

Title: TensorNetwork: accelerating tensor network computations and improving the coding experience

Speakers: Ashley Milsted

Collection: Many-Body States and Dynamics Workshop II

Date: June 13, 2019 - 10:00 AM

URL: <http://pirsa.org/19060029>

Abstract: Tensor networks are powerful computational tools, widely used in condensed matter physics, and increasingly in high-energy physics, with promising applications to machine learning problems. Developed in collaboration with Google and X, we present TensorNetwork: a new software package that makes it easier to code tensor network algorithms and, by using a framework like TensorFlow as a backend, to accelerate computations using specialized hardware (GPUs, TPUs) and integrate tensor networks into machine-learning projects.

TensorNetwork

accelerating tensor network computations and
improving the coding experience

Many-Body States and Dynamics Workshop II
13th June 2019

Perimeter Institute

Ashley Milsted, Martin Ganahl, Yijian Zou, Guifre Vidal

Google and X

Chase Roberts, Stefan Leichenauer, Adam Zalcman, Bruce Fontaine, Jack Hidary

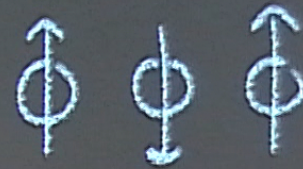


TENSOR
NETWORKS
INITIATIVE



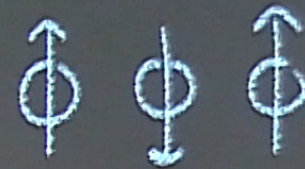


Quantum State



2

Quantum State

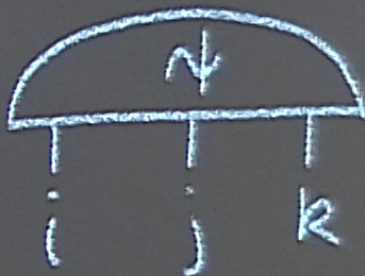


$$|\Psi\rangle = \sum_{ijk} \psi_{ijk} |ijk\rangle$$

2

Quantum State Tensor

$$|\psi\rangle = \sum_{ijk} \psi_{ijk} |ijk\rangle$$

$$\psi_{ijk} = \text{diagram}$$


"Matrix Product State"

$$\psi_{ijk} = \begin{array}{c} \text{---}\psi\text{---} \\ | \quad | \quad | \\ i \quad j \quad k \end{array}$$

4

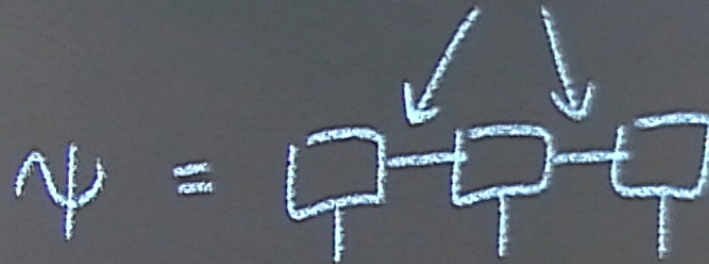
"Matrix Product State"

$$\psi_{ijk} = \begin{array}{c} \text{---} \psi \text{---} \\ | \\ i \quad j \quad k \end{array}$$

$$\psi_{ijk} = \sum_{\alpha/\beta} \begin{array}{c} \boxed{}^{\alpha} \text{---} \boxed{}^{\beta} \text{---} \boxed{} \\ | \quad \quad | \quad \quad | \\ i \quad \quad j \quad \quad k \end{array}$$

"Matrix Product State"

