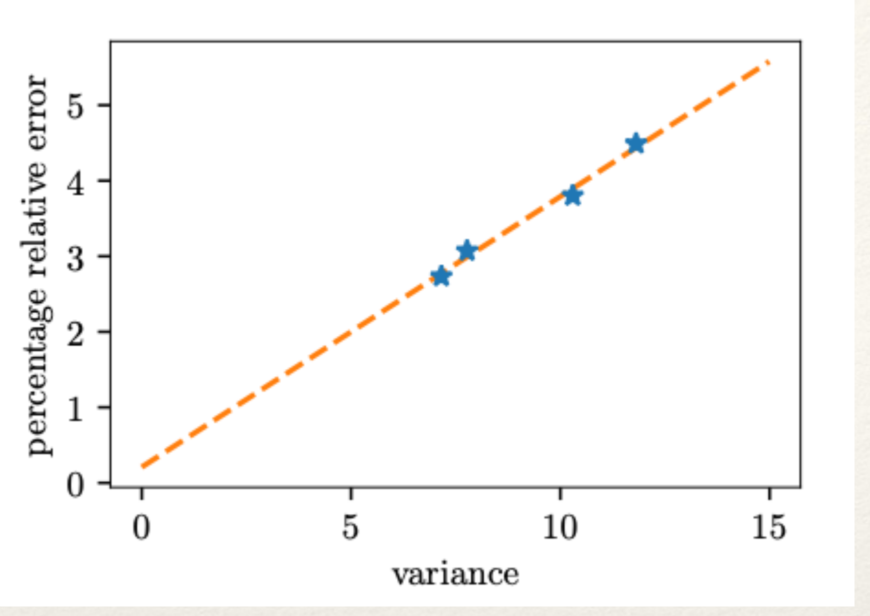
and change the signs…

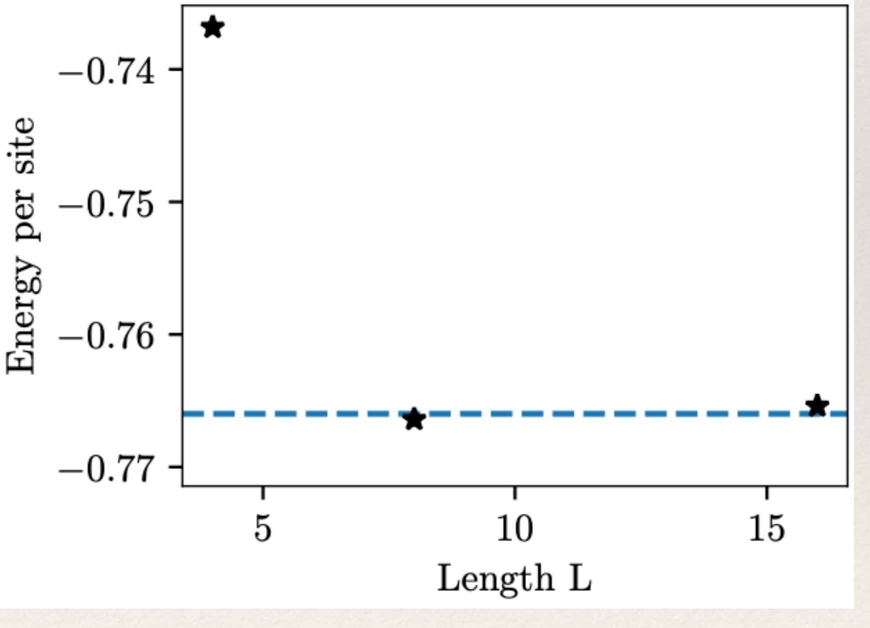
$$\frac{\int |\Psi_{\mathrm{SO}}(x)|^2 \mathrm{sgn}(\Psi_{\mathrm{SN}}(x))\mathrm{sgn}(\Psi_{\mathrm{SO}}(x))dx}{\int |\Psi_{\mathrm{SO}}(x)|^2 dx} = 0.815$$

9% difference between signs…

Bigger systems….

Relative energy error	Slater-Jastrow	NNB	NNB variance extrapolation
16×4 Hubbard, n=0.875	$(5.9 \pm 2 \times 10^{-3})\%$	$(2.734 \pm 8 \times 10^{-3})\%$	0.209%
12×8 Hubbard, n=0.875	$(6.3 \pm 3 \times 10^{-3})\%$	$(3.94 \pm 10^{-2})\%$	0.655%
$4 \times 4 \times 3$ Kagome	$(1.8 \pm 10^{-5})\%$	$(1.093 \pm 4 \times 10^{-3})\%$	0.286%





iDMRG answer

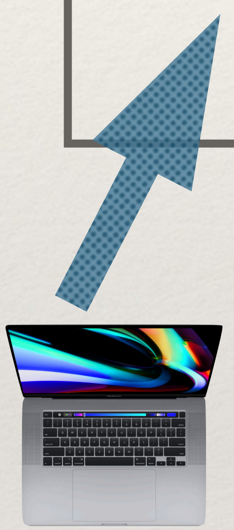
Solutions of the Two-Dimensional Hubbard Model: Benchmarks and Results from a Wide Range of Numerical Algorithms

