Symmetries and ML

Andreas Ipp

Institute for Theoretical Physics, TU Wien, Austria

SLAC Summer Institute (SSI 2023)

August 18, 2023

INSTITUTE FOR THEORETICAL PHYSICS

$\int dk\Pi$

Der Wissenschaftsfonds.

Symmetries

Symmetry in physics: a property that remains unchanged under some transformation.

Discrete symmetries
- finite groups, permutation group
  (Graph neural network, Transformer)
- parity / mirror
- time inversion

Continuous symmetries
- Lie groups

Global symmetries
- translation (Convolutional neural network)
- rotation (Group-equivariant CNN, Steerable CNN)
- time translation (Recurrent neural network)

Local symmetries
- gauge symmetries
  (Lattice gauge equivariant CNN)

Noether’s theorem

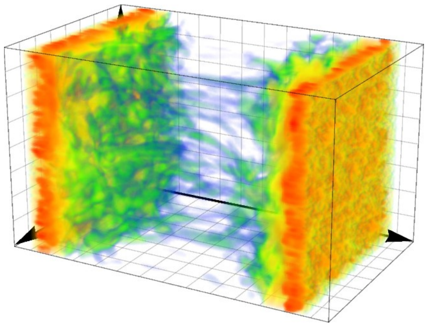
Every continuous symmetry of the action corresponds to a conservation law.

Symmetry	Conservation law
Translation in space	Conservation of momentum (CNN)
Translation in time	Conservation of energy (Hamiltonian NN, Lagrangian NN)
Rotation in space	Conservation of angular momentum (G-CNN, Steerable CNN)
Gauge invariance	Conservation of charge (L-CNN, equivariant coupling layers)

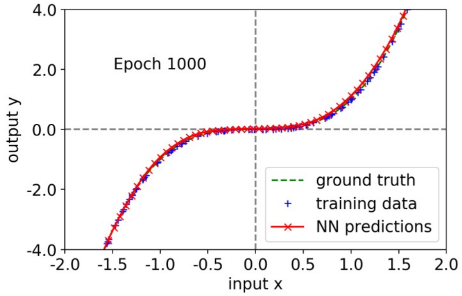
Emmy Noether
(source: Wikipedia)

Outline

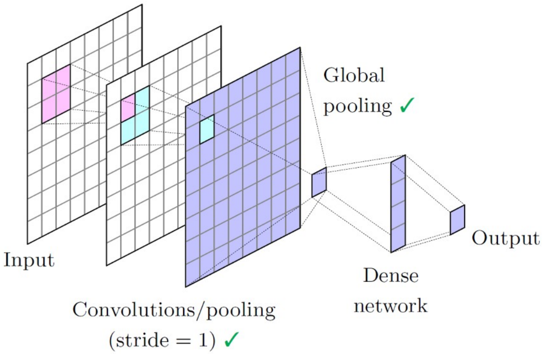
Motivation



Toy model

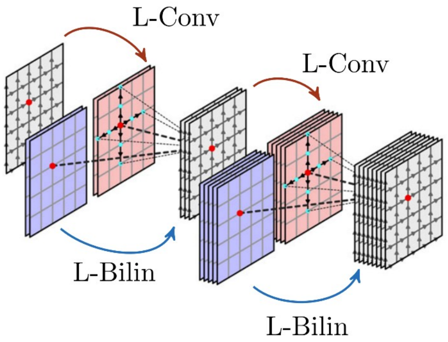


Translational symmetry



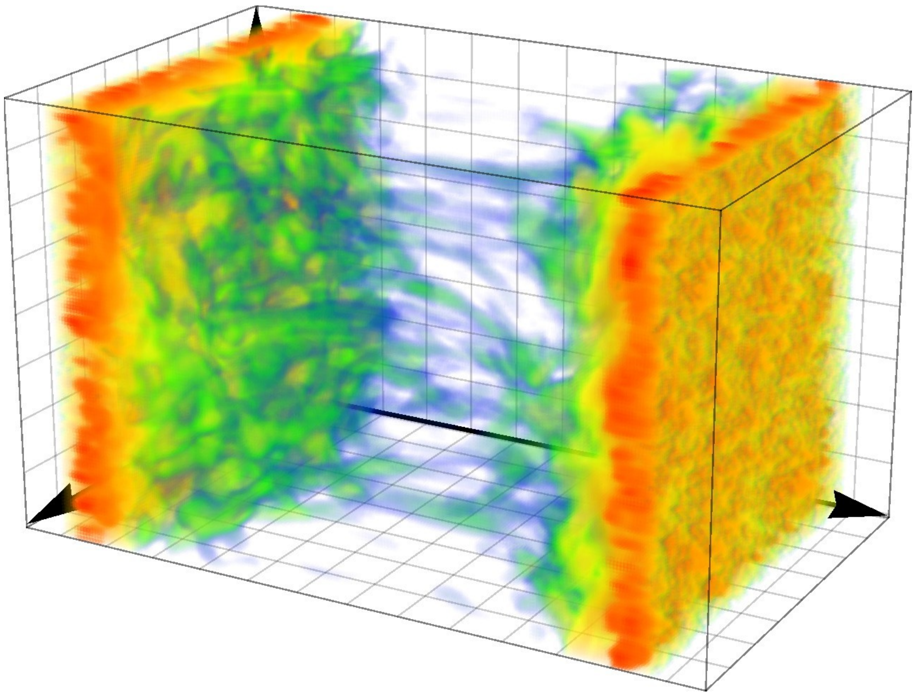
Bulusu, Favoni, AI, Müller, Schuh, Phys. Rev. D 104 (2021) 074504

Lattice gauge symmetry

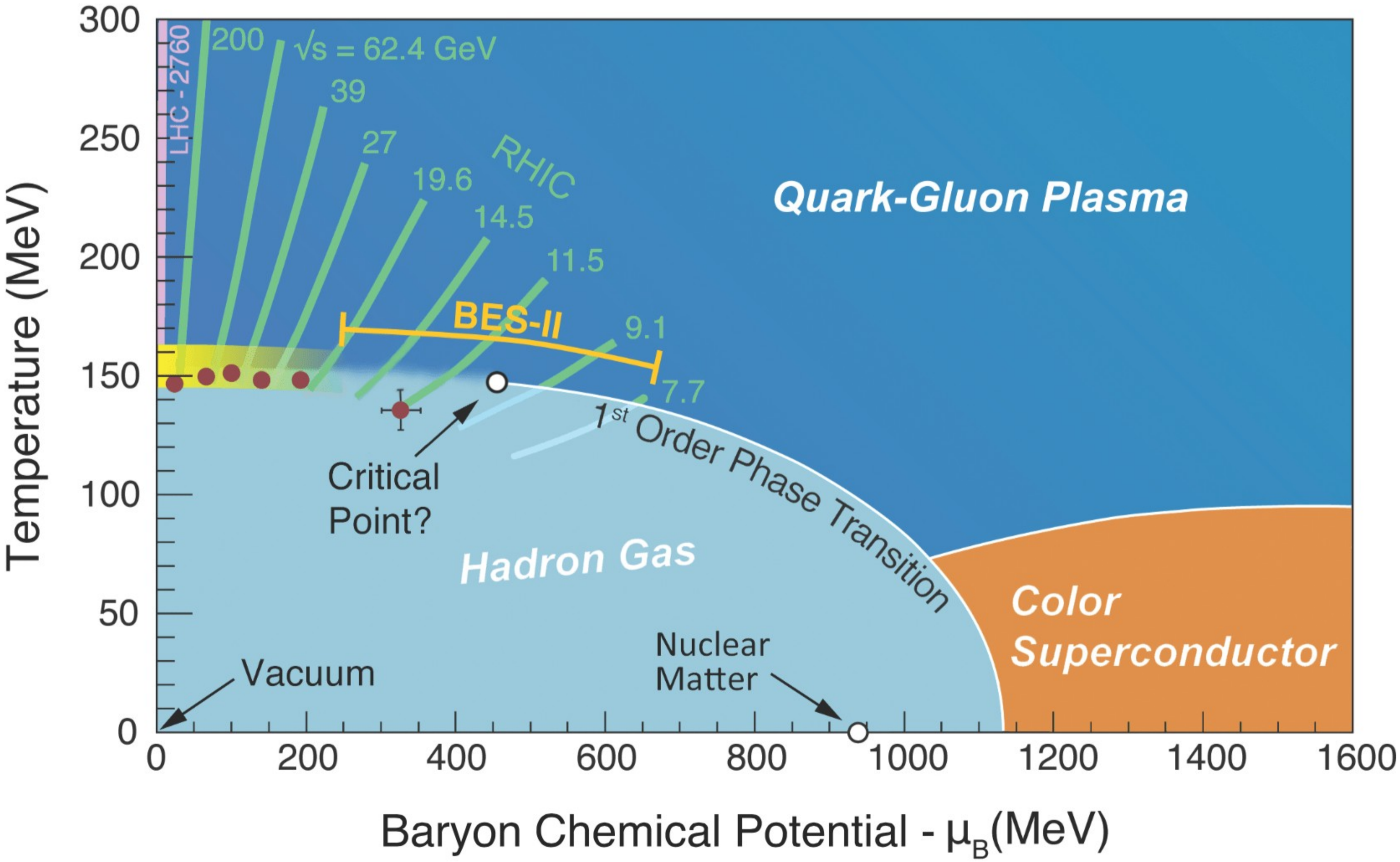


Favoni, AI, Müller, Schuh, Phys.Rev.Lett. 128 (2022) 032003

Motivation



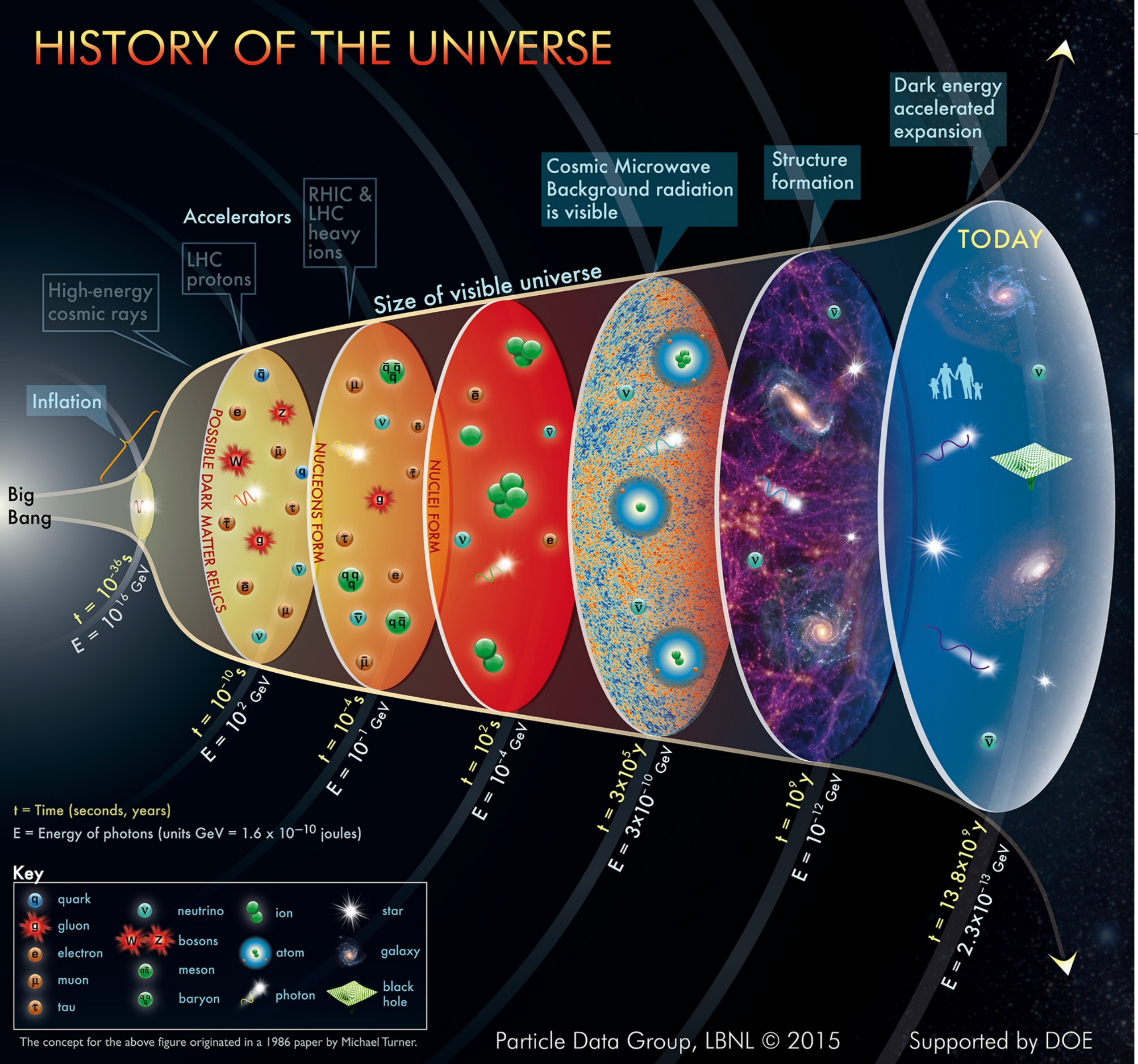
QCD phase diagram



Sun (surface):
6000°C ≈ 0.5 eV

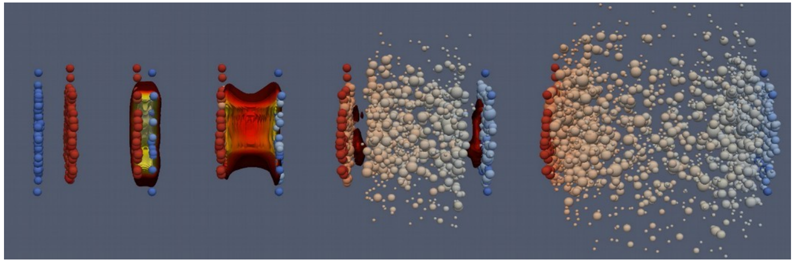
Sun (core):
15 million °C ≈ 1.3 keV

Quark-Gluon Plasma:
1.7×10^{12} °C ≈ 150 MeV

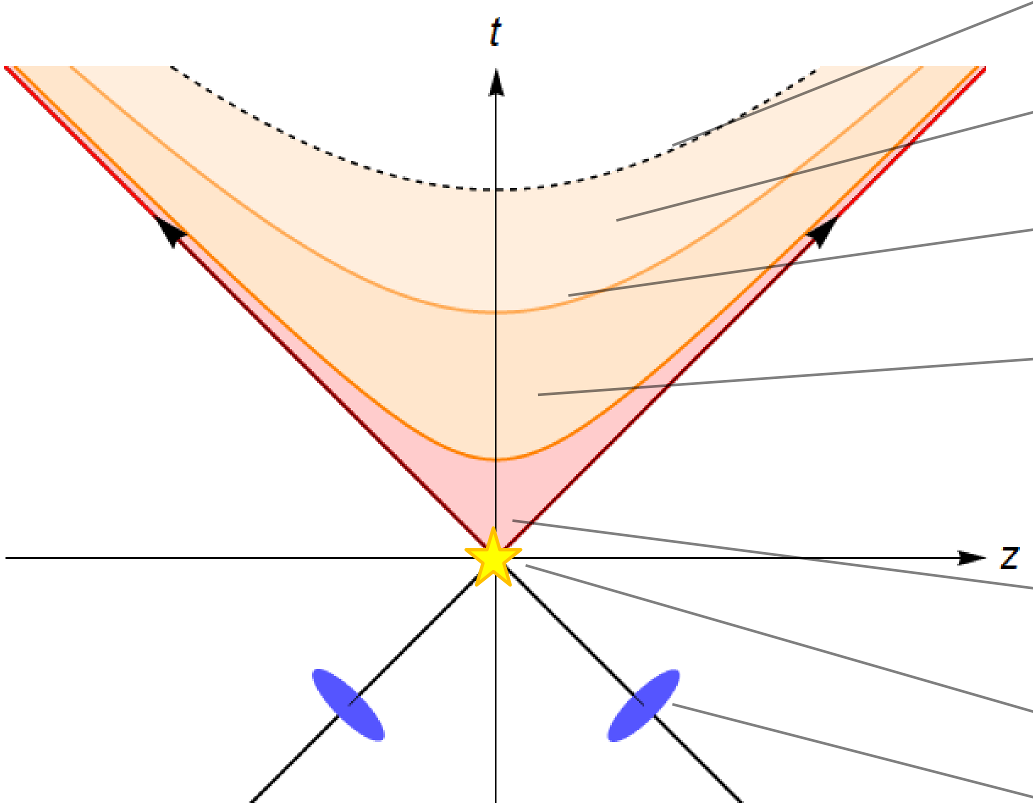


Quark-gluon plasma

- Existed in the early universe
- Produced in heavy ion collisions



Stages of a heavy-ion collision



Freeze-out (τ ≈ 15 fm/*c*):
interactions stop

Hadronic gas (τ ≈ 10 - 15 fm/*c*)**:** hadrons
(kinetic transport theory)

Hadronization (τ ≈ 10 fm/*c*):
confinement transition → hadron formation

QGP (τ ≈ 1 - 10 fm/*c*)**:** quarks and gluons
(relativistic viscous hydrodynamics)

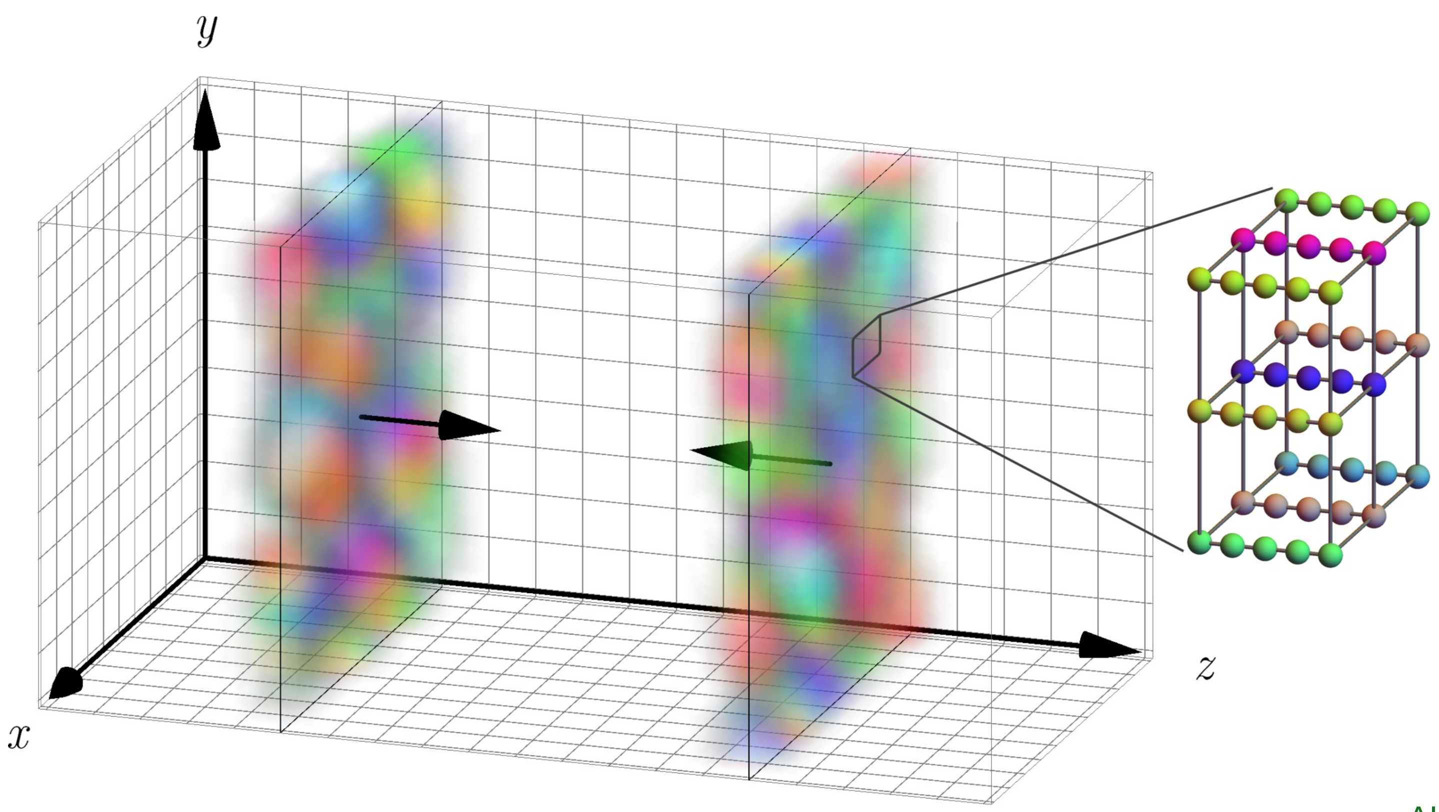
Glasma (τ ≈ 0 - 1 fm/*c*)**:** quasi-classical fields
(classical field equations)

