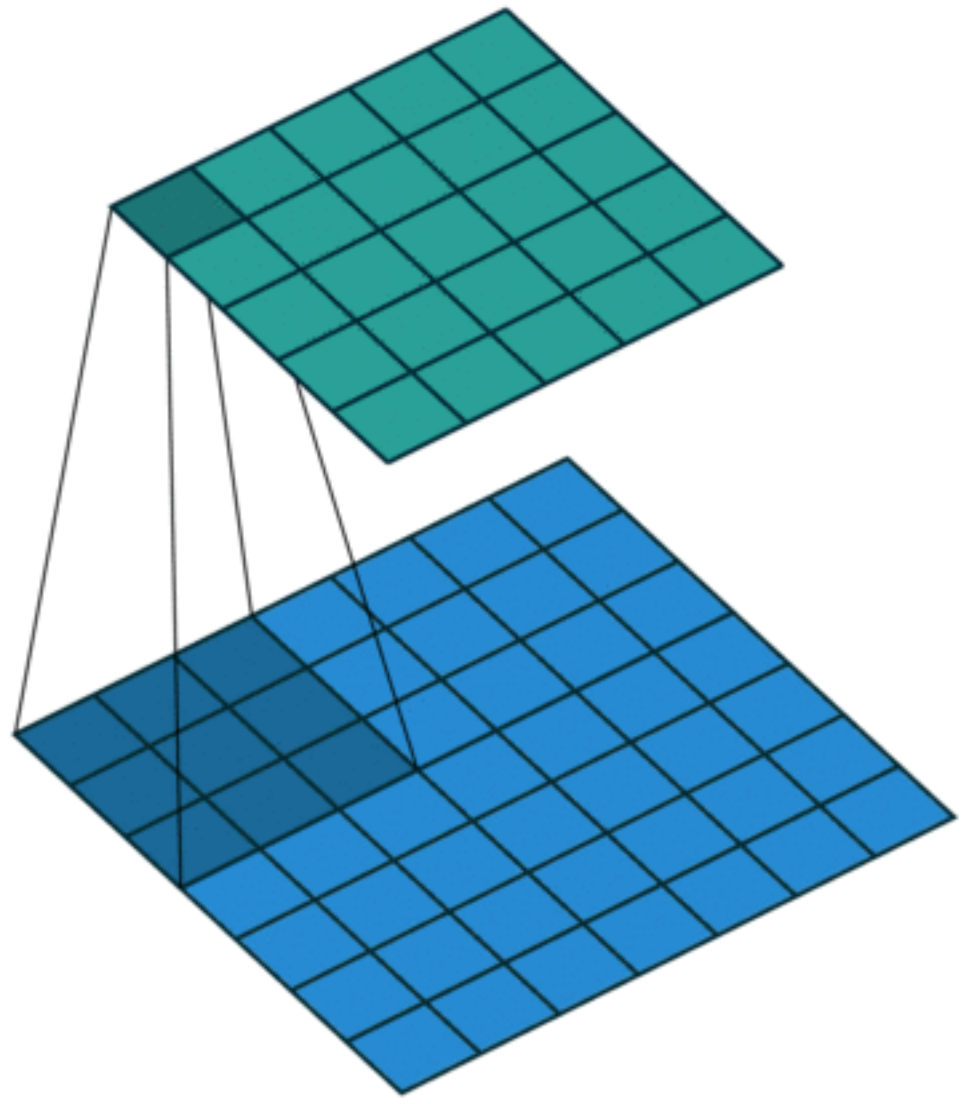
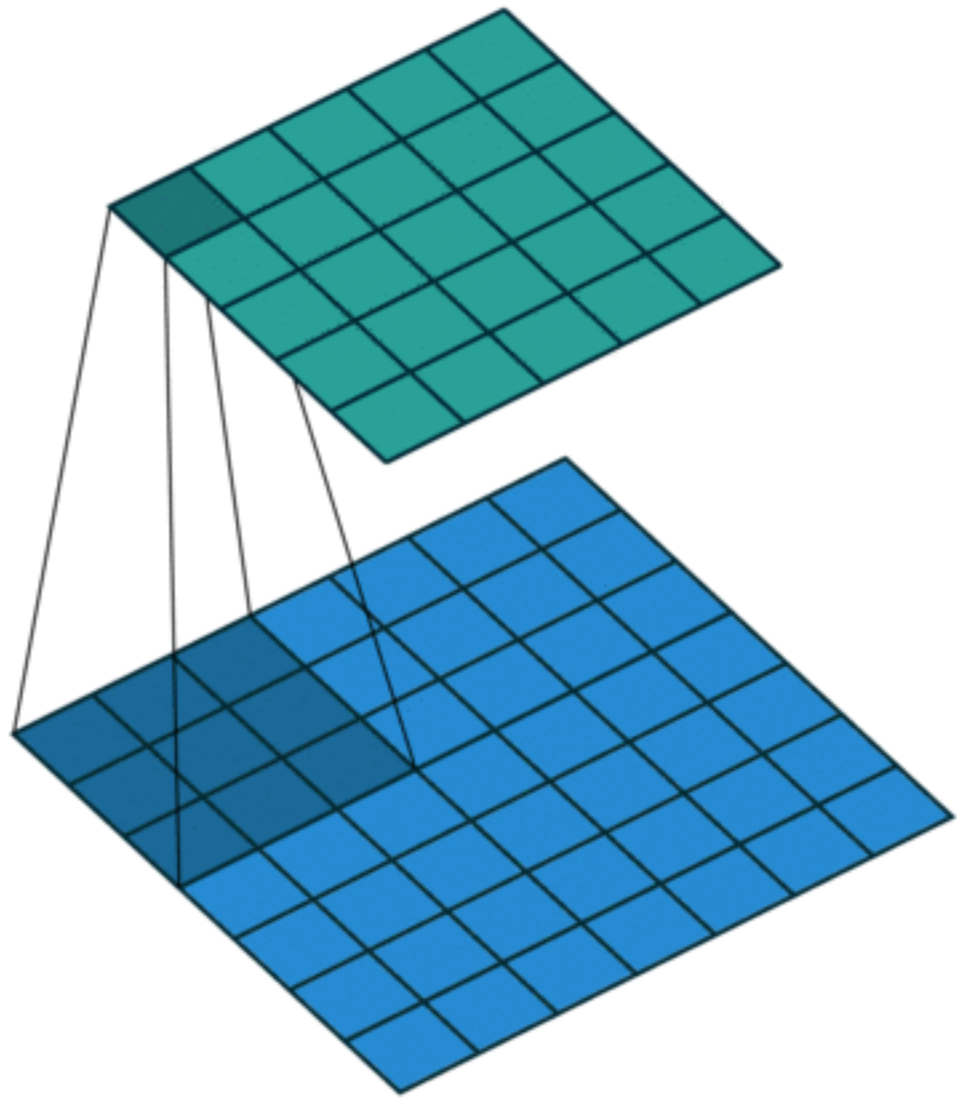
Physics-Informed Inductive Biases in Deep Learning



Miles Cranmer (Princeton)
Shirley Ho (Flatiron, NYU, CMU, Princeton)

Physics-Informed Inductive Biases in Deep Learning



Miles Cranmer (Princeton)
Shirley Ho (Flatiron, NYU, CMU, Princeton)

Main Ideas

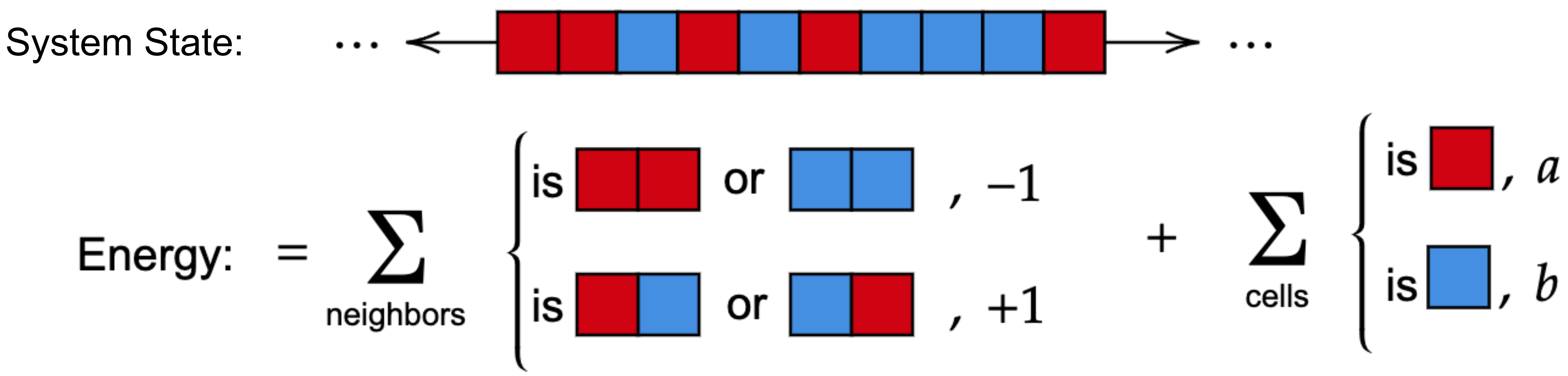
- Physics has informed many inductive biases in deep learning, both **explicitly and implicitly**
    - Success of these often due to fact that deep learning seeks models of the physical world; using physics as a prior can directly or indirectly benefit these models.
- Formalizing these in a physics language often leads to **new insights**

First, some history

- An early physics-motivated inductive bias: **Hopfield networks**
- Based on **Ising Model**

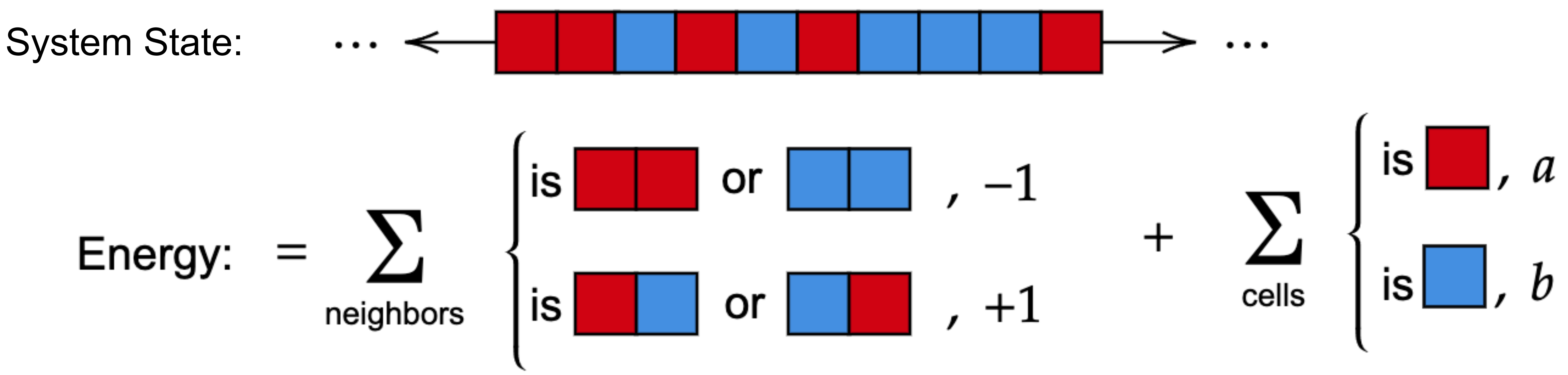
First, some history

- An early physics-motivated inductive bias: **Hopfield networks**
- Based on **Ising Model**



First, some history

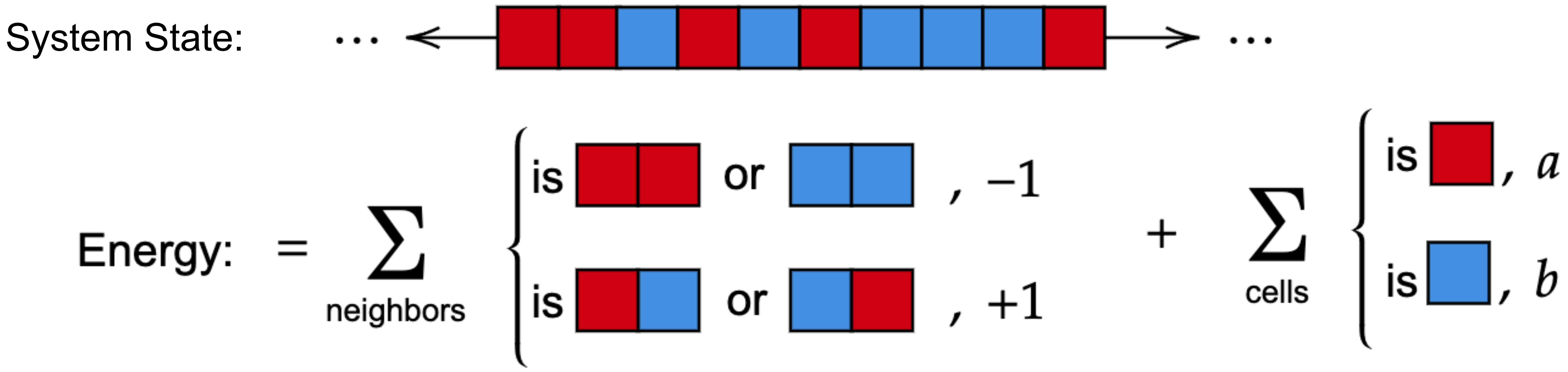
- An early physics-motivated inductive bias: **Hopfield networks**
- Based on **Ising Model**



$p(\textbf{State}) \propto \exp(-\textbf{Energy}/\textbf{Temperature})$

First, some history

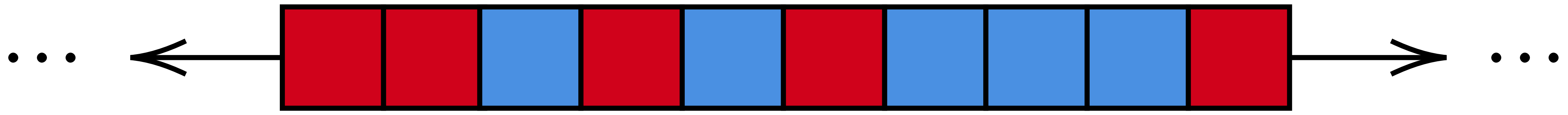
- An early physics-motivated inductive bias: **Hopfield networks**
- Based on **Ising Model**



$$p(\textbf{State}) \propto \exp(-\textbf{Energy}/\textbf{Temperature})$$

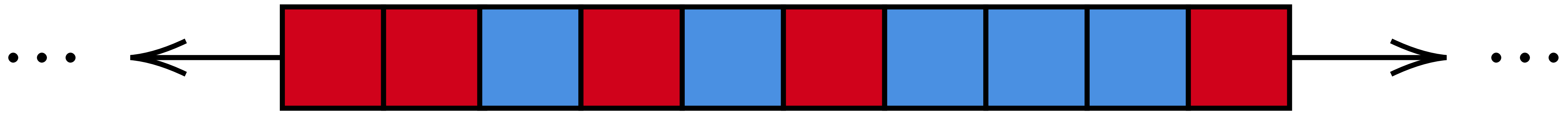
In higher dimension - additional neighbors!

Ising Model

System State: 

Ising Model

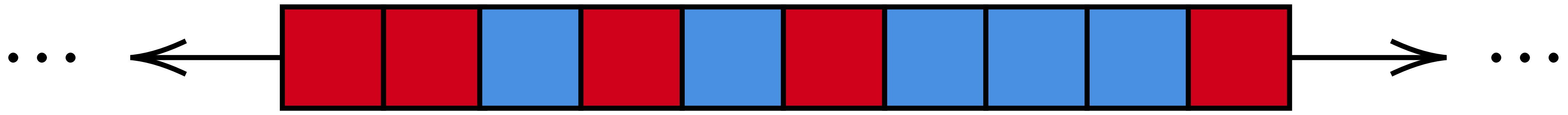
System State:



- To simulate (Monte Carlo)

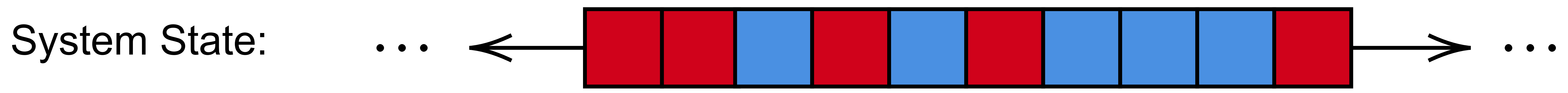
Ising Model

System State:



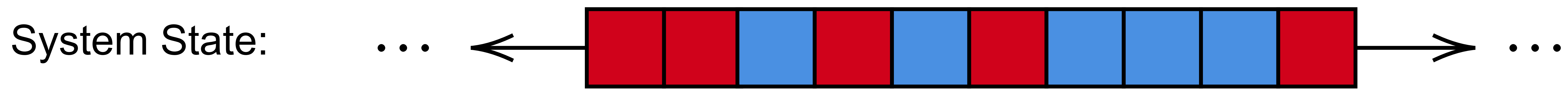
- To simulate (Monte Carlo)
  - Pick random grid cell.

Ising Model



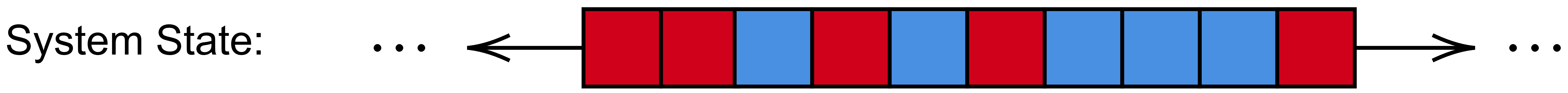
- To simulate (Monte Carlo)
  - Pick random grid cell.
  - Swap color.

Ising Model



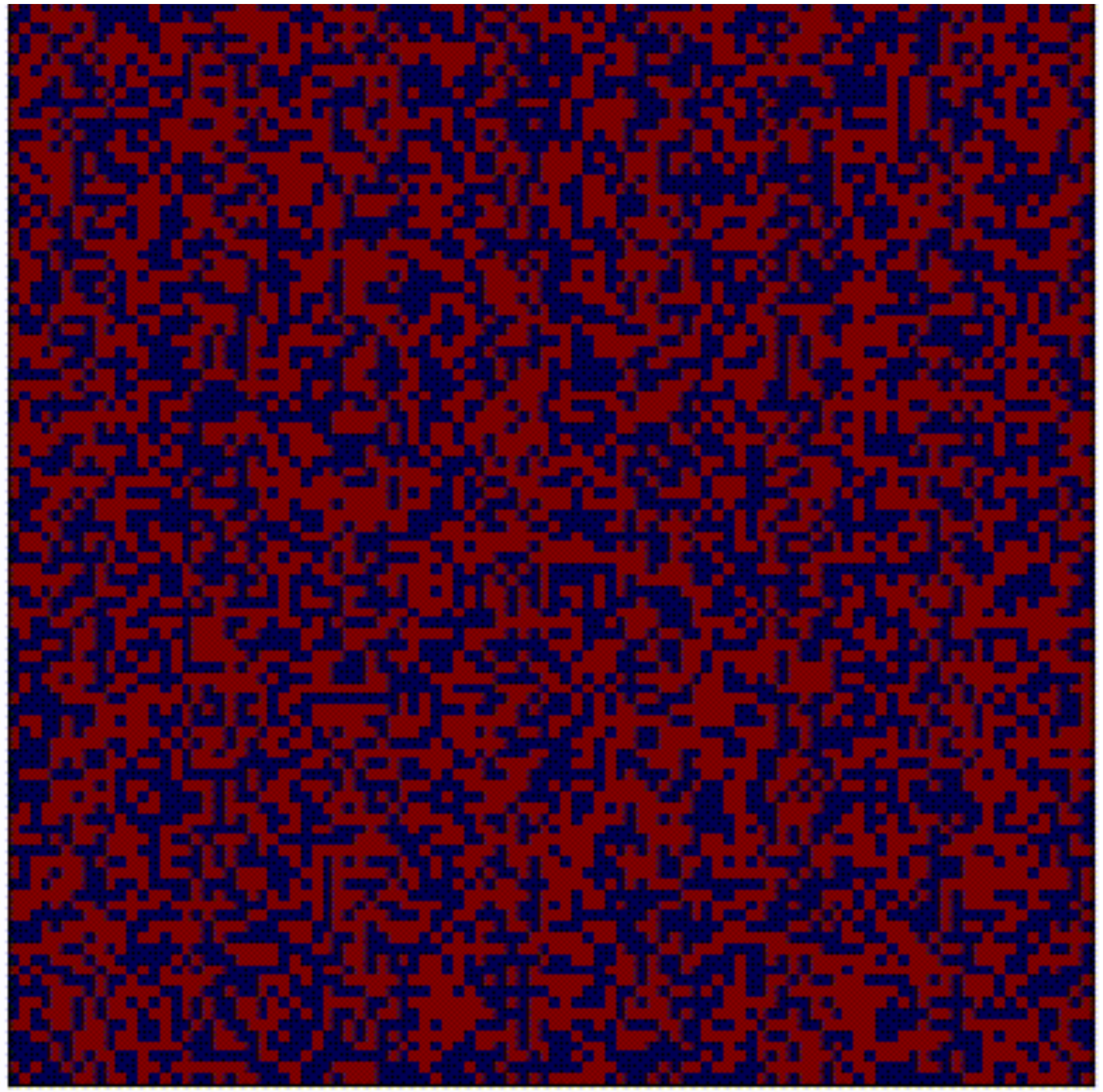
- To simulate (Monte Carlo)
  - Pick random grid cell.
  - Swap color.
    - If energy decreases ⇒ keep change

Ising Model



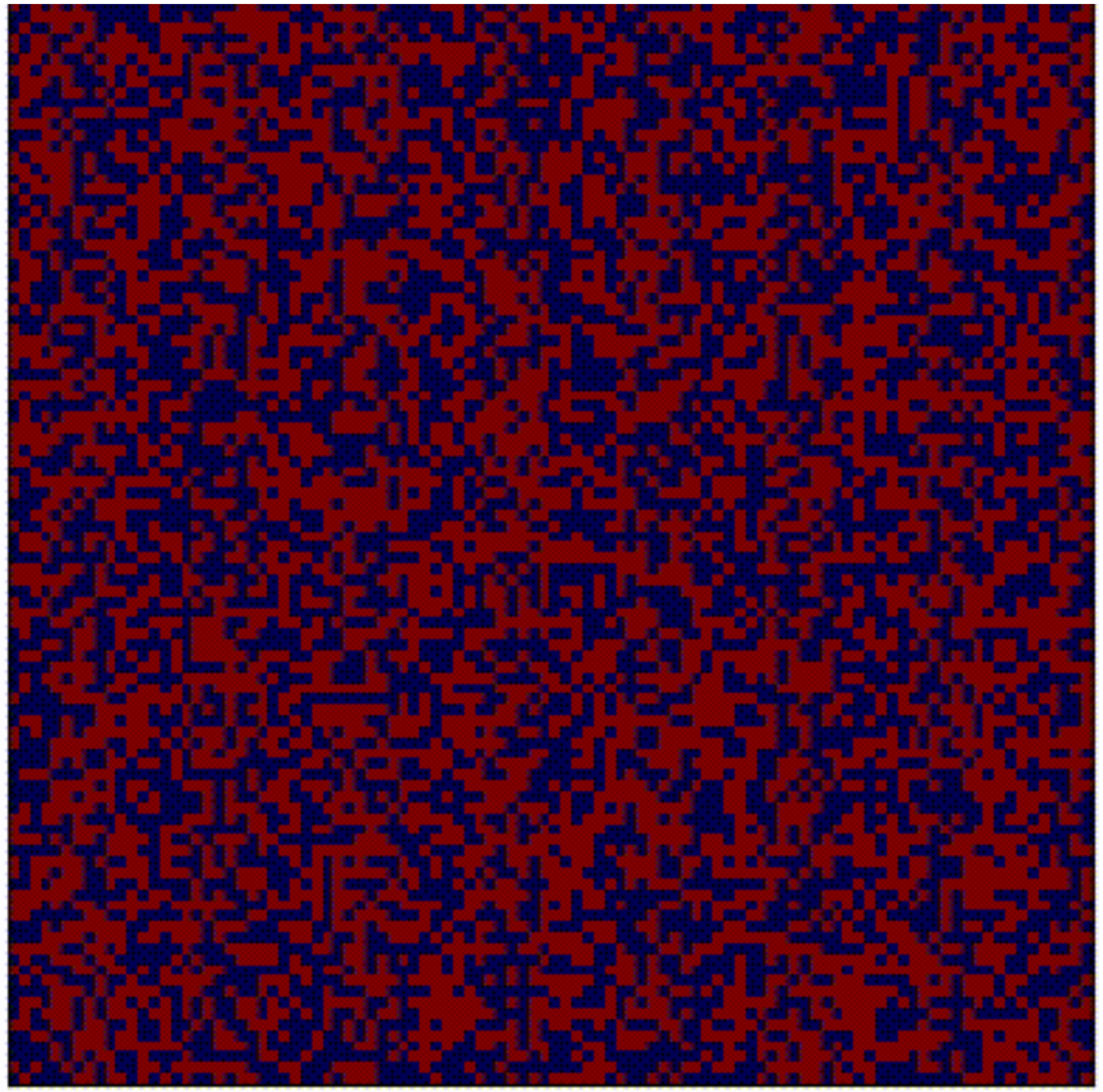
- To simulate (Monte Carlo)
  - Pick random grid cell.
  - Swap color.
    - If energy decreases ⇒ keep change
    - If energy increases ⇒ keep change with $p = \exp(-(E_{\text{new}} - E_{\text{old}})/\text{Temperature})$