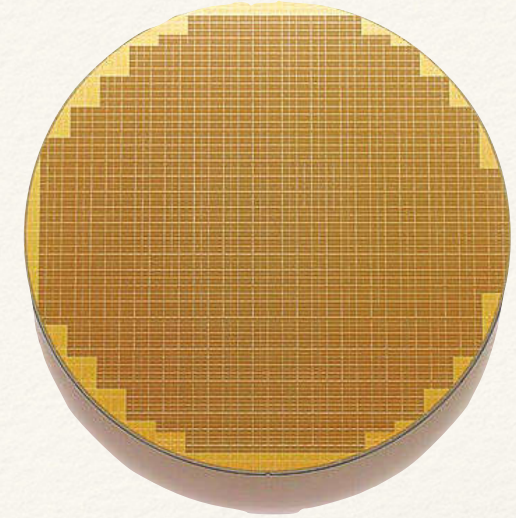
J. P. F. LeBlanc, Andrey E. Antipov, Federico Becca, Ireneusz W. Bulik, Garnet Kin-Lic Chan, Chia-Min Chung, Youjin Deng, Michel Ferrero, Thomas M. Henderson, Carlos A. Jiménez-Hoyos, E. Kozik, Xuan-Wen Liu, Andrew J. Millis, N. V. Prokof'ev, Mingpu Qin, Gustavo E. Scuseria, Hao Shi, B. V. Svistunov, Luca F. Tocchio, I. S. Tupitsyn, Steven R. White, Shiwei Zhang, Bo-Xiao Zheng, Zhenyue Zhu, and Emanuel Gull (Simons Collaboration on the Many-Electron Problem)

Phys. Rev. X **5**, 041041 – Published 14 December 2015

Beyond the silver age…

(and how close are we to reaching it?)





Silicon Age

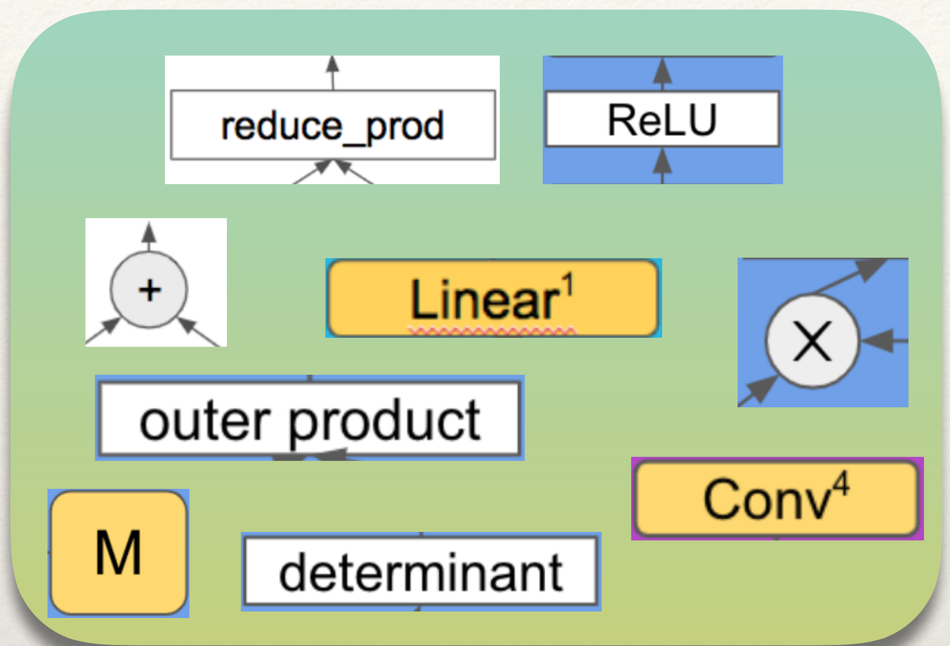
We'd like to consider the wave-functions accessible `quickly' by computers.

Q: What's a good way to represent this space?

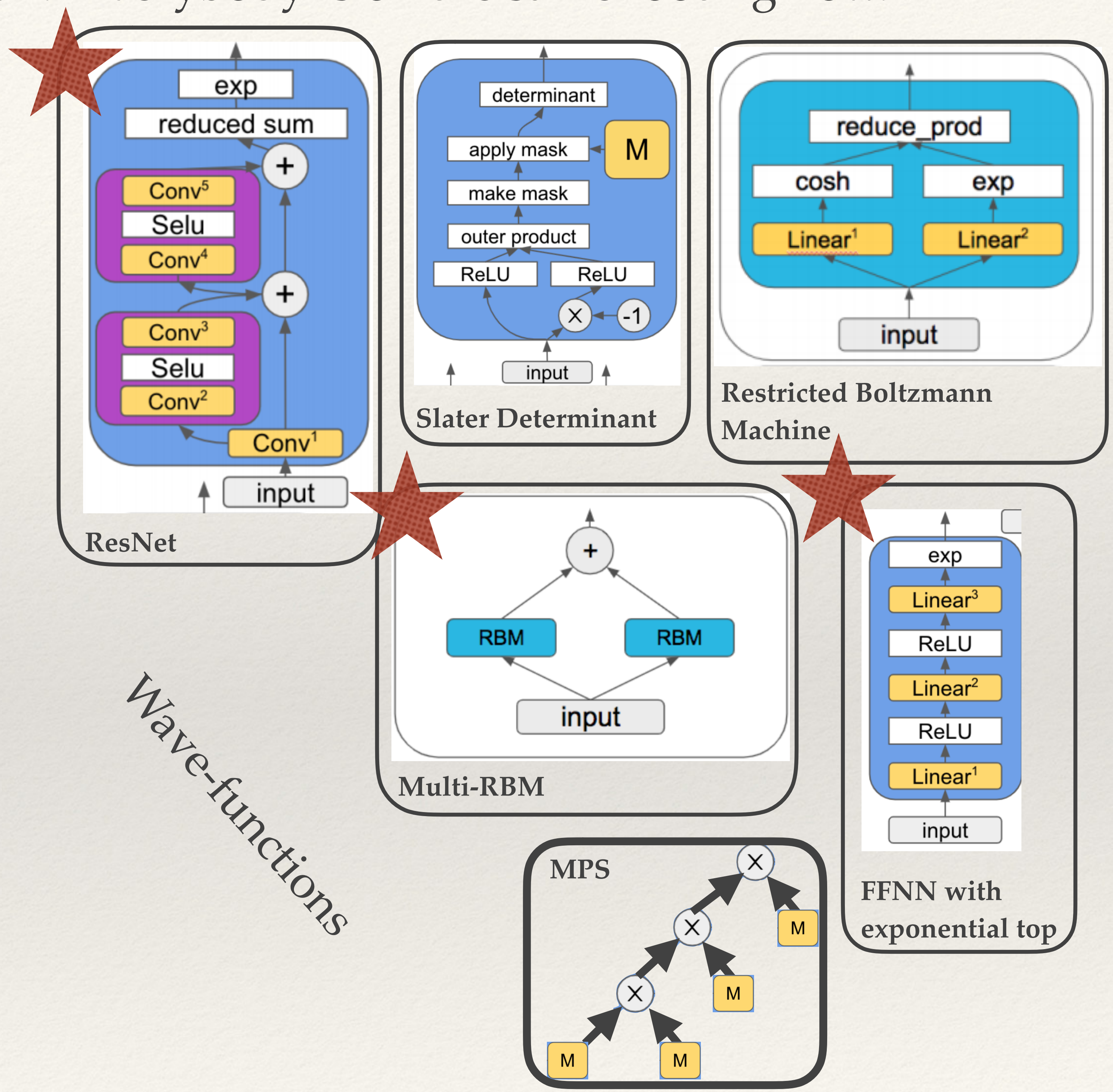
Our answer (stolen from machine-learning): computational graph states…