Collision event

Initial state: Lorentz-contracted nuclei
(color glass condensate)

$$1\,\mathrm{fm}/c \approx 3.3\cdot 10^{-24}\,\mathrm{s} \approx 3.3\,\mathrm{ys}$$

Colored particle-in-cell method



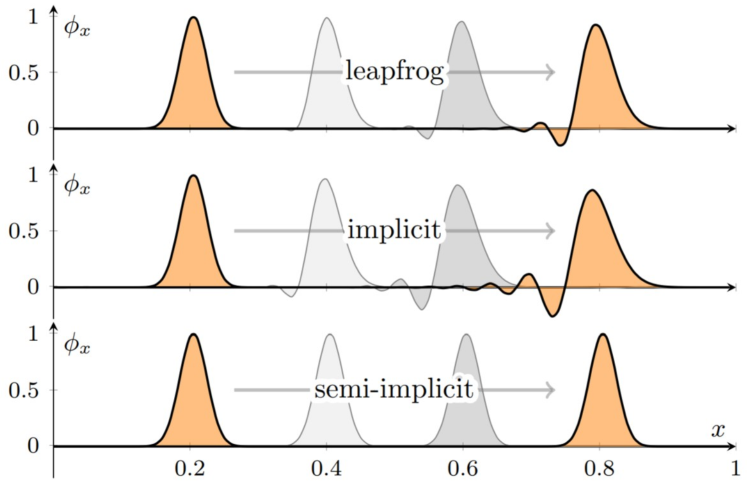
Generalization of the particle-in-cell method from plasma physics for strong interactions.

[A. Dumitru, Y. Nara, M. Strickland: PRD75:025016 (2007)]

Based on real-time lattice gauge theory in a classical regime.

AI, D. Müller, Phys. Lett. B 771 (2017) 74

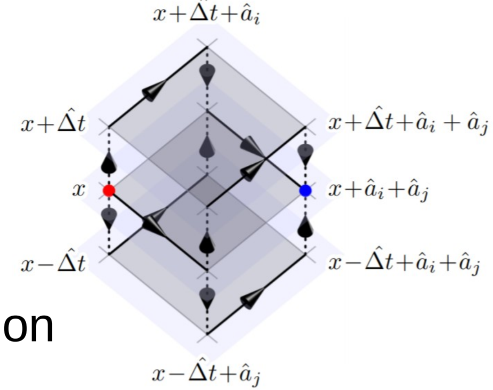
Dispersion-free propagation



Standard **Wilson action**:

$$S[U]=\frac{V}{g^2}\sum_x\left(\sum_i\frac{1}{(a^0a^i)^2}\,\mathrm{tr}\left(2-U_{x,0i}-U^\dagger_{x,0i}\right)-\frac{1}{2}\sum_{i,j}\frac{1}{(a^ia^j)^2}\,\mathrm{tr}\left(2-U_{x,ij}-U^\dagger_{x,ij}\right)\right)$$

Variational integrator: Discretized equations of motion from discretized action



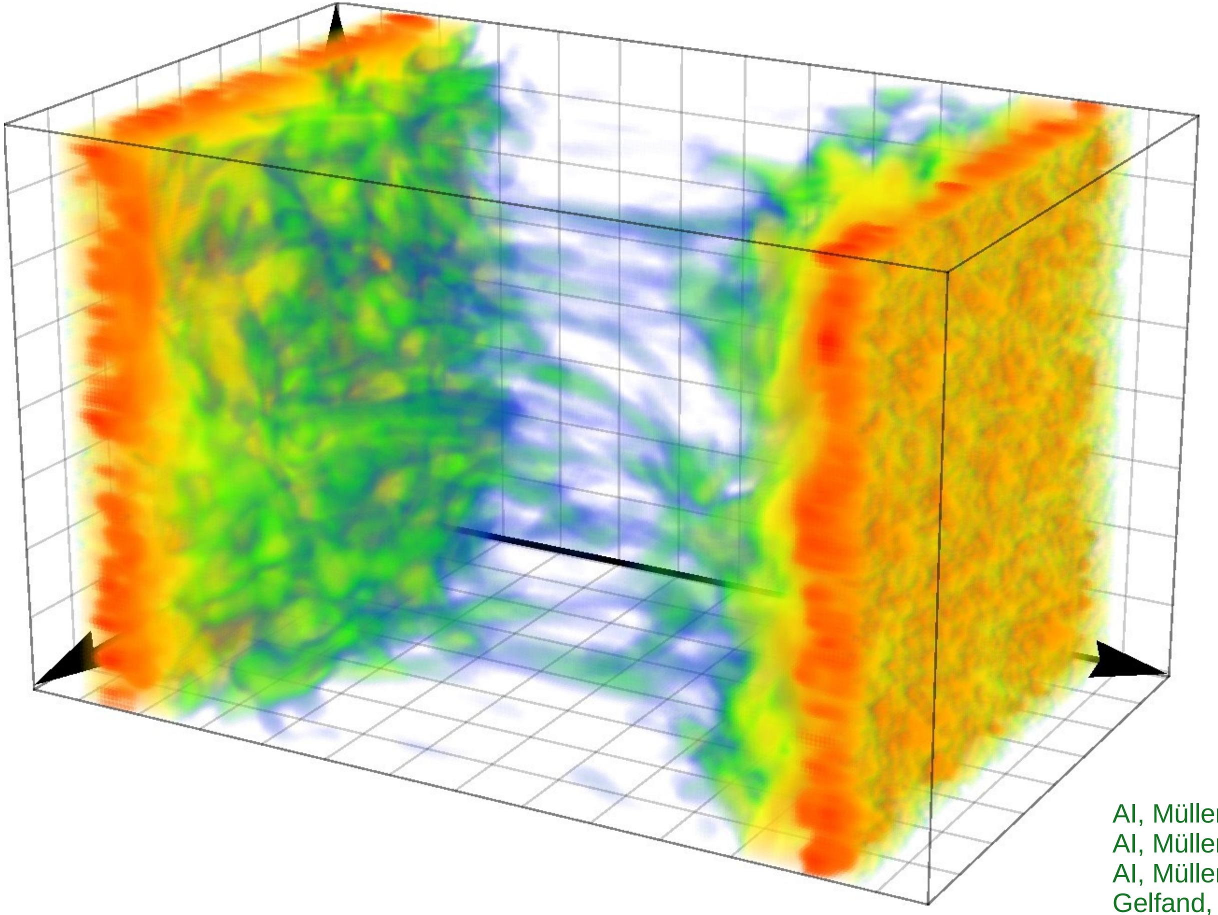
Discretized action for the **semi-implicit scheme**:

$$S[U]=\frac{V}{g^2}\sum_x\left(\frac{1}{(a^0a^1)^2}\,\mathrm{tr}\left(C_{x,01}C^\dagger_{x,01}\right)+\sum_i\frac{1}{(a^0a^i)^2}\,\mathrm{tr}\left(C_{x,0i}C^\dagger_{x,0i}\right)\right.$$
$$\left.-\underbrace{\frac{1}{4}\sum_{i,|j|}\frac{1}{(a^ia^j)^2}\,\mathrm{tr}\left(C_{x,ij}M^\dagger_{x,ij}\right)}_{\text{implicit part}}-\underbrace{\frac{1}{4}\sum_{|j|}\frac{1}{(a^1a^j)^2}\,\mathrm{tr}\left(C_{x,1j}W^\dagger_{x,1j}+\mathrm{h.c.}\right)}_{\text{semi-implicit part}}\right)$$

with $C_{x,\mu\nu}\equiv U_{x,\mu}U_{x+\mu,\nu}-U_{x,\nu}U_{x+\nu,\mu}$

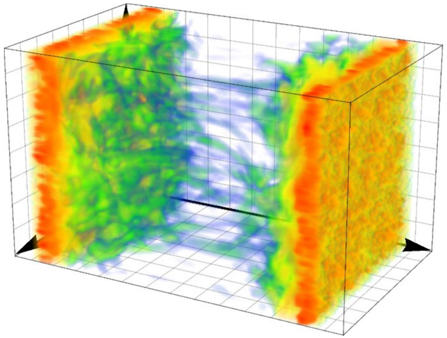
For details see:
AI, D. Müller, Eur.Phys.J. C78 (2018) no.11, 884

Simulations of the collision process



AI, Müller, Eur.Phys.J.A 56 (2020) 9, 243
AI, Müller, Eur.Phys.J. C78 (2018) no.11, 884
AI, Müller, Phys. Lett. B 771 (2017) 74
Gelfand, AI, Müller, Phys. Rev. D94 (2016) no.1, 014020

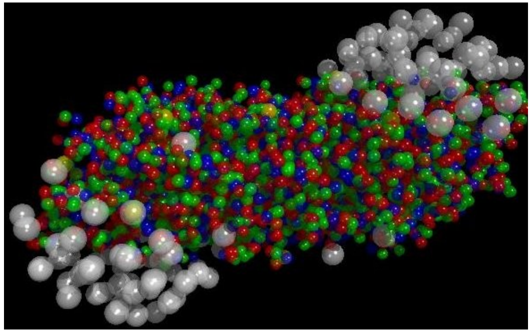
Computational challenges



Simulating small part of nuclei
at RHIC energies:

γ-factor: 100
Lattice: 2048×192^2 cells
Gauge group: SU(2)
Color sheets: 1
Simulation box: $(6\ \mathrm{fm})^3$

→ **25 GB** simulation data
→ **192 core hours** on
Vienna Scientific Cluster (VSC-3)



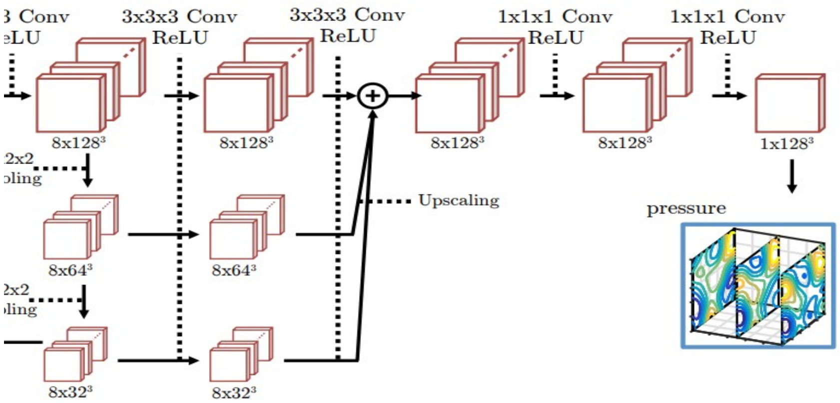
Simulating realistic off-central full size nuclei
at LHC energies:

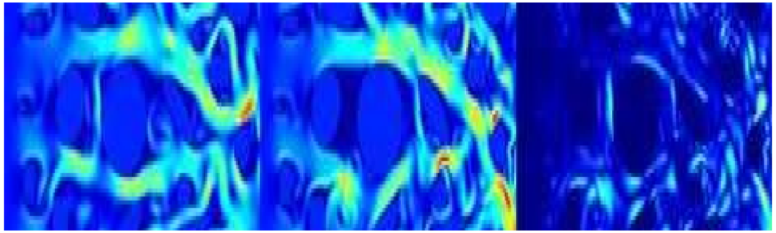
γ-factor: 2500
Lattice: $(25\times20480) \times 1920^2$ cells
Gauge group: SU(3)
Color sheets: 100
Simulation box: $(60\ \mathrm{fm})^3$

→ **25 PB** simulation data
→ **5 million core years** on VSC-3
(2020: 150 years on VSC-3; but only 130 TB RAM available)
(2023: 55 years on VSC-5; 355 TB RAM available)

Machine learning in fluid dynamics

Accelerating Eulerian Fluid Simulation With Convolutional Networks
Tompson et al, arxiv:1607.03597



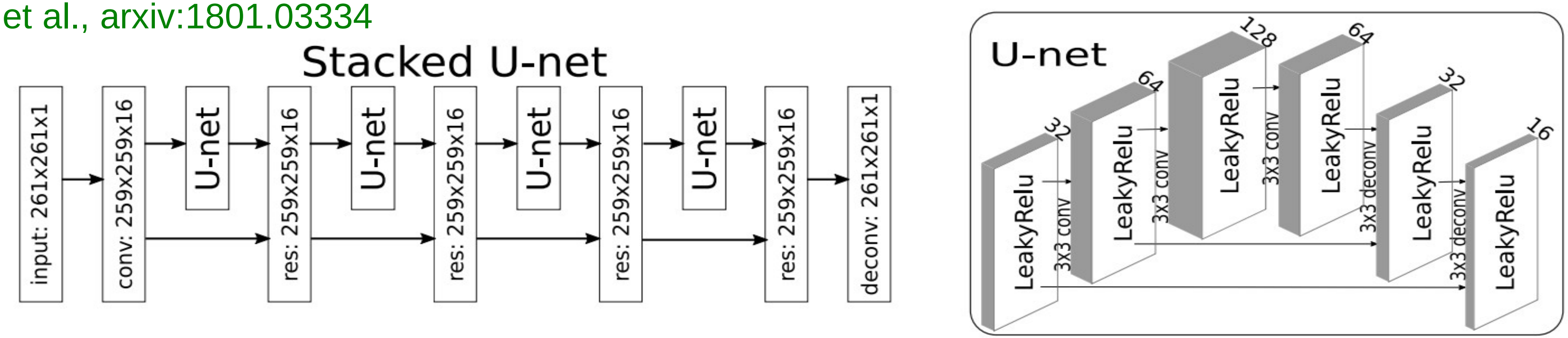


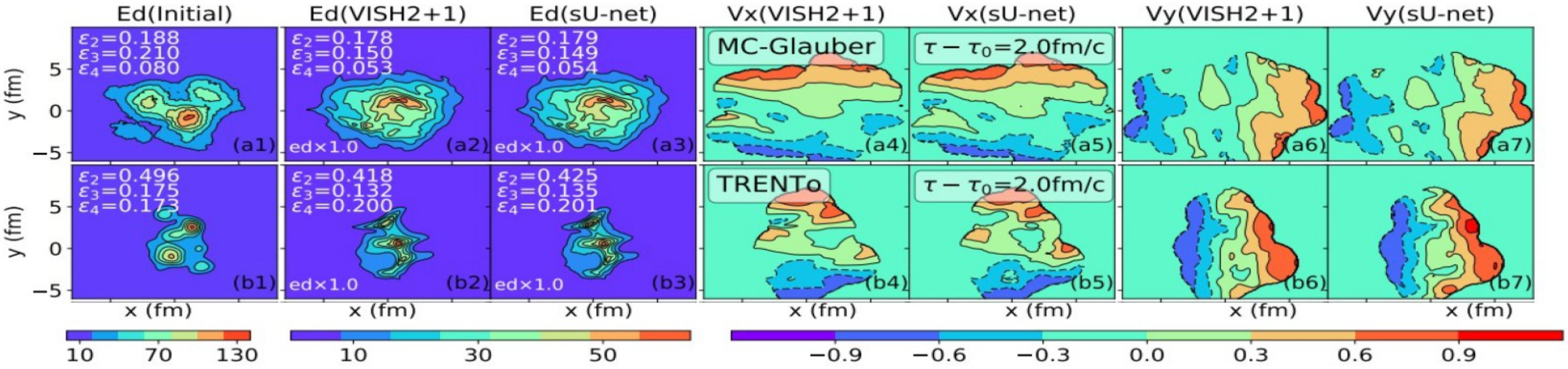
- Compress computation time and memory usage
- Use convolutional autoencoders to compress state size
- Learn dynamics on compressed form
- Can generalize to larger grid sizes

Lat-Net: Compressing Lattice Boltzmann Flow Simulations using Deep Neural Networks
Hennigh, arxiv:1705.09036

Relativistic hydrodynamics

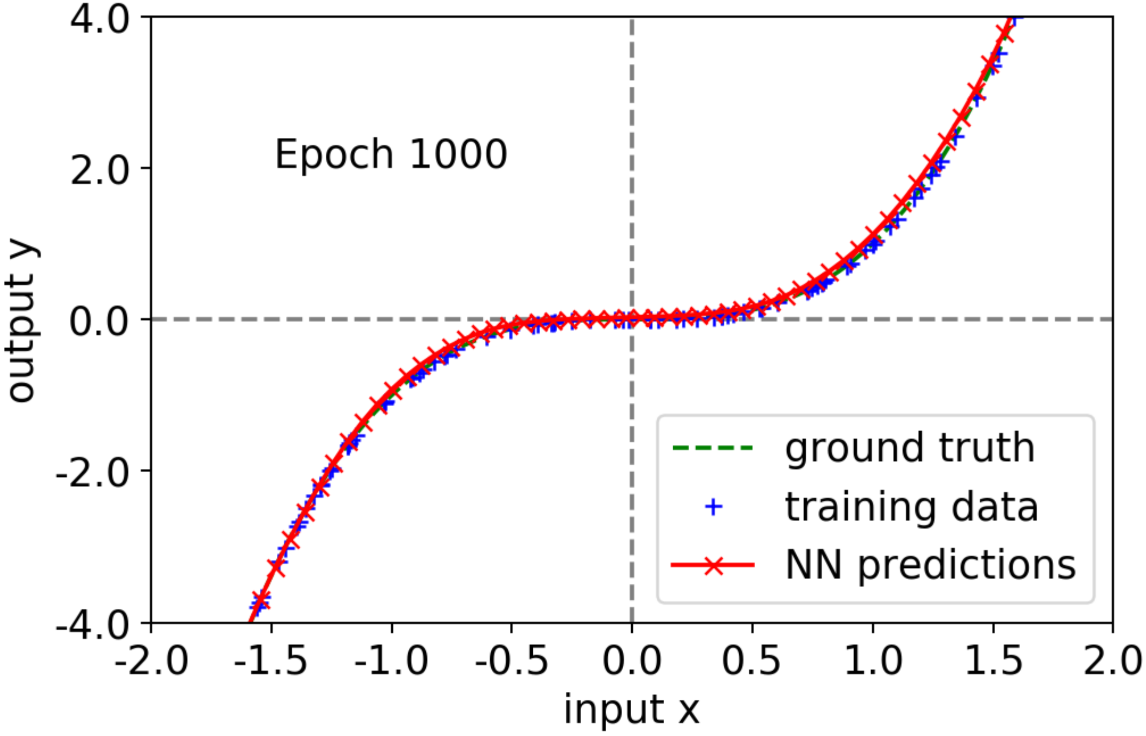
Applications of deep learning to relativistic hydrodynamics
Huang et al., arxiv:1801.03334