2D (4 neighbors for every cell)



(Alex Pettitt)

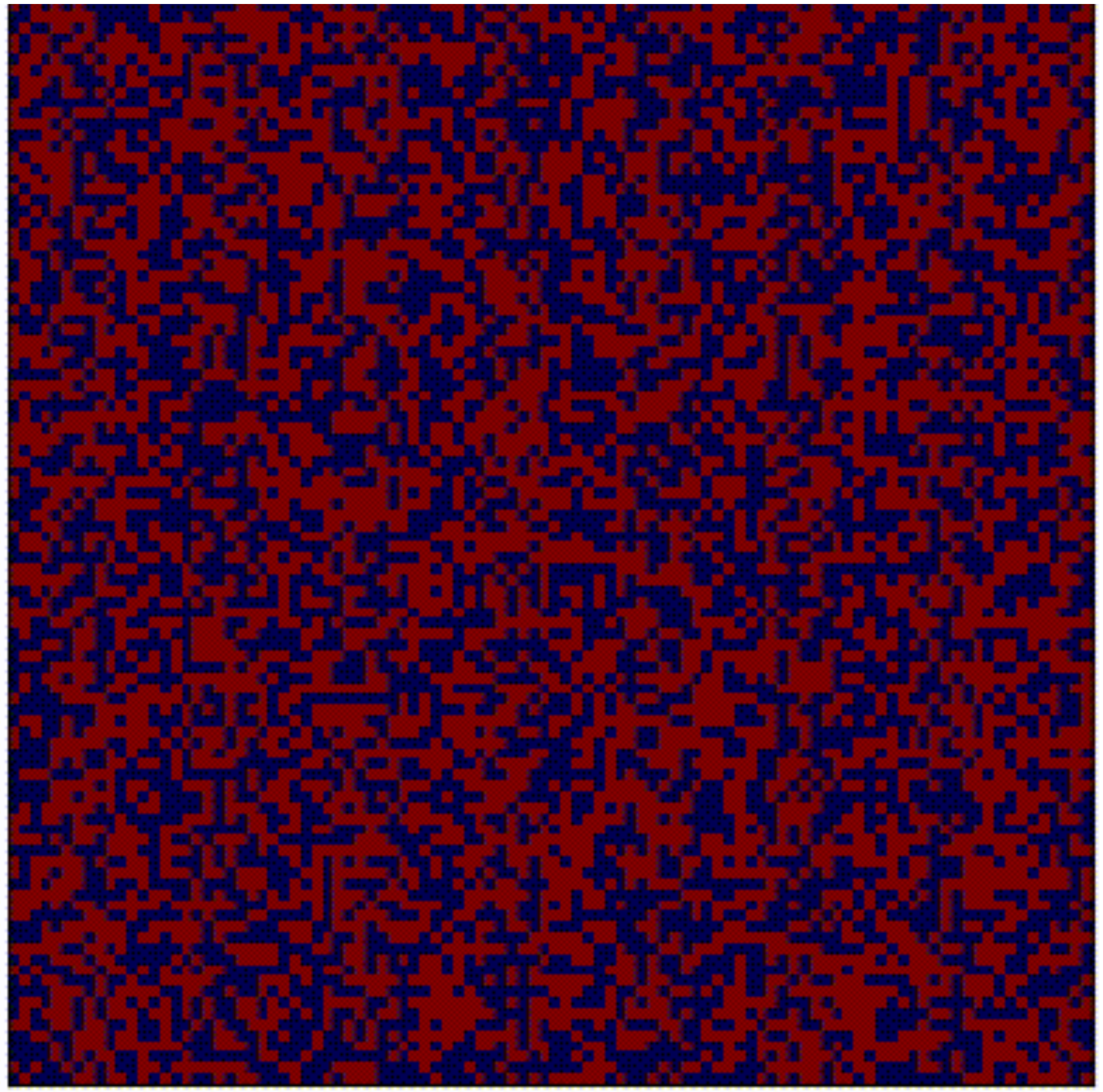
2D (4 neighbors for every cell)



(Alex Pettitt)

- Simple system; but can be used to model many phenomena:

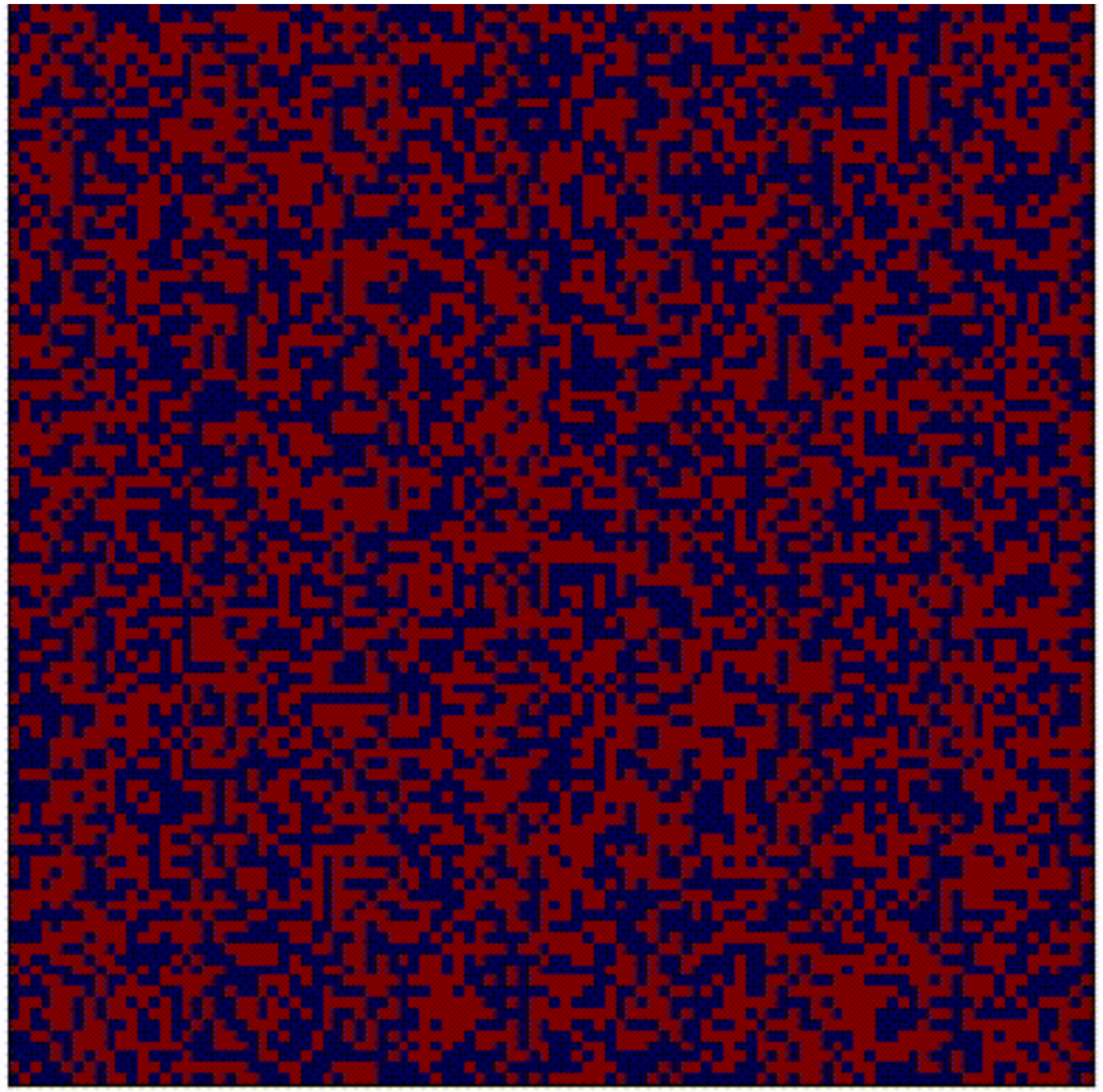
2D (4 neighbors for every cell)



(Alex Pettitt)

- Simple system; but can be used to model many phenomena:
  - Ferromagnets, chemical equilibrium, crystals, ice, etc.

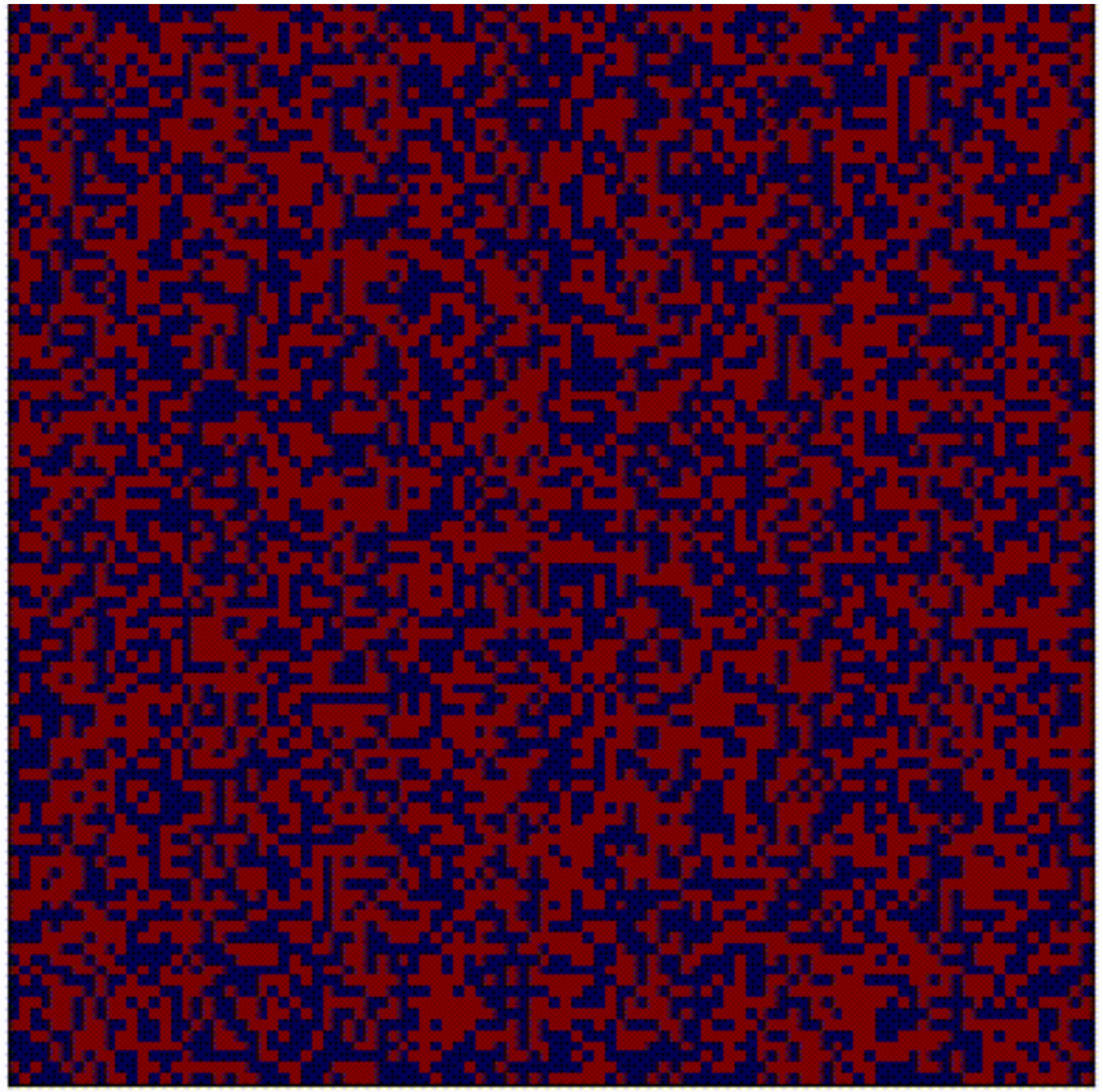
2D (4 neighbors for every cell)



(Alex Pettitt)

- Simple system; but can be used to model many phenomena:
  - Ferromagnets, chemical equilibrium, crystals, ice, etc.
  - Non-physics: social networks, human memory (Hopfield network!), etc.

2D (4 neighbors for every cell)



(Alex Pettitt)

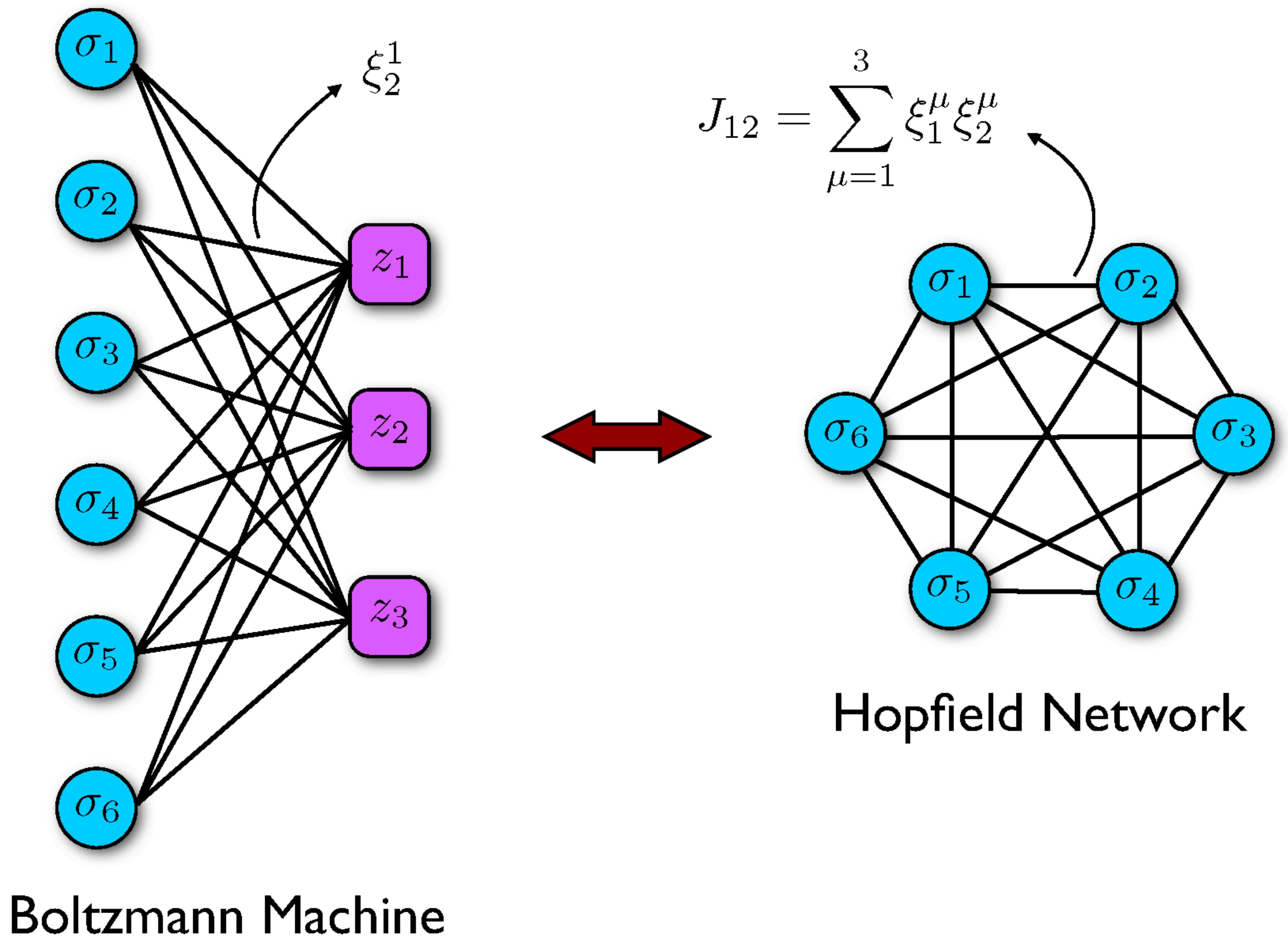
Hopfield Networks & Boltzmann Machines

“Neighbor” $\Rightarrow$”Connection”

One can think about updating neurons as if they were cells in an Ising Model!

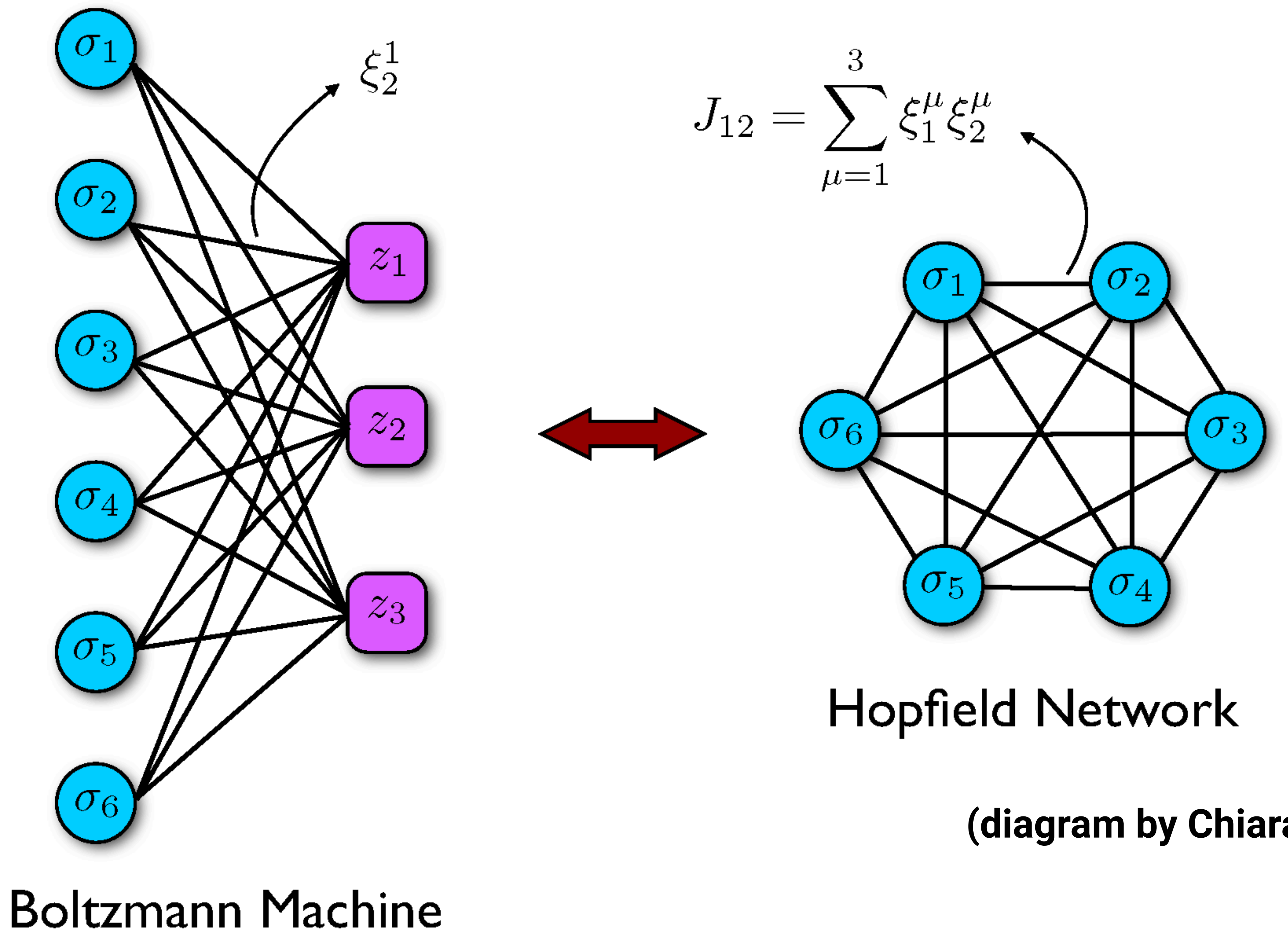
Hopfield Networks & Boltzmann Machines

“Neighbor” ⇒”Connection”
One can think about updating neurons as if they were cells in an Ising Model!



Hopfield Networks & Boltzmann Machines

"Neighbor" ⇒"Connection"
One can think about updating neurons as if they were cells in an Ising Model!