Computational Graph States

Taking a page, from the machine learning play-book, we can represent all "fast wave-functions" as a computational graph. Everybody is on the same footing now.



Box of operations…



WF	Hidden Layers	Hidden Units	Params 1d - 2d
FCNN	2	80	9800 - 11744
RBM	0	80	3320 - 5264
FC-RBM	2	80	16280 - 18224
Conv1D	5	16 filters (size 5)	5280 -
Conv2D	5	16 filters (5 by 5)	- 26080
ResNet	1 + 2(2)	16 filters (size 5)	5280 - 25760
Multi-RBM	2	16 filters (5 by 5)	6640 - 10528
Multi-FCNN	2	80	19600 - 23488
P-BDG			1600 - 4096
MPS			4800 -

Kochkov, BKC
arxiv.org/abs/1811.12423

Good representation….still a question of optimization

```python
class ProjectedBDG(Wavefunction):
  """P-BDG module."""

  def __init__(
      self,
      num_sites: int,
      name: str = 'projected_bdg'):
    """Constructs a projected BDG module.

    Args:
      num_sites: Number of sites.
      name: Name of the module.
    """
    super(ProjectedBDG, self).__init__(name=name)
    self._num_sites = num_sites
    with self._enter_variable_scope():
      self._pairing_matrix = tf.get_variable(
          'pairing_matrix', shape=[1, num_sites, num_sites], dtype=tf.float32)

  def _build(self, inputs: tf.Tensor) -> tf.Tensor:
    """Connects the P-BDG module into the graph with input `inputs`.

    Args:
      inputs: Tensor with input values of shape=[batch] and values +/- 1.

    Returns:
      Wave-function amplitudes of shape=[batch].
    """
    batch_size = inputs.shape[0]
    n_sites = self._num_sites
    mask = tf.einsum('ij,ik->ijk', tf.nn.relu(inputs), tf.nn.relu(-inputs))
    bool_mask = tf.greater(mask, tf.zeros([batch_size, n_sites, n_sites]))
    tiled_pairing = tf.tile(self._pairing_matrix, [batch_size, 1, 1])
    det_size = [batch_size, n_sites // 2, n_sites // 2]
    pre_det = tf.reshape(tf.boolean_mask(tiled_pairing, bool_mask), det_size)

    sign, ldet = tf.linalg.slogdet(pre_det)
    det_value = tf.exp(self.add_exp_normalization(ldet))
    return sign * det_value

  @classmethod
  def from_hparams(
      cls,
      hparams: tf.contrib.training.HParams,
      name: str = ''
  ) -> 'Wavefunction':
    """Constructs an instance of a class from hparams."""
    pbdg_params = {
        'num_sites': hparams.num_sites,
    }
    if name:
      pbdg_params['name'] = name
    return cls(**pbdg_params)
```

```python
    super(Conv2DNetwork, self).__init__(name=name)
    self._num_layers = num_layers
    self._num_filters = num_filters
    self._kernel_size = kernel_size
    self._nonlinearity = nonlinearity
    self._output_activation = output_activation
    self._size_x = size_x
    self._size_y = size_y

    reduction = functools.partial(tf.reduce_sum, axis=[1, 2, 3])
    self._components = []
    with self._enter_variable_scope():
      for layer in range(num_layers):
        self._components.append(layers.Conv2dPeriodic(num_filters, kernel_size))
        if layer + 1 != num_layers:
          self._components.append(nonlinearity)
      if output_activation == tf.exp:
        self._components += [reduction, self.add_exp_normalization, tf.exp]
      else:
        self._components += [reduction, output_activation]

  def _build(
      self,
      inputs: tf.Tensor,
  ) -> tf.Tensor:
    """Builds computational graph evaluating the wavefunction on inputs.

    Args:
      inputs: Input tensor, must have shape (batch, num_sites, ...).

    Returns:
      Tensor holding values of the wavefunction on `inputs`.

    Raises:
      ValueError: Input tensor has wrong shape.
    """
    inputs_new_shape = [-1, self._size_x, self._size_y, 1]
    inputs = tf.reshape(inputs, shape=inputs_new_shape)
    return snt.Sequential(self._components)(inputs)

  @classmethod
  def from_hparams(
      cls,
      hparams: tf.contrib.training.HParams,
      name: str = ''
  ) -> 'Wavefunction':
    """Constructs an instance of a class from hparams."""
    conv_2d_params = {
        'num_layers': hparams.num_conv_layers,
        'num_filters': hparams.num_conv_filters,
        'kernel_size': hparams.kernel_size,
        'size_x': hparams.size_x,
        'size_y': hparams.size_y,
        'output_activation': layers.NONLINEARITIES[hparams.output_activation],
        'nonlinearity': layers.NONLINEARITIES[hparams.nonlinearity],
    }
    if name:
      conv_2d_params['name'] = name
    return cls(**conv_2d_params)
```