- Speed up simulation time from 20 min to few seconds
- Network had to be trained with 10,000 initial and final state pairs

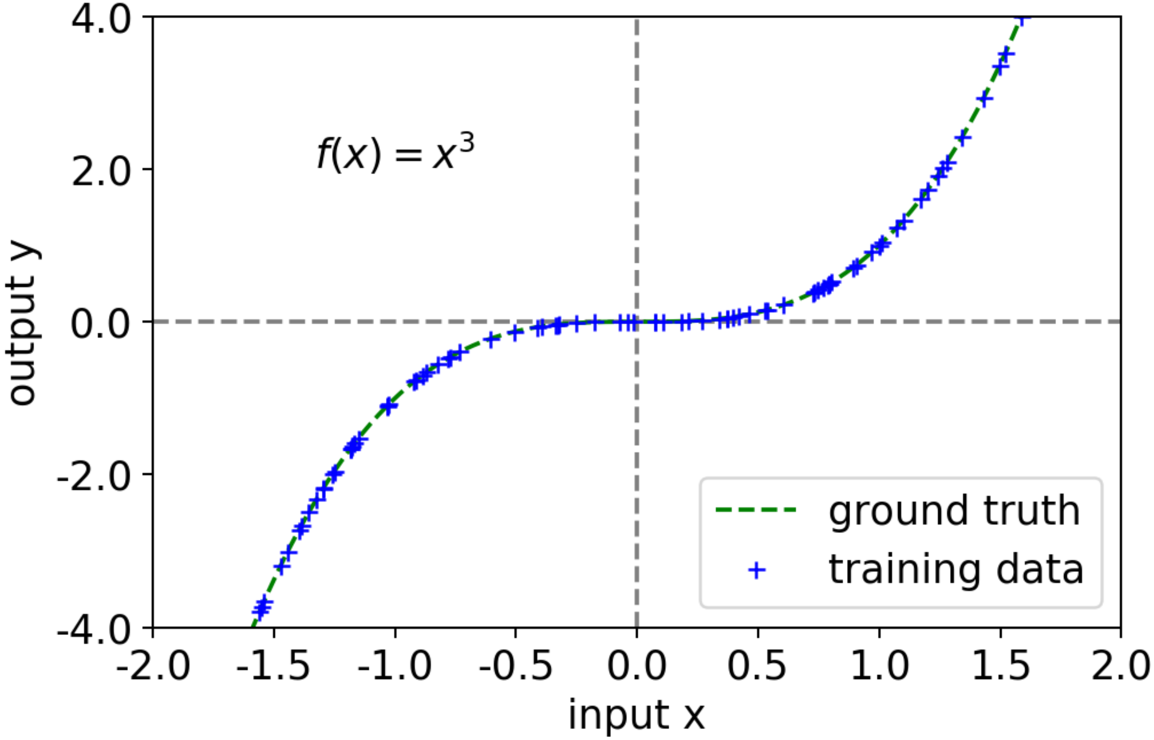
Toy model



Epoch 1000
output y
input x
4.0
2.0
0.0
-2.0
-4.0
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
ground truth
training data
NN predictions

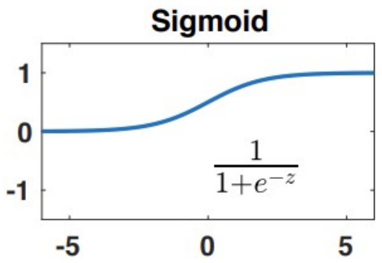
Learning a simple function

Learn $f(x) = x^3$



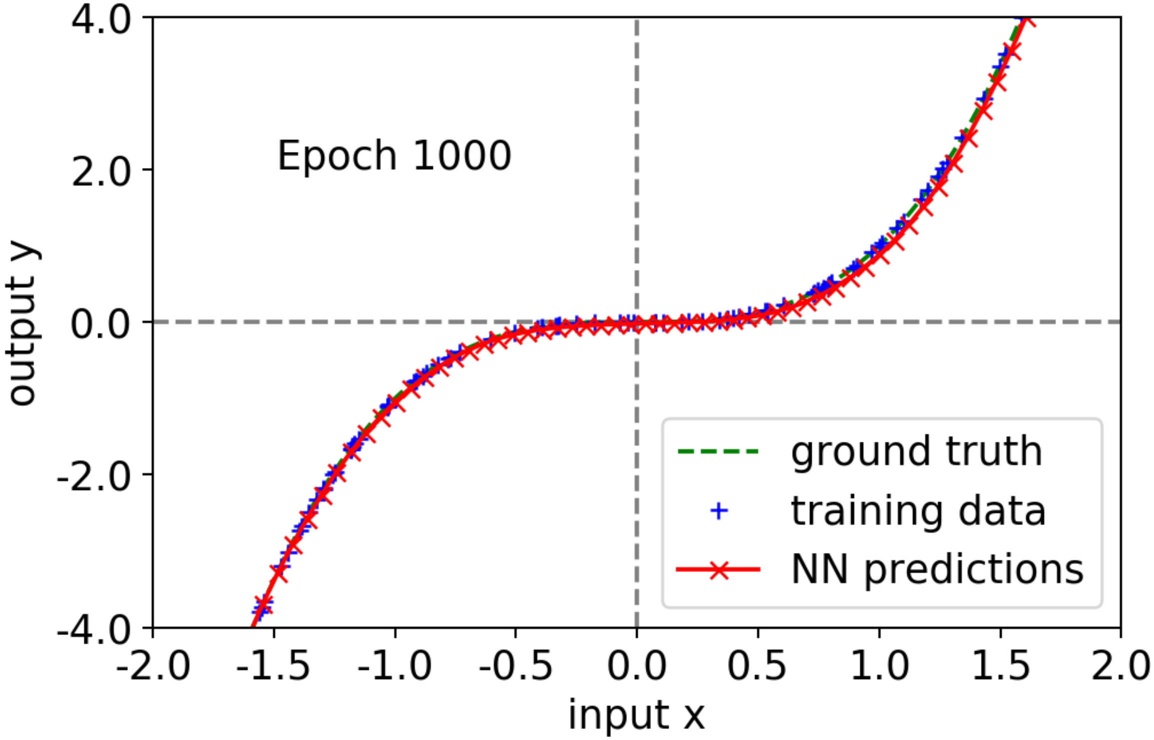
Example in PyTorch

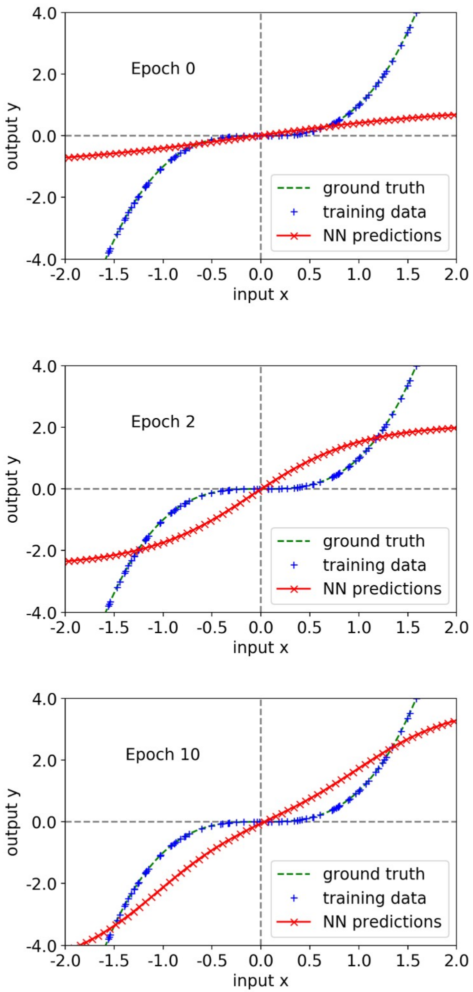
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.l1 = nn.Linear(1, 128)
        self.l2 = nn.Linear(128, 32)
        self.l3 = nn.Linear(32, 1)

    def forward(self, x):
        x = torch.sigmoid(self.l1(x))
        x = torch.sigmoid(self.l2(x))
        x = self.l3(x)
        return x
```



Learning a simple function

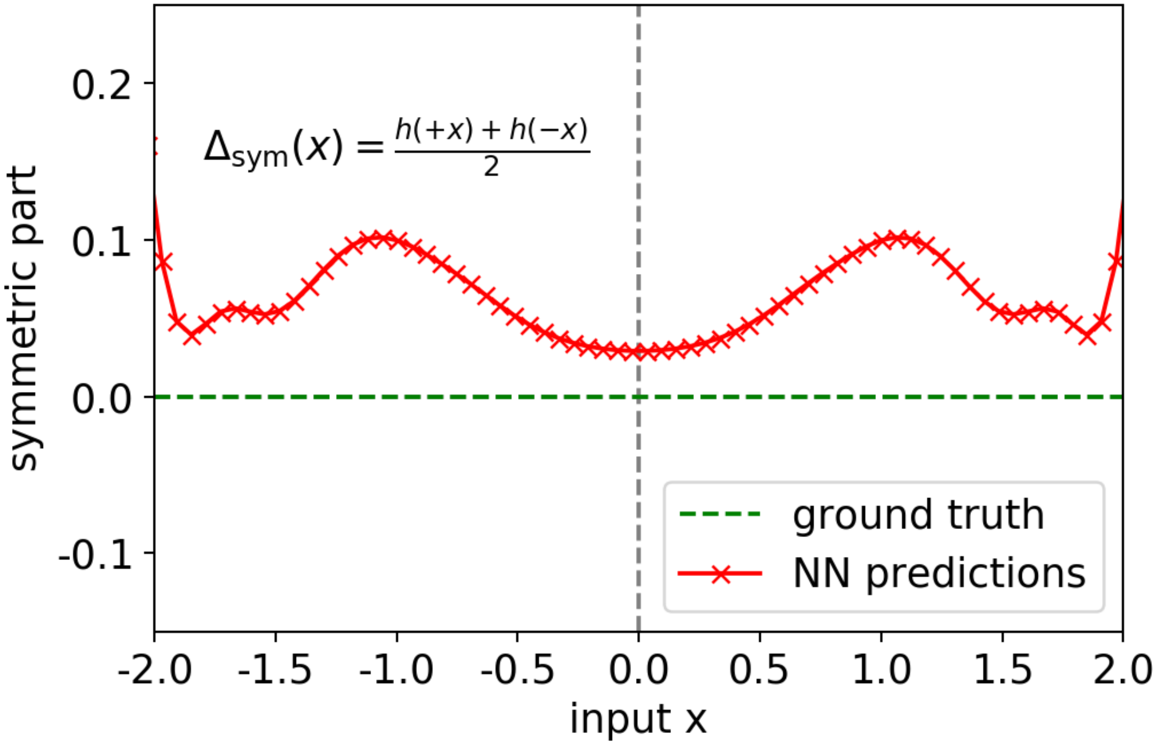
Learn $f(x) = x^3$

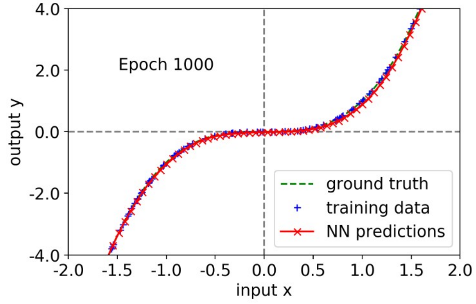




Learning a simple function

Learn $f(x) = x^3$



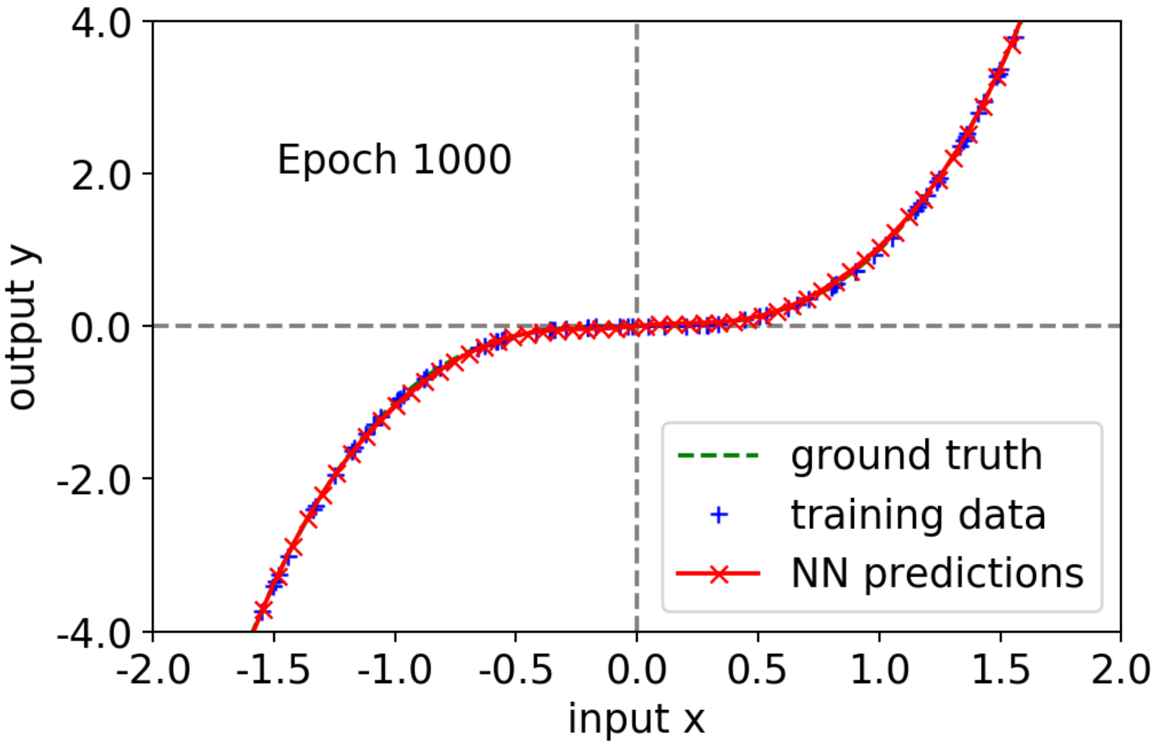


Symmetry: $f(-x) = -f(x)$

learned approximately,
but small deviation remains

Imposing the symmetry

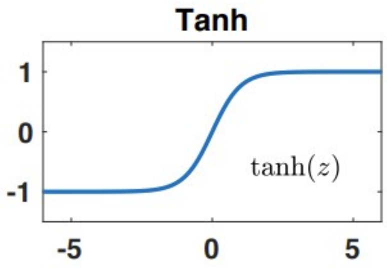
Learn $f(x) = x^3$



Example in PyTorch

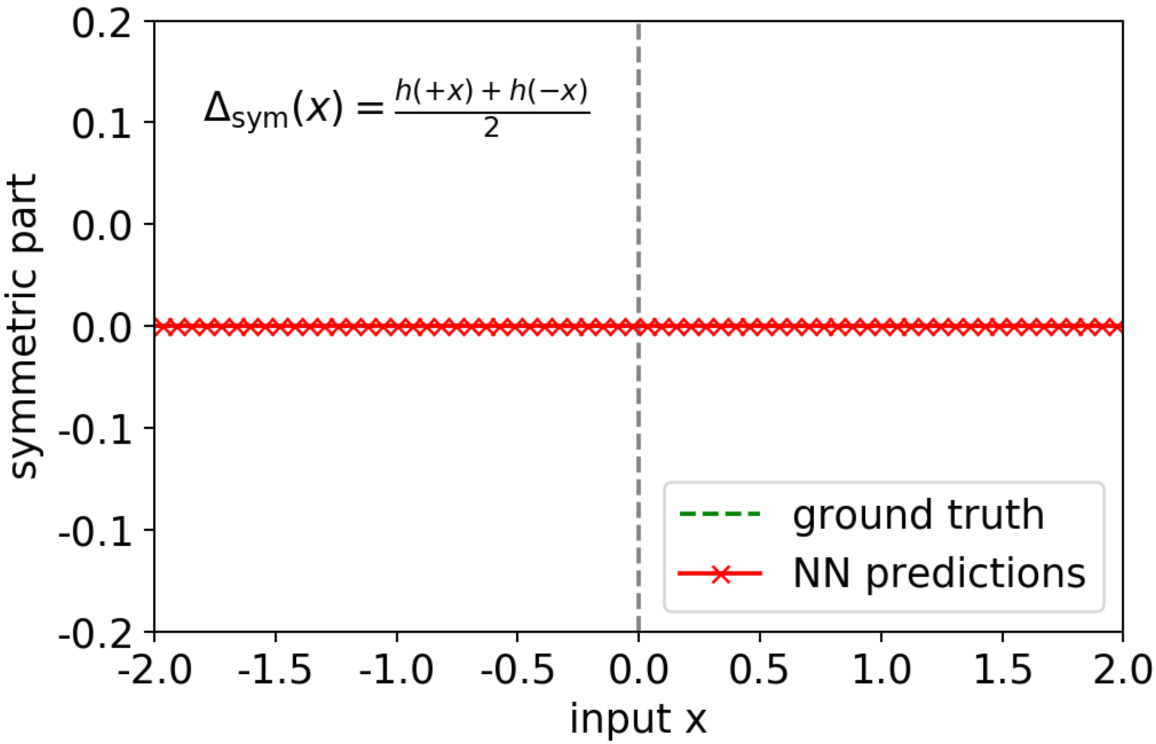
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.l1 = nn.Linear(1, 128, bias=False)
        self.l2 = nn.Linear(128, 32, bias=False)
        self.l3 = nn.Linear(32, 1, bias=False)

    def forward(self, x):
        x = torch.tanh(self.l1(x))
        x = torch.tanh(self.l2(x))
        x = self.l3(x)
        return x
```

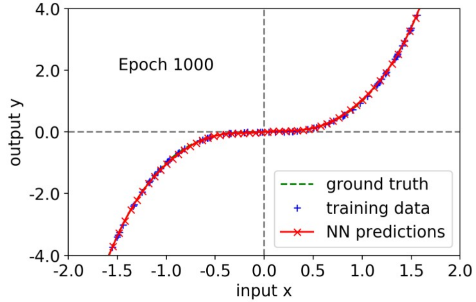
Remove bias and use antisymmetric activation function → every layer is antisymmetric



