Universität Augsburg
Fakultät für Angewandte Informatik

Colloquium Bachelor Thesis

Investigation of transformer architectures for geometrical graph structures and their application to two-dimensional spin systems

Jonas Kell

Augsburg, 08.11.2022

Outline

The Ising Model & Quantum Mechanical Ground State

Physical Theory

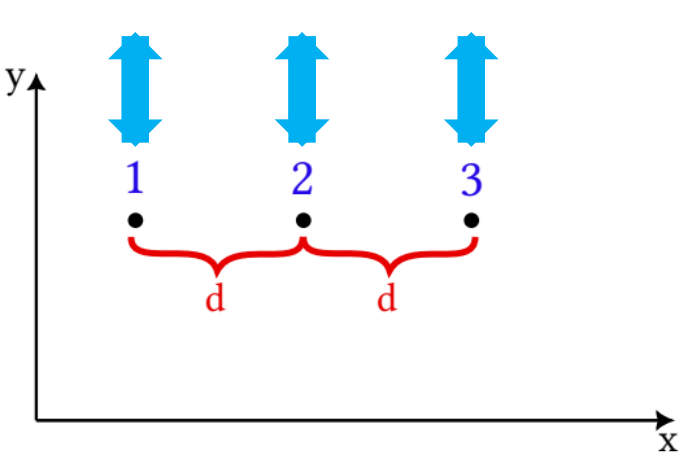
- ***Ising Model***: System of positionally fixed ***spins*** that may freely rotate

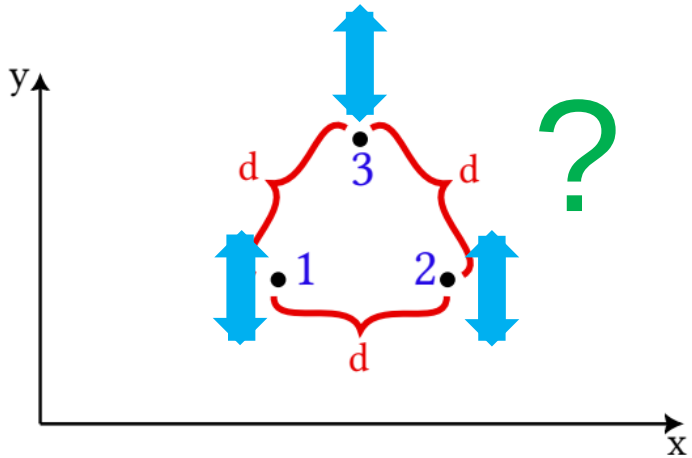
$$\mathcal{H}\,|\Psi\rangle = E \cdot |\Psi\rangle$$

- Physics of the system described by the ***Schrödinger Equation***

$$\mathcal{H} = J \cdot \sum_{\langle i,j \rangle} \hat{s}_i^z \hat{s}_j^z + h \cdot \sum_i \hat{s}_i^x$$

- Goal is to find the system‘s eigenstate with the ***smallest eigenvalue*** (= ***ground state***)





$$E(\theta) = \frac{\langle \Psi_\theta | \mathcal{H} | \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle} \approx \frac{1}{N_{\mathrm{MC}}} \sum_{n=1}^{N_{\mathrm{MC}}} \mathcal{H}_{\mathrm{loc}}^{\theta}(s_{(n)})$$

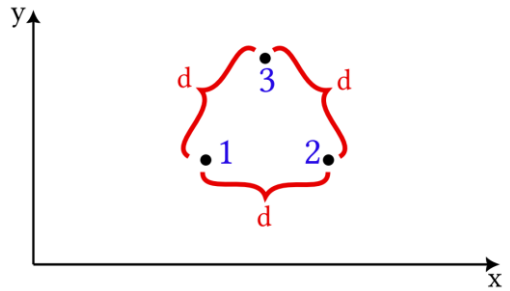
$$= \frac{1}{N_{\mathrm{MC}}} \sum_{n=1}^{N_{\mathrm{MC}}} \left(\sum_{s'} \langle s_{(n)} | \mathcal{H} | s' \rangle \frac{\langle s' | \Psi_\theta \rangle}{\langle s_{(n)} | \Psi_\theta \rangle} \right)$$