"bonds," dimension χ

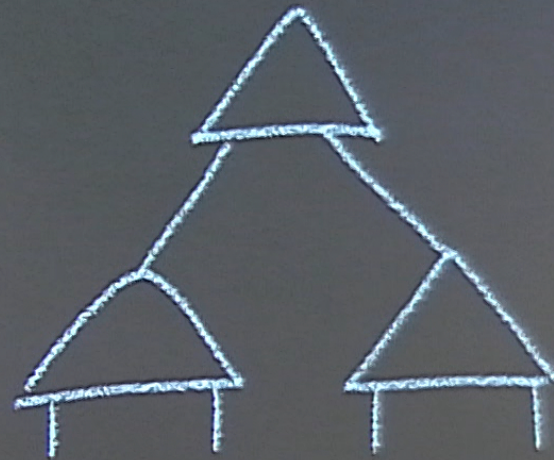


5

Tree Tensor Network State

ψ

approx.
state

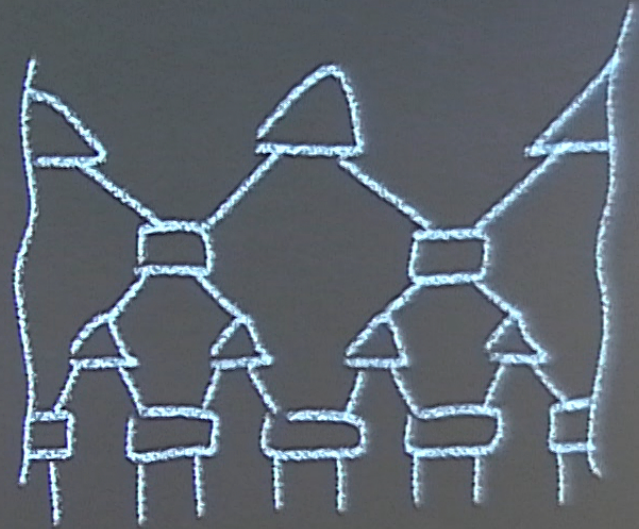


6

NERA

2

=



7

Tensor Network States

typical properties

efficient approximation of ground states
(of Hamiltonians with local interactions)

efficient computation of correlators,
entanglement entropies, etc.

8

Tensor Network States

Matrix Product States (DMRG)

- (Quasi) 1-dimensional systems
- Local interactions
- Relatively simple to code!
- $O(\chi^3)$

9

Tensor Network States

Matrix Product States (DMRG)

- (Quasi) 1-dimensional systems
- Local interactions
- Relatively simple to code!
- $O(\chi^3)$

Tree Tensor Networks

- Finite 1D and 2D systems
- Local interactions
- Harder to code...
- $O(\chi^4)$

MERA

- 1D systems (for now...)
- Local interactions
- Precisely captures many properties of critical systems
- Harder to code...
- $O(\chi^7)$, $O(\chi^9)$, ... depending on variant

9

Tensor Network States

Matrix Product States (DMRG)

- (Quasi) 1-dimensional systems
- Local interactions
- Relatively simple to code!
- $O(\chi^3)$

MERA

- 1D systems (for now...)
- Local interactions
- Precisely captures many properties of critical systems
- Harder to code...
- $O(\chi^7), O(\chi^9), \dots$ depending on variant

Tree Tensor Networks

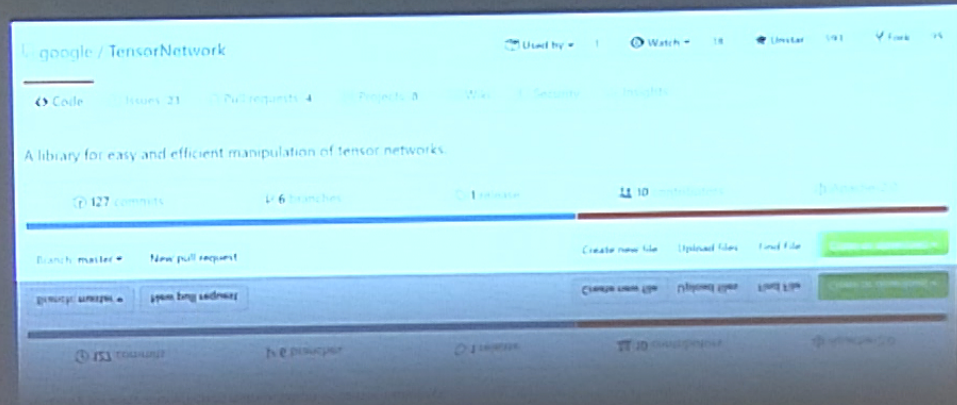
- Finite 1D and 2D systems
- Local interactions
- Harder to code...
- $O(\chi^4)$

Basically anything beyond MPS

- Harder to code...
- Computationally more demanding...
- Might be the one you really want for your favorite problem!