Imposing the symmetry

Learn $f(x) = x^3$





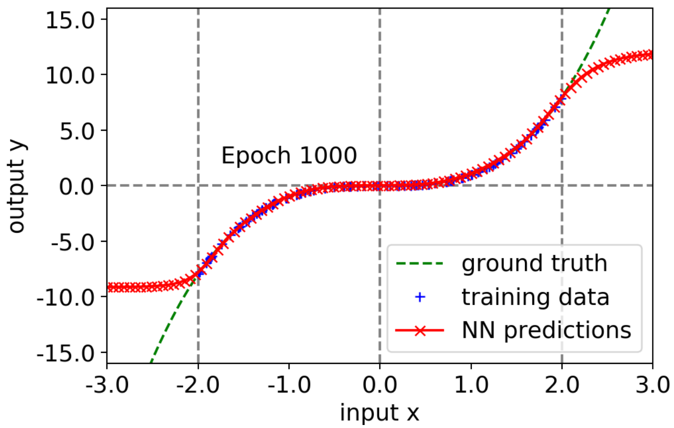
Symmetry: $f(-x) = -f(x)$

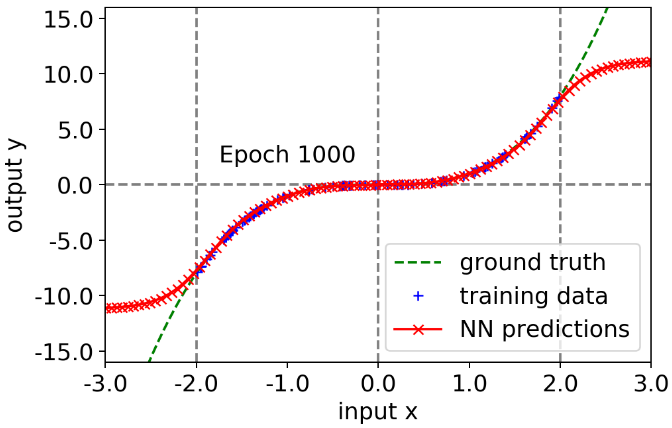
exactly preserved by construction
(without bias, tanh)

Generalization beyond training domain

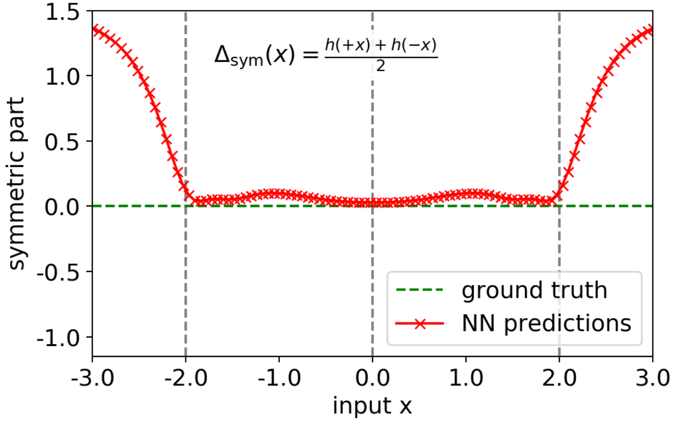
With bias, sigmoid

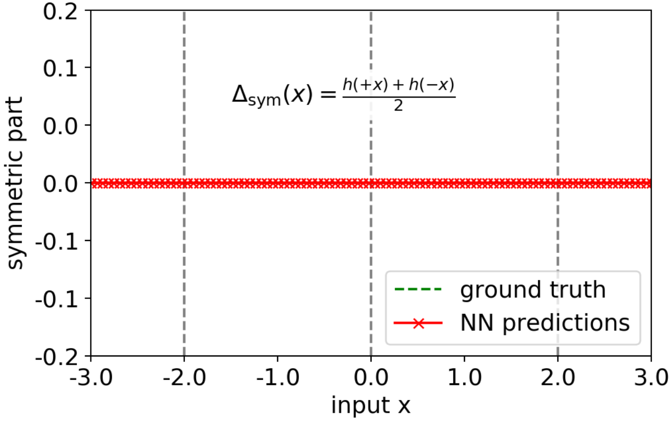
Without bias, tanh





Generalization is difficult.



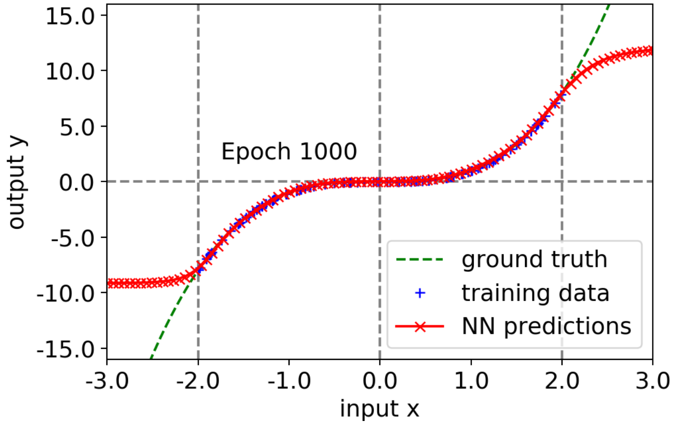


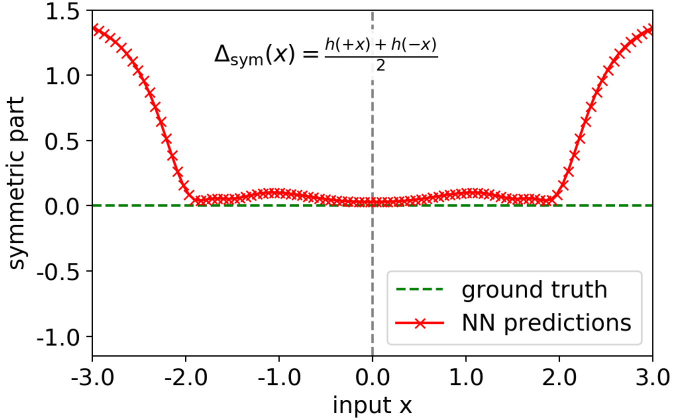
Imposed symmetry is preserved everywhere.

(This does not mean one can trust the result everywhere.)

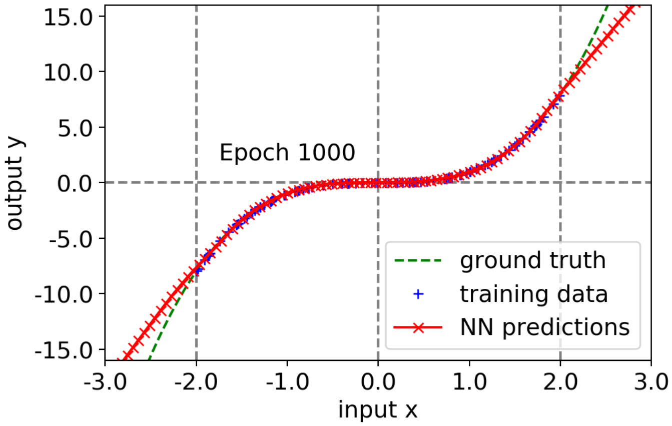
Generalization beyond training domain

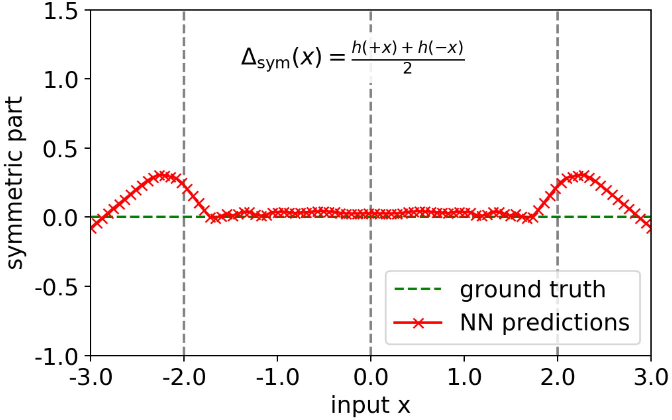
With bias, sigmoid



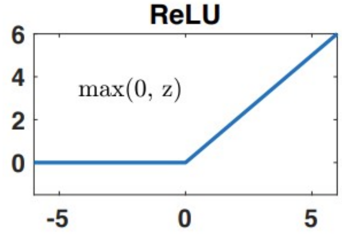


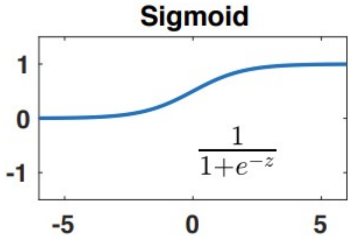
With bias, ReLU





Commonly used activation function:

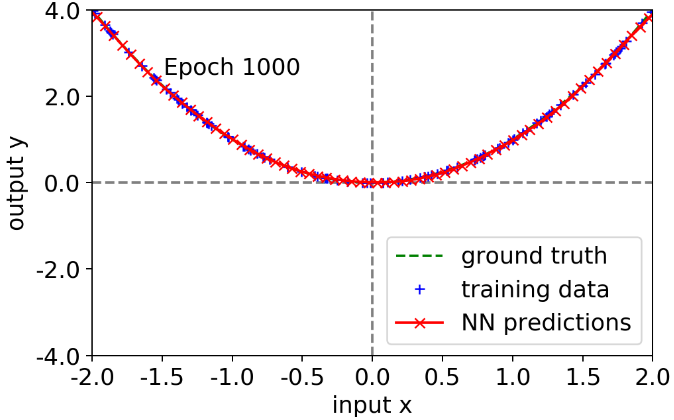


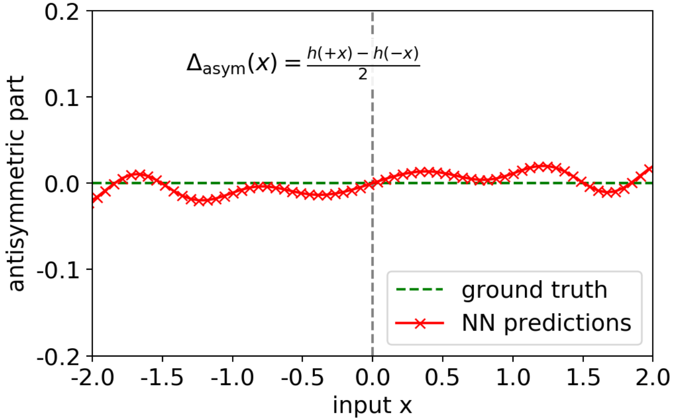


Slightly better generalization possible, but symmetry not exactly preserved

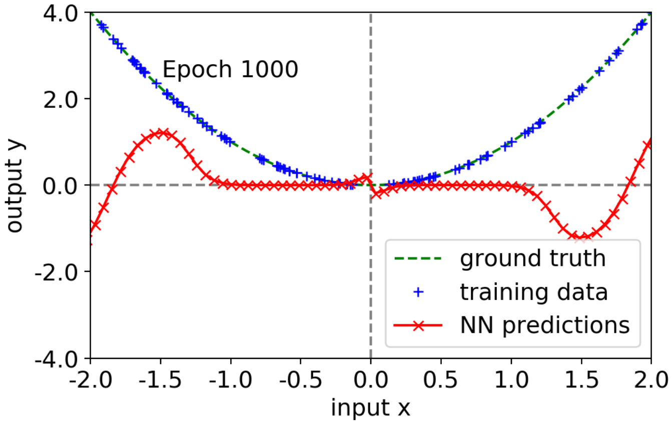
Trying to solve for wrong symmetry

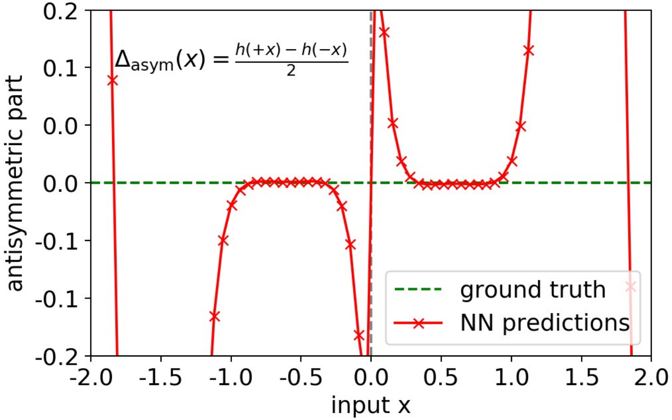
With bias, sigmoid





Without bias, tanh



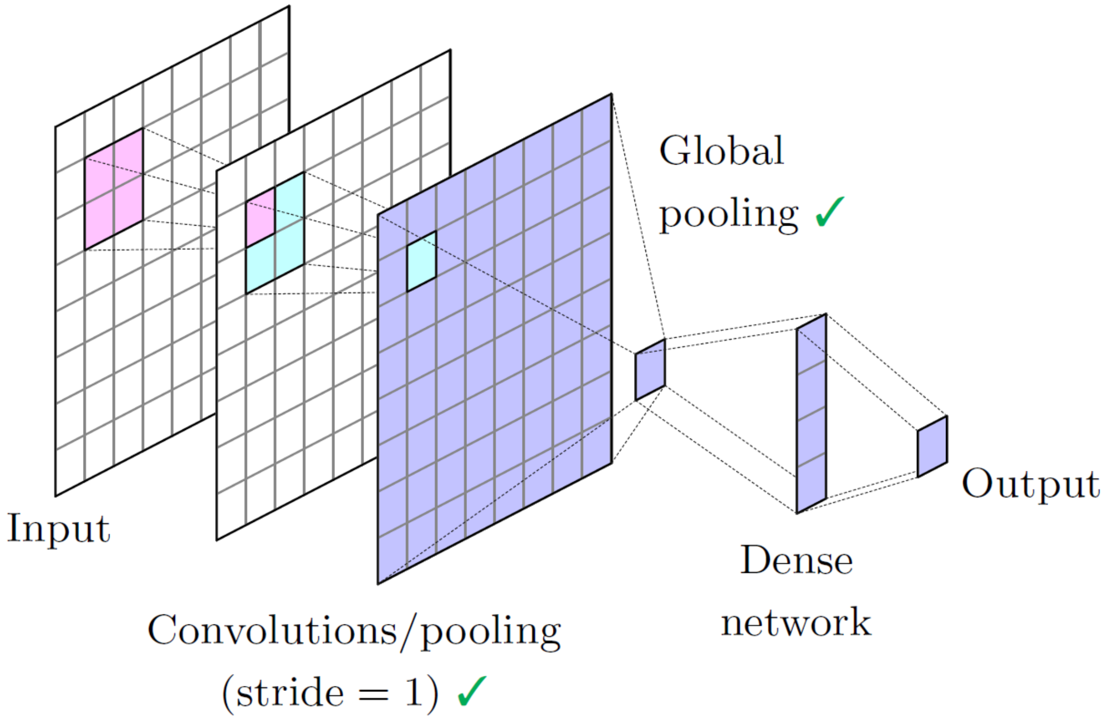


Learn $f(x) = x^2$

Generic network can also learn symmetric function approximately.

Antisymmetric fit to a symmetric solution fails.

Translational symmetry



Convolutional neural networks

Dense neural network:

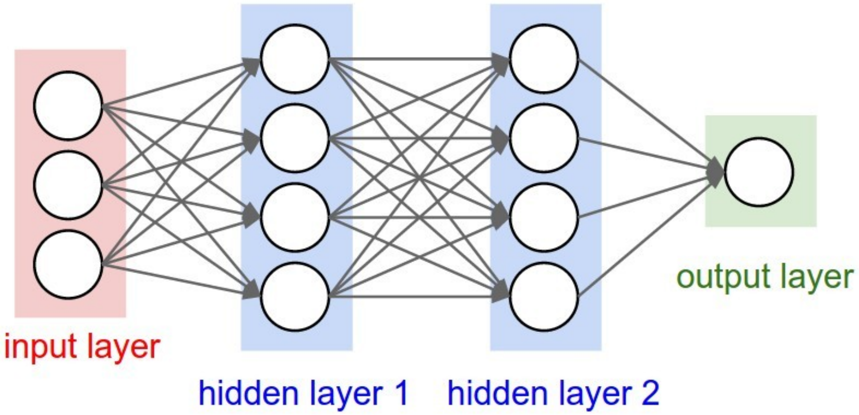


Image: https://cs231n.github.io/neural-networks-1/

→ Every input node connected to every output node

Convolutional Neural Network (CNN):

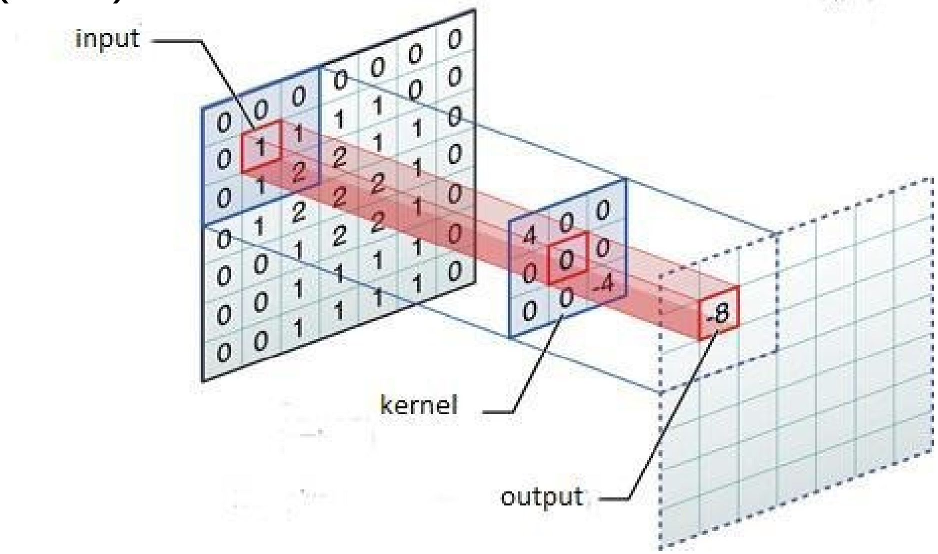
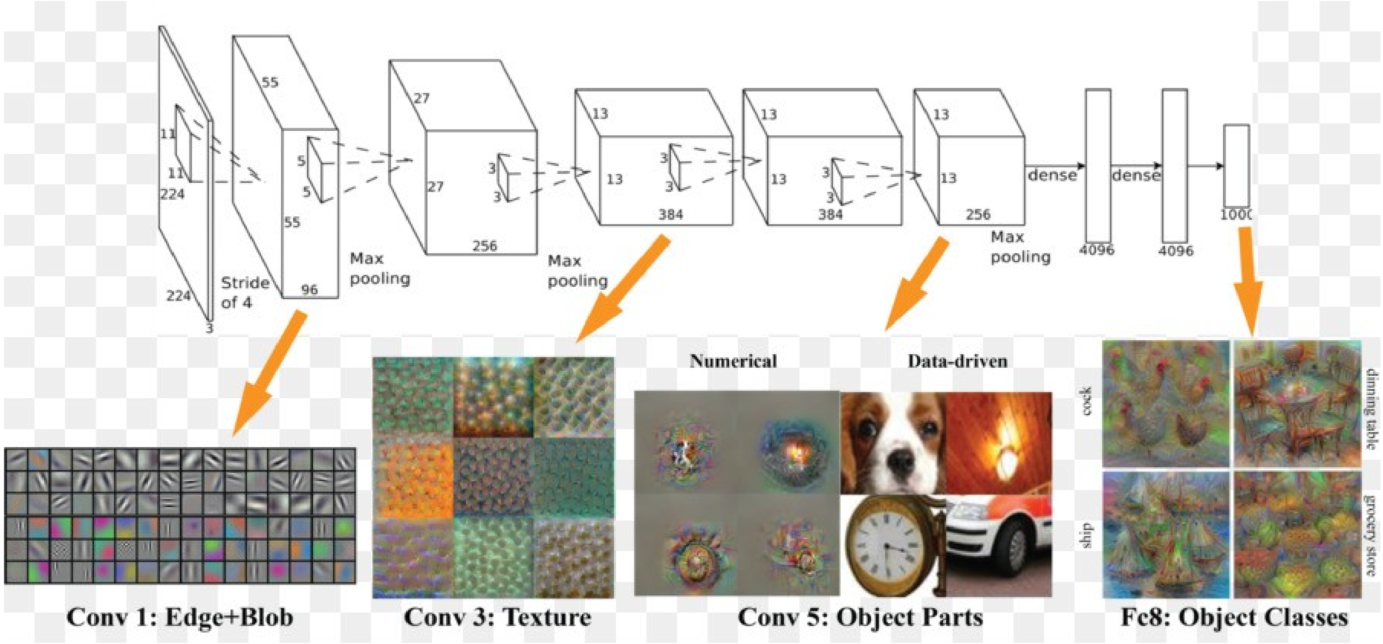