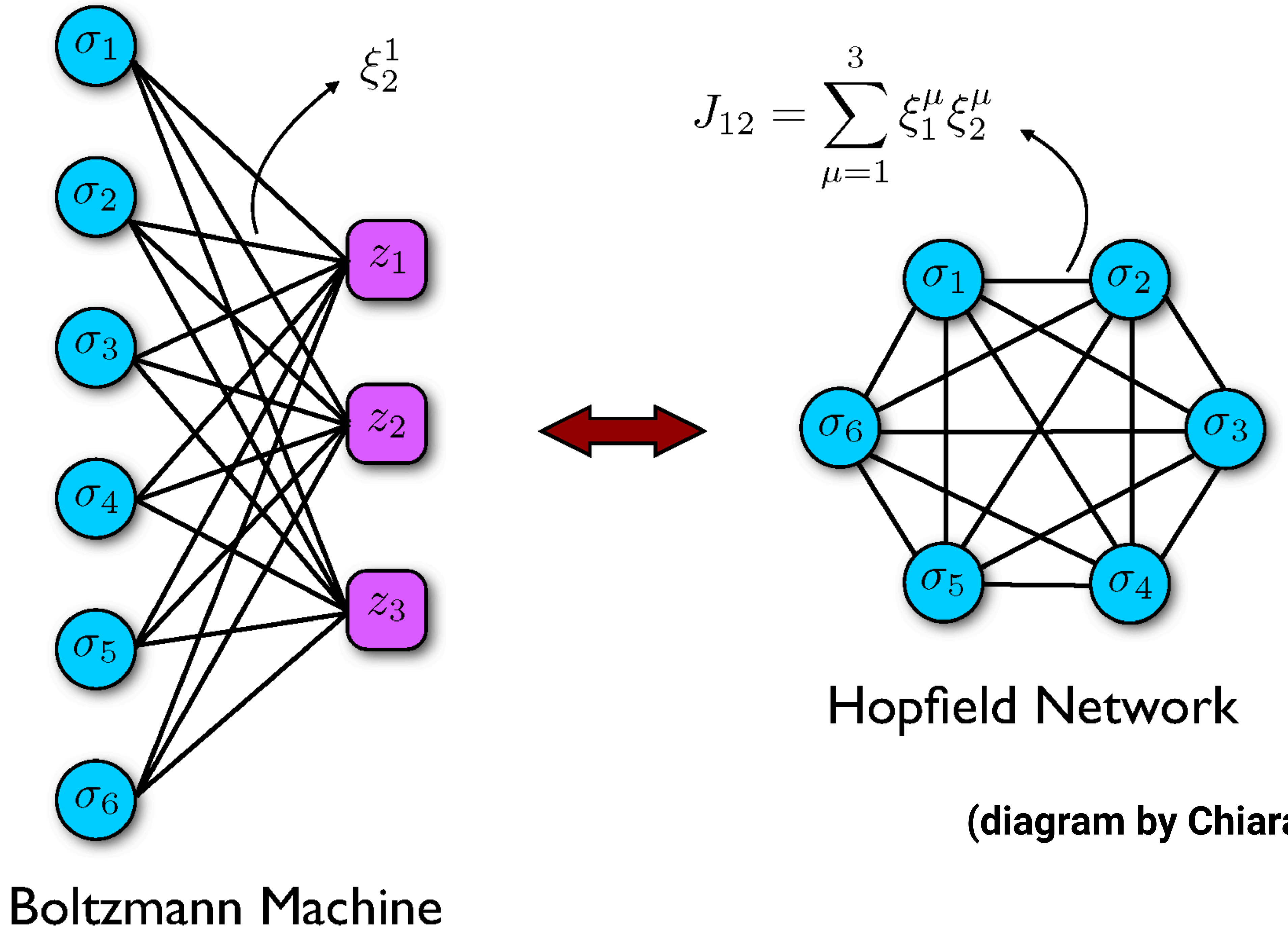


(diagram by Chiara Marullo)

Hopfield Networks & Boltzmann Machines

“Neighbor” ⇒”Connection”
One can think about updating neurons as if they were cells in an Ising Model!

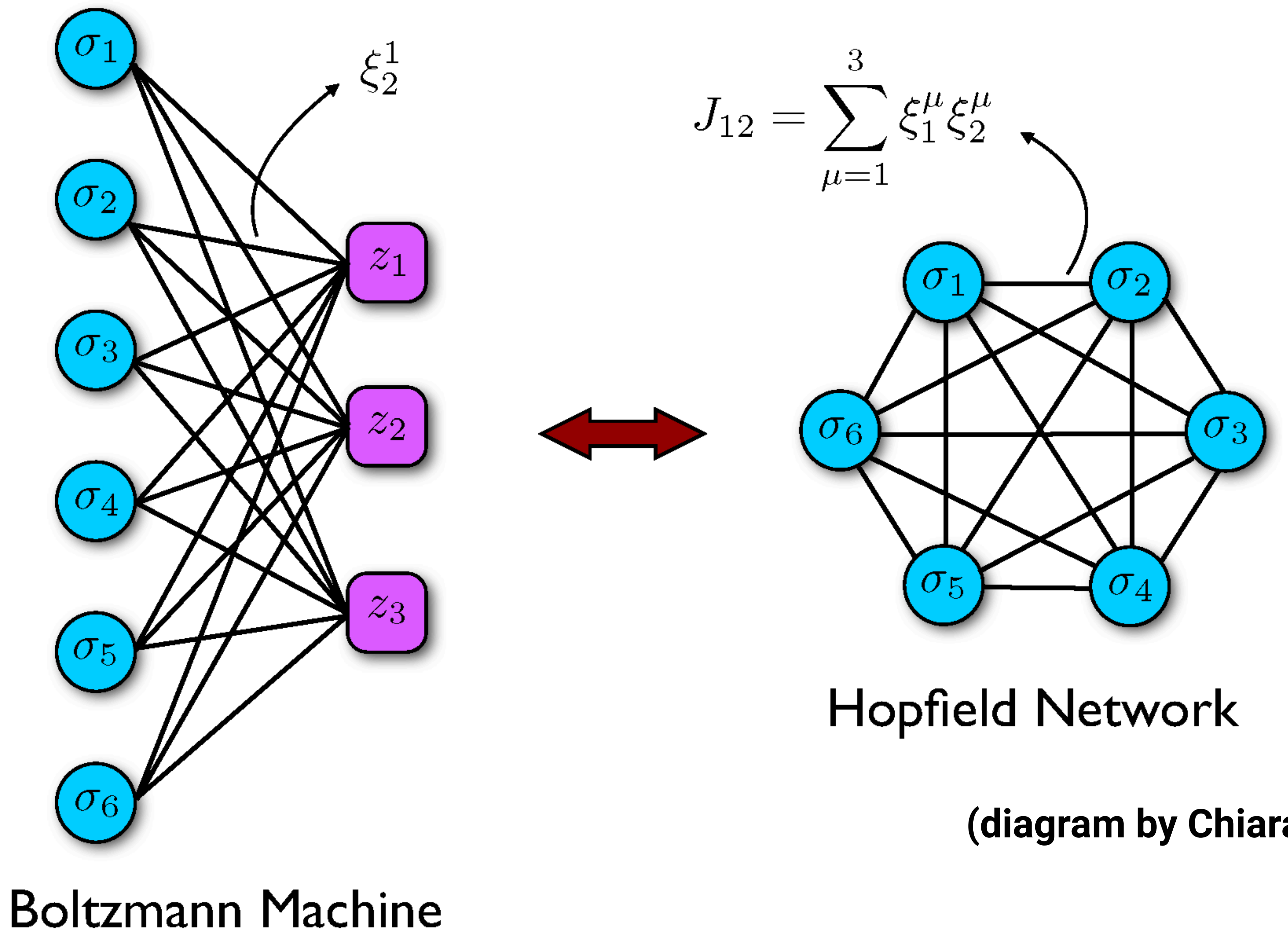
- Applying Ising Model to Neural Networks



(diagram by Chiara Marullo)

Hopfield Networks & Boltzmann Machines

“Neighbor” ⇒”Connection”
One can think about updating neurons as if they were cells in an Ising Model!

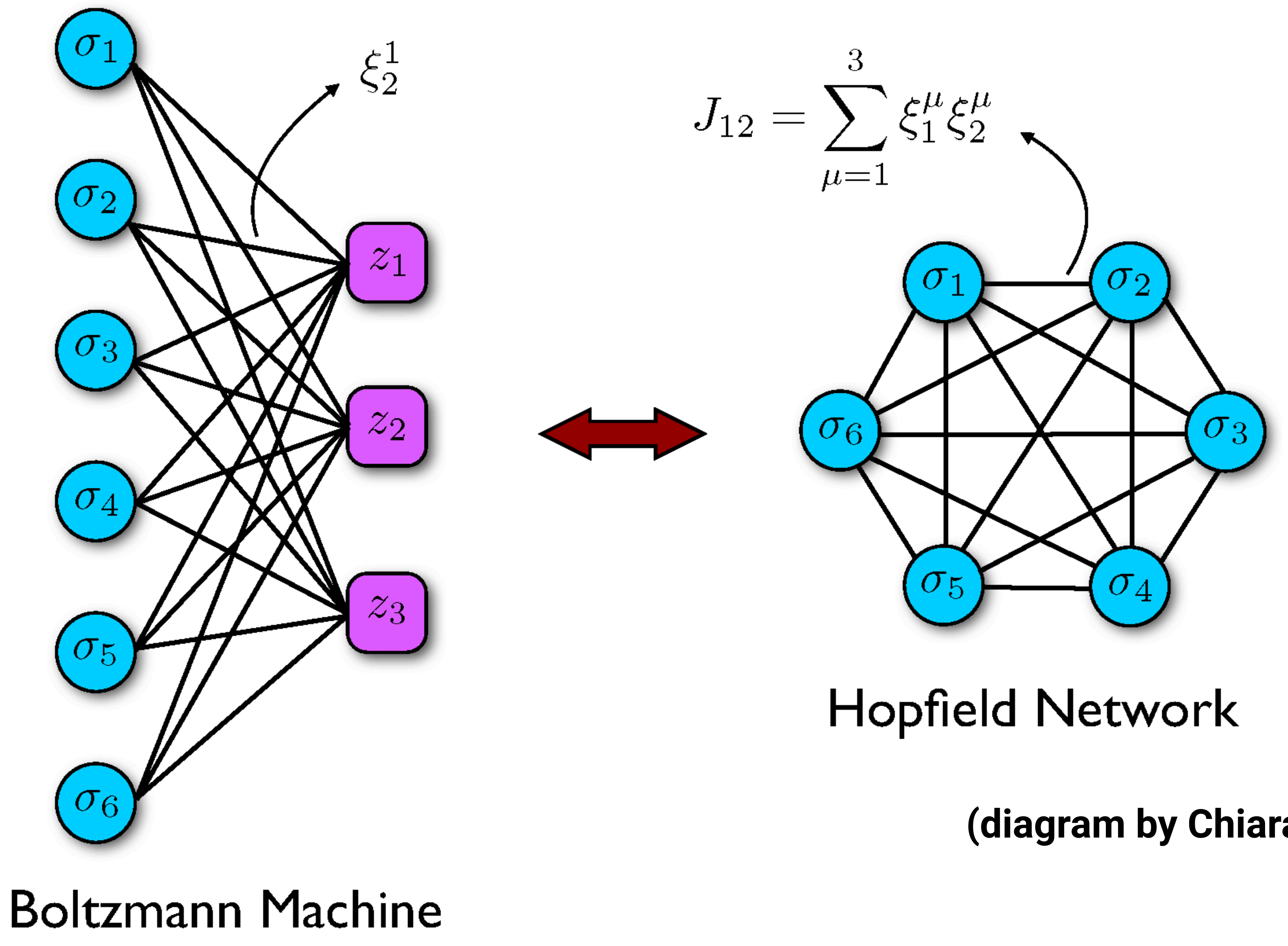


(diagram by Chiara Marullo)

- Applying Ising Model to Neural Networks
    - Originally proposed by Little (1974); then Hopfield (1982), generalized to continuous variables in Hopfield (1984)

Hopfield Networks & Boltzmann Machines

“Neighbor” ⇒”Connection”
One can think about updating neurons as if they were cells in an Ising Model!

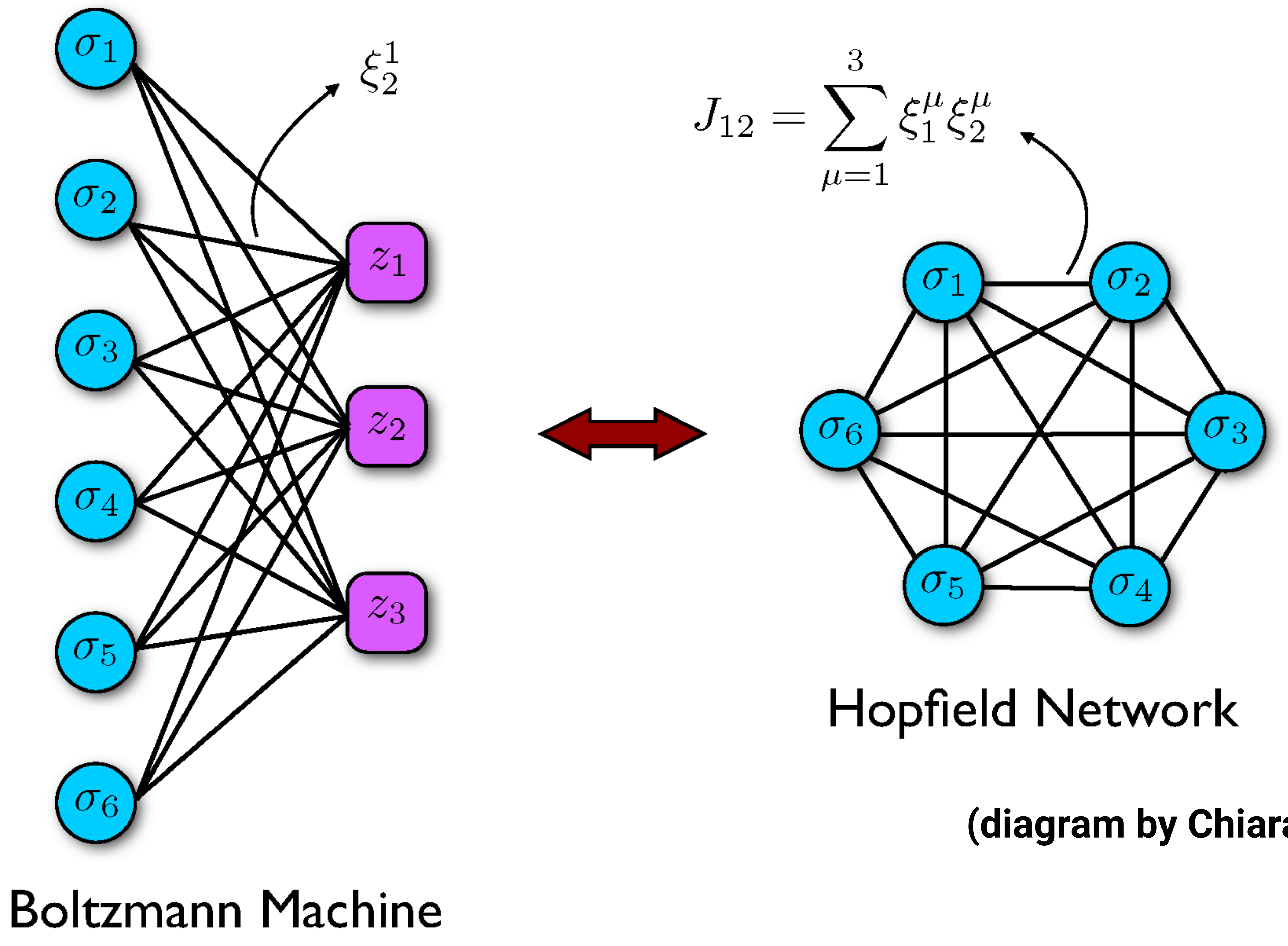


(diagram by Chiara Marullo)

- Applying Ising Model to Neural Networks
  - Originally proposed by Little (1974); then Hopfield (1982), generalized to continuous variables in Hopfield (1984)
- Modern developments include:

Hopfield Networks & Boltzmann Machines

“Neighbor” ⇒”Connection”
One can think about updating neurons as if they were cells in an Ising Model!



(diagram by Chiara Marullo)

- Applying Ising Model to Neural Networks
  - Originally proposed by Little (1974); then Hopfield (1982), generalized to continuous variables in Hopfield (1984)
- Modern developments include:
  - **Hopfield Networks is All You Need** (2020)**,** Hubert Ramsauer, et al., successfully applies a variant of modern Hopfield Networks to classification, NLP, and drug design problems, with great performance.

(A Small Selection of)

Physics-Informed Inductive Biases in the modern era

(A Small Selection of)

Physics-Informed Inductive Biases in the modern era

Categories:

1. Energy
2. Geometry
3. Differential Equations

Why Inductive Biases?

Why Inductive Biases?

- To beat the curse of dimensionality, inductive biases define a **prior** on the **space of learnable functions**

Why Inductive Biases?

- To beat the curse of dimensionality, inductive biases define a **prior** on the **space of learnable functions**

- As a simple example, limiting the L2 norm of neural network's weights places an upper bound on its Lipschitz constant.

Why Inductive Biases?

- To beat the curse of dimensionality, inductive biases define a **prior** on the **space of learnable functions**

- As a simple example, limiting the L2 norm of neural network's weights places an upper bound on its Lipschitz constant.

  - This is a prior which favors smooth functions; which is assumed for nearly every machine learning problem.

Why Inductive Biases?

- To beat the curse of dimensionality, inductive biases define a **prior** on the **space of learnable functions**

- As a simple example, limiting the L2 norm of neural network's weights places an upper bound on its Lipschitz constant.

  - This is a prior which favors smooth functions; which is assumed for nearly every machine learning problem.

- However, this prior is not enough. Physically-motivated inductive biases define additional priors on this function space.

Energy-Based Models (EBMs)

Energy-Based Models (EBMs)

- Formalism similarly based on framework of **statistical physics**

Energy-Based Models (EBMs)

- Formalism similarly based on framework of **statistical physics**
- Parameterize **energy function**

Energy-Based Models (EBMs)

- Formalism similarly based on framework of **statistical physics**
- Parameterize **energy function**
  - Seek to **minimize energy** over **positive pairs** of data, **maximize energy** over **negative pairs** of data.

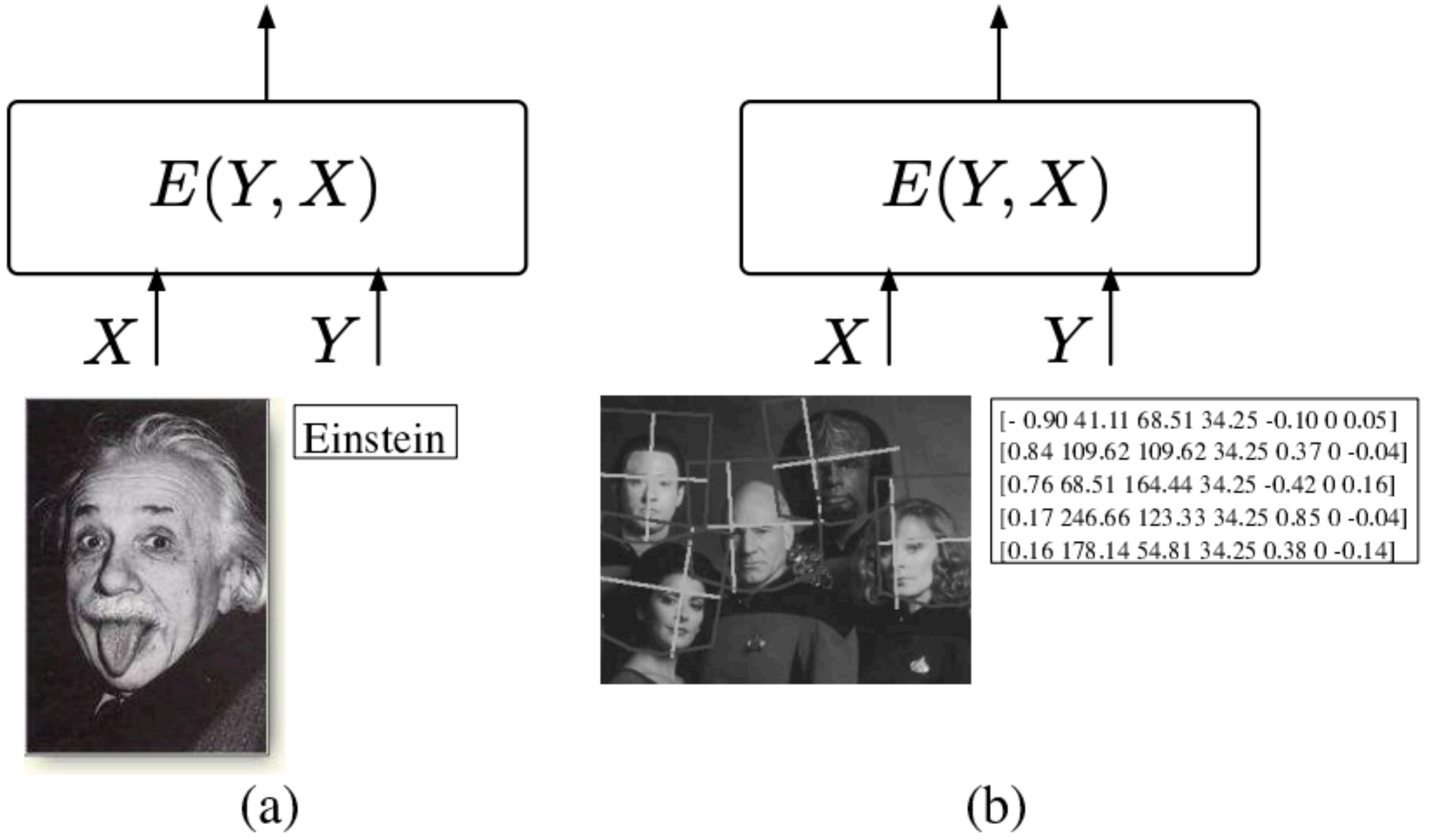
Energy-Based Models (EBMs)

- Formalism similarly based on framework of **statistical physics**

- Parameterize **energy function**

  - Seek to **minimize energy** over **positive pairs** of data, **maximize energy** over **negative pairs** of data.