9

TensorNetwork



arXiv: 1905.01330

<https://github.com/google/TensorNetwork>

10

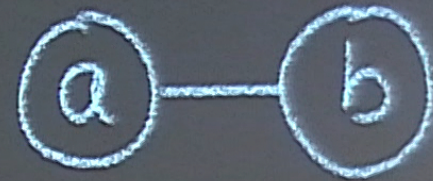
Example: explicit syntax

```
import numpy as np
import tensorflow as tf
tf.enable_v2_behavior()
import tensornetwork

my_vector = np.ones((10,))

# Create the network
net = tensornetwork.TensorNetwork()
# Add the nodes
a = net.add_node(my_vector)
b = net.add_node(my_vector)
# Connect the nodes
edge = net.connect(a[0], b[0])

final_node = net.contract(edge)
print(final_node.tensor.numpy()) # Prints "10".
```



11

Example: explicit syntax

```
import numpy as np
import tensorflow as tf
tf.enable_v2_behavior()
import tensornetwork
```

```
my_vector = np.ones((10,))
```

```
# Create the network
```

```
net = tensornetwork.TensorNetwork()
```

```
# Add the nodes
```

```
a = net.add_node(my_vector)
```

```
b = net.add_node(my_vector)
```

```
# Connect the nodes
```

```
edge = net.connect(a[0], b[0])
```

```
l_edges = net.contract(edge)
print(l_edges[0].tensor.numpy()) # Prints "10".
```



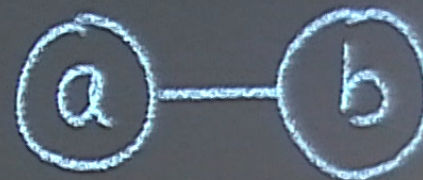
11

Example: Compact syntax (NCON)

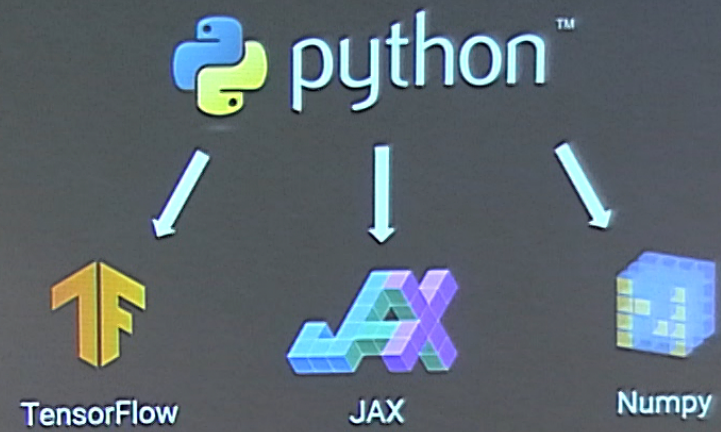
```
import numpy as np
import tensorflow as tf
tf.enable_v2_behavior()
import tensornetwork
```

```
my_vector = np.ones((10,))
```