Image: https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1

→ local information: only nodes „nearby“ are connected
→ Weight sharing by sliding the same kernel across the whole image

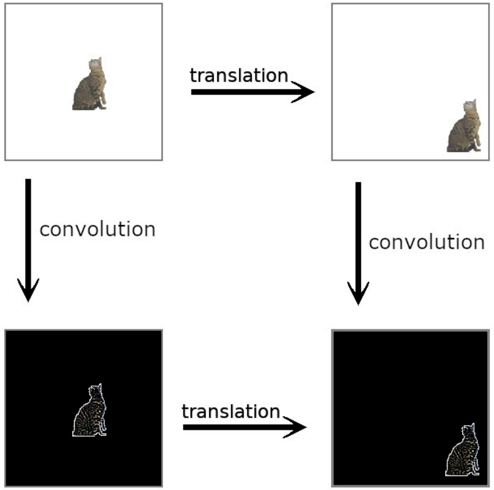
Deep learning



1960s: shallow neural networks
1960-70s: backpropagation
1980s: convolutional networks (CNN)

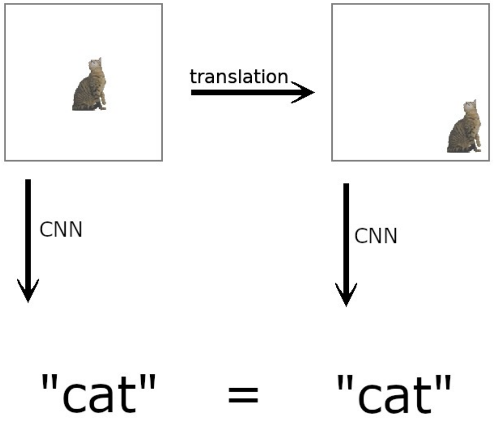
1990s: supervised deep learning
2006s: modern deep learning
2012: AlexNet (first GPU CNN)

Equivariance (covariance) vs. invariance



Equivariance

$$\Phi(L_g\, x) = L_g'\, \Phi(x)$$



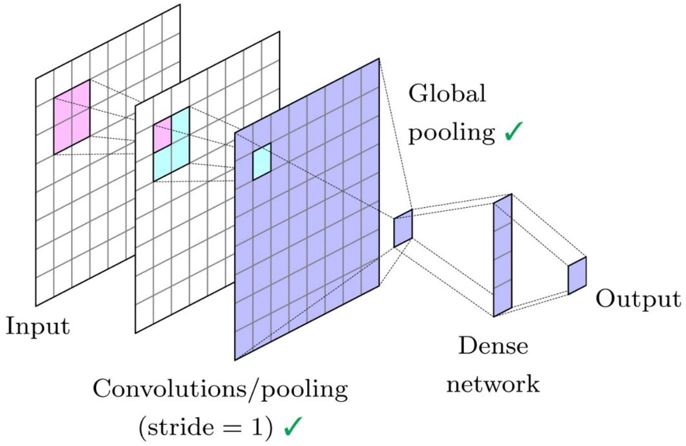
Invariance

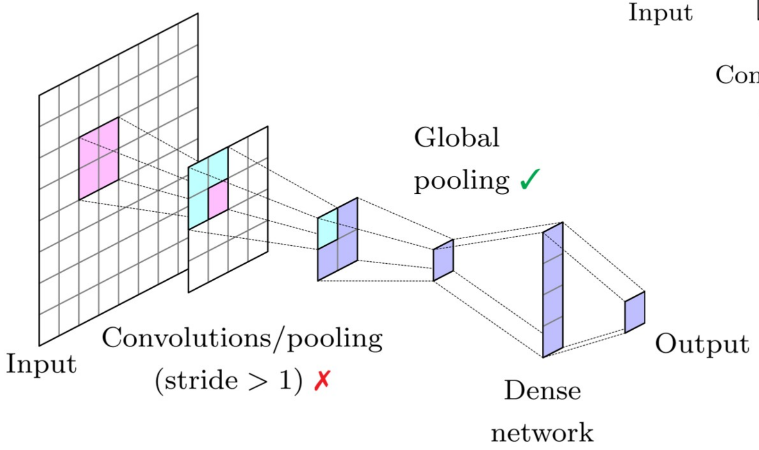
$$\Phi(L_g\, x) = \Phi(x)$$

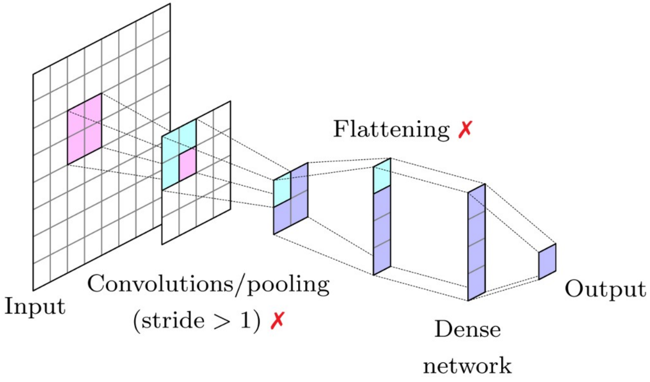
Adapted from: https://towardsdatascience.com/sesn-cec766026179

Translational symmetry

Equivariant architecture (EQ)







Strided architecture (ST)

Flattening architecture (FL)

Bulusu, Favoni, AI, Müller, Schuh, Phys. Rev. D 104 (2021) 074504

Complex scalar field in 1+1D

Complex scalar field action:
with chemical potential μ

$$S = \int \mathrm{d}x_0 \mathrm{d}x_1 \left(|D_0\phi|^2 - |\partial_1\phi|^2 - m^2|\phi|^2 - \lambda|\phi|^4 \right)$$

$$D_0 = \partial_0 - i\mu$$

Lattice formulation:

$$S_{lat} = \sum_x \left(\eta|\phi_x|^2 + \lambda|\phi_x|^4 - \sum_{\nu=1}^{2} \left(e^{\mu\,\delta_{\nu,2}} \phi_x^* \phi_{x+\hat{\nu}} + e^{-\mu\,\delta_{\nu,2}} \phi_x^* \phi_{x-\hat{\nu}} \right) \right)$$

$$\eta = 2D + m^2$$

Dual formulation with integer fields *k*, *l* solves sign problem Gattringer, Kloiber (2013)

$$\phi_x \to \{k_{x,\nu}, l_{x,\nu}\}$$

Observables:

Particle number density:

$$n = \frac{1}{N} \sum_x k_{x,2}$$

Squared absolute value of field:

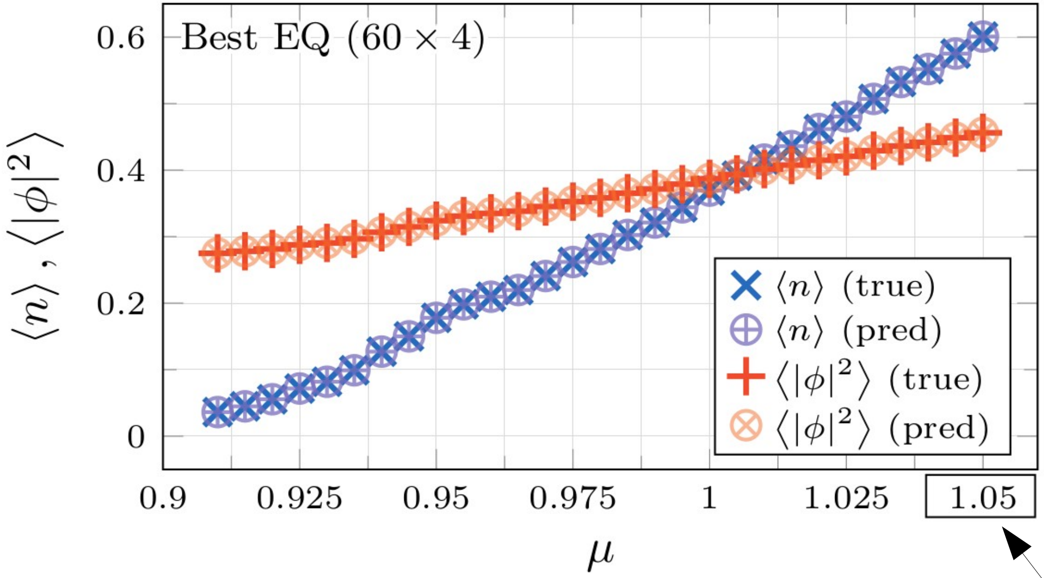
$$|\phi|^2 = \frac{1}{N} \sum_x \frac{W(f_x + 2)}{W(f_x)}$$

$$f_x = \sum_\nu [|k_{x,\nu}| + |k_{x-\hat{\nu},\nu}| + 2(l_{x,\nu} + l_{x-\hat{\nu},\nu})]$$

$$W(f_x) = \int_0^\infty \mathrm{d}x\, x^{f_x+1} \mathrm{e}^{-\eta x^2 - \lambda x^4}$$

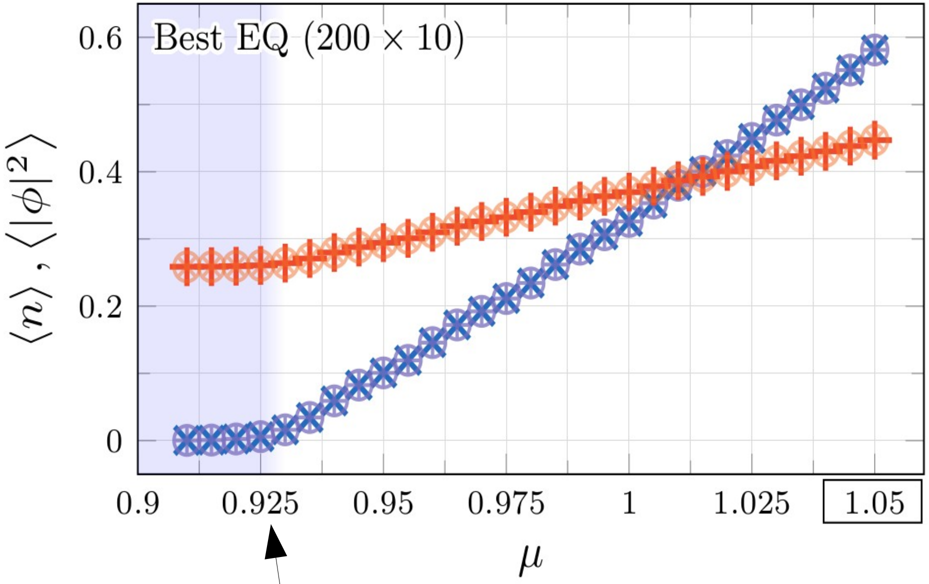
Predicted vs. true values

Generalization to smaller chemical potential:



Training point

Generalization to larger lattices:



Predicts Silver Blaze phase transition

Ensemble averages for each μ:

$$\langle n \rangle = \frac{T}{V} \frac{\partial \ln Z}{\partial \mu} = \frac{1}{N_x N_t} \left\langle \sum_x k_{x,2} \right\rangle$$

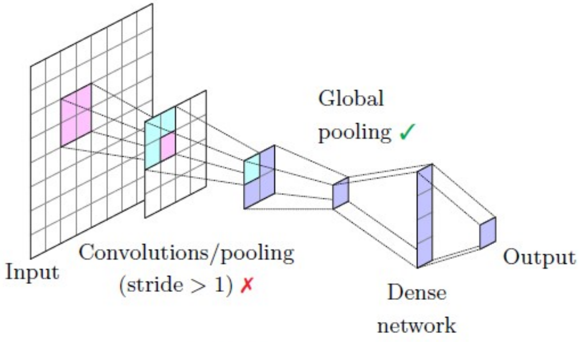
$$\langle |\phi|^2 \rangle = -\frac{T}{V} \frac{\partial \ln Z}{\partial \eta} = \frac{1}{N_x N_t} \left\langle \sum_x \frac{W(f_x + 2)}{W(f_x)} \right\rangle$$

Bulusu, Favoni, AI, Müller, Schuh,
Phys. Rev. D 104 (2021) 074504

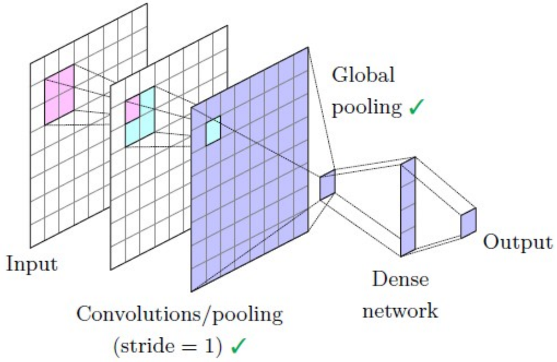
Comparison of architecture types

For fair comparison, best architectures for each type have been obtained by an Optuna optimization (scanning through various kernel sizes, number of layers, number of channels, …)

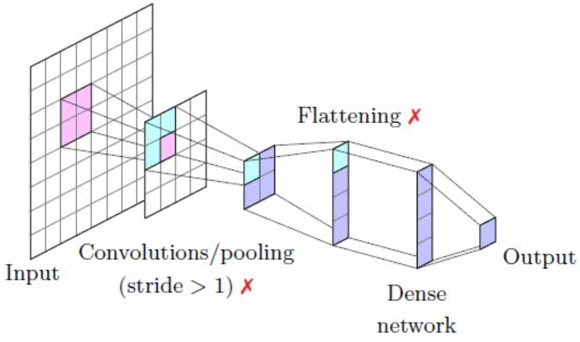
Best architectures are retrained 10 times and evaluated on the validation set.



(b)Strided architecture (ST)

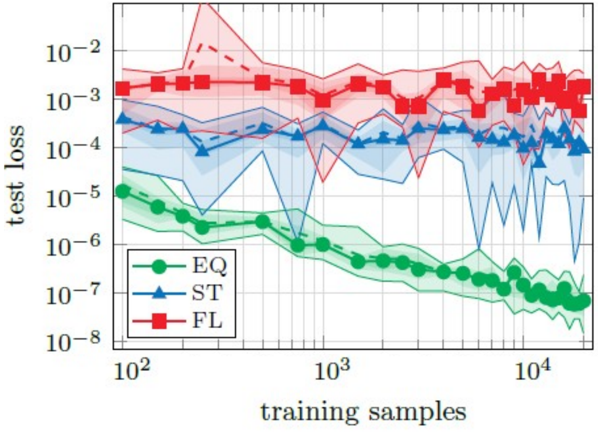


(a)Equivariant architecture (EQ)



(c)Flattening architecture (FL)

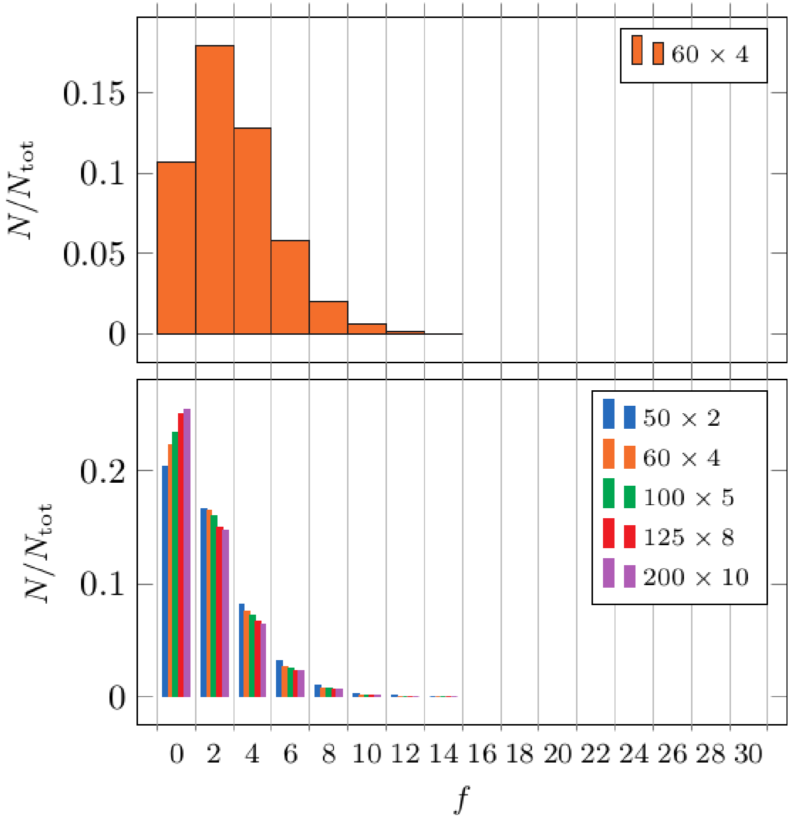
Test regression tasks on observables of a scalar field model in 2 dimensions:



Bulusu, Favoni, AI, Müller, Schuh, Phys. Rev. D 104 (2021) 074504

Why can the models generalize so well?