Part 2 Optimizing wave-functions in the age of AI

Wave-Function optimization through history

Ancient History

Stochastic Gradient Descent

Given an objective function, walk downhill.

$$E(\overrightarrow{\theta}) \equiv \langle \Psi(\overrightarrow{\theta}) | H | \Psi(\overrightarrow{\theta}) \rangle$$

$$\theta_i \leftarrow \theta_i - \delta \frac{\partial E}{\partial \theta_i}$$

$O(p)$

Slow convergence

(actually, pre-ancient history optimized the variance)

Imaginary Time Evolution

Stochastic Reconfiguration

iTEBD for VMC

$O(p^3)$

Numerically unstable

Undersampling problem

$$|\Psi_{new}\rangle = \mathscr{P}(1 - \tau H) | \Psi_{old}\rangle$$

Projection into manifold…

Need: S_{ij}^{-1} where $S_{ij} \equiv \left\langle \frac{\partial \Psi}{\partial \theta_i} \middle| \frac{\partial \Psi}{\partial \theta_j} \right\rangle$

Linear Method

DMRG for VMC

$O(p^3)$

Numerically unstable

Complicated

$$\widetilde{H} | \Psi \rangle = ES | \Psi \rangle$$

$$\widetilde{H}_{ij} \equiv \left\langle \frac{\partial \Psi}{\partial \theta_i} \middle| H \middle| \frac{\partial \Psi}{\partial \theta_j} \right\rangle$$

Supervised Learning…

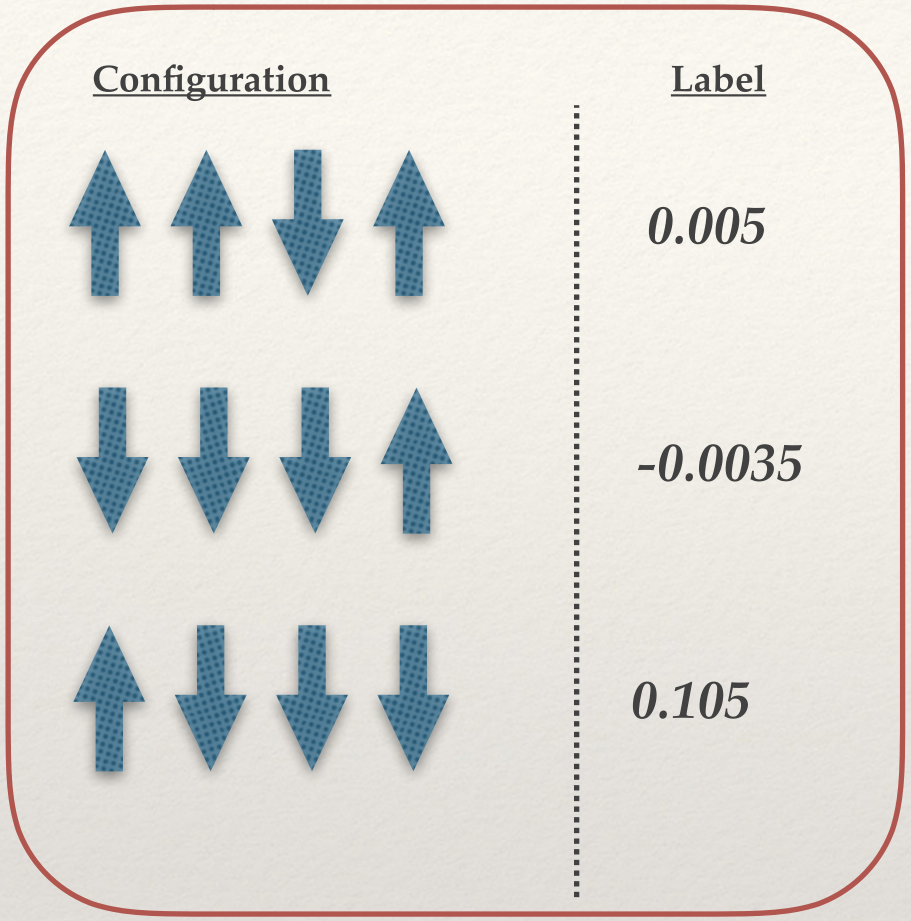
IMAGE	LABEL
	*CAT*
	*DOG*
	*CAT*

Train your supervised algorithm on these labels.

If you give it a new image, it should be able to tell if it's a cat or dog.

