Exploiting symmetries to achieve fast model convergence

Simon Pfahler[1], Tilo Wettig[1]

[1]Department of Physics, University of Regensburg

Scan for digital version

Studienstiftung des deutschen Volkes

Elitenetzwerk Bayern

Summary

- Neural networks with very simple architectures can be proven to be universal function approximators [1]
  - $\Rightarrow$ Such simple networks are in theory sufficient to solve arbitrary tasks
- However, customizing the network architecture can be beneficial to achieve faster convergence
- Symmetries in the input data can lead to customized networks that are as expressive as the original one
- We investigate how symmetries can be exploited to find suitable network architectures

Motivation

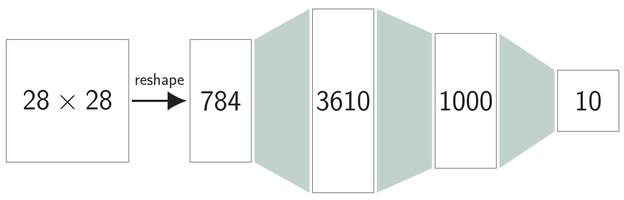
- Consider a function $f : A \to B$ which satisfies the symmetry $f(x) = f(g_\lambda(x))$ for some family of functions $g_\lambda : A \to A$ with $\lambda \in \Lambda$
  - $\Rightarrow$ f is fully determined through its restriction on the equivalence classes $[x] = \{y \in A \,|\, f(x) = f(y)\} = \{g_\lambda(x) \,|\, \lambda \in \Lambda\}$
- To train a neural network to mimic f, it would be sufficient to train it on the equivalence classes and exploit the symmetry afterwards
- This approach is usually infeasible, as representatives of an equivalence class are hard to define and the training data would have to be transformed to their corresponding representatives first

Introductory example: digit recognition

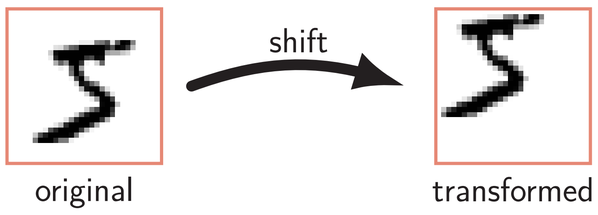
- The simplest building block for neural network architectures is the dense layer between N_i input features φ_a and N_o output features ψ_b, defined by

$$\psi_a = \sum_b w_{ab} \varphi_b$$

  with $a \in \{1, \dots, N_i\}$ and $b \in \{1, \dots, N_o\}$ and scalar weights w_{ab}
- A possible network architecture would then be



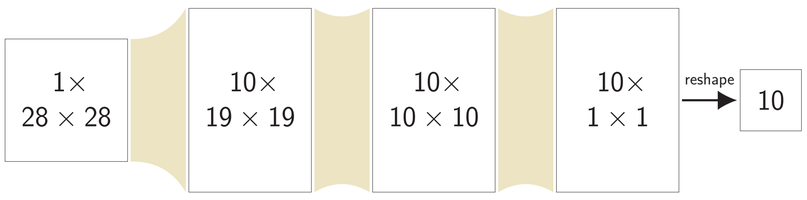
  This network architecture does not use any insights into the specific problem, therefore a lot of weights (6454860) are necessary
- We do have some prior knowledge: Regardless of where the digit is written within the image, it should be recognized in the same way



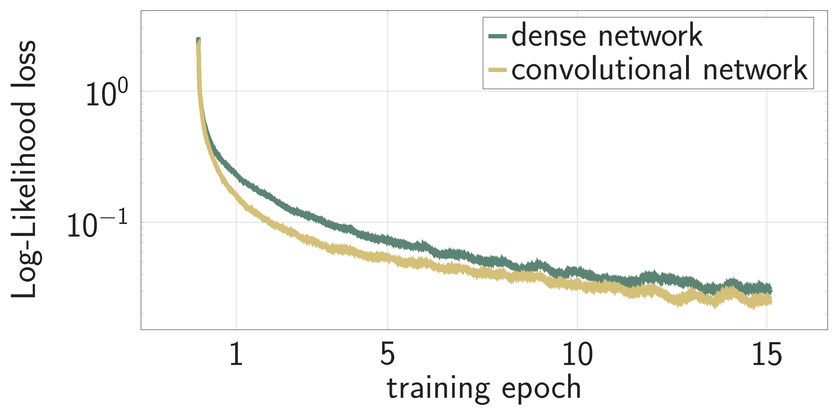
- This can be used in the network architecture by using using convolutional layers:

$$\psi^a_{x,y} = \sum_b \sum_{\delta_x, \delta_y} w^{ab}_{\delta_x \delta_y} \varphi^b_{x+\delta_x, y+\delta_y}$$

  where x and y denote coordinates within the image, a is the number of input channels and b is the number of output channels
- Convolutional layers satisfy translational invariance (up to boundary terms)
- This leads to architectures like