Input distribution



Train at

$$\mu = 1.05$$

and 60 × 4

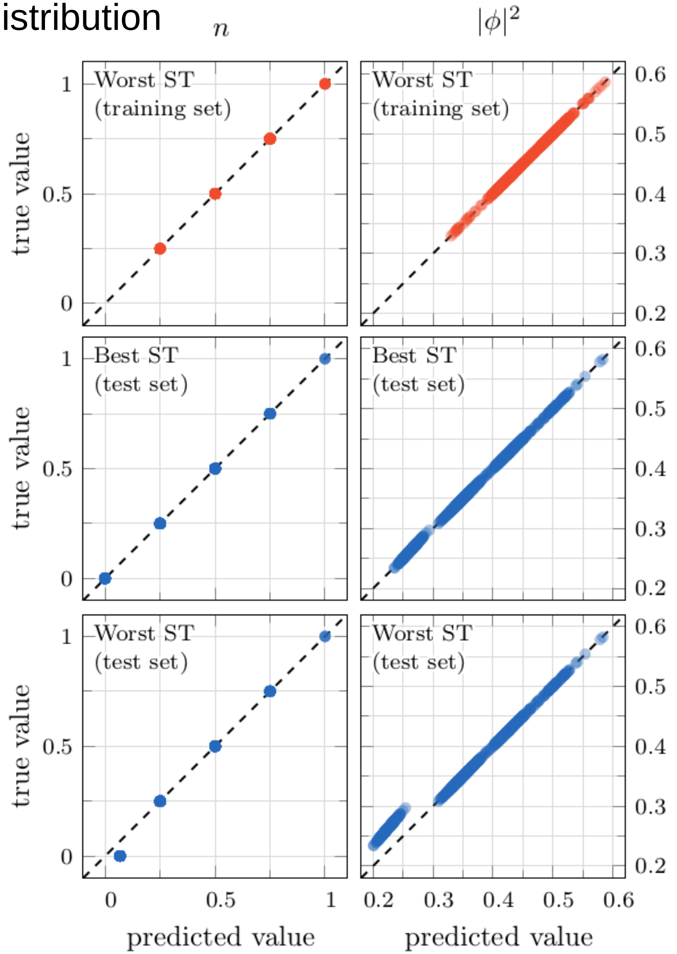
Test for

$$\mu \in [0.91, 1.05]$$

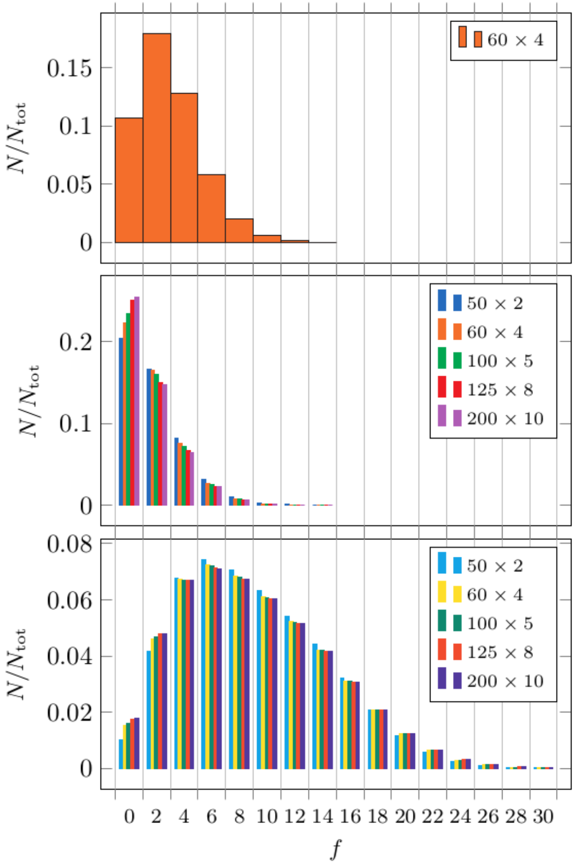
and various lattice sizes

Without ensemble average, individual configurations cover a large range of possible output values.

Output distribution



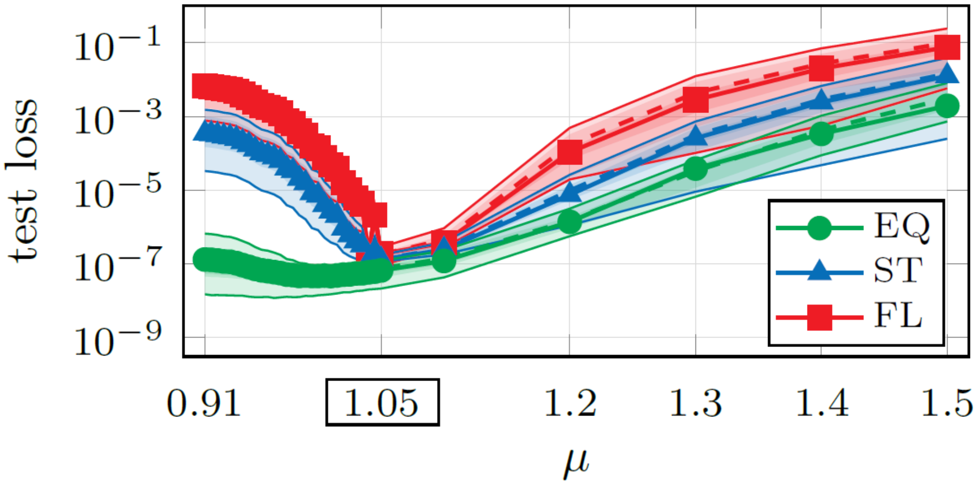
Generalization to larger μ



Train at
$\mu = 1.05$

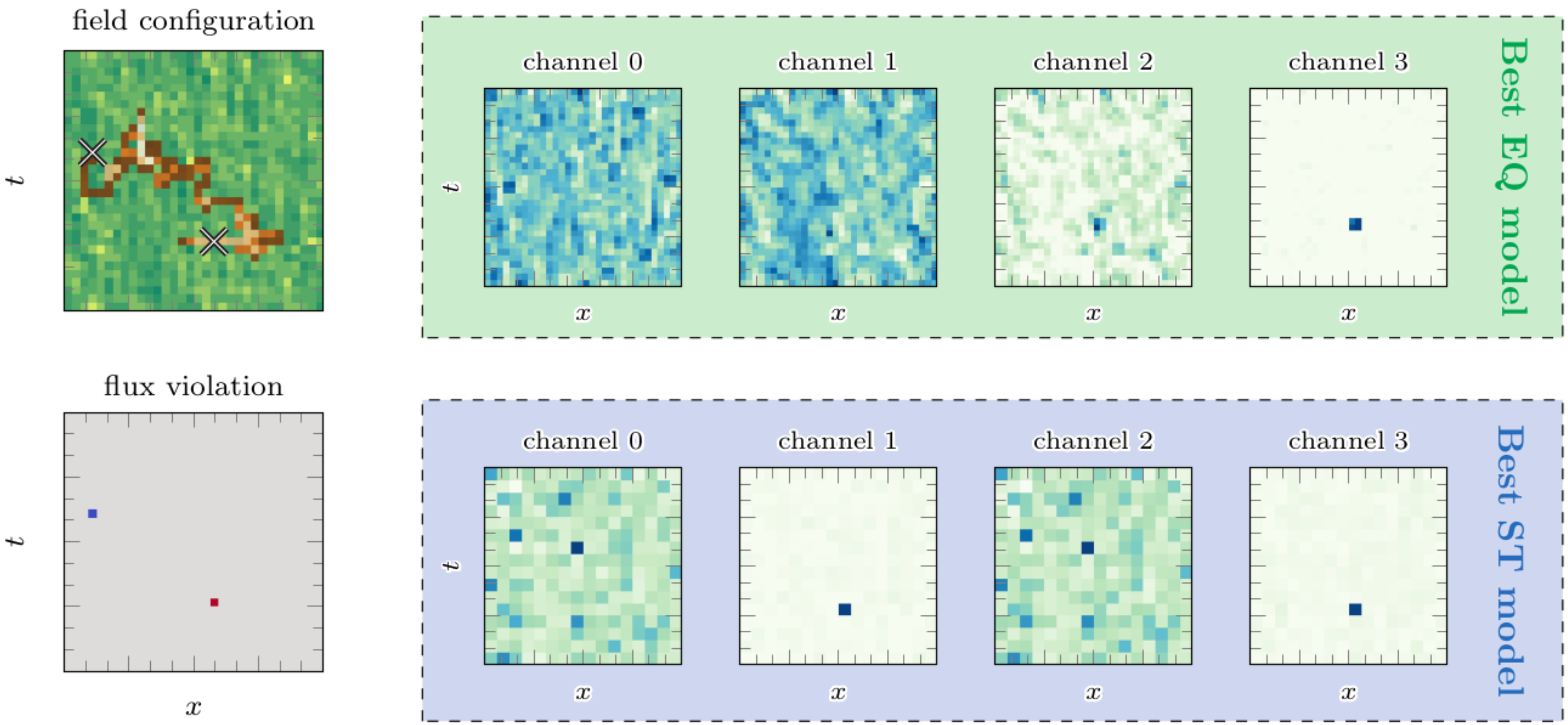
Test for
$\mu \in [0.91, 1.05]$

Test for
$\mu \in [1.1, 1.5]$



(No ensemble average)

Detection of flux violation

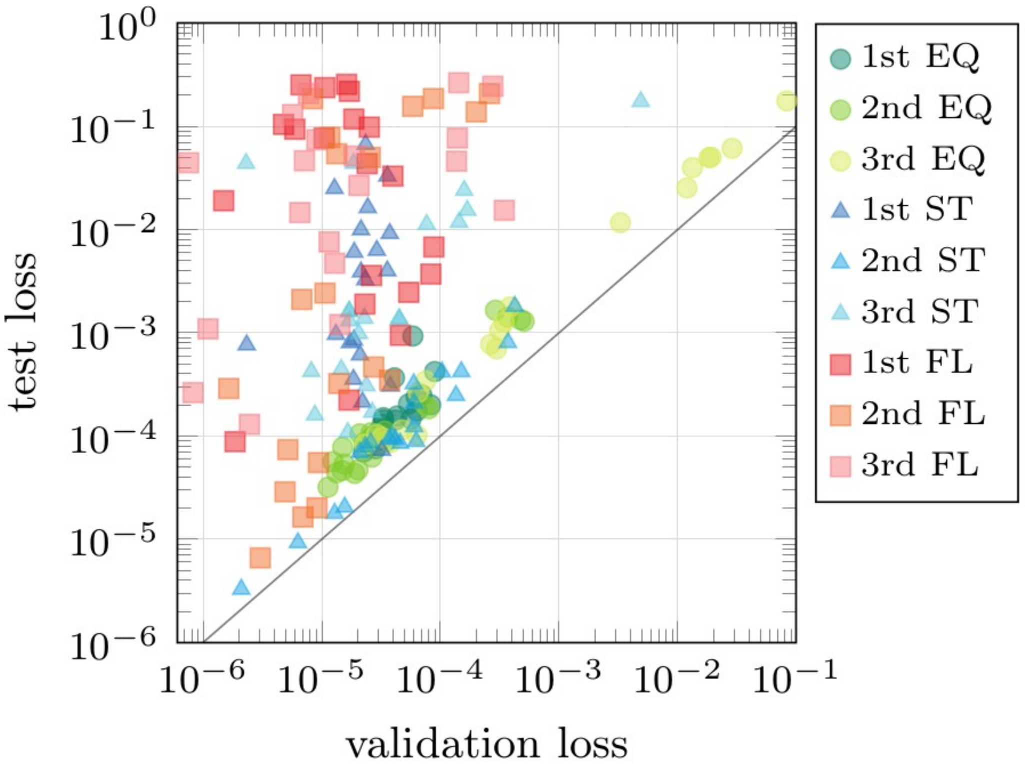


(a)Example field configuration

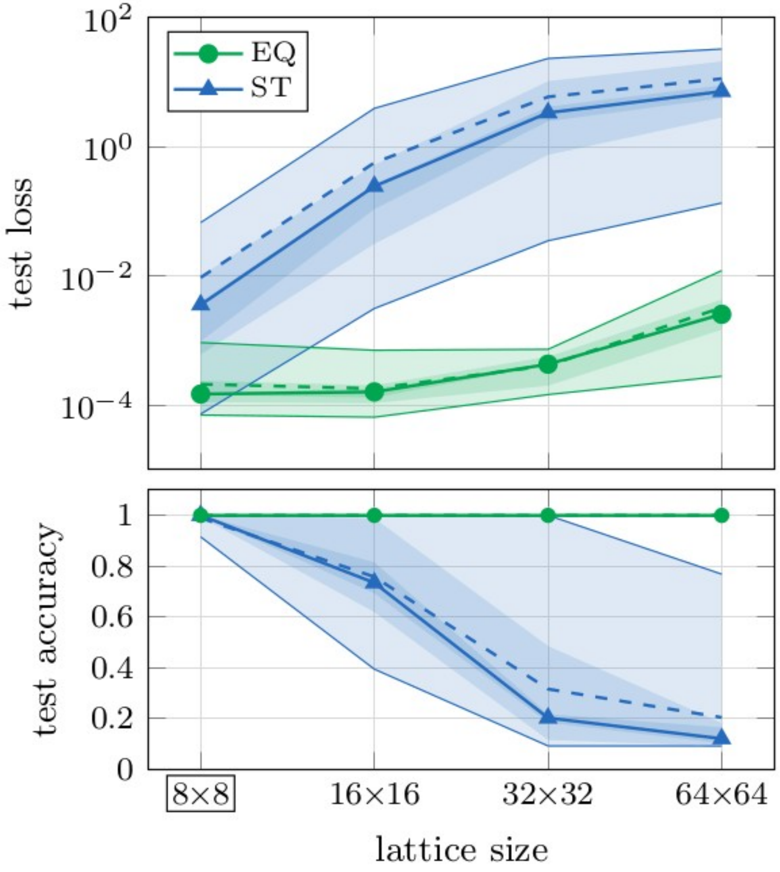
(b)Feature maps of convolutional network in best EQ and ST models

Detection of flux violation

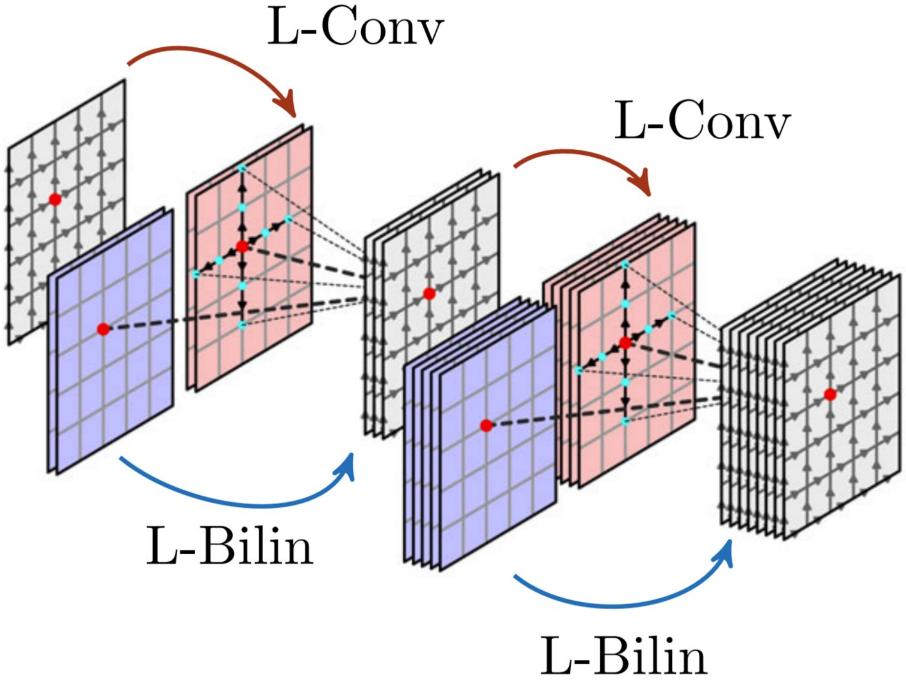
Generalization to different number of worms



Generalization to different lattice sizes

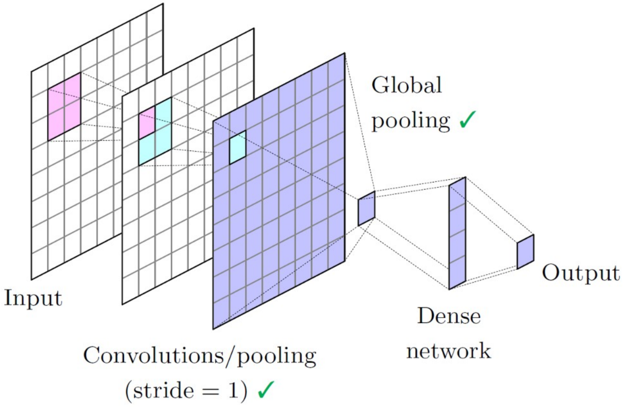


Lattice gauge symmetry



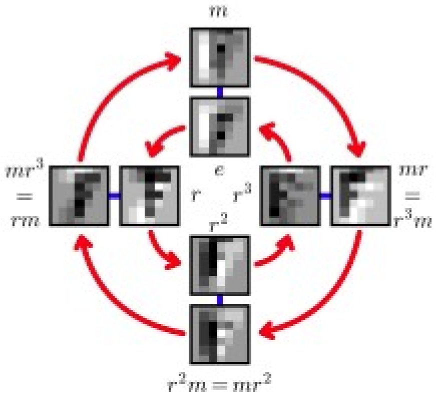
Symmetries on the lattice

Translational symmetry
→ Convolutional neural networks (CNNs)



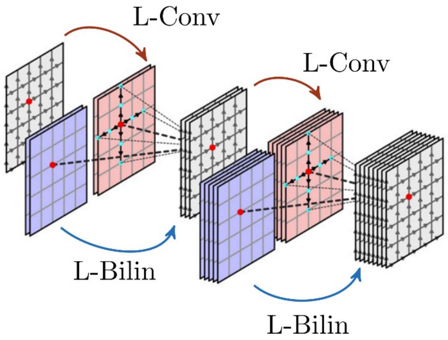
Bulusu, Favoni, AI, Müller, Schuh, Phys. Rev. D 104 (2021) 074504

Rotation, mirror symmetry
→ Group equivariant CNNs (G-CNNs)



Cohen, Welling, ICML 2016

Lattice gauge symmetry
→ Lattice gauge equivariant CNNs (L-CNNs)



Favoni, AI, Müller, Schuh, Phys.Rev.Lett. 128 (2022) 032003

Yang-Mills action vs. Wilson action

Continuum formulation

Yang-Mills action

$$S_G[A] = \frac{1}{2\,g^2}\int \mathrm{d}^4x \;\mathrm{tr}\left[F_{\mu\nu}(x)F_{\mu\nu}(x)\right]$$

Field strength tensor

$$F_{\mu\nu}(x) = -\mathrm{i}[D_\mu(x), D_\nu(x)]$$
$$= \partial_\mu A_\nu(x) - \partial_\nu A_\mu(x) + \mathrm{i}[A_\mu(x), A_\nu(x)]$$

Covariant derivative

$$D_\mu(x) = \partial_\mu + \mathrm{i}\,A_\mu(x)$$

SU(3) gauge fields

$$A_\mu(x) = \sum_{i=1}^{8} A_\mu^{(i)}(x)\,T_i$$

Gauge transformation

$$A_\mu(x) \to A'_\mu(x) = \Omega(x)A_\mu(x)\Omega(x)^\dagger + \mathrm{i}\left(\partial_\mu\Omega(x)\right)\Omega(x)^\dagger$$

Discrete formulation

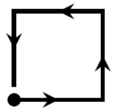
Wilson action

$$S_W[U] = \frac{2}{g^2}\sum_{x\in\Lambda}\sum_{\mu<\nu}\mathrm{Tr}\left[\mathbb{1} - U_{x,\mu\nu}\right]$$

Wilson (1974)

Plaquette

$$U_{x,\mu\nu} = U_{x,\mu}U_{x+\mu,\nu}U^\dagger_{x+\nu,\mu}U^\dagger_{x,\nu} =$$



Link variable

$$U_\mu(n) = \exp\left(\mathrm{i}\,aA_\mu(n)\right)$$

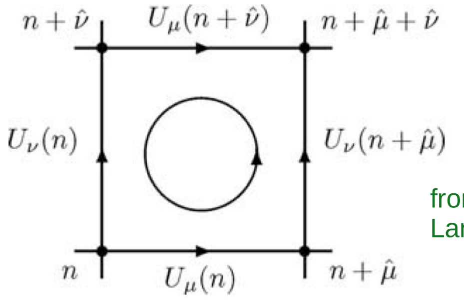
Gauge transformation

$$U_\mu(n) \;\to\; U'_\mu(n) = \Omega(n)\,U_\mu(n)\,\Omega(n+\hat{\mu})^\dagger$$