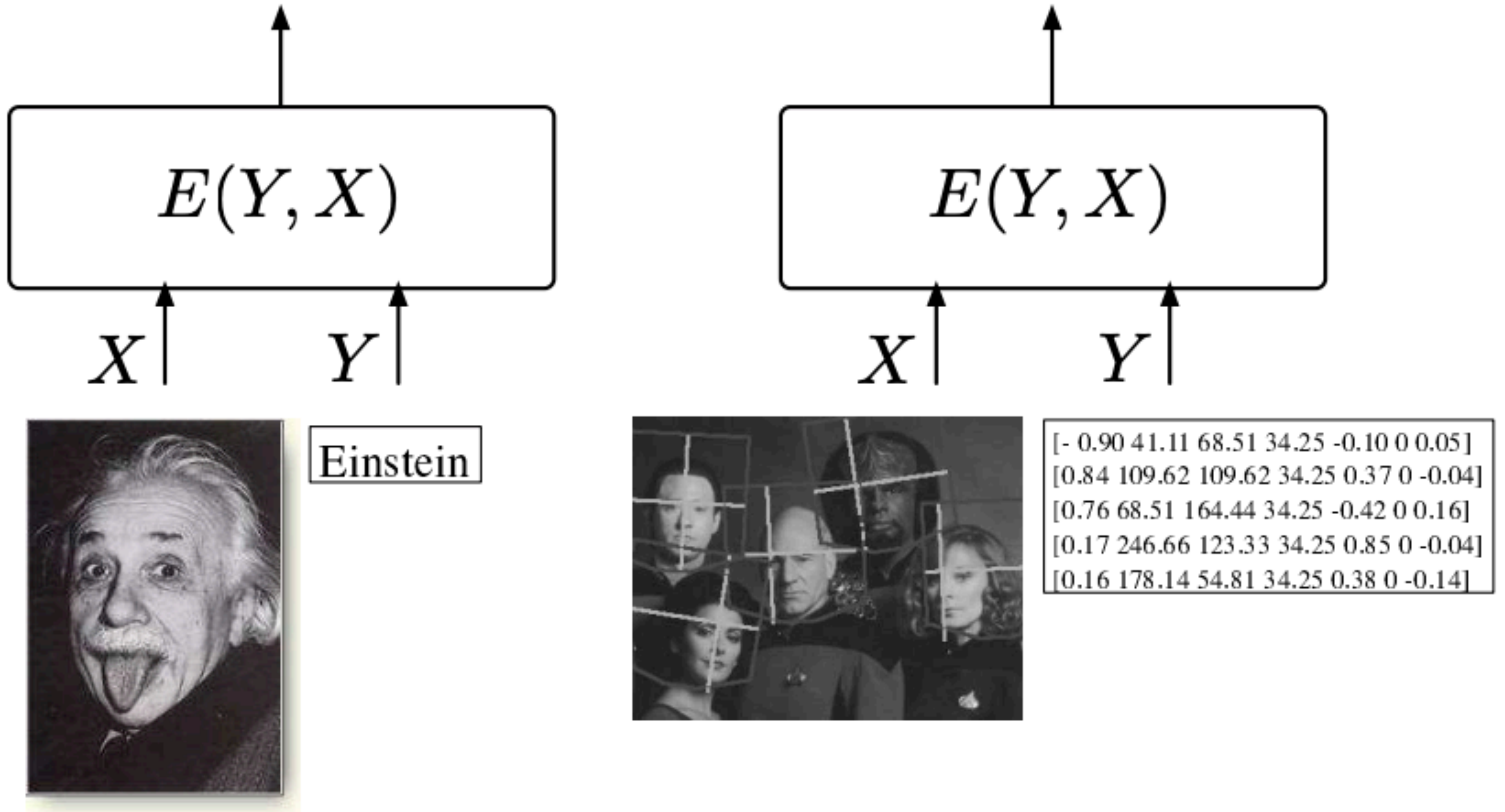
LeCun et al., (2006)



Energy-Based Models (EBMs)

- Formalism similarly based on framework of **statistical physics**
- Parameterize **energy function**
  - Seek to **minimize energy** over **positive pairs** of data, **maximize energy** over **negative pairs** of data.
  - Early work includes "Contrastive Divergence" by Hinton (2000); formalized into energy-based framework by LeCun

LeCun et al., (2006)



(a) (b)

Energy-Based Models (EBMs)

- Formalism similarly based on framework of **statistical physics**

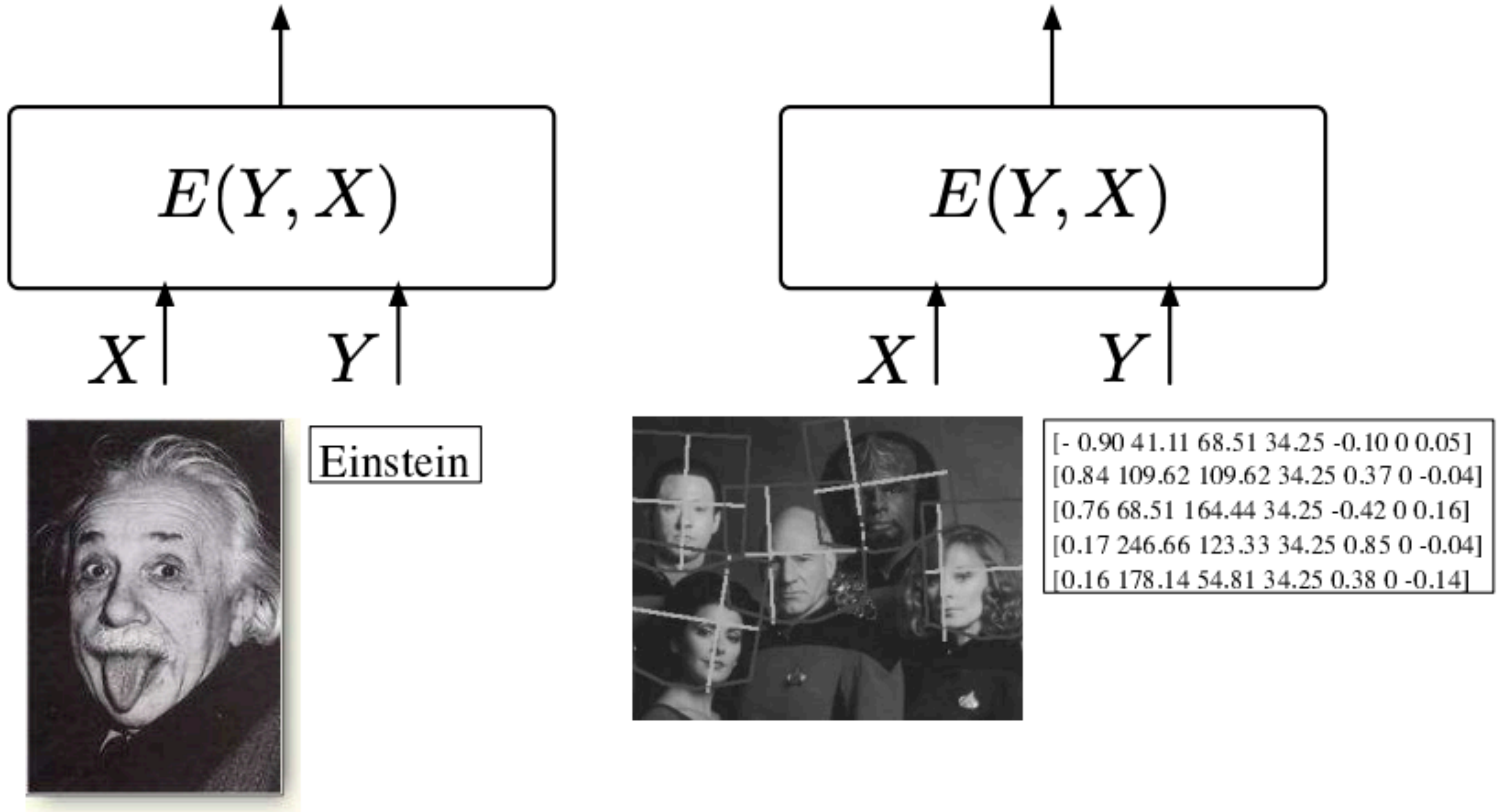
- Parameterize **energy function**

  - Seek to **minimize energy** over **positive pairs** of data, **maximize energy** over **negative pairs** of data.

  - Early work includes “Contrastive Divergence” by Hinton (2000); formalized into energy-based framework by LeCun

- Many ML problems can be easily rephrased in this **unified** energy-based framework!

LeCun et al., (2006)

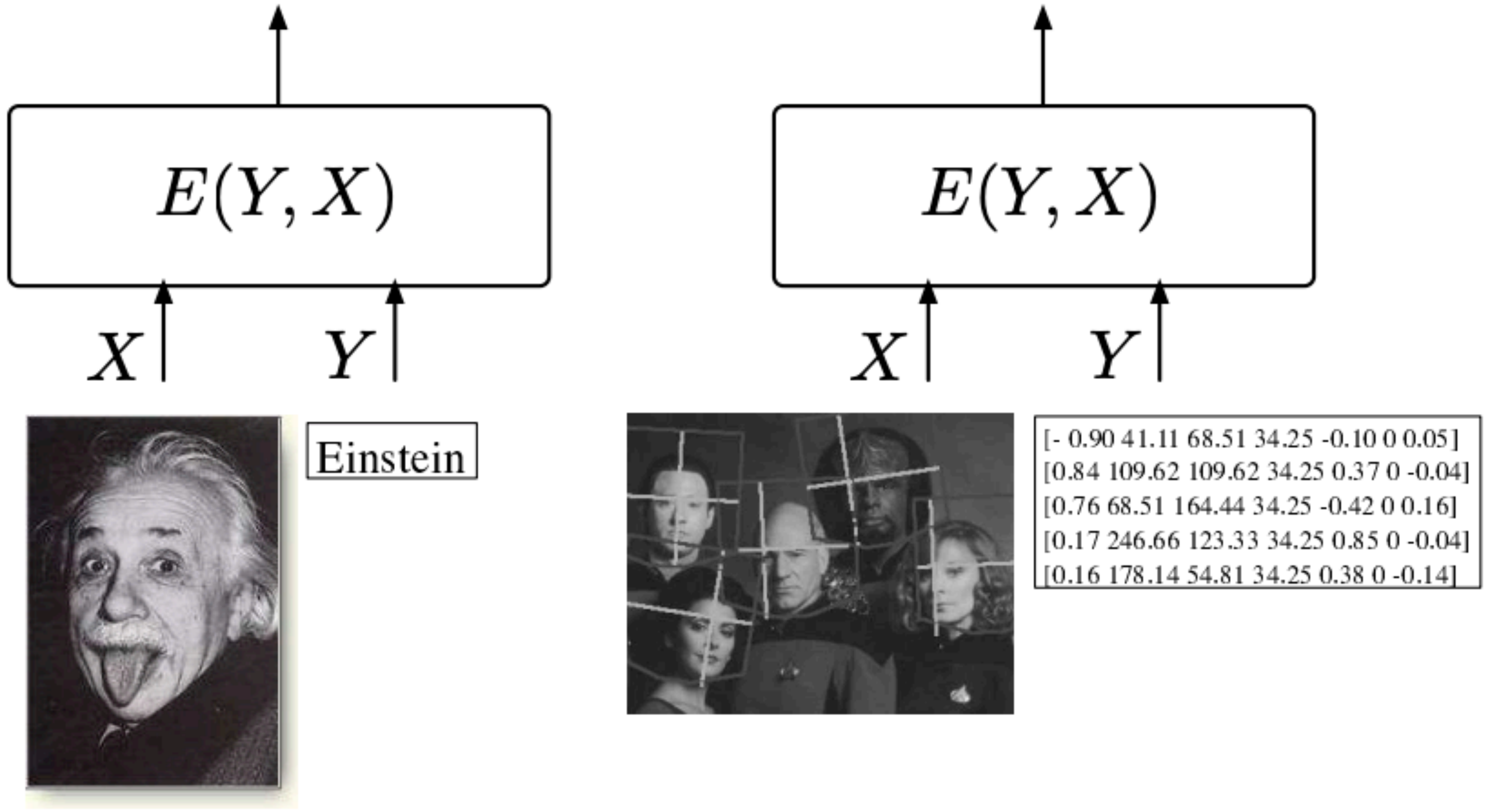


(a) (b)

Energy-Based Models (EBMs)

- Formalism similarly based on framework of **statistical physics**
- Parameterize **energy function**
  - Seek to **minimize energy** over **positive pairs** of data, **maximize energy** over **negative pairs** of data.
  - Early work includes “Contrastive Divergence” by Hinton (2000); formalized into energy-based framework by LeCun
- Many ML problems can be easily rephrased in this **unified** energy-based framework!

LeCun et al., (2006)



(a) (b)

Energy surface

Hamiltonian Neural Networks

Hamiltonian Neural Networks

- Greydanus et al., 2019

Hamiltonian Neural Networks

- Greydanus et al., 2019
- Learn energy, and apply Hamilton’s equations to get dynamical predictions.

Hamiltonian Neural Networks

- Greydanus et al., 2019
- Learn energy, and apply Hamilton's equations to get dynamical predictions.

$$\dot{q} = \frac{\partial H}{\partial p} \qquad \dot{p} = -\frac{\partial H}{\partial q}$$

Hamiltonian Neural Networks

- Greydanus et al., 2019
- Learn energy, and apply Hamilton’s equations to get dynamical predictions.

Learned Function

$$\dot{q} = \frac{\partial H}{\partial p} \qquad \dot{p} = -\frac{\partial H}{\partial q}$$

Hamiltonian Neural Networks

- Greydanus et al., 2019
- Learn energy, and apply Hamilton's equations to get dynamical predictions.

Learned Function

$$\dot{q} = \frac{\partial H}{\partial p} \qquad \dot{p} = -\frac{\partial H}{\partial q}$$

Time derivative of position and momentum

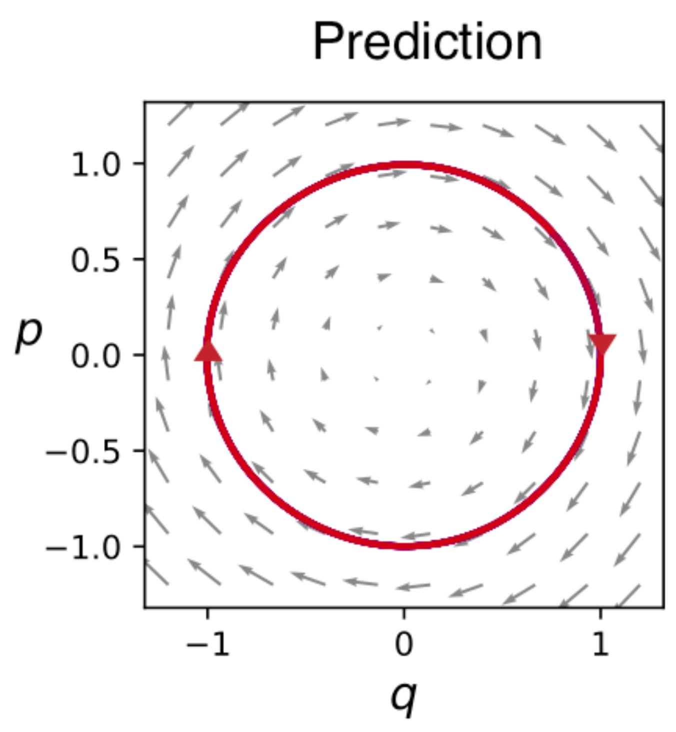
Hamiltonian Neural Networks

- Greydanus et al., 2019
- Learn energy, and apply Hamilton's equations to get dynamical predictions.

Learned Function

$$\dot{q} = \frac{\partial H}{\partial p} \qquad \dot{p} = -\frac{\partial H}{\partial q}$$

Time derivative of position and momentum



Hamiltonian Neural Networks

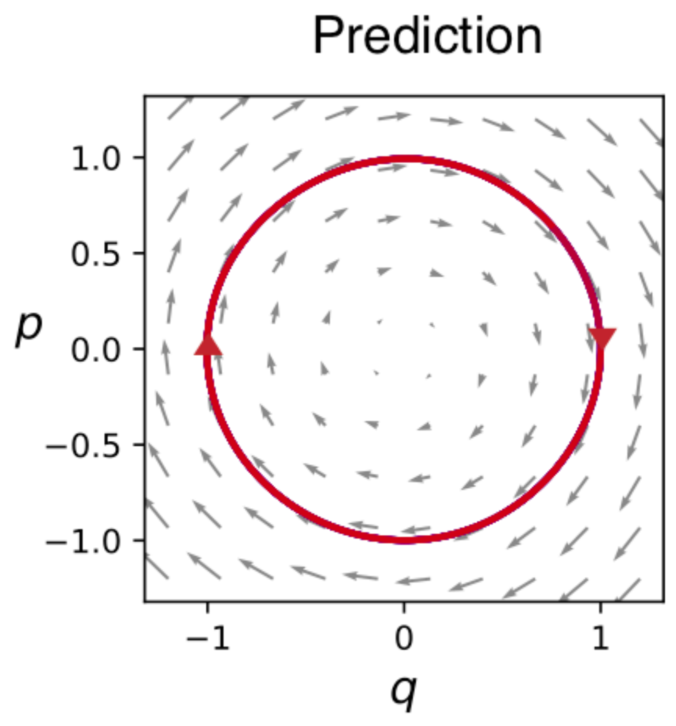
- Greydanus et al., 2019
- Learn energy, and apply Hamilton's equations to get dynamical predictions.

- This gives the model **explicit** and **exact** energy conservation
- Can even apply to a latent representation of an video:

Learned Function

$$\dot{q} = \frac{\partial H}{\partial p} \qquad \dot{p} = -\frac{\partial H}{\partial q}$$

Time derivative of position and momentum



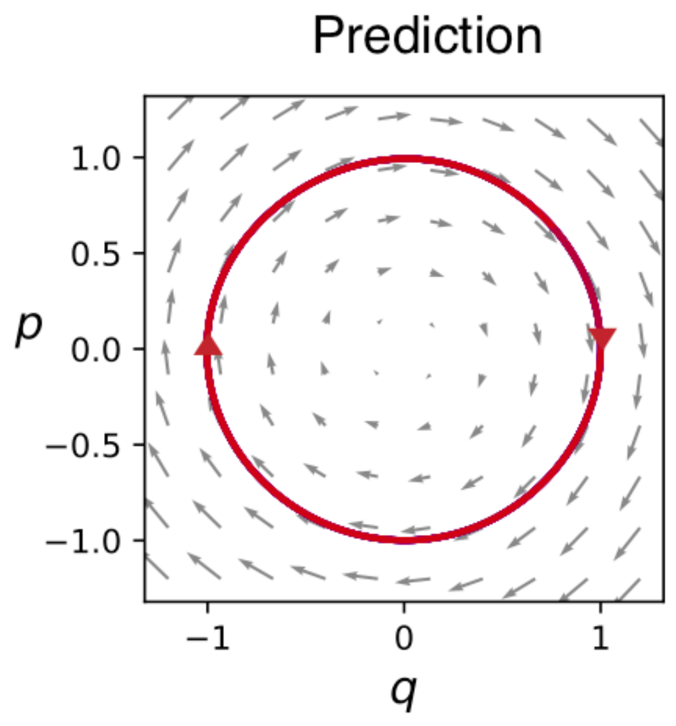
Hamiltonian Neural Networks

- Greydanus et al., 2019
- Learn energy, and apply Hamilton's equations to get dynamical predictions.

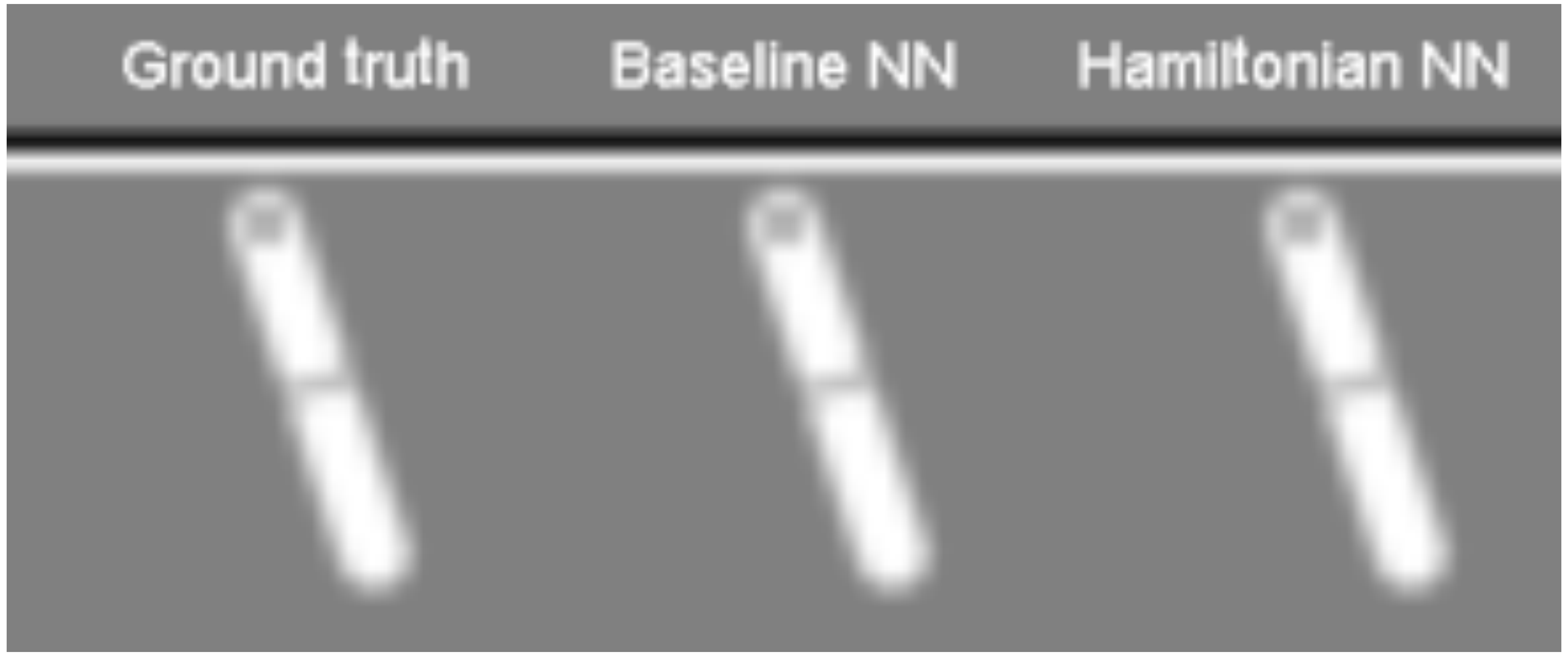
Learned Function

$$\dot{q} = \frac{\partial H}{\partial p} \qquad \dot{p} = -\frac{\partial H}{\partial q}$$

Time derivative of position and momentum



- This gives the model **explicit** and **exact** energy conservation
- Can even apply to a latent representation of an video:



Lagrangian Neural Networks

Lagrangian Neural Networks

- Generalized energy-conserving model:
  the LNN (Cranmer et al., 2020)

Lagrangian Neural Networks

- Generalized energy-conserving model:
  the LNN (Cranmer et al., 2020)

- Precursor work: DeLaN (Lutter et al., 2019).

Lagrangian Neural Networks

- Generalized energy-conserving model:
  the LNN (Cranmer et al., 2020)

- Precursor work: DeLaN (Lutter et al., 2019).

  - Issue with HNNs and DeLaN: require
    known functional form of **kinetic energy**

Lagrangian Neural Networks

- Generalized energy-conserving model: the LNN (Cranmer et al., 2020)
- Precursor work: DeLaN (Lutter et al., 2019).
  - Issue with HNNs and DeLaN: require **known** functional form of **kinetic energy**

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} = \frac{\partial \mathcal{L}}{\partial q_j} \qquad \text{Euler-Lagrange (5)}$$

$$\frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} = \nabla_q \mathcal{L} \qquad \text{vectorize (6)}$$

$$\nabla_q \mathcal{L} = (\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L})\ddot{q} + (\nabla_q \nabla_{\dot{q}}^{\top}\mathcal{L})\dot{q} \qquad \text{expand } \frac{d}{dt} \text{ (7)}$$

$$\ddot{q} = (\nabla_{\dot{q}}\nabla_{\dot{q}}^{\top}\mathcal{L})^{-1}[\nabla_q \mathcal{L} - (\nabla_q \nabla_{\dot{q}}^{\top}\mathcal{L})\dot{q}] \qquad \text{solve for } \ddot{q} \text{ (8)}$$

Lagrangian Neural Networks

- Generalized energy-conserving model:
  the LNN (Cranmer et al., 2020)

- Precursor work: DeLaN (Lutter et al., 2019).

  - Issue with HNNs and DeLaN: require
    known functional form of **kinetic energy**

Lagrangian Neural Networks

- Generalized energy-conserving model: the LNN (Cranmer et al., 2020)

- Precursor work: DeLaN (Lutter et al., 2019).