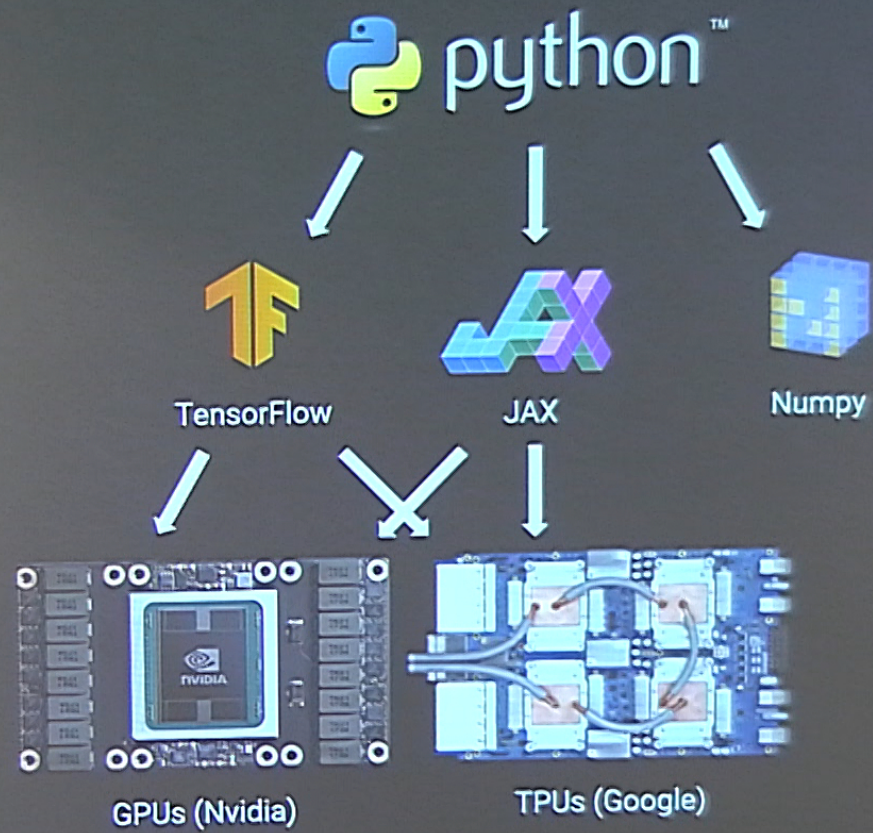
```
result = tensornetwork.ncon([my_vector, my_vector], [(0,), (0,)])
print(result) # Prints "10".
```



12

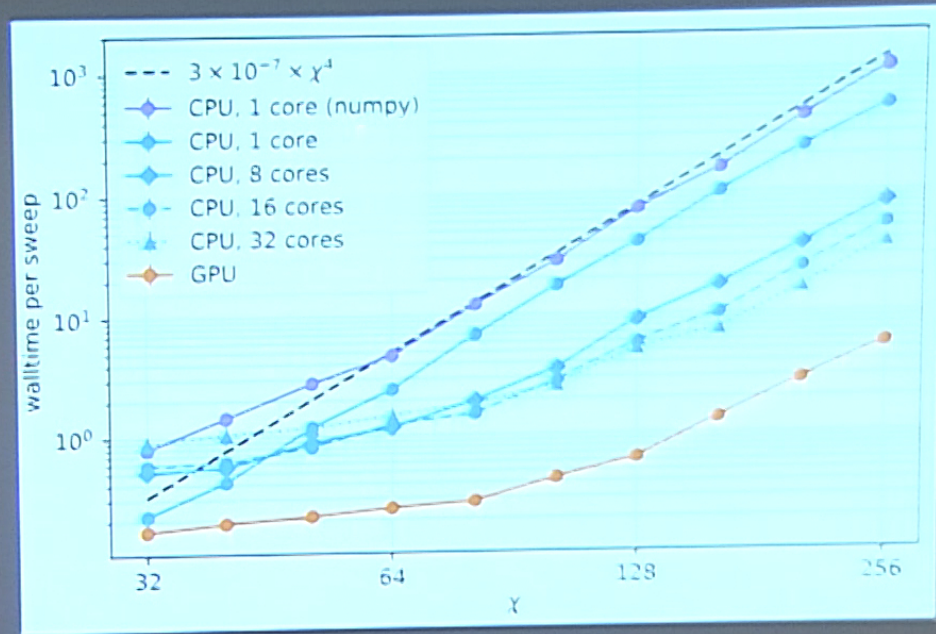


13



13

Tree Tensor Network Acceleration



arXiv:1905.01331

14

example code included

Trotter-evolving full wavefunctions

Matrix Product States (finite and infinite)

Tree Tensor Networks

MERA

...

15

TensorNetwork

lots of features to come
(more automatic parallelism, optimal contraction orders, ...)

exploring TPU acceleration

algorithms using automatic differentiation

applications to Machine Learning

arXiv: 1905.01330

<https://github.com/google/TensorNetwork>

16