Cost: $O(p)$

Supervised Learning…



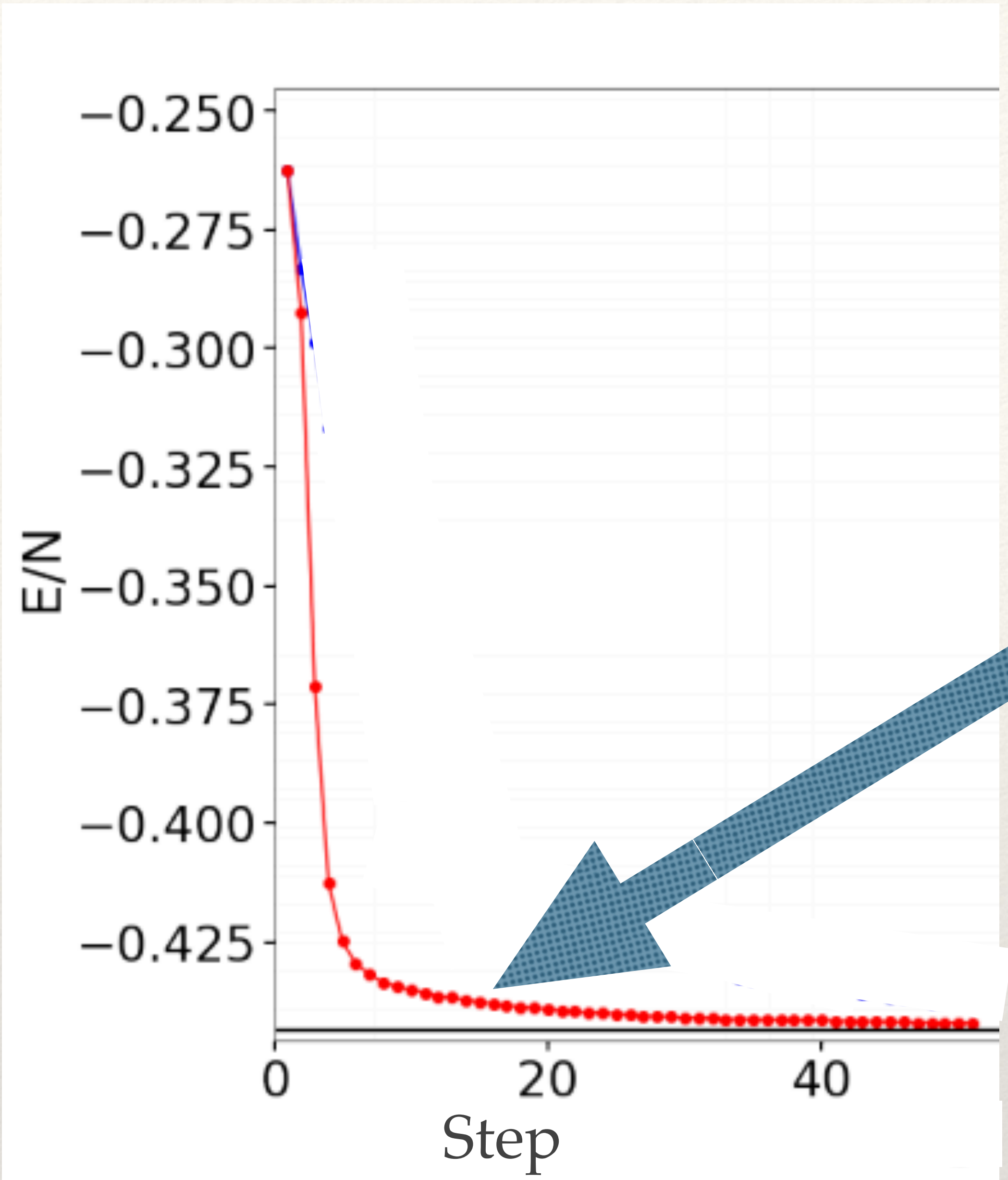
Train your supervised algorithm on these labels.

If you give it a new image, it should be able to tell if it's a cat or dog.

Cost: $O(p)$

We call this approach, supervised wave-function optimization (**SWO**)

All one needs now is a labelled wave-function to match….



Quickly match the correct wave-function.

Unfortunately, getting the exact wave-function is difficult….

Instead…our goal will be to get a better wave-function.

Better Wave-functions:

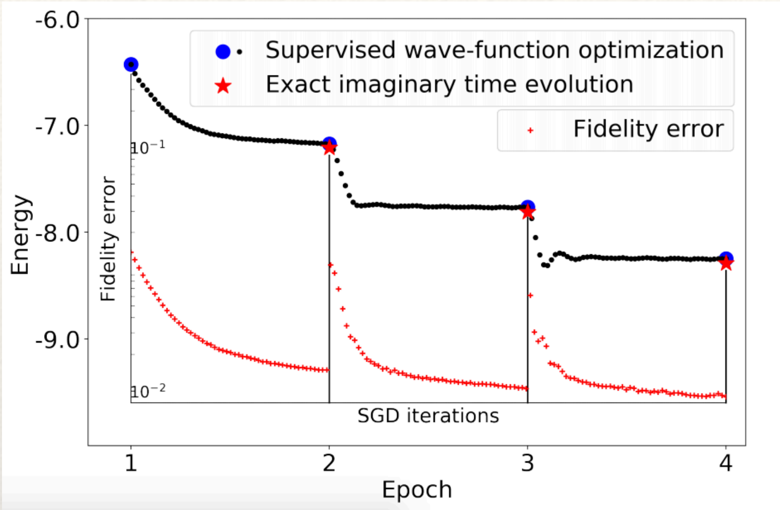
Imaginary Time Evolution $(1 - \tau H)|\Psi_{NN}\rangle$

IT-SWO

Current best wave-function

This gives us an $O(p)$ approach which avoids lots of other problems.