  - Issue with HNNs and DeLaN: require **known** functional form of **kinetic energy**

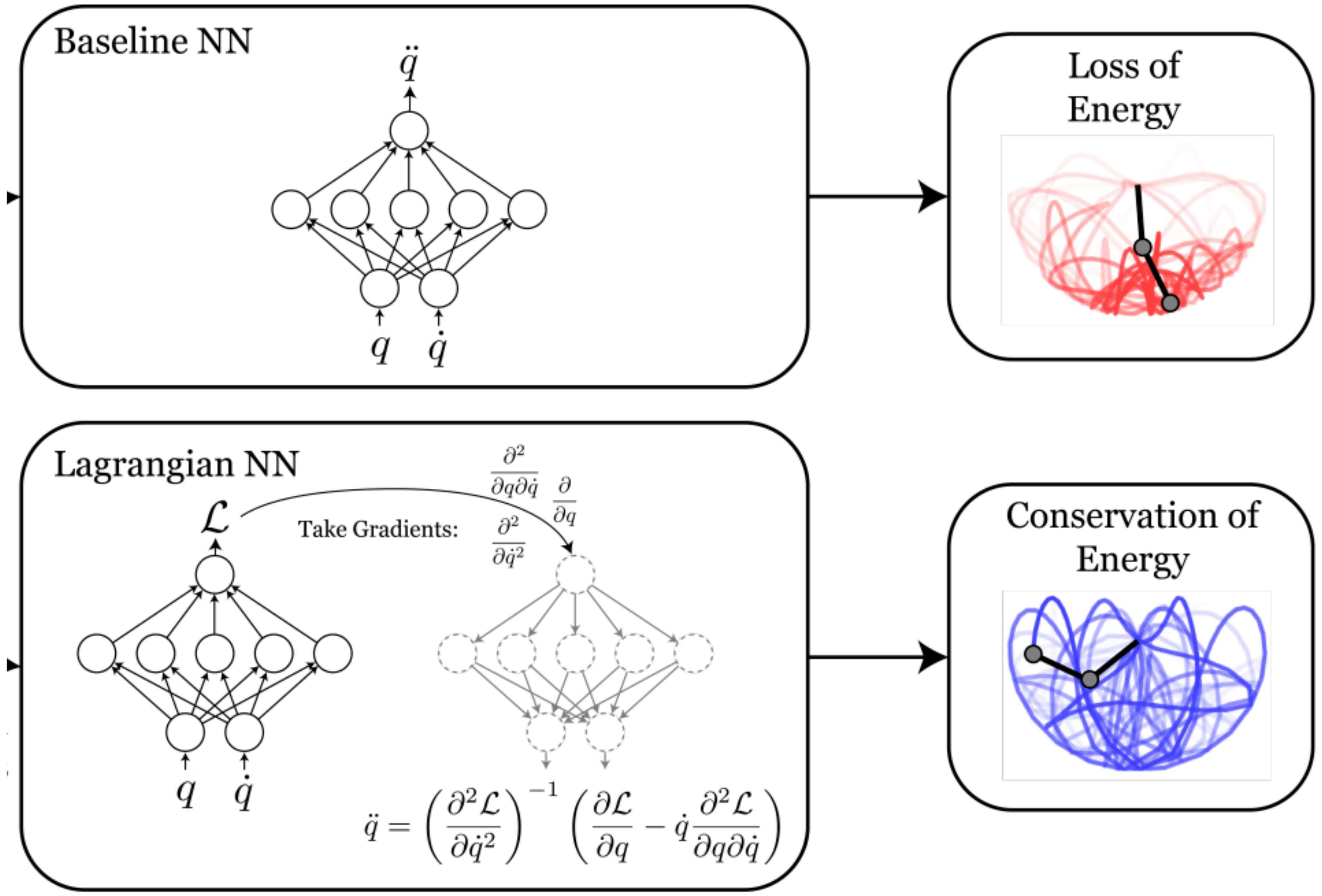
Learned Function

$$\ddot{q} = (\nabla_{\dot{q}} \nabla_{\dot{q}}^{\top} L)^{-1} (\nabla_q L - (\nabla_q \nabla_{\dot{q}}^{\top} L)\dot{q})$$

Second order gradient ⇒matrix inverse

Lagrangian Neural Networks

- Generalized energy-conserving model: the LNN (Cranmer et al., 2020)
- Precursor work: DeLaN (Lutter et al., 2019).
  - Issue with HNNs and DeLaN: require **known** functional form of **kinetic energy**



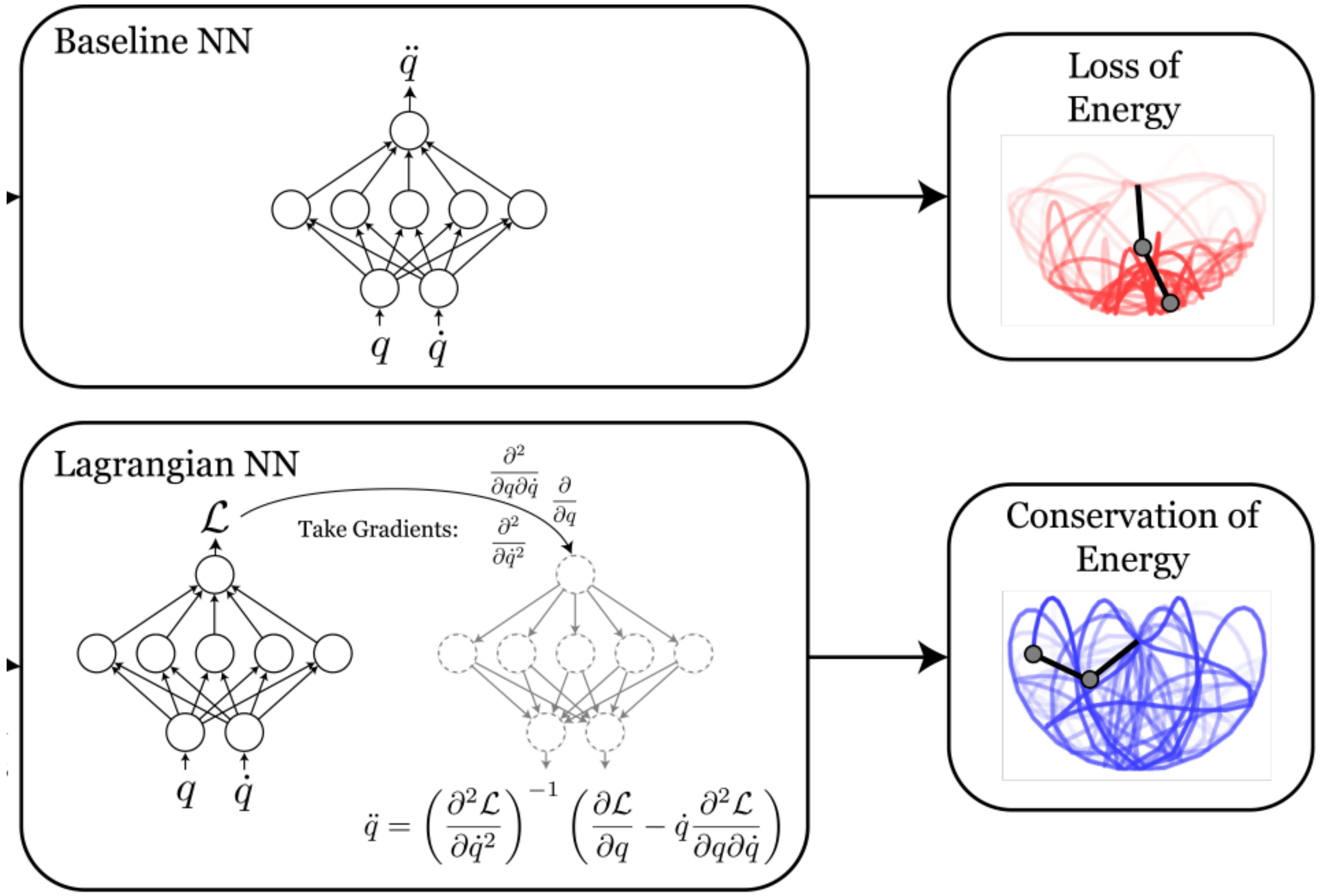
Learned Function

$$\ddot{q} = (\nabla_{\dot{q}} \nabla_{\dot{q}}^{\top} L)^{-1} (\nabla_q L - (\nabla_q \nabla_{\dot{q}}^{\top} L)\dot{q})$$

Second order gradient ⇒matrix inverse

Lagrangian Neural Networks

- Generalized energy-conserving model: the LNN (Cranmer et al., 2020)
- Precursor work: DeLaN (Lutter et al., 2019).
    - Issue with HNNs and DeLaN: require **known** functional form of **kinetic energy**

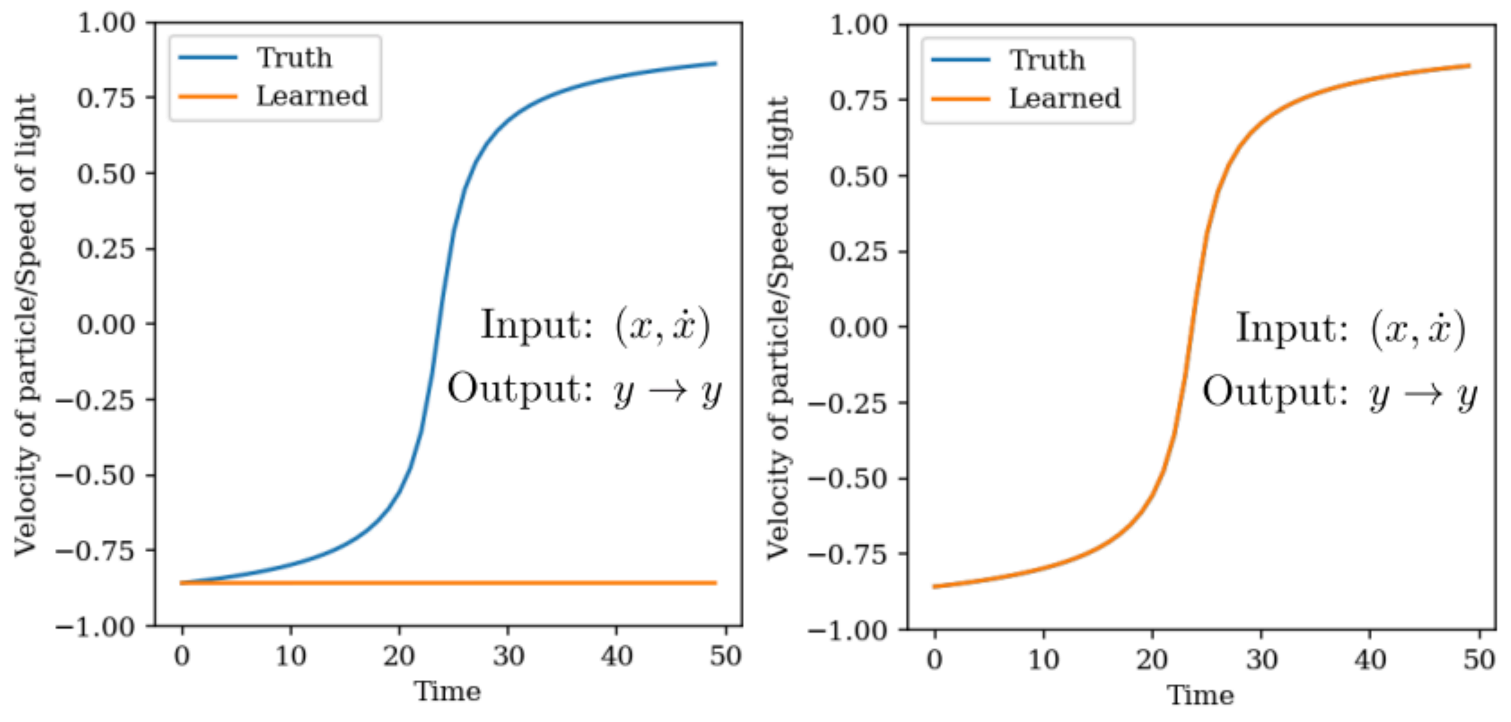


Learned Function

$$\ddot{q} = (\nabla_{\dot{q}} \nabla_{\dot{q}}^{\top} L)^{-1} (\nabla_q L - (\nabla_q \nabla_{\dot{q}}^{\top} L)\dot{q})$$

Second order gradient ⇒matrix inverse

Without known kinetic energy:



HNN LNN

Hamiltonian Generative Networks

Hamiltonian Generative Networks

- Toth et al., (2019)

Hamiltonian Generative Networks

- Toth et al., (2019)
  - Hamiltonian dynamics conserves **energy**.

Hamiltonian Generative Networks

- Toth et al., (2019)
  - Hamiltonian dynamics conserves **energy**.
  - Define probability = energy; then we conserve total probability! Use to define a normalizing flow.

Hamiltonian Generative Networks

- Toth et al., (2019)
  - Hamiltonian dynamics conserves **energy**.
  - Define probability = energy; then we conserve total probability! Use to define a normalizing flow.
- Can apply to regular dynamical problems with a probabilistic model:

Hamiltonian Generative Networks

- Toth et al., (2019)
  - Hamiltonian dynamics conserves **energy**.
  - Define probability = energy; then we conserve total probability! Use to define a normalizing flow.
- Can apply to regular dynamical problems with a probabilistic model:

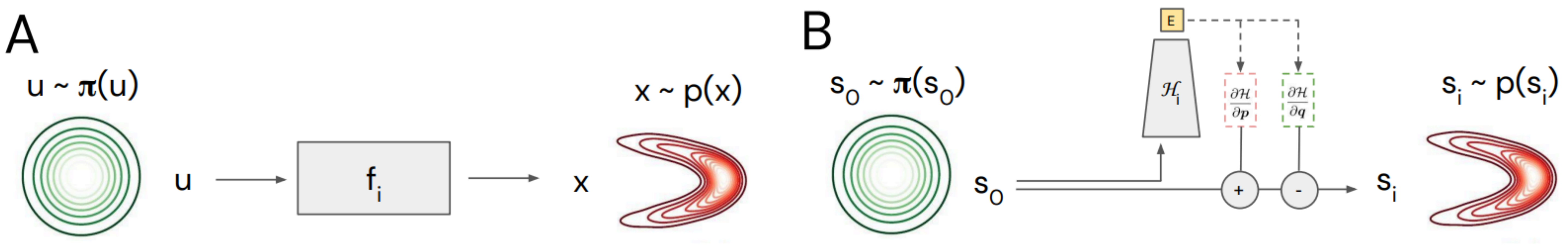
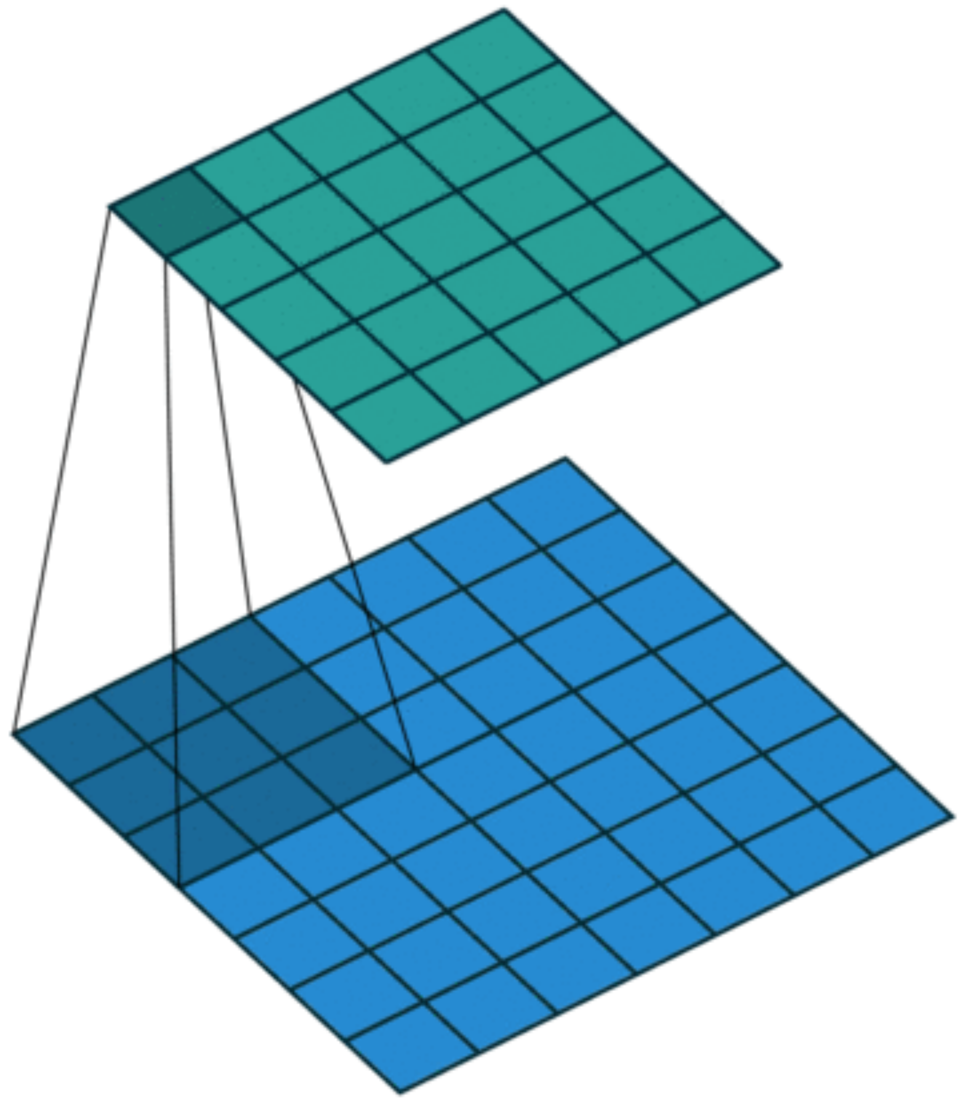
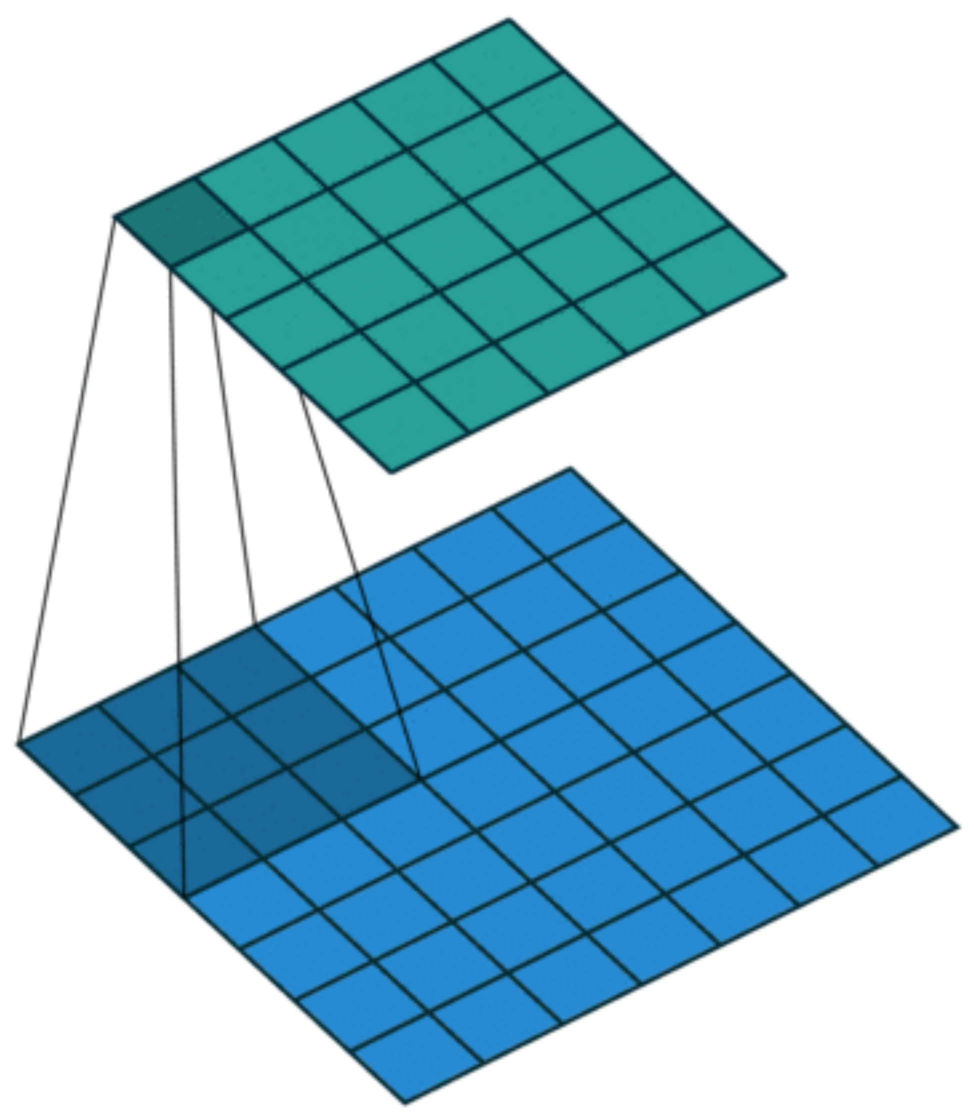


Figure 3: **A**: standard normalising flow, where the invertible function f_i is implemented by a neural network. **B**: Hamiltonian flows, where the initial density is transformed using the learned Hamiltonian dynamics. Note that we depict Euler updates of the state for schematic simplicity, while in practice this is done using a leapfrog integrator.

Inductive Biases which Specify <u>Geometry</u>

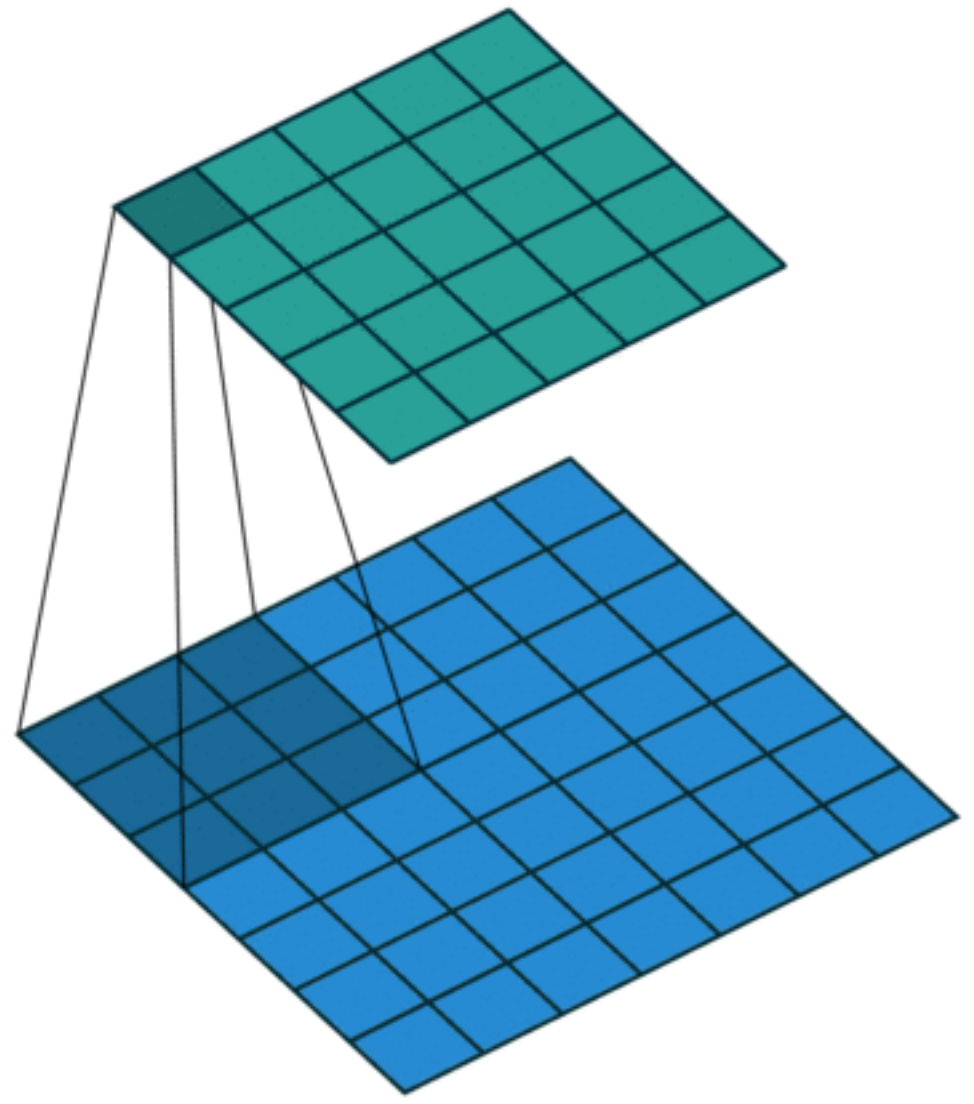


Inductive Biases which Specify <u>Geometry</u>



Inductive Biases which Specify Geometry

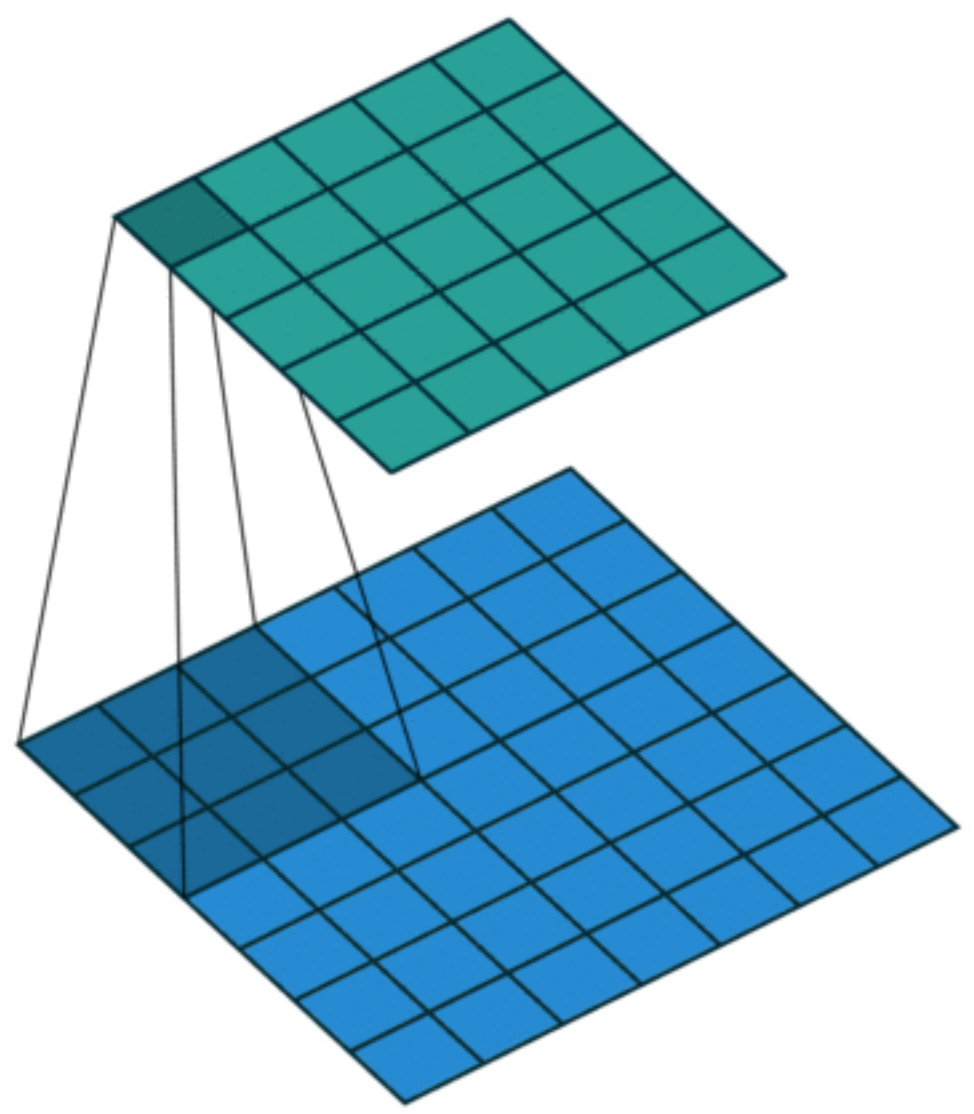
- Long been known that symmetries are important for machine learning. Much of this is rooted in fact that ML implicitly and explicitly models the physical world: and so the universe's symmetries make for good inductive biases.