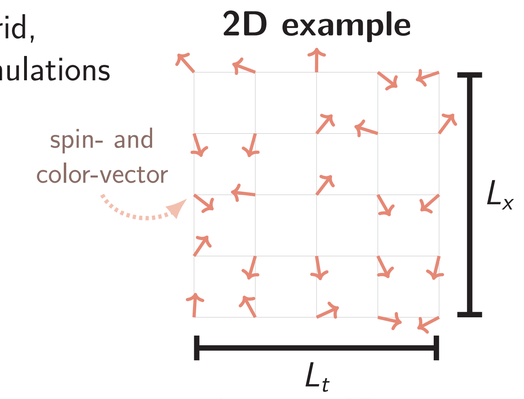
  By exploiting translational symmetry in the network architecture, the number of weights is drastically reduced to 21030
- Both networks have the same expressivity
- We can now compare their Log-Likelihood loss during training:



- Better network architecture leads to faster convergence due to reduced number of weights

Real-world example: Preconditioners in lattice QCD

- Lattice QCD discretizes space-time on a grid, wich can be used to perform numerical simulations
- A quark field on the lattice is defined as an object $\varphi \in \mathbb{R}^{L_x \times L_y \times L_z \times L_t \times 4 \times 3}$, where the different sizes are:
  - L_x, L_y, L_z, L_t: lattice sizes in all four space-time directions
  - 4: spin degrees of freedom
  - 3: color degrees of freedom



- One of the most computationally expensive steps in many lattice QCD calcaulations is the solution of the discretized Dirac equation [2]

$$D\psi = \varphi$$

  where ψ and φ are quark fields and D is a discretized Dirac operator (linear)
- To accelerate the solution of the Dirac equation, one can use preconditioners [3], i.e. an operator M which satisfies $DM \approx \mathbb{1}$
- Using M, the Dirac equation can be reformulated as

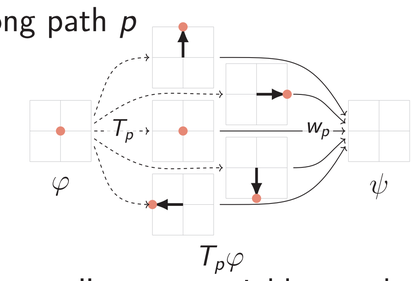
$$(DM)\chi = \varphi \qquad \text{with} \qquad \psi = M\chi$$

  which is easier to solve if MD has a smaller condition number than D
- We want to find such a preconditioner M using neural networks
  - Approach 1 – Brute force:
    - Without further knowledge, we can simply use a sequence of dense layers to learn a preconditioner
    - Every layer has one quark field as an input and one quark field as an output $\Rightarrow (4 \times 3 \times L_x L_y L_z L_t)^2$ weights per layer
  - Approach 2 – Utilize symmetries:
    - While we do not have simple translational symmetry in this case, the network should be gauge-equivariant under gauge transformations, which leads to the symmetry

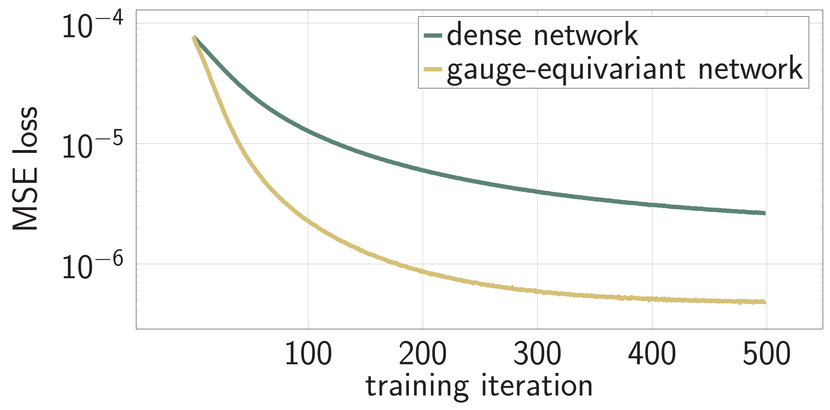
$$DT_p\psi = T_p D\psi$$

      where T_p is a parallel transport of the field along path p
    - Define a local parallel-transport-convolution (PTC) layer [2] as

$$\psi^a(x) = \sum_b \sum_{\text{paths } p} w^{ab}(x)_p T_p \varphi^b(x)$$



    - The parallel transports act on the color space $\Rightarrow (9 \times 4 \times L_x L_y L_z L_t)^2$ weights per layer if we use all nearest-neighbor paths



- Faster convergence even though number of parameters is larger!

Takeaway messages

- Symmetries are often found within data used to train neural networks
- Incorporating symmetries into the network architecture can reduce the number of parameters without reducing expressivity
- Networks obeying symmetry in the data can train more efficiently as the symmetry does not have to be learned

References

[1] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[2] C. Lehner and T. Wettig. "Gauge-equivariant neural networks as preconditioners in lattice QCD". In: *Phys. Rev. D* 108 (3 Aug. 2023), p. 034503.

[3] A. J. Wathen. "Preconditioning". In: *Acta Numerica* 24 (2015), pp. 329–376.

Supported by the German Academic Scholarship Foundation and the Marianne-Plehn Programme

Correspondence: simon.pfahler@ur.de