(no ill-defined inverse; no over-parameterization; no under sampling)

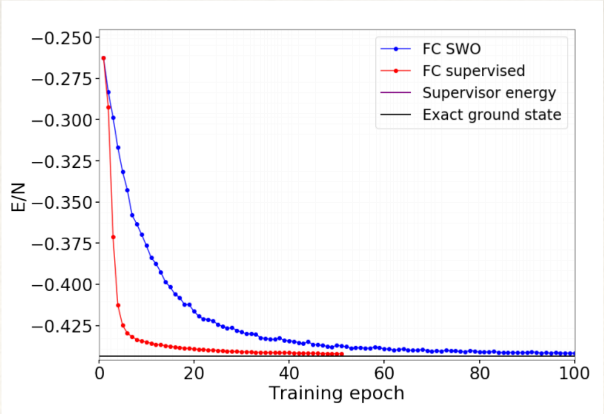


Lanczos Steps $|\Psi_{NN}\rangle + \alpha H|\Psi_{NN}\rangle + \beta H^2|\Psi_{NN}\rangle$

Lanczos-SWO

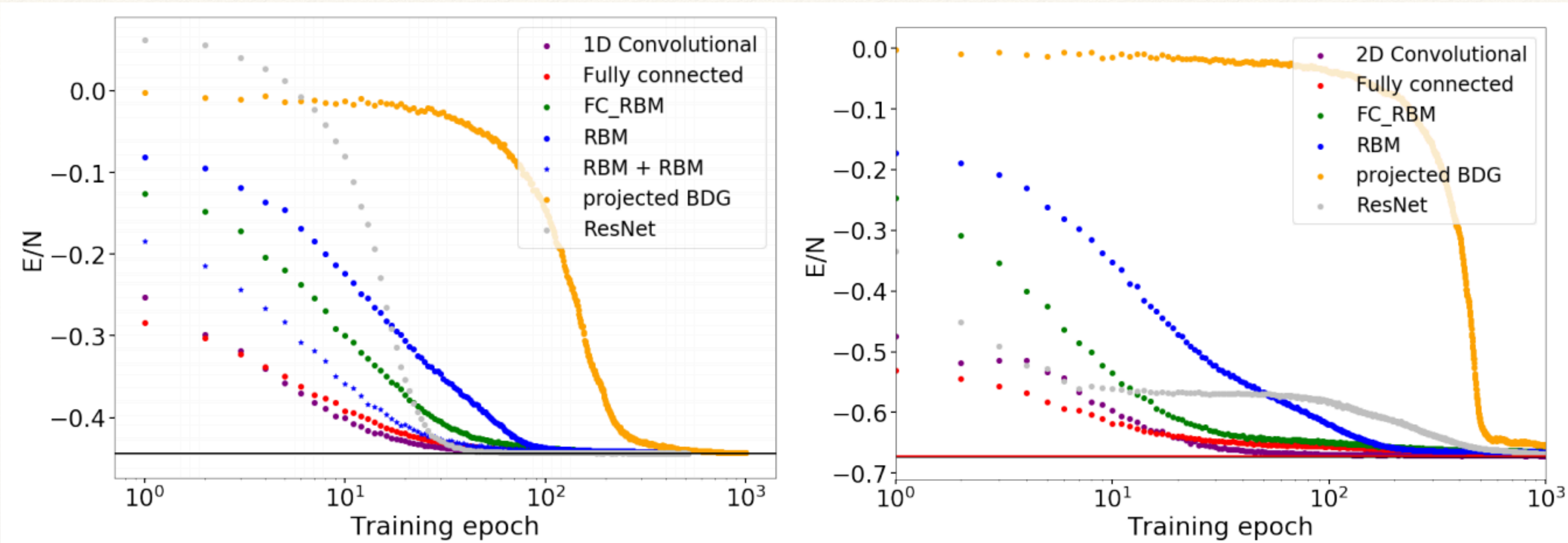
Previously Optimized (but stuck) states… MPS, simpler NN, etc.

Better tunneling, ….



Matching-SWO

Putting it all together…

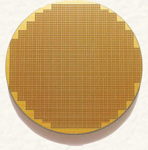


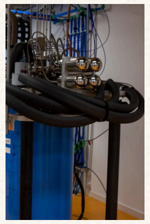
WF	Hidden Layers	Hidden Units	Params 1d - 2d
FCNN	2	80	9800 - 11744
RBM	0	80	3320 - 5264
FC-RBM	2	80	16280 - 18224
Conv1D	5	16 filters (size 5)	5280 -
Conv2D	5	16 filters (5 by 5)	- 26080
ResNet	1 + 2(2)	16 filters (size 5)	5280 - 25760
Multi-RBM	2	16 filters (5 by 5)	6640 - 10528
Multi-FCNN	2	80	19600 - 23488
P-BDG			1600 - 4096
MPS			4800 -

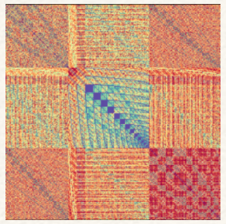
Can play with this yourself:

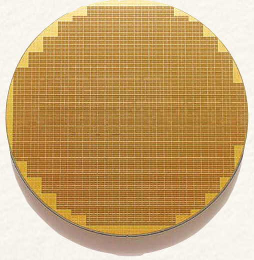
github.com/ClarkResearchGroup/cgs-vmc/

The next steps (for the variational approach)….









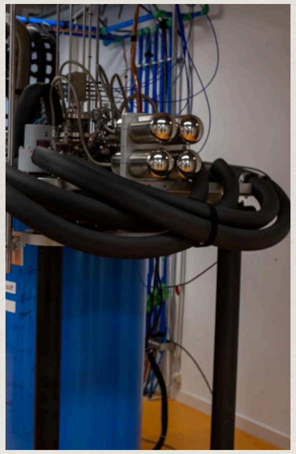
Fulfill the promise of the silicon age….

Computational graphs are a great structure, but how do you pick the correct graph?

Current Solution: Graph optimization by graduate student.

Future Solution: Graph optimization by computer.

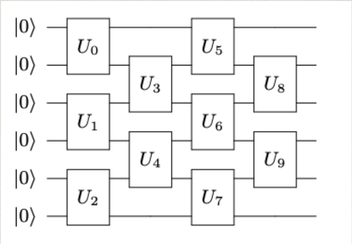
Matching SWO is a key piece of this puzzle. Once you have a new graph you want to quickly get its parameters to match the old graph. You can't afford to do this through Hamiltonian optimization.