Taylor expansion in small lattice spacing reproduces continuum action:

$$U_{\mu\nu}(n) = \exp\left(\mathrm{i}\,a^2 F_{\mu\nu}(n) + \mathcal{O}(a^3)\right)$$

$$S_G[U] = \frac{2}{g^2}\sum_{n\in\Lambda}\sum_{\mu<\nu}\mathrm{Re\;tr}\left[\mathbb{1} - U_{\mu\nu}(n)\right] = \frac{a^4}{2\,g^2}\sum_{n\in\Lambda}\sum_{\mu,\nu}\mathrm{tr}[F_{\mu\nu}(n)^2] + \mathcal{O}(a^2)$$



from Gattringer, Lang (2010)

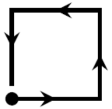
Wilson loops

Wilson action

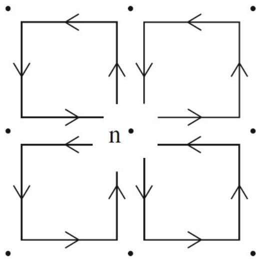
$$S_W[U] = \frac{2}{g^2} \sum_{x \in \Lambda} \sum_{\mu < \nu} \mathrm{Tr}\left[\mathbb{1} - U_{x,\mu\nu}\right]$$

Plaquette

$$U_{x,\mu\nu} = U_{x,\mu} U_{x+\mu,\nu} U^\dagger_{x+\nu,\mu} U^\dagger_{x,\nu} =$$

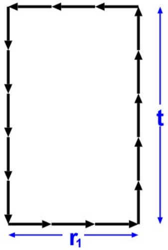


Symanzik improved clover action



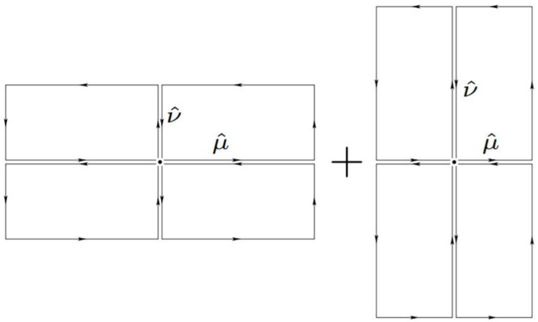
from: Gattringer, Lang (2010)

Potential of static quark pair



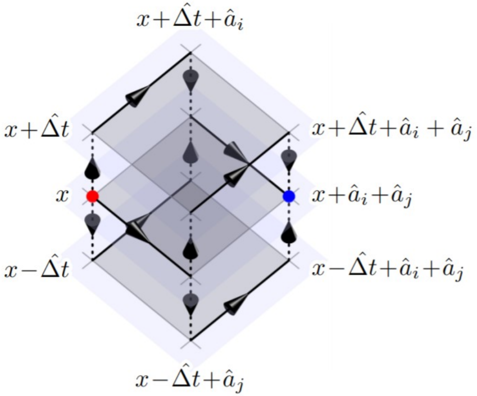
from: Bali, Phys.Rept. 343:1 (2001)

Improved topological charge



from: Alexandrou et al., Eur.Phys.J.C 80 (2020) 5, 424

Improved real-time lattice actions



AI, Müller, Eur.Phys.J. C78 (2018) no.11, 884

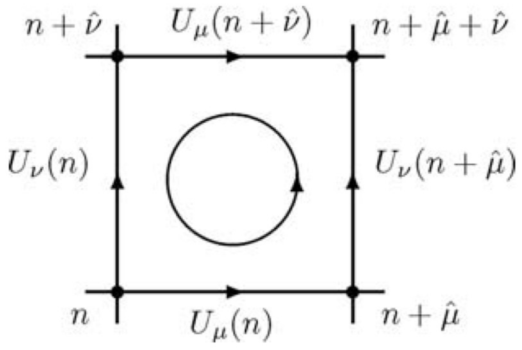
L-CNN data

Combine lattice links *U*
and locally transforming objects *W*

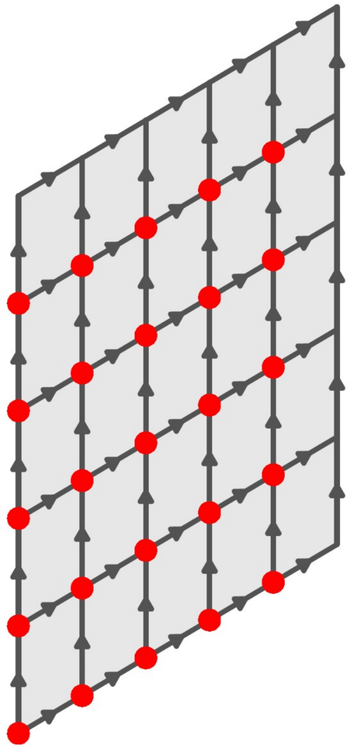
$$\text{tuple} \;\; (\mathcal{U}, \mathcal{W})$$

$$\mathcal{U} \;=\; \{U_{x,\mu}\} \;\; \mathrm{SU}(N_c) \text{ matrices}$$

$$\mathcal{W} \;=\; \{W_{x,i}\} \;\text{ with }\; W_{x,i} \;\in\; \mathbb{C}^{N_c \times N_c}$$



from: Gattringer, Lang (2010)



$$\mathcal{U} = \{U_{\mathbf{x},\mu}\}$$

$$\mathcal{W} = \{W_{\mathbf{x},\mu\nu}\}$$

Gauge transformation

$$T_\Omega U_{x,\mu} = \Omega_x U_{x,\mu} \Omega^\dagger_{x+\mu}$$

$$T_\Omega W_{x,i} = \Omega_x W_{x,i} \Omega^\dagger_x$$

Gauge equivariant (gauge covariant) function

$$f(T_\Omega \mathcal{U}, T_\Omega \mathcal{W}) \;=\; T'_\Omega f(\mathcal{U}, \mathcal{W})$$

Gauge invariant function

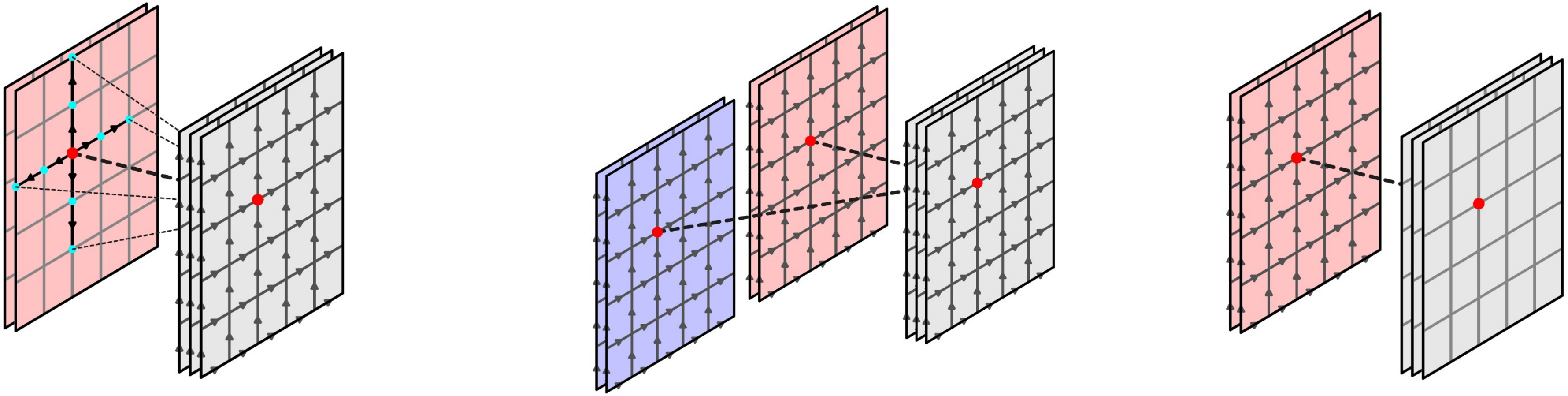
$$f(T_\Omega \mathcal{U}, T_\Omega \mathcal{W}) \;=\; f(\mathcal{U}, \mathcal{W})$$

Lattice gauge equivariant layers

Convolution (L-Conv)

Bilinear layer (L-Bilin)

Trace layer



Convolution wish shared weights and proper parallel transport along coordinate axes

$$(\mathcal{U},\mathcal{W}) \to (\mathcal{U},\mathcal{W}')$$

$$W'_{\mathbf{x},i} = \sum_{j,\mu,k} \omega_{i,j,\mu,k} U_{\mathbf{x},k\cdot\mu} W_{\mathbf{x}+k\cdot\mu,j} U^\dagger_{\mathbf{x},k\cdot\mu}$$

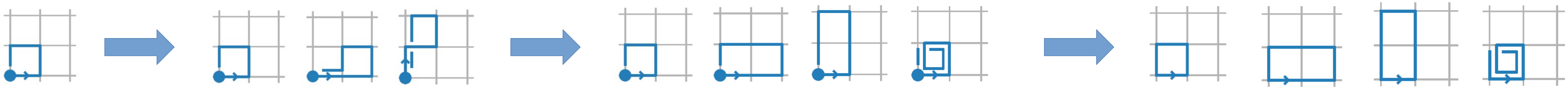
Multiply W at each lattice point

$$(\mathcal{U},\mathcal{W}) \times (\mathcal{U},\mathcal{W}') \to (\mathcal{U},\mathcal{W}'')$$

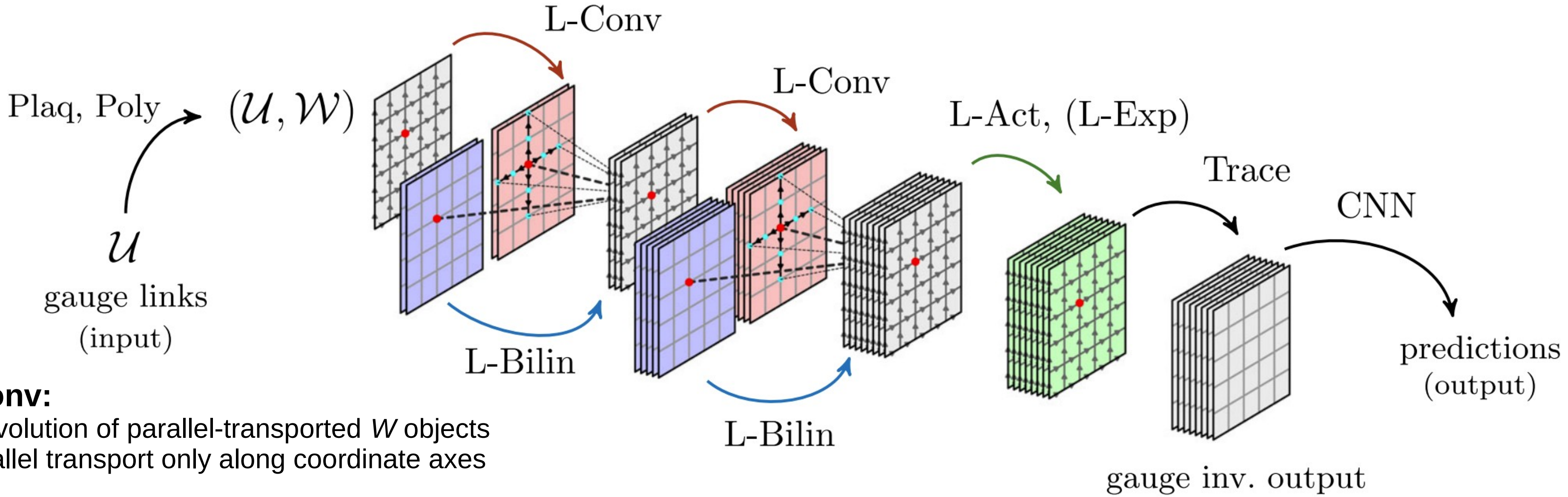
$$W''_{\mathbf{x},i} = \sum_{j,k} \alpha_{ijk} W_{\mathbf{x},j} W'_{\mathbf{x},k}$$

Generate gauge invariant output

$$w_{\mathbf{x},i} = \operatorname{Tr} W_{\mathbf{x},i} \in \mathbb{C}$$



Generic L-CNN



L-Conv:
* convolution of parallel-transported *W* objects
* parallel transport only along coordinate axes

L-Bilin:
* bilinear layer, product of locally transforming objects

L-Act:
* activation functions multiply *W* objects by scalar, gauge-invariant functions

L-Exp:
* update link variables using exponential map

Trace:
* calculate gauge invariant trace

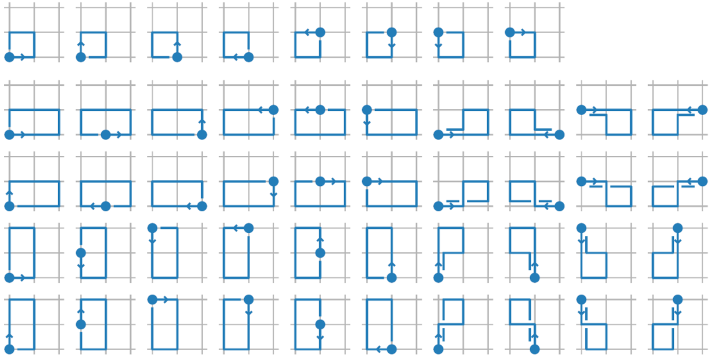
Plaq:
* generate all possible plaquettes

Poly:
* generate all possible Polyakov loops

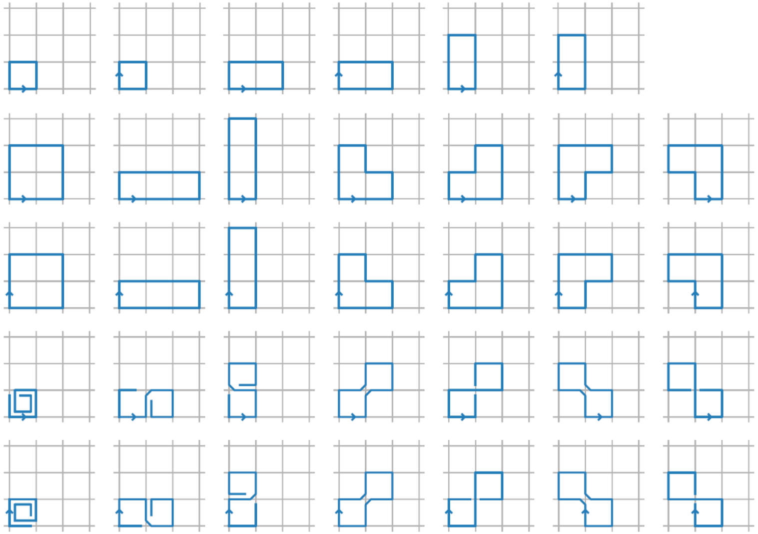
Favoni, AI, Müller, Schuh,
Phys.Rev.Lett. 128 (2022) 032003

L-CNNs generate Wilson loops

Untraced loops



Traced loops



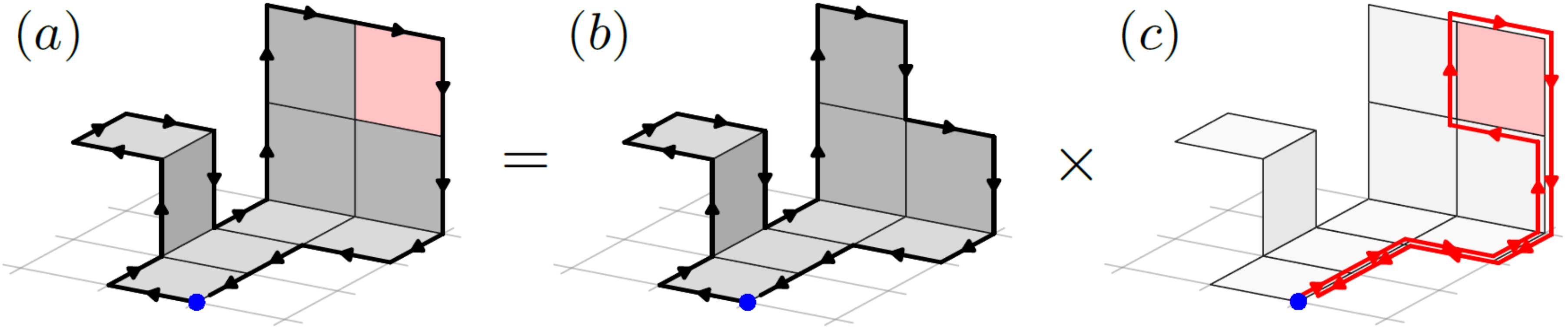
Number of traced Wilson loops covered by L-CNN architectures of various sizes in 1+1 D

Length	Max	$W^{(1\times1)}$ S	$W^{(1\times2)}$ S	$W^{(1\times2)}$ M	$W^{(1\times2)}$ L	$W^{(2\times2)}$ S	$W^{(2\times2)}$ M	$W^{(2\times2)}$ L
0	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
4	2	2	2	2	2	2	2	2
6	4		4	4	4	4	4	4
8	28		4	4	4	22	22	22
10	152			8	8	48	76	76
12	1,010				8	92	204	220
14	6,772					120	412	532
16	47,646					100	712	1,080
18	343,168					136	928	1,896
20	2,529,890					32	1,056	2,620
22	18,982,172					64	768	3,152
$\geq$24							800	7,210
Total		3	11	19	27	621	4,985	16,725
Max.Len		4	8	10	12	22	28	34

Architectures differ in number of layers, kernel size, and number of channels.

Favoni, AI, Müller, Schuh, Phys.Rev.Lett. 128 (2022) 032003

Sketch of proof for arbitrary Wilson loops



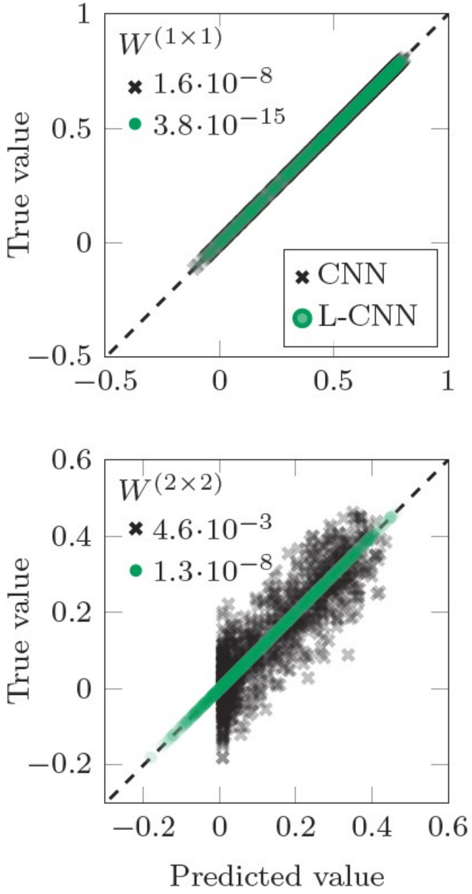
Favoni, AI, Müller, Schuh, Phys.Rev.Lett. 128 (2022) 032003

(a) An arbitrary contractible Wilson loop of *n* tiles ...
(b) … is composed (L-Bilin) of a Wilson loop of (*n*-1) tiles …
(c) … and a parallel-transported (L-Conv) plaquette (Plaq).