Inductive Biases which Specify <u>Geometry</u>

- Long been known that symmetries are important for machine learning. Much of this is rooted in fact that ML implicitly and explicitly models the physical world: and so the universe's symmetries make for good inductive biases.

- Convolutional Neural Networks are translationally **equivariant**

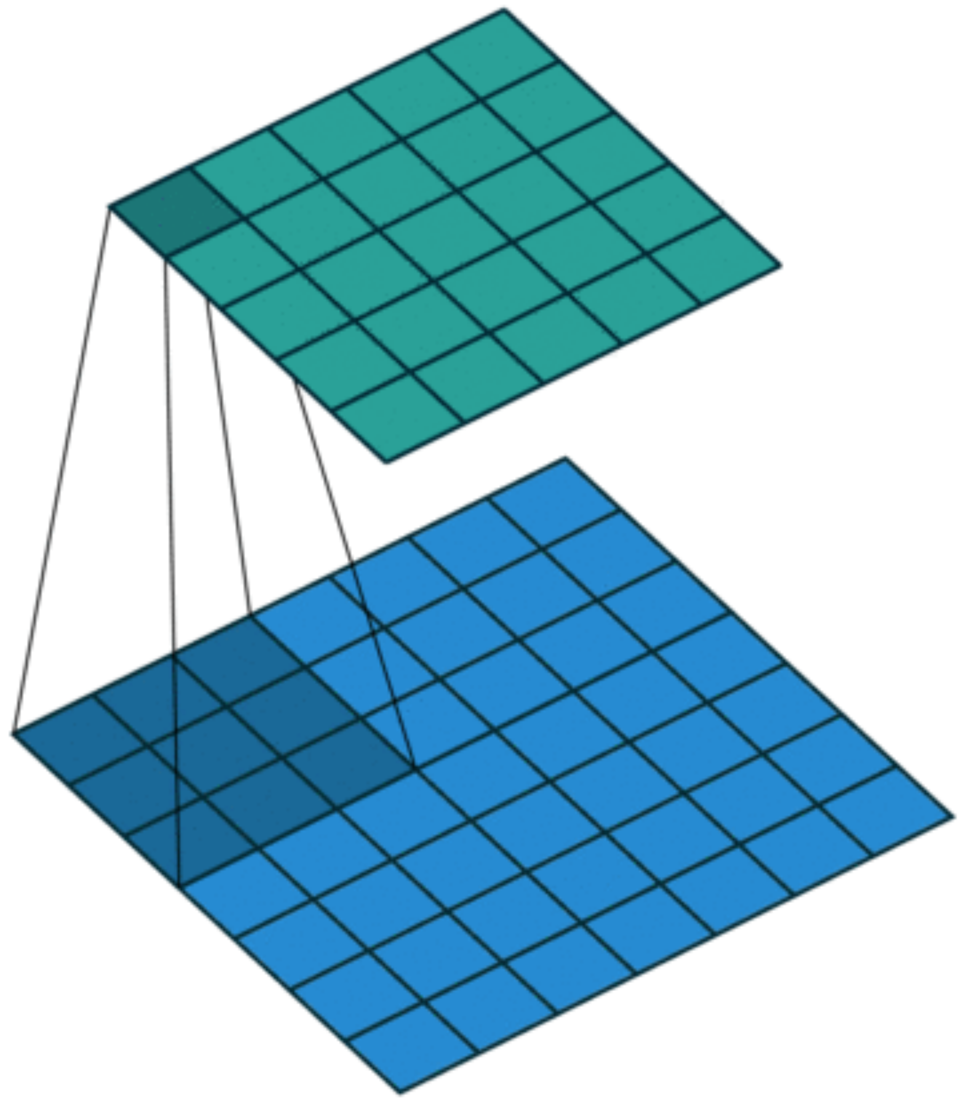


Inductive Biases which Specify Geometry

- Long been known that symmetries are important for machine learning. Much of this is rooted in fact that ML implicitly and explicitly models the physical world: and so the universe's symmetries make for good inductive biases.

- Convolutional Neural Networks are translationally **equivariant**

  - Invariance: $h(x) = h(g \circ x) \forall g \in G$, a group.



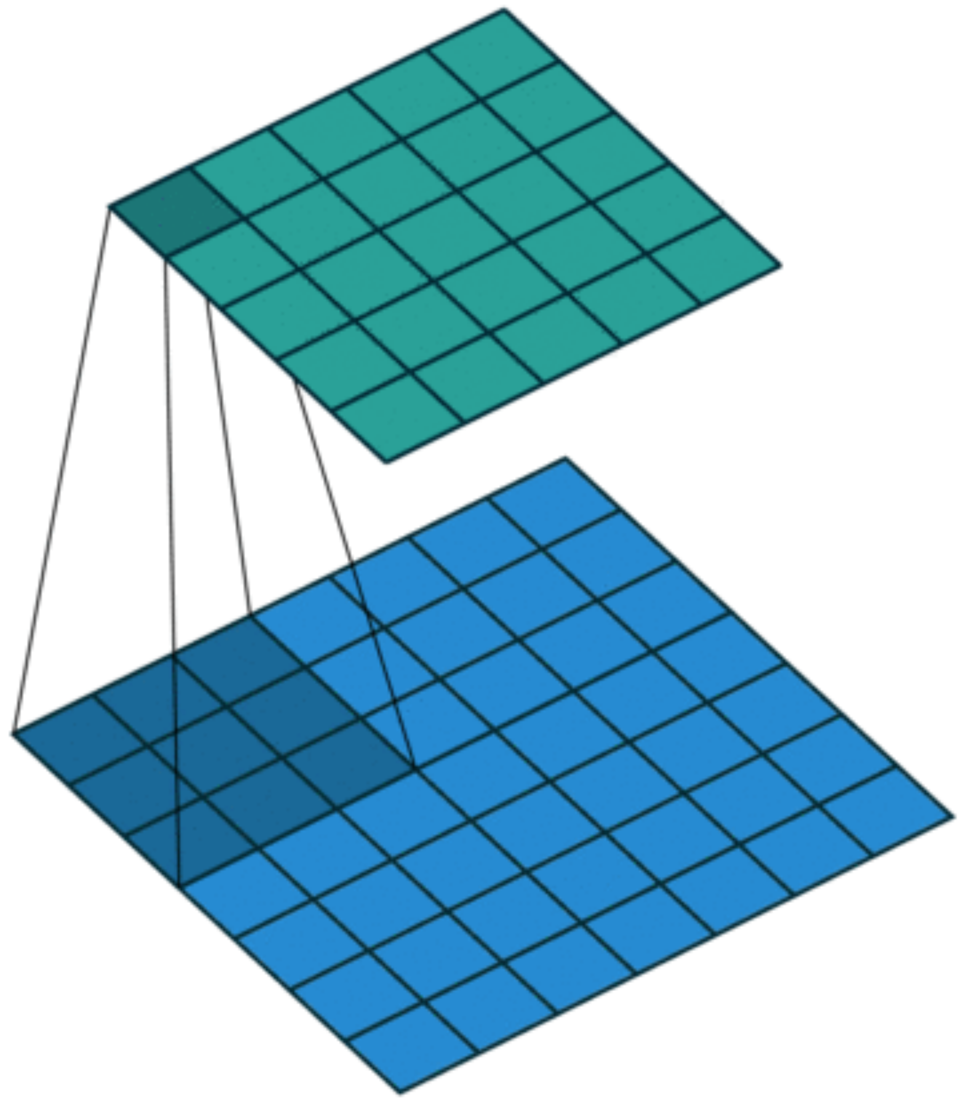
Inductive Biases which Specify <u>Geometry</u>

- Long been known that symmetries are important for machine learning. Much of this is rooted in fact that ML implicitly and explicitly models the physical world: and so the universe's symmetries make for good inductive biases.

- Convolutional Neural Networks are translationally **equivariant**

  - Invariance: $h(x) = h(g \circ x) \forall g \in G$, a group.

  - Equivariance: $g \circ f(x) = f(g \circ x) \forall g \in G$.



Inductive Biases which Specify Geometry

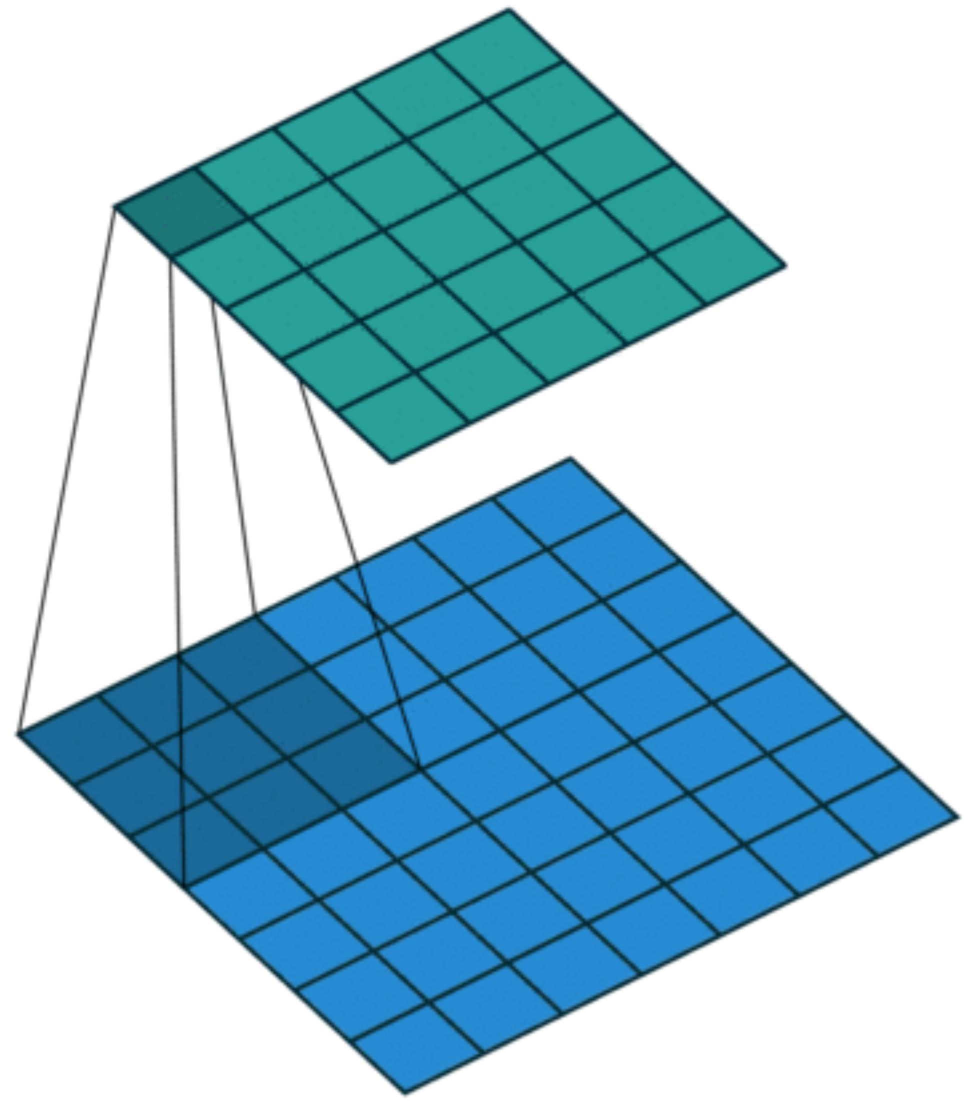
- Long been known that symmetries are important for machine learning. Much of this is rooted in fact that ML implicitly and explicitly models the physical world: and so the universe's symmetries make for good inductive biases.

- Convolutional Neural Networks are translationally **equivariant**

  - Invariance: $h(x) = h(g \circ x) \forall g \in G$, a group.

  - Equivariance: $g \circ f(x) = f(g \circ x) \forall g \in G$.

- The universe obeys translational symmetry. This is equivalent to **momentum conservation**.



Inductive Biases which Specify Geometry

- Long been known that symmetries are important for machine learning. Much of this is rooted in fact that ML implicitly and explicitly models the physical world: and so the universe's symmetries make for good inductive biases.

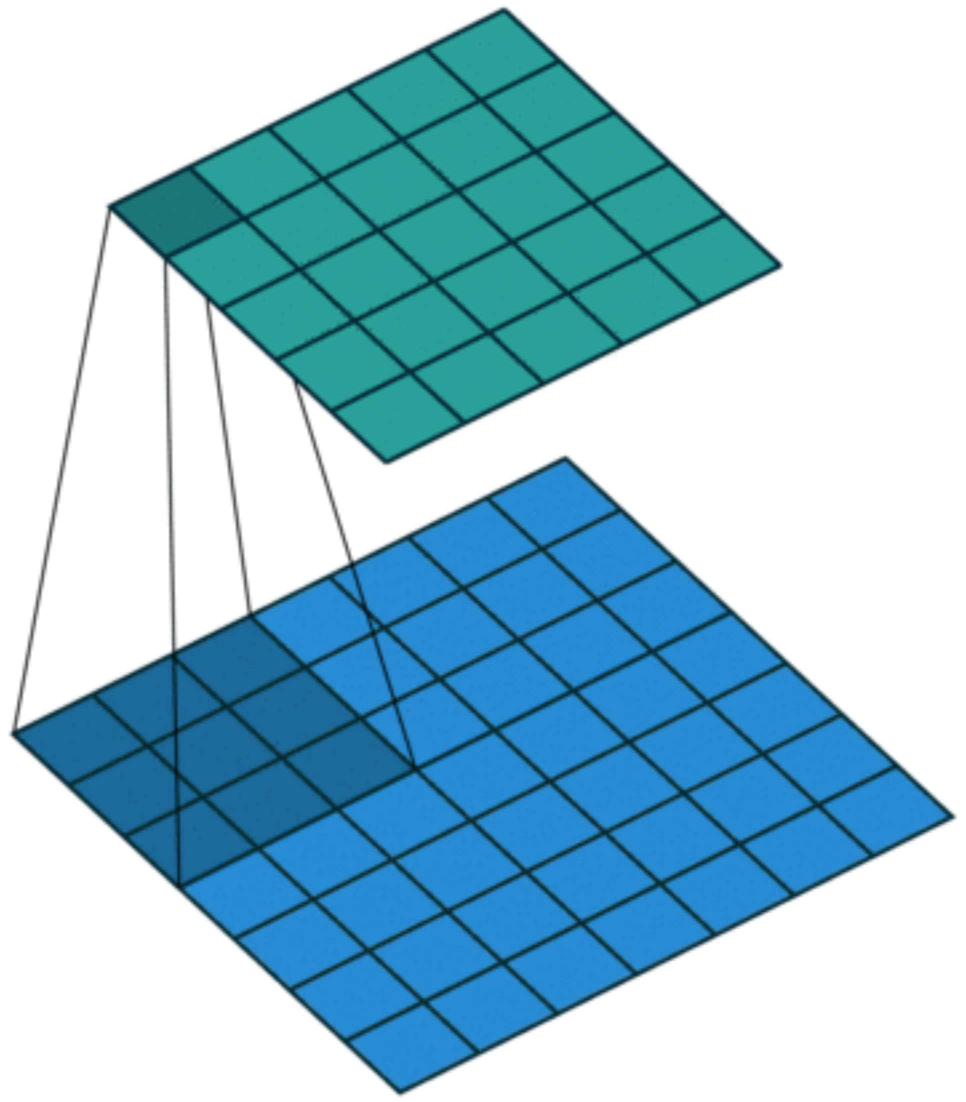
- Convolutional Neural Networks are translationally **equivariant**

  - Invariance: $h(x) = h(g \circ x) \forall g \in G$, a group.

  - Equivariance: $g \circ f(x) = f(g \circ x) \forall g \in G$.

- The universe obeys translational symmetry. This is equivalent to **momentum conservation**.

- This symmetry is intuitive because we have been living with these physical laws. Perhaps it would not be as intuitive if the laws of physics changed at every point of space!



Why is it good to formalize this?

Why is it good to formalize this?

- Describing ConvNet’s equivariance in a formal framework like this lets you consider other symmetries.
  - For example, ConvNets do not by default have rotational symmetry.
  - Taco Cohen & Max Welling (2016) derived this: the Group Equivariant-CNN. Makes the CNN rotationally invariant.

Why is it good to formalize this?

- Describing ConvNet's equivariance in a formal framework like this lets you consider other symmetries.
  - For example, ConvNets do not by default have rotational symmetry.
  - Taco Cohen & Max Welling (2016) derived this: the Group Equivariant-CNN. Makes the CNN rotationally invariant.

Generalize a convolution to any group convolution:

Why is it good to formalize this?

- Describing ConvNet's equivariance in a formal framework like this lets you consider other symmetries.
  - For example, ConvNets do not by default have rotational symmetry.
  - Taco Cohen & Max Welling (2016) derived this: the Group Equivariant-CNN. Makes the CNN rotationally invariant.

Generalize a convolution to any group convolution:

Latent pixel coordinate — Image — Filter

$$[f \star \psi](x) = \sum_{y \in \mathbb{Z}^2} \sum_k f_k(y)\psi_k(x - y)$$

Translation

Input pixel coordinate

Why is it good to formalize this?

- Describing ConvNet's equivariance in a formal framework like this lets you consider other symmetries.
  - For example, ConvNets do not by default have rotational symmetry.
  - Taco Cohen & Max Welling (2016) derived this: the Group Equivariant-CNN. Makes the CNN rotationally invariant.

Generalize a convolution to any group convolution:

Latent pixel coordinate
Image
Filter

$$[f \star \psi](x) = \sum_{y \in \mathbb{Z}^2} \sum_k f_k(y)\psi_k(x - y)$$

Translation
Input pixel coordinate

Discrete group

$$[f \star \psi](g) = \sum_{y \in \mathbb{Z}^2} \sum_k f_k(y)\psi_k(g^{-1}y)$$

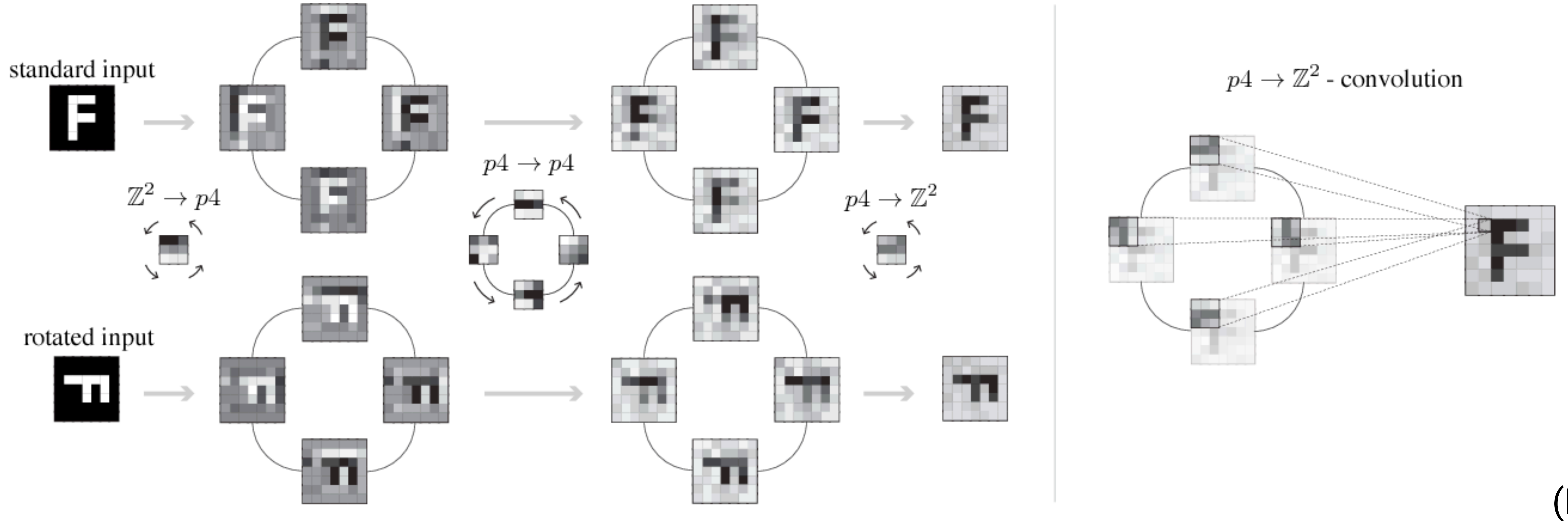
Can have this be a rotation group!