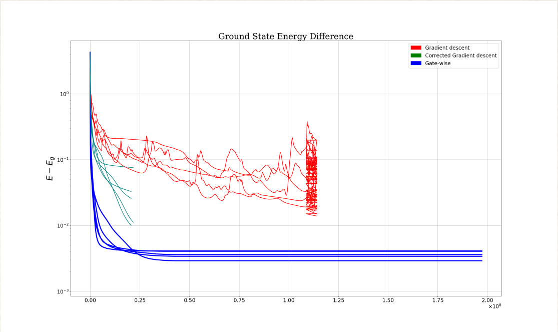
The quantum age…

Variational Quantum Eigensolvers

A standard quantum circuit is the moral equivalent of computational graph states.



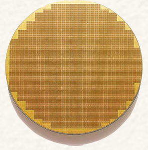
But we are currently in ancient history as far as optimization is concerned…

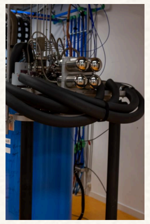


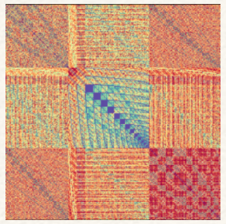
DMRG (or linear method) for quantum circuits….

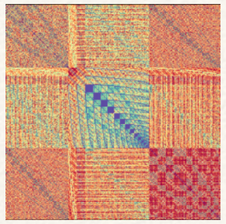
Cherktov, BKC, …..
Cherktov, BKC, ….

The next steps (for the variational approach)….









Inverting the variational approach…

Typical approach: $H \rightarrow |\Psi\rangle$

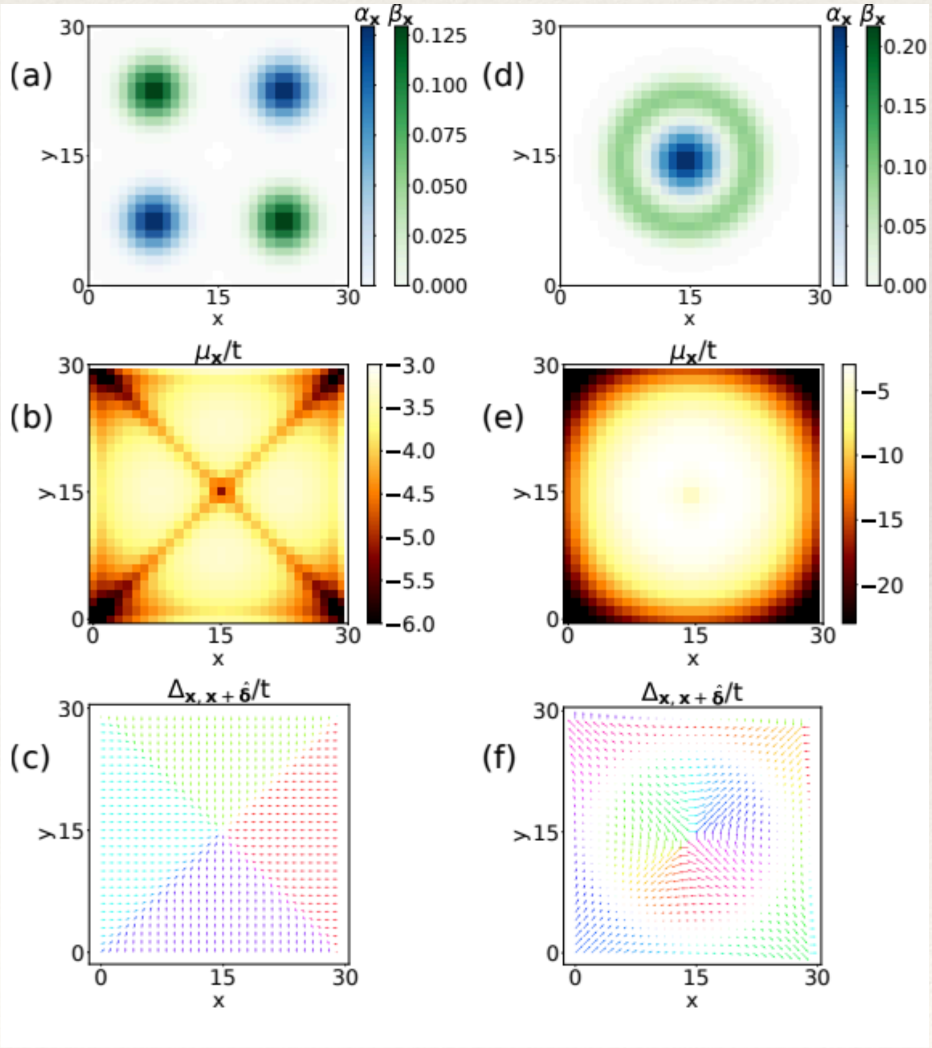
New approach: $|\Psi\rangle \rightarrow H$

symmetry $O \rightarrow H$

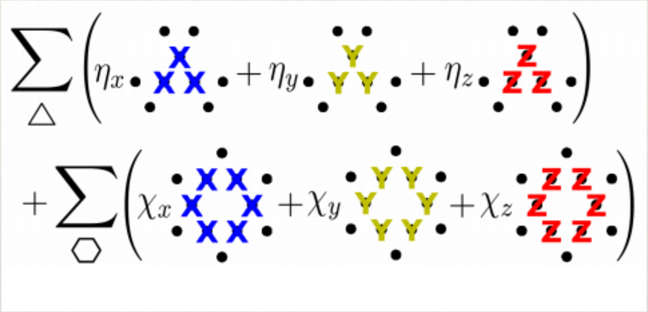
Zero mode

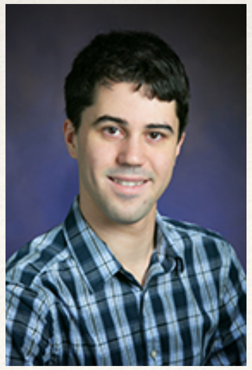
Chemical Potential

Superconducting Order



Zoo of new spin liquids….