Non-contractible loops (like Polyakov loops) have to be added (Poly).

Numerical results

1+1D



Regression task to learn value of rectangular Wilson loops:

$$W_{x,\mu\nu}^{(m\times n)} = \mathrm{Re}\,\mathrm{Tr}\left[U_{x,\mu\nu}^{(m\times n)}\right]$$

Lattice gauge equivariant CNN (L-CNNs, green) can learn the relation, while traditional convolutional neural networks (CNNs, black) struggle to find the solution.

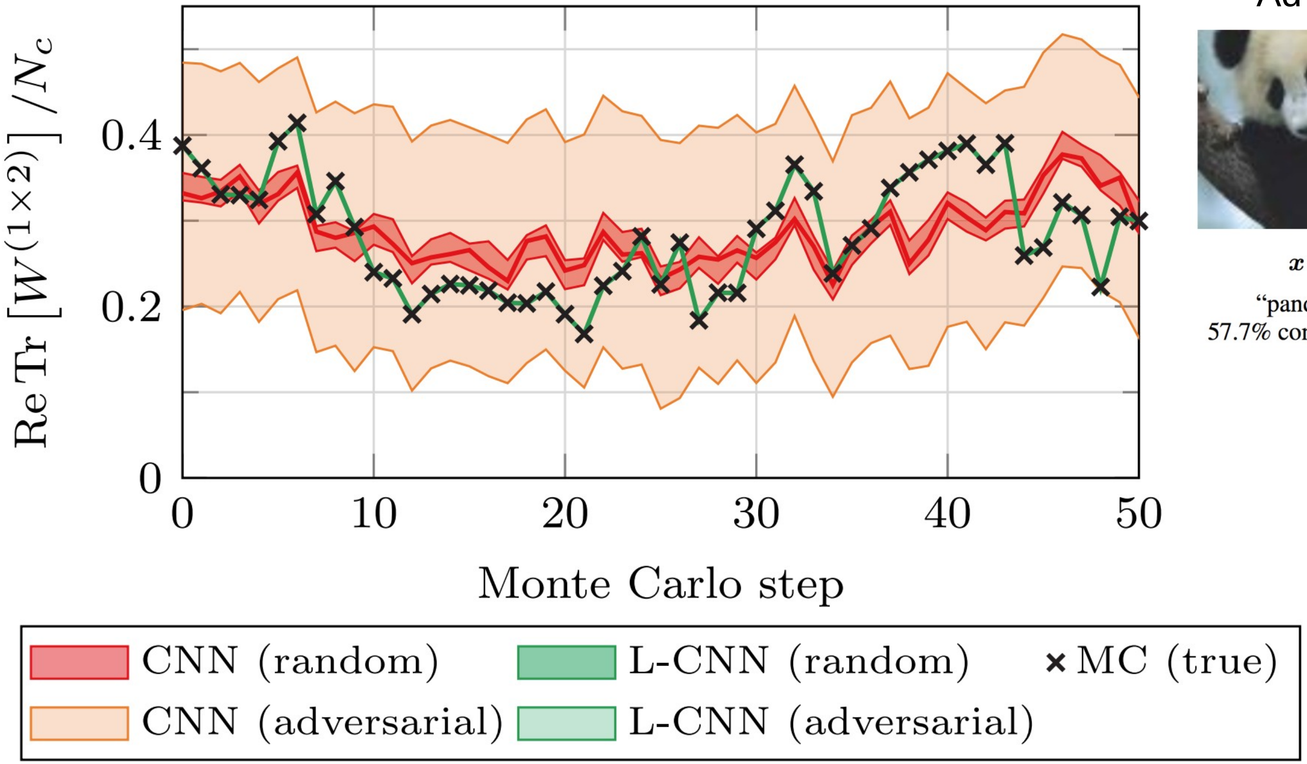
Training on 8 × 8, testing from 8 × 8 up to 64 × 64

Compared best from:
100 L-CNN models (10 – 10^4 trainable parameters, up to 4 L-Conv+L-Bilin)

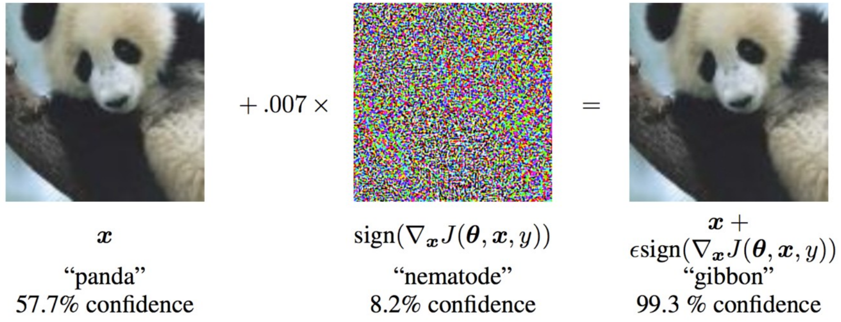
2840 CNN models (100 – 10^5 trainable parameters up to 6 layers, 512 channels, 4 activation functions)

Favoni, AI, Müller, Schuh, Phys.Rev.Lett. 128 (2022) 032003

Adversarial attacks



Adversarial attack:



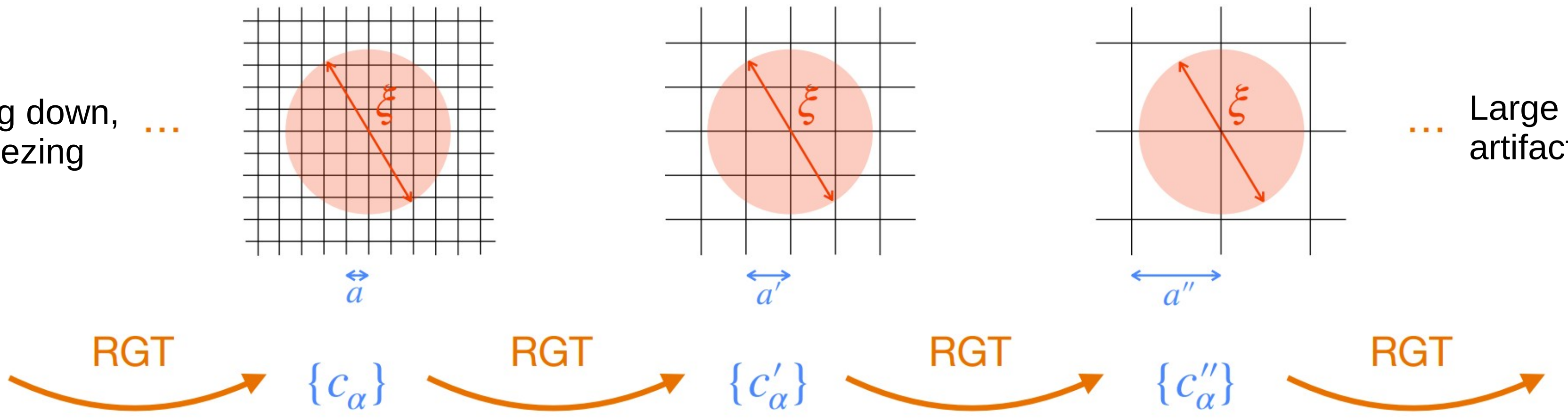
From Goodfellow, Shlens, Szegedy ICLR 2015

L-CNNs are insensitive to random or adversarial gauge transformations

Favoni, AI, Müller, Schuh, Phys.Rev.Lett. 128 (2022) 032003

Fixed point action

Critical slowing down, topological freezing ...



... Large lattice artifacts

Introduce a renormalization group transformation (RGT)

$$\exp\left\{-\beta' A'[V]\right\} = \int \mathscr{D}U \exp\left\{-\beta\left(A[U] + T[U,V]\right)\right\}$$

Blocking kernel

The effective action $\beta' A'[V]$ is described

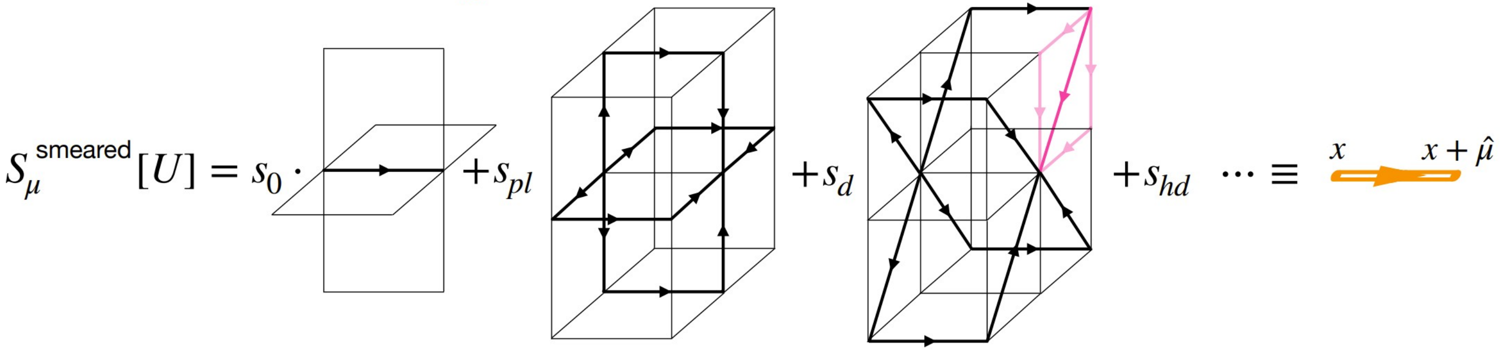
by infinitely many couplings $\{c'_\alpha\}$

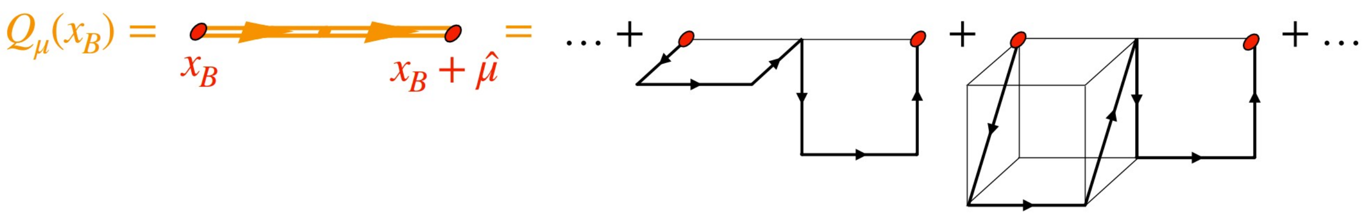
The fixed point is the saddle point in the classical limit $\beta \to \infty$, which can be found by a minimization condition.

P. Hasenfratz, F. Niedermayer, Nucl.Phys.B 414 (1994) 785

Blocking kernel

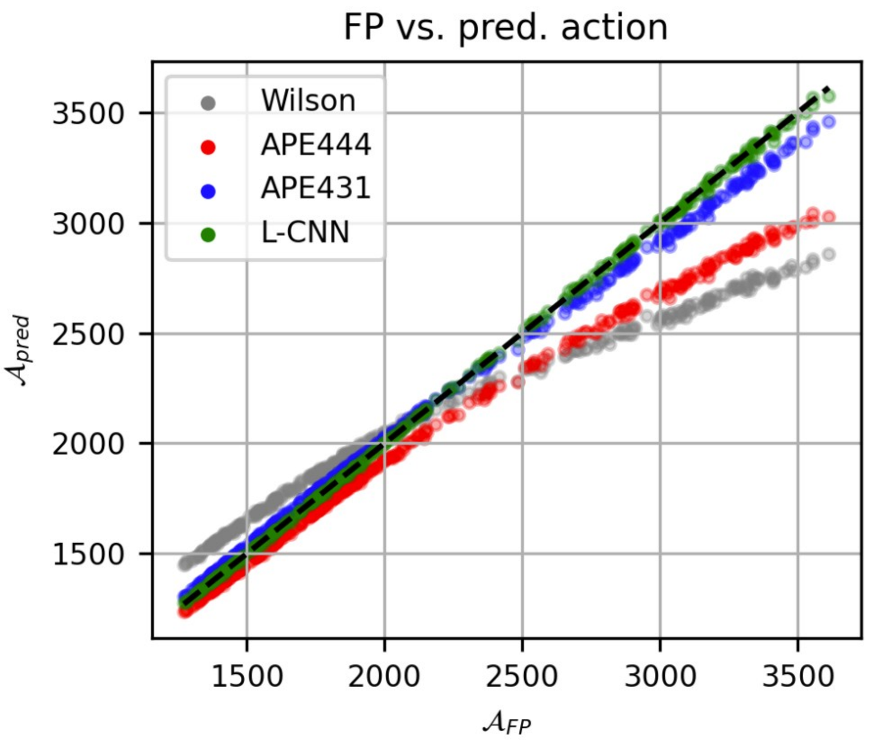
$$T[U,V] = -\frac{\kappa}{N_c}\sum_{x_B,\mu}\left\{\mathrm{ReTr}\left(V_\mu(x_B)\cdot Q_\mu^\dagger(x_B)\right) - \mathcal{N}_\mu^\beta\right\}$$

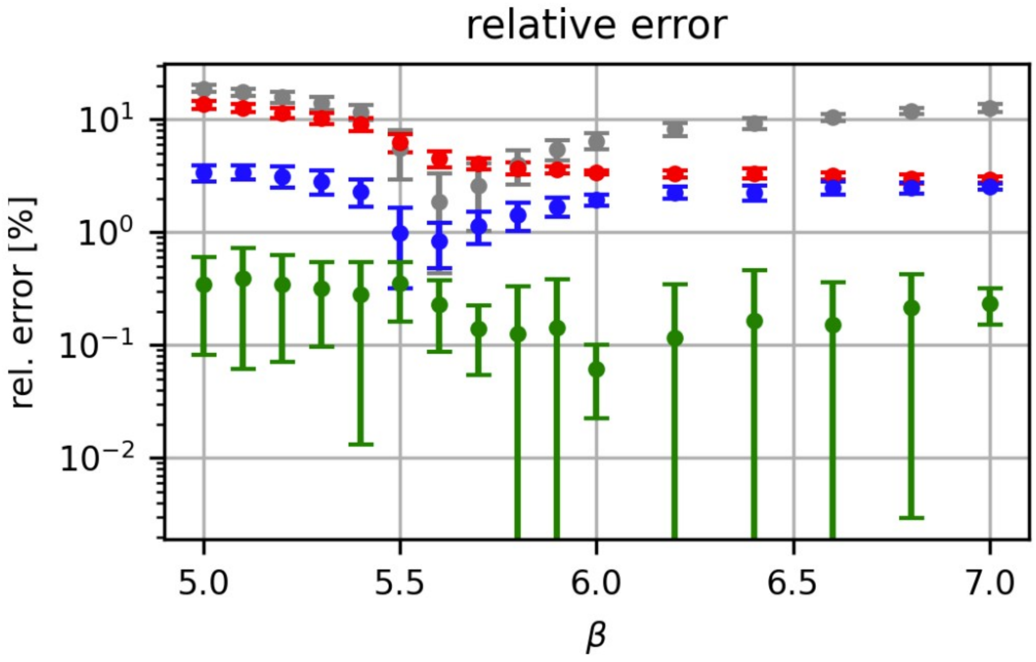




Many possibilities to construct a blocking kernel manually,
fit corresponding parameters.

Learning the fixed point action with L-CNNs





Training example: L-CNN model with 3 layers with 12, 24, 24 channels and kernel size 2, 2, 1.

L-CNN superior to older parametrizations of FP action.

Holland, Al, Müller, Wenger, in preparation

Continuous formulation of L-CNNs

Define a continuous version of a gauge equivariant convolution:

$$[\psi * \mathcal{W}]^a(\mathbf{x}) = \sum_b \int_{\mathbb{R}^D} \mathrm{d}\mathbf{y}^D \, \psi^{ab}(\mathbf{y}-\mathbf{x}) U_{\mathbf{x}\to\mathbf{y}} W^b(\mathbf{y}) U^\dagger_{\mathbf{x}\to\mathbf{y}}$$

with kernel components $\psi^{ab} : \mathbb{R}^D \to \mathbb{R}$

and parallel transporter $U_{\mathbf{x}\to\mathbf{y}} = \mathcal{P} \exp \left\{ i \int_0^1 \mathrm{d}s \frac{\mathrm{d}x^\nu(s)}{\mathrm{d}s} A_\nu(x(s)) \right\}$

that map $\mathcal{W} = (W^1, \ldots, W^m)$ objects to new objects

in a gauge equivariant manner:

$$[\psi * T_\Omega \mathcal{W}]^a(\mathbf{x}) = T_\Omega [\psi * \mathcal{W}]^a(\mathbf{x})$$

Similarly define continuous bilinear layer, trace layer, …

Discretize this to obtain previous formulation.

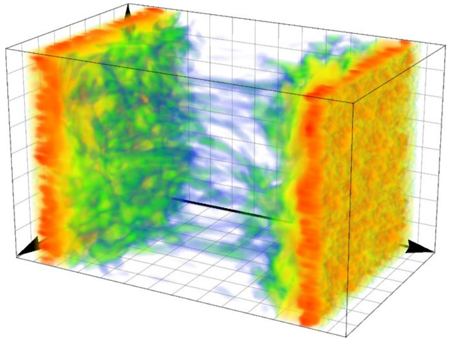
Compatible with G-CNNs.

Generalizable to vectors and tensors.

Aronsson, Müller, Schuh, arxiv:2303.11448

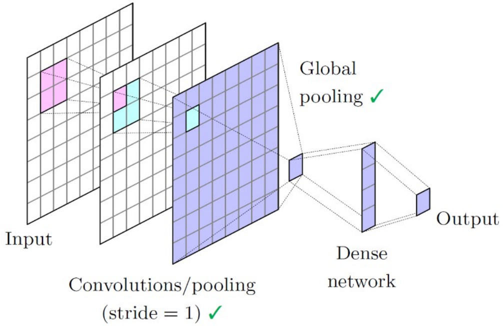
Summary

Glasma simulations



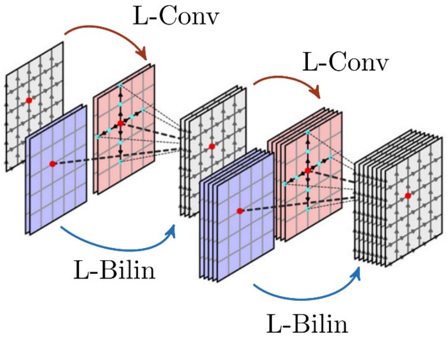
AI, Müller, Phys. Lett. B 771 (2017) 74
Gelfand, AI, Müller, Phys. Rev. D94 (2016) no.1, 014020

Translational equivariance



Bulusu, Favoni, AI, Müller, Schuh,
Phys. Rev. D 104 (2021) 074504

L-CNNs

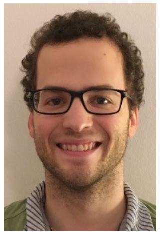


Favoni, AI, Müller, Schuh,
Phys.Rev.Lett. 128 (2022) 032003

“Upscaling Glasma simulations using machine learning”
Austrian Science Fund FWF No. P32446-N27

Open source: https://gitlab.com/openpixi/lge-cnn

David Müller



Daniel Schuh

Matteo Favoni