Group Equivariant CNN

Discrete group

$$[f \star \psi](g) = \sum_{y \in \mathbb{Z}^2} \sum_k f_k(y) \psi_k(g^{-1}y)$$

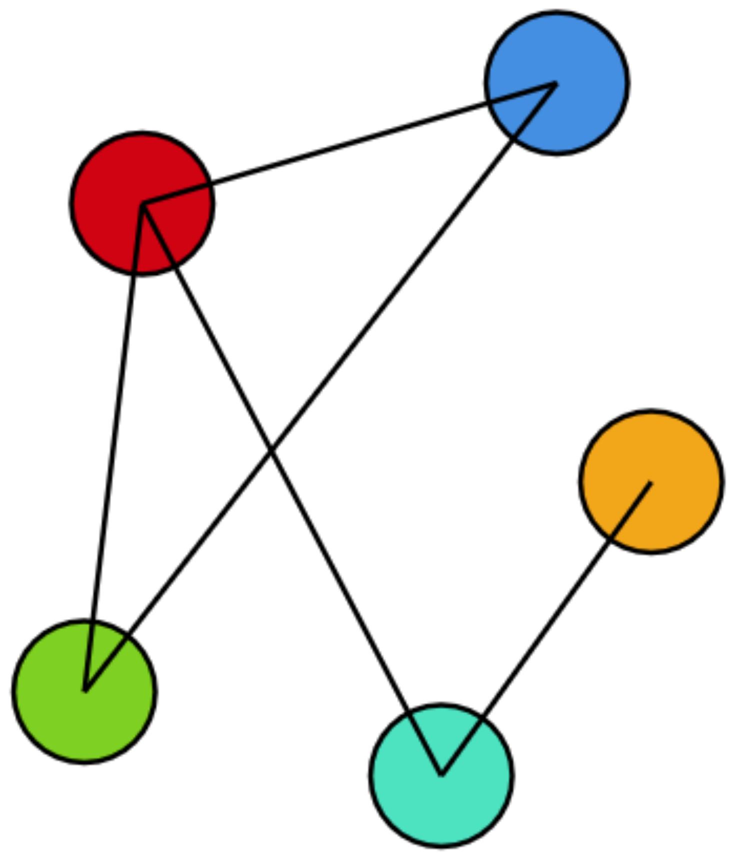
Can have this be a rotation group!



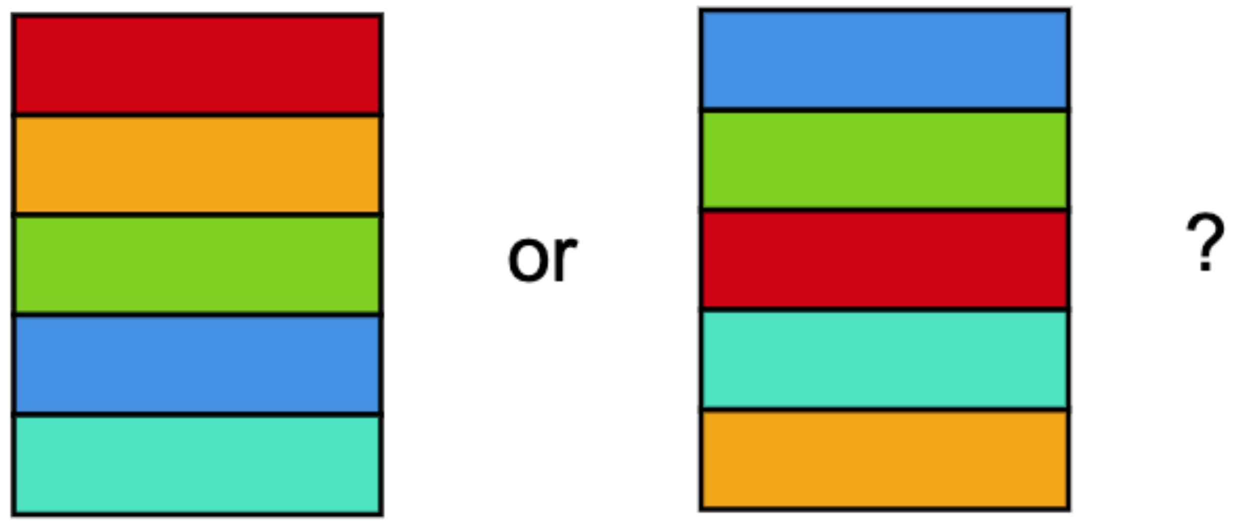
(Note that rotational symmetry is also a symmetry of the universe)

Graph Nets

Graph Nets

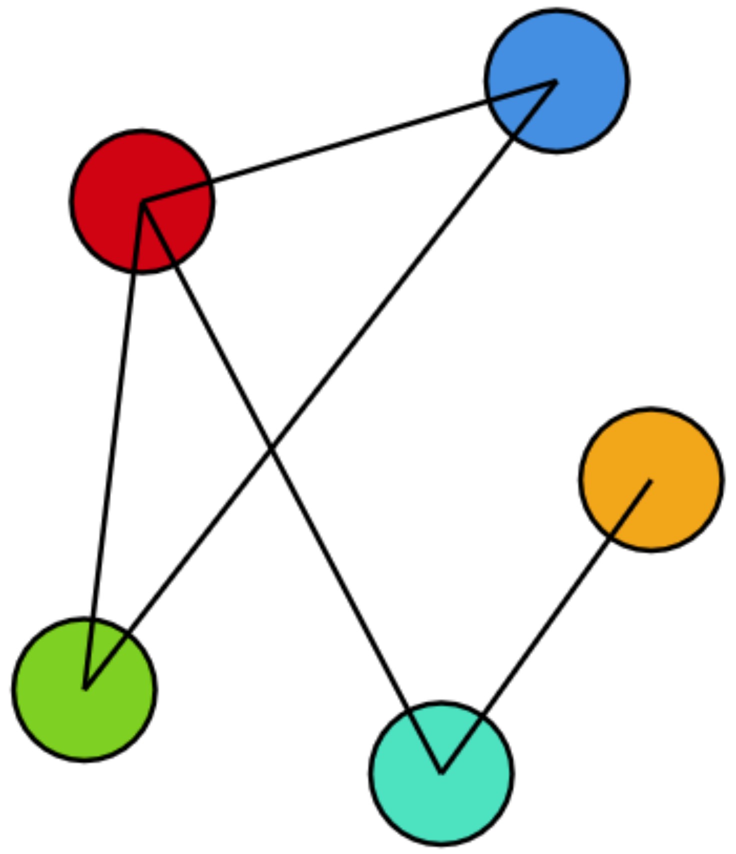


How should we pass this graph to a model?

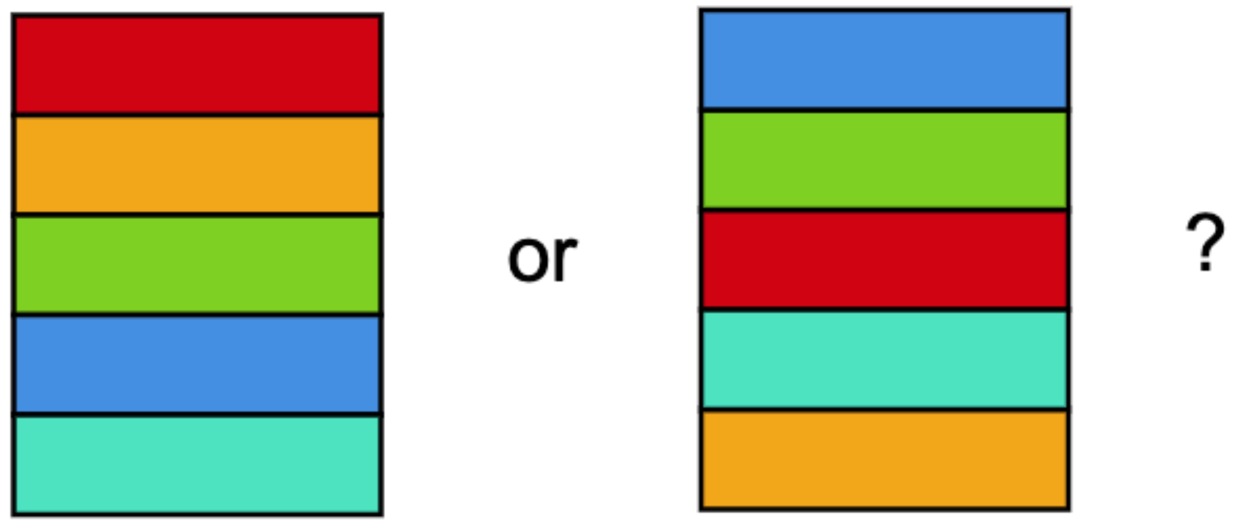


DeepSets, Graph Neural Networks have permutation invariance and equivariance, respectively.
Don't need to make such a choice!

Graph Nets



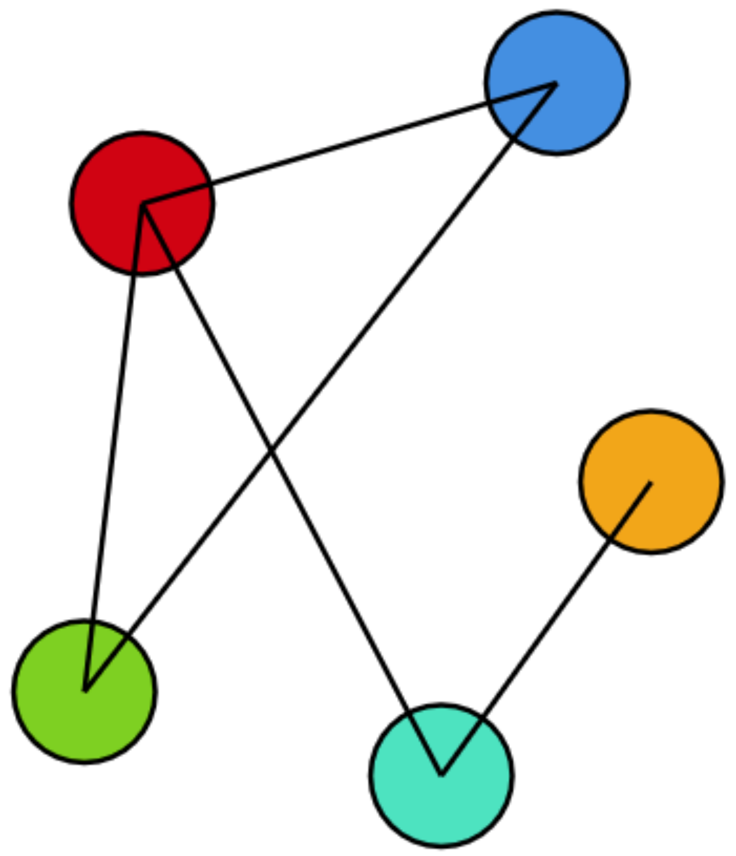
How should we pass this graph to a model?



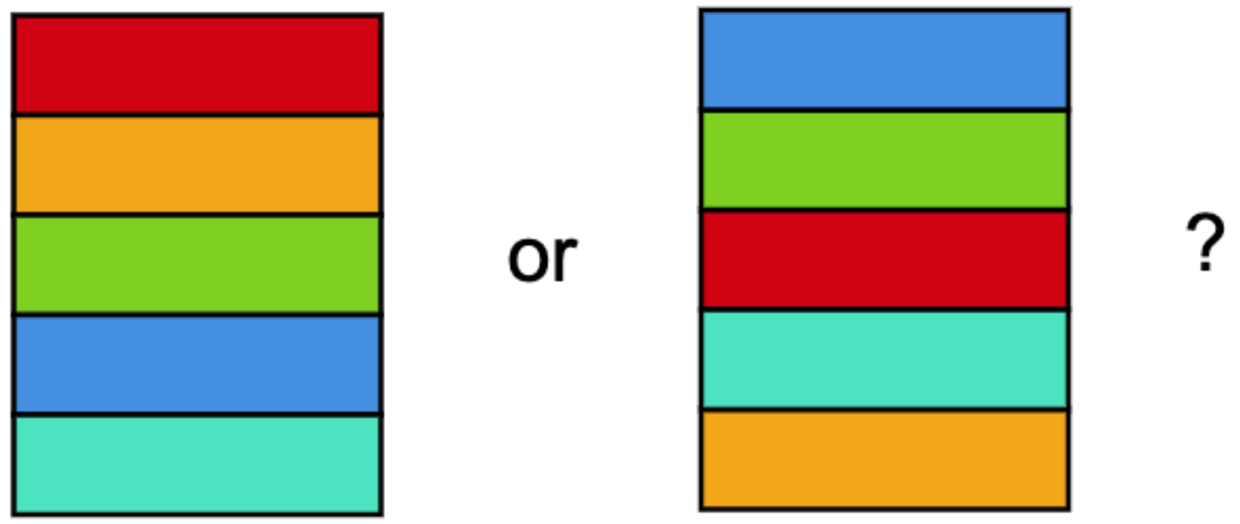
DeepSets, Graph Neural Networks have permutation invariance and equivariance, respectively.
Don't need to make such a choice!

See Battaglia et al., 2018 for a good review on GNNs

Graph Nets



How should we pass this graph to a model?

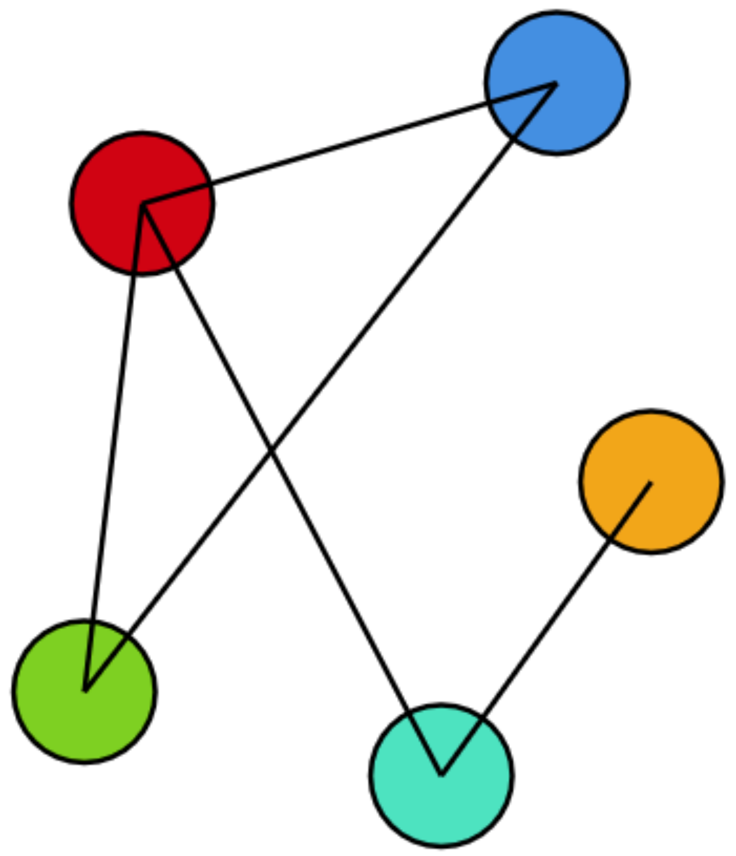


DeepSets, Graph Neural Networks have permutation invariance and equivariance, respectively.
Don't need to make such a choice!

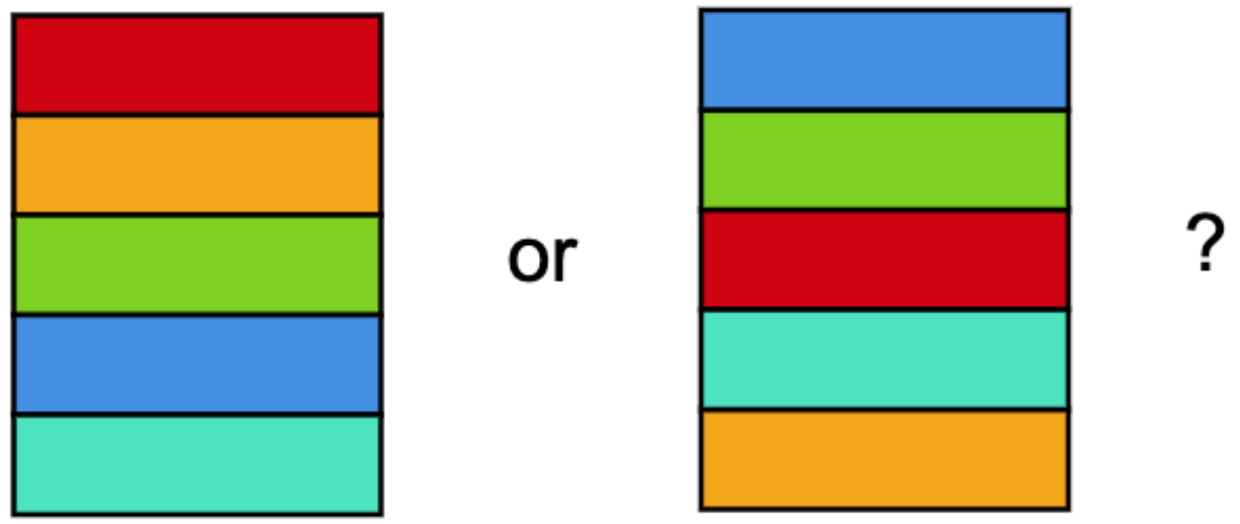
See Battaglia et al., 2018 for a good review on GNNs

- The Universe (mostly) shares this permutation symmetry as well; and many laws are equivariant to exchange of particles.

Graph Nets



How should we pass this graph to a model?

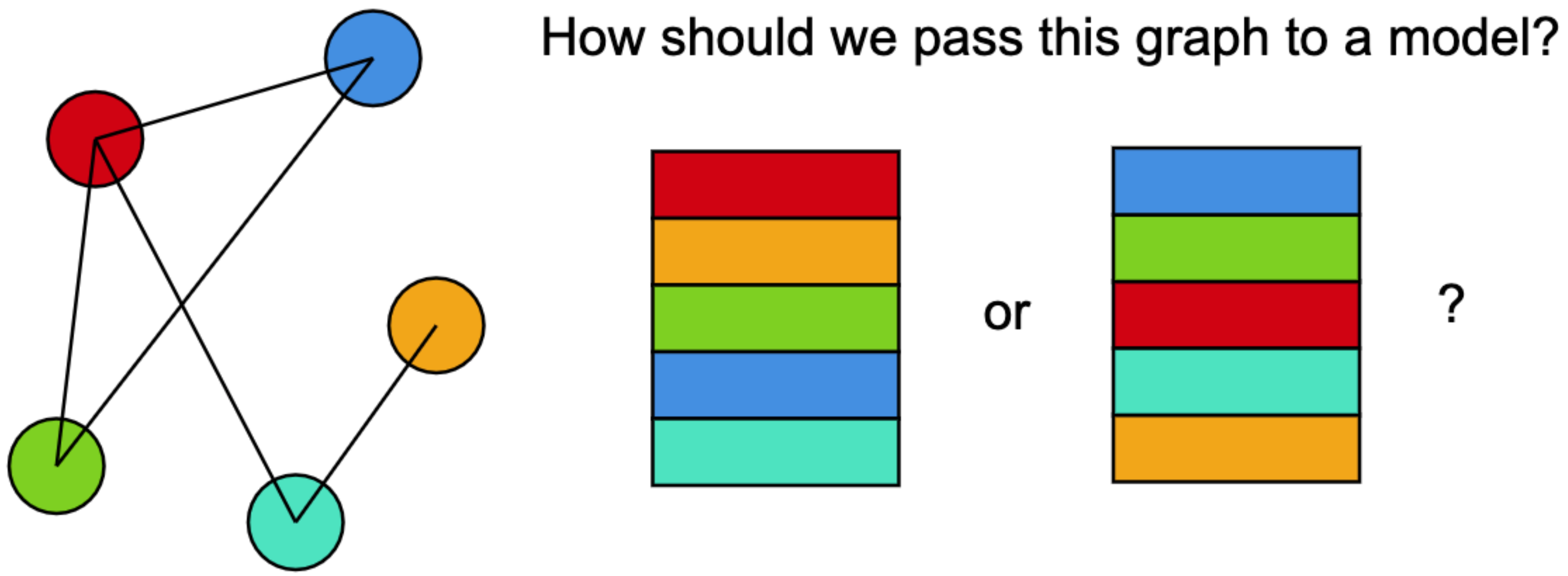


DeepSets, Graph Neural Networks have permutation invariance and equivariance, respectively.
Don't need to make such a choice!

See Battaglia et al., 2018 for a good review on GNNs

- The Universe (mostly) shares this permutation symmetry as well; and many laws are equivariant to exchange of particles.
- Graph Network inductive biases are loosely based on classical mechanics

Graph Nets

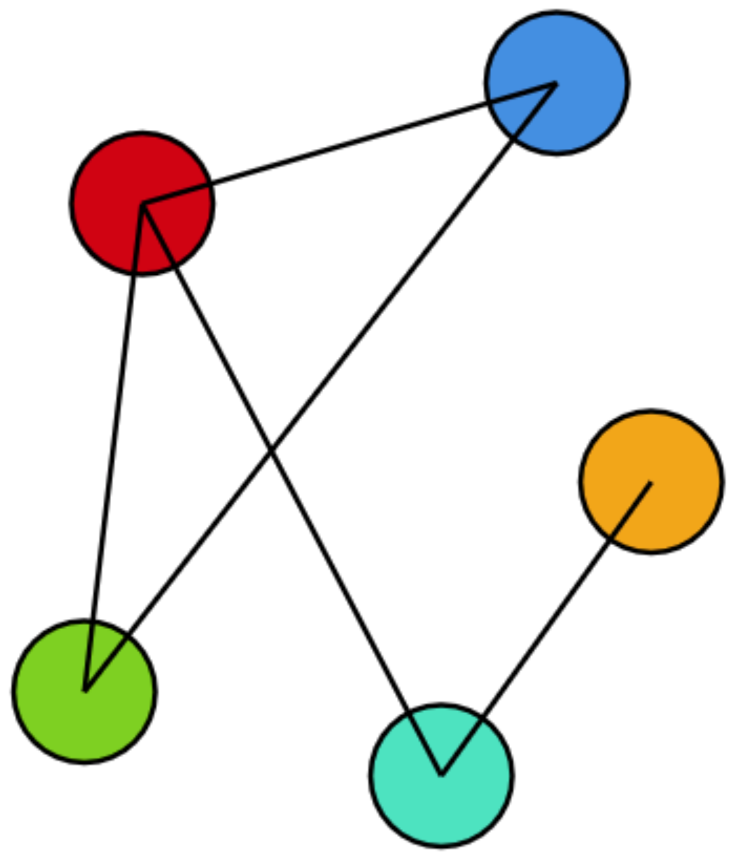


DeepSets, Graph Neural Networks have permutation invariance and equivariance, respectively.
Don't need to make such a choice!