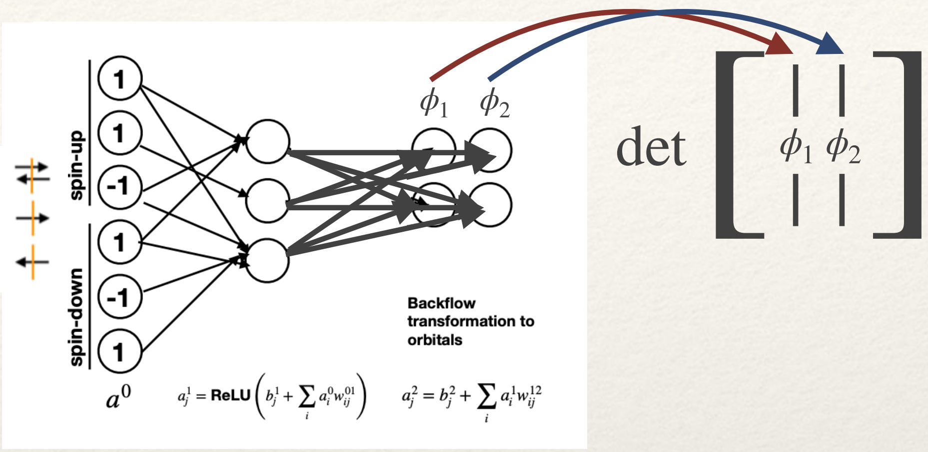
Summary (the main pieces)…

There's a lot of hype in variational wave-functions + AI. Despite what you might have heard, they don't solve all problems.

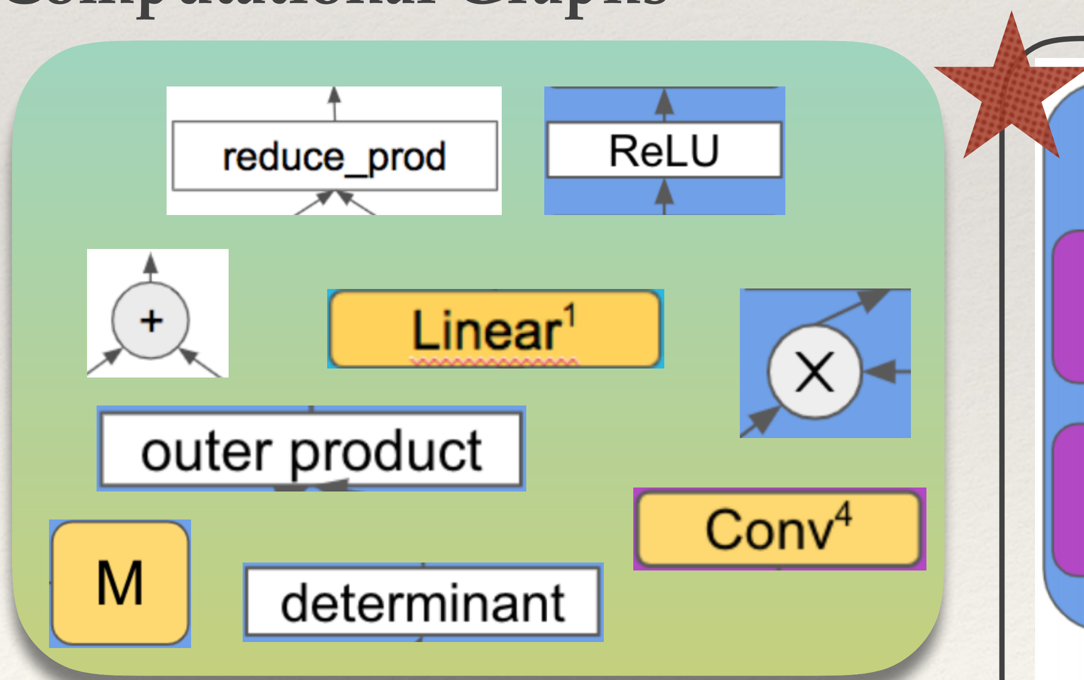
That said, smart choices are making real progress, optimization is improving, etc.

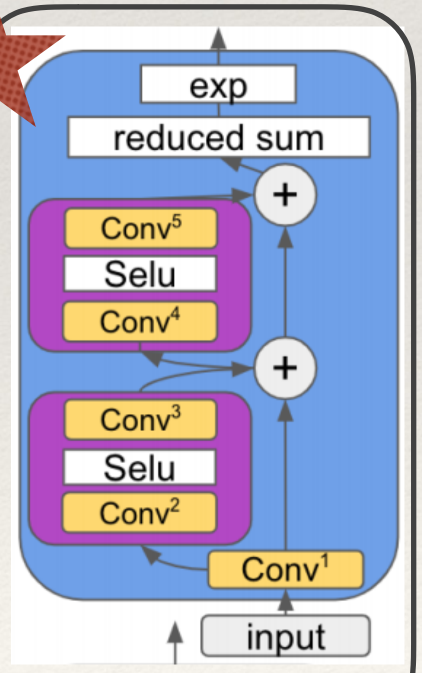
Neural Net Backflow

Fermions with Neural Net @ DMRG accuracy



Computational Graphs





ResNet

Box of operations…

SWO