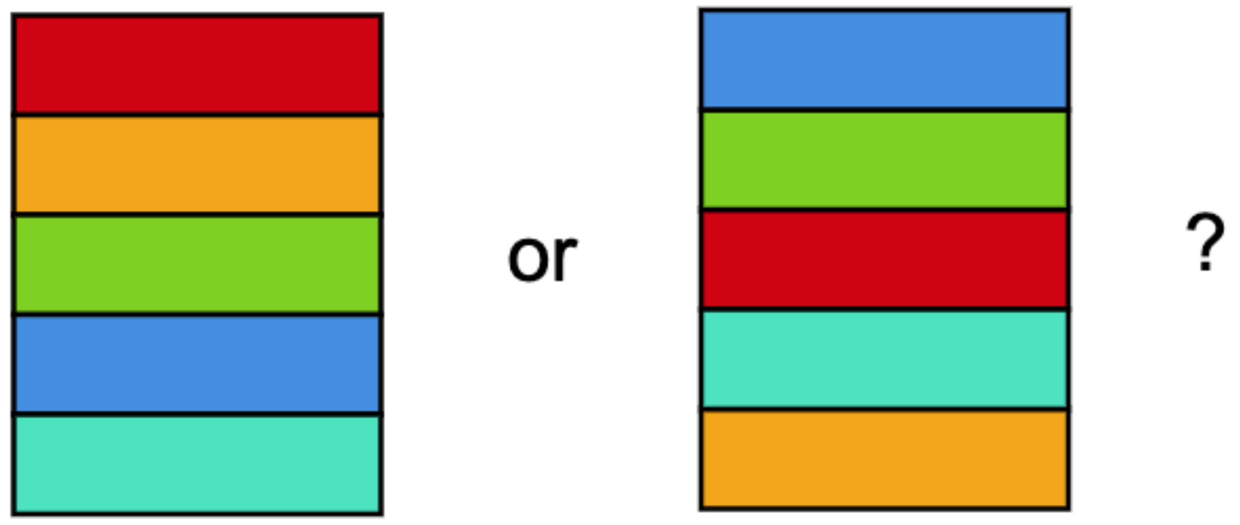
See Battaglia et al., 2018 for a good review on GNNs

- The Universe (mostly) shares this permutation symmetry as well; and many laws are equivariant to exchange of particles.
- Graph Network inductive biases are loosely based on classical mechanics
  - Another example of a formal framework from physics which can be applied to learning!

Graph Nets



How should we pass this graph to a model?

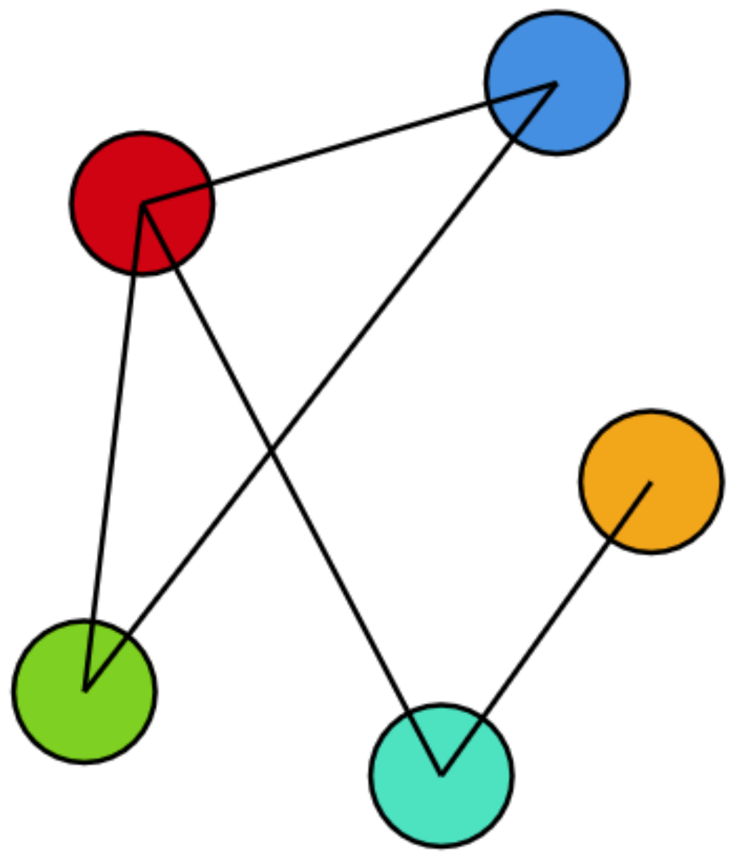


DeepSets, Graph Neural Networks have permutation invariance and equivariance, respectively.
Don't need to make such a choice!

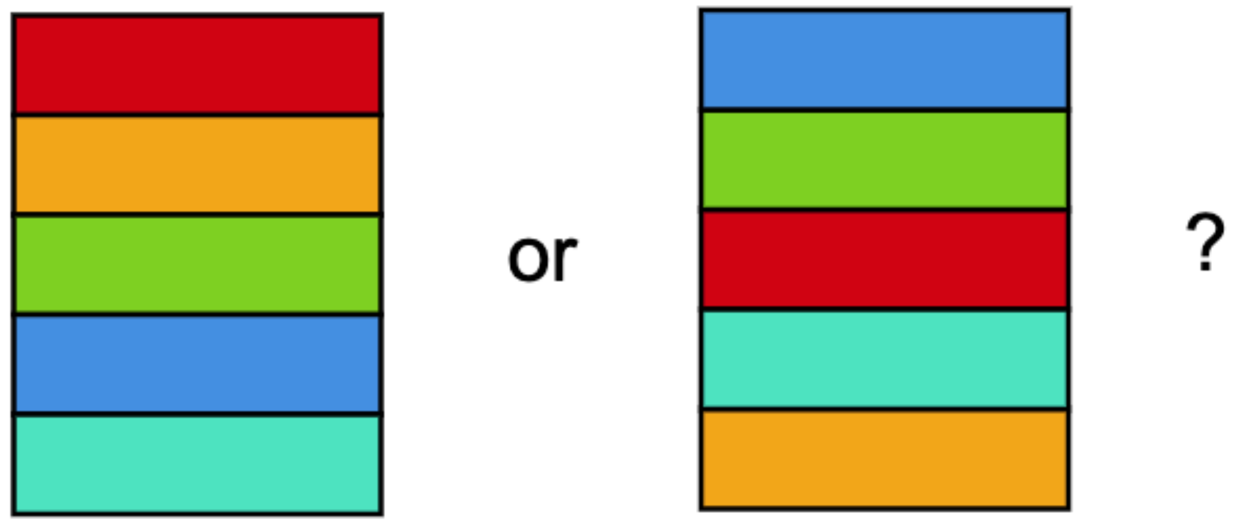
See Battaglia et al., 2018 for a good review on GNNs

- The Universe (mostly) shares this permutation symmetry as well; and many laws are equivariant to exchange of particles.
- Graph Network inductive biases are loosely based on classical mechanics
    - Another example of a formal framework from physics which can be applied to learning!

Graph Nets



How should we pass this graph to a model?



DeepSets, Graph Neural Networks have permutation invariance and equivariance, respectively.
Don't need to make such a choice!

See Battaglia et al., 2018 for a good review on GNNs

- The Universe (mostly) shares this permutation symmetry as well; and many laws are equivariant to exchange of particles.
- Graph Network inductive biases are loosely based on classical mechanics
  - Another example of a formal framework from physics which can be applied to learning!

(Can even exploit this relation to classical mechanics, and distill force laws - see M Cranmer et al., 2020)

For the ultimate book on geometry in deep learning, see geometricdeeplearning.com (Bronstein, Bruna, Cohen, Veličković)

Differential Equations - Neural ODEs

Chen et al., 2018

Differential Equations - Neural ODEs

- Differential equations first created to model the rate of change in physical systems (Newton/Leibniz)

Chen et al., 2018

Differential Equations - Neural ODEs

- Differential equations first created to model the rate of change in physical systems (Newton/Leibniz)

- In a regular Neural ODE, one optimizes a learned function $f(y, t; \theta)$ such to optimize a predictive model:

$$y(t) = y(0) + \int_0^t f(y, \tau; \theta) d\tau$$

Chen et al., 2018

Differential Equations - Neural ODEs

- Differential equations first created to model the rate of change in physical systems (Newton/Leibniz)

- In a regular Neural ODE, one optimizes a learned function $f(y, t; \theta)$ such to optimize a predictive model:

$$y(t) = y(0) + \int_0^t f(y, \tau; \theta) d\tau$$

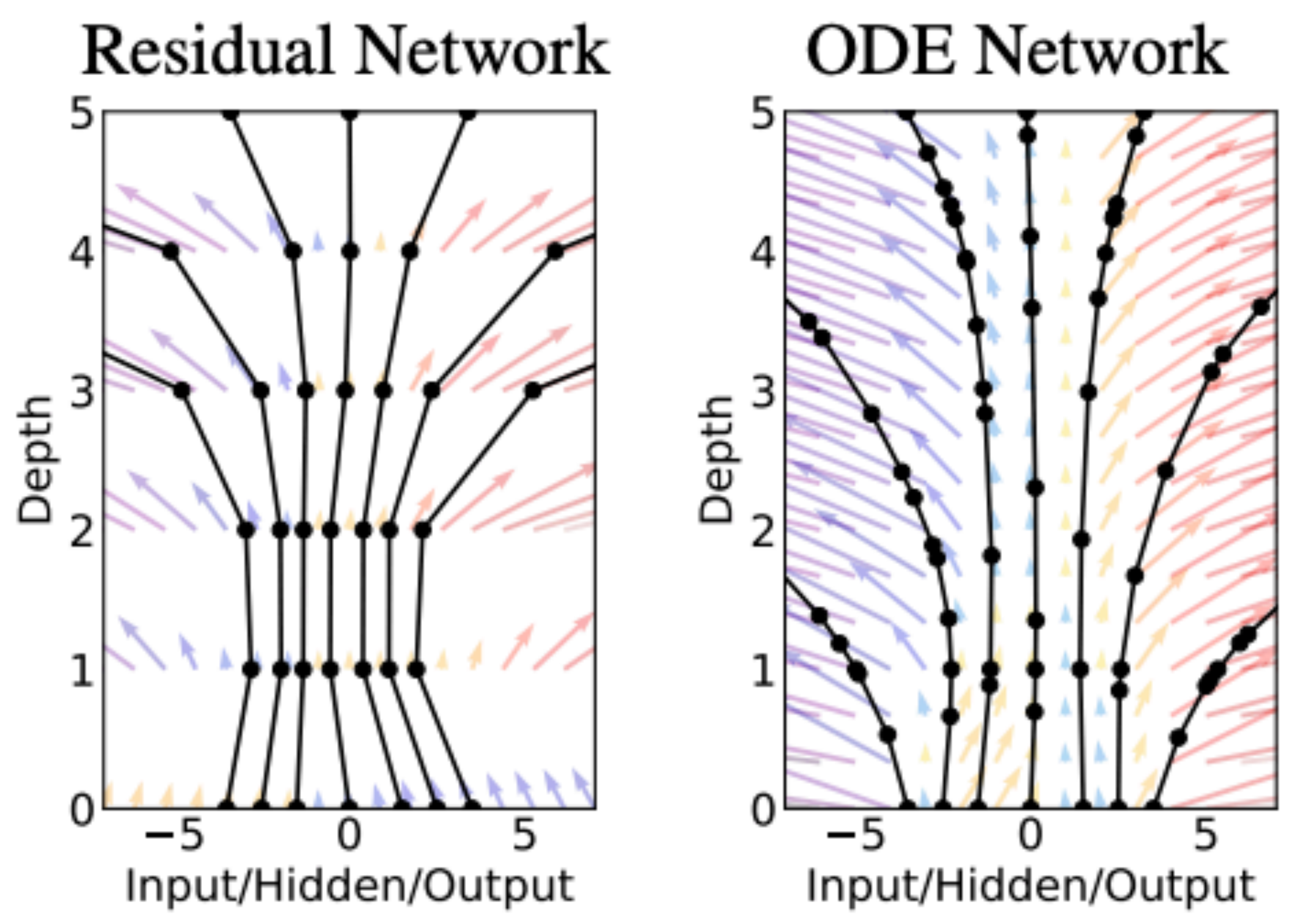


Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

Chen et al., 2018

Differential Equations - Neural ODEs

- Differential equations first created to model the rate of change in physical systems (Newton/Leibniz)

- In a regular Neural ODE, one optimizes a learned function $f(y, t; \theta)$ such to optimize a predictive model:

$$y(t) = y(0) + \int_0^t f(y, \tau; \theta) d\tau$$

- With the obvious applicability to learning time series, can be applied to learning for general problems

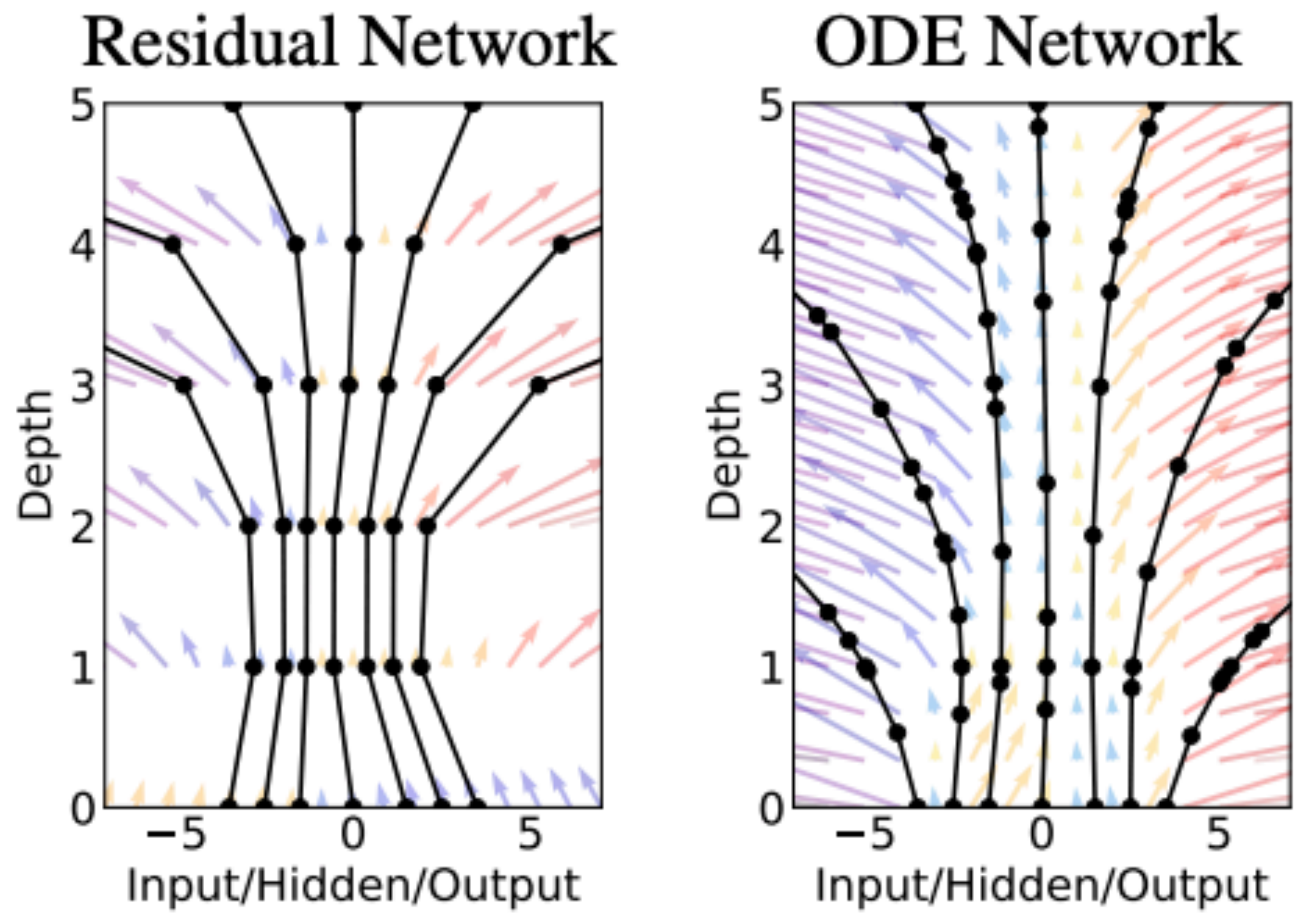


Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

Chen et al., 2018

Differential Equations - PINN

Differential Equations - PINN

- Learn physical variable: $u(x, t; \theta)$, given some observations.

Differential Equations - PINN

- Learn physical variable: $u(x, t; \theta)$, given some observations.

- Explicitly assume u is governed by a specific PDE.

Differential Equations - PINN

- Learn physical variable: $u(x, t; \theta)$, given some observations.
- Explicitly assume u is governed by a specific PDE.
- Regularize the solution u such that is satisfies **both** the PDE and data.

Differential Equations - PINN

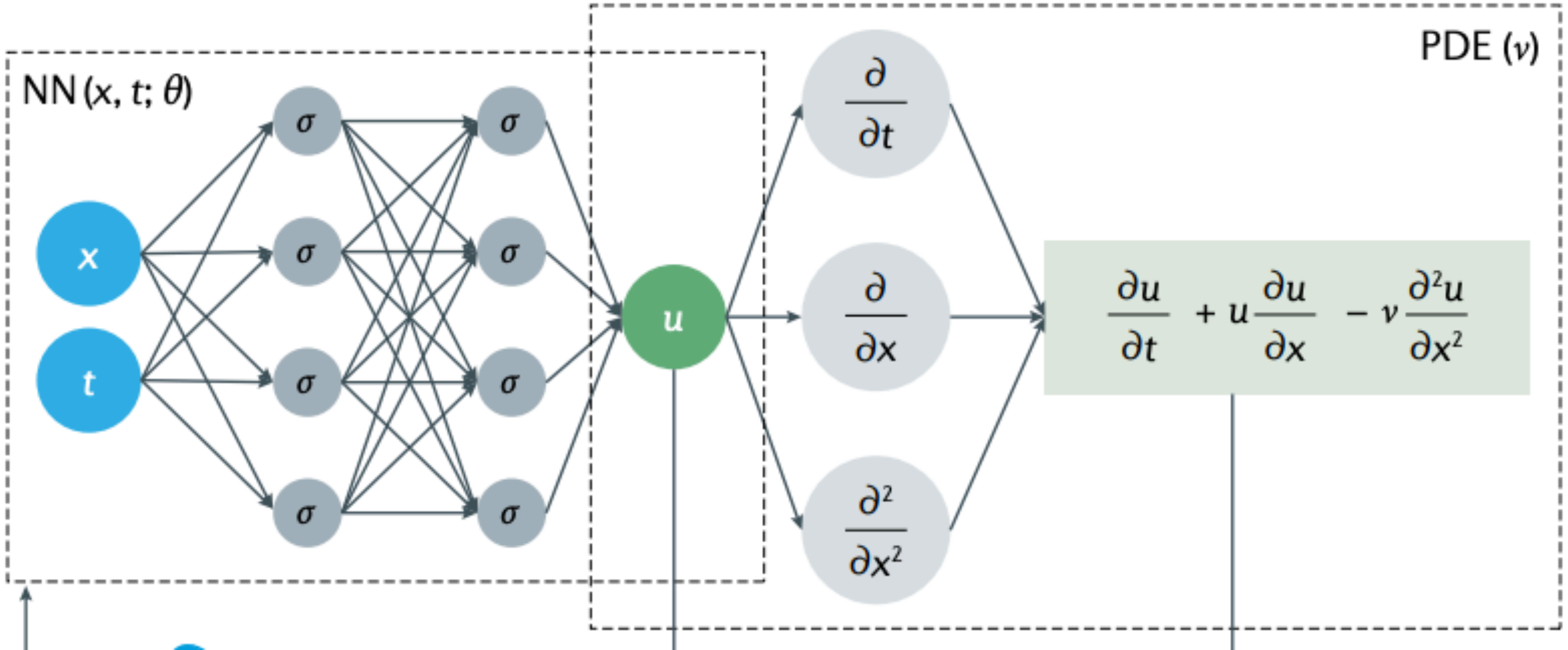
- Learn physical variable: $u(x, t; \theta)$, given some observations.
- Explicitly assume u is governed by a specific PDE.
- Regularize the solution u such that is satisfies **both** the PDE and data.
- Unlike LNN and HNN, this is a **soft** inductive bias.

Differential Equations - PINN

- Learn physical variable: $u(x, t; \theta)$, given some observations.
- Explicitly assume u is governed by a specific PDE.
- Regularize the solution u such that is satisfies **both** the PDE and data.
- Unlike LNN and HNN, this is a **soft** inductive bias.

Truth: $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = v\frac{\partial^2 u}{\partial x^2}$

Compute error in PDE



See Karniadakis, et al., (2021) for a good review.

Soft vs Hard

Soft vs Hard

- Soft: it is difficult for the model to deviate from the inductive bias.

Soft vs Hard

- Soft: it is difficult for the model to deviate from the inductive bias.
  - Data augmentation is a type of soft inductive bias.

Soft vs Hard

- Soft: it is difficult for the model to deviate from the inductive bias.
  - Data augmentation is a type of soft inductive bias.
- Hard: it is **impossible** for the model to deviate.

Soft vs Hard

- Soft: it is difficult for the model to deviate from the inductive bias.
  - Data augmentation is a type of soft inductive bias.
- Hard: it is **impossible** for the model to deviate.
  - e.g., a CNN cannot learn absolute positions

Soft vs Hard

- Soft: it is difficult for the model to deviate from the inductive bias.
    - Data augmentation is a type of soft inductive bias.
- Hard: it is **impossible** for the model to deviate.
    - e.g., a CNN cannot learn absolute positions
- For example, an LNN is a hard constraint on the dynamics, whereas a PINN is a soft constraint.

Soft vs Hard

- Soft: it is difficult for the model to deviate from the inductive bias.
  - Data augmentation is a type of soft inductive bias.
- Hard: it is **impossible** for the model to deviate.
  - e.g., a CNN cannot learn absolute positions
- For example, an LNN is a hard constraint on the dynamics, whereas a PINN is a soft constraint.
- For some inductive biases, hard constraints may be intractable to create. Soft constraints are useful when a symmetry might be slightly violated.

Explicit vs Implicit

Explicit vs Implicit

- Explicit: an inductive bias created to define a particular functional prior.

Explicit vs Implicit

- Explicit: an inductive bias created to define a particular functional prior.
- Implicit: an inductive bias is present which was not intended.

Explicit vs Implicit

- Explicit: an inductive bias created to define a particular functional prior.
- Implicit: an inductive bias is present which was not intended.
    - e.g., large learning rates and small batch sizes define an implicit regularization term (e.g., Sam Smith et al., 2021 and references therein)

Explicit vs Implicit

- Explicit: an inductive bias created to define a particular functional prior.
- Implicit: an inductive bias is present which was not intended.
  - e.g., large learning rates and small batch sizes define an implicit regularization term (e.g., Sam Smith et al., 2021 and references therein)
- Generally, it seems that making an inductive bias **explicit** in a formal framework, such as physics, leads to new insights, and allows one to use existing methods. Also allows one to control it.

General vs Application-Specific

General vs Application-Specific

- General: an inductive bias that can be used for many different problems

General vs Application-Specific

- General: an inductive bias that can be used for many different problems
- Application-specific: an inductive bias created for a particular physical problem

General vs Application-Specific

- General: an inductive bias that can be used for many different problems
- Application-specific: an inductive bias created for a particular physical problem
  - For example, a PINN’s inductive bias is the ODE describing the underlying data; whereas some Neural ODE regularizations are very general (e.g., J Kelly et al., 2020 and C Finlay et al., 2020)

Summary

Summary

- Many successful inductive biases in deep learning are explicitly or implicitly informed by physics. Additional insights can be gained when making this connection explicit!

Summary

- Many successful inductive biases in deep learning are explicitly or implicitly informed by physics. Additional insights can be gained when making this connection explicit!

- One should directly consider inductive biases, and what choices to make, given the following categories:

Summary

- Many successful inductive biases in deep learning are explicitly or implicitly informed by physics. Additional insights can be gained when making this connection explicit!

- One should directly consider inductive biases, and what choices to make, given the following categories:

    - Explicit vs Implicit

Summary

- Many successful inductive biases in deep learning are explicitly or implicitly informed by physics. Additional insights can be gained when making this connection explicit!
- One should directly consider inductive biases, and what choices to make, given the following categories:
  - Explicit vs Implicit
  - General vs Application-Specific

Summary

- Many successful inductive biases in deep learning are explicitly or implicitly informed by physics. Additional insights can be gained when making this connection explicit!
- One should directly consider inductive biases, and what choices to make, given the following categories:
  - Explicit vs Implicit
  - General vs Application-Specific
  - Hard vs Soft

Code tutorial

https://astroautomata.com/inductive_biases_tutorial.html