Solution through exact Diagonalization

Physical Theory

$$\mathcal{H} = J \cdot \sigma_1^z \sigma_2^z + J \cdot \sigma_2^z \sigma_3^z + J \cdot \sigma_1^z \sigma_3^z + h\sigma_1^x + h\sigma_2^x + h\sigma_3^x$$

$$1: |\uparrow,\uparrow,\uparrow\rangle \quad 2: |\uparrow,\uparrow,\downarrow\rangle \quad 3: |\uparrow,\downarrow,\uparrow\rangle \quad 4: |\uparrow,\downarrow,\downarrow\rangle$$
$$5: |\downarrow,\uparrow,\uparrow\rangle \quad 6: |\downarrow,\uparrow,\downarrow\rangle \quad 7: |\downarrow,\downarrow,\uparrow\rangle \quad 8: |\downarrow,\downarrow,\downarrow\rangle$$



- Algorithm
  - Write down system of base functions
  - Evaluate effect of the Hamiltonian on them
  - Diagonalize resulting matrix (smallest eigenvalue)

$$\begin{array}{c} \\ \langle\uparrow,\uparrow,\uparrow| \\ \langle\uparrow,\uparrow,\downarrow| \\ \langle\uparrow,\downarrow,\uparrow| \\ \langle\uparrow,\downarrow,\downarrow| \\ \langle\downarrow,\uparrow,\uparrow| \\ \langle\downarrow,\uparrow,\downarrow| \\ \langle\downarrow,\downarrow,\uparrow| \\ \langle\downarrow,\downarrow,\downarrow| \end{array} \begin{array}{c} \begin{array}{cccccccc} |\uparrow,\uparrow,\uparrow\rangle & |\uparrow,\uparrow,\downarrow\rangle & |\uparrow,\downarrow,\uparrow\rangle & |\uparrow,\downarrow,\downarrow\rangle & |\downarrow,\uparrow,\uparrow\rangle & |\downarrow,\uparrow,\downarrow\rangle & |\downarrow,\downarrow,\uparrow\rangle & |\downarrow,\downarrow,\downarrow\rangle \end{array} \\ \begin{pmatrix} 3\cdot J & h & h & 0 & h & 0 & 0 & 0 \\ h & -1\cdot J & 0 & h & 0 & h & 0 & 0 \\ h & 0 & -1\cdot J & h & 0 & 0 & h & 0 \\ 0 & h & h & -1\cdot J & 0 & 0 & 0 & h \\ h & 0 & 0 & 0 & -1\cdot J & h & h & 0 \\ 0 & h & 0 & 0 & h & -1\cdot J & 0 & h \\ 0 & 0 & h & 0 & h & 0 & -1\cdot J & h \\ 0 & 0 & 0 & h & 0 & h & h & 3\cdot J \end{pmatrix} \end{array}$$

- Simple mathematical calculation
  - Exponential memory requirement
  - Exponential time complexity

Solutions with Neural Quantum States

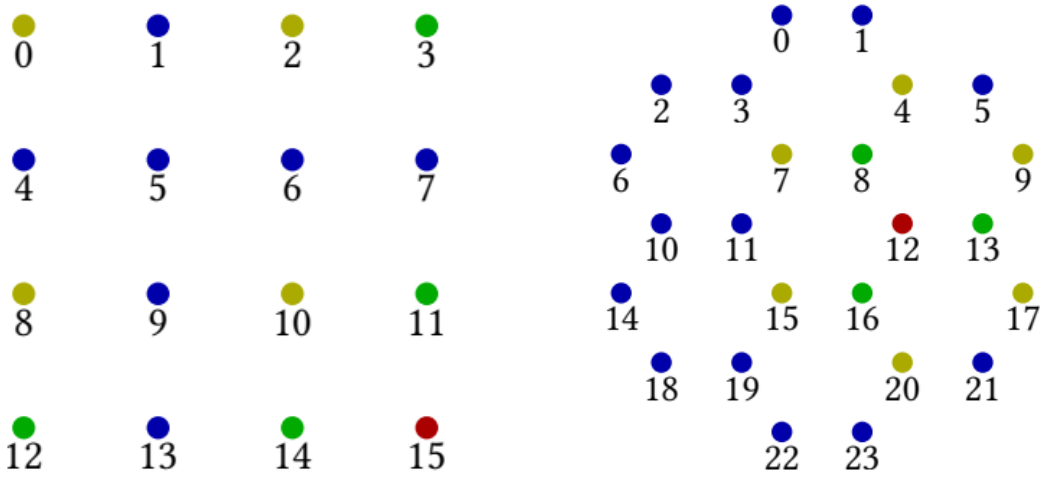
Physical Theory

- The wavefunction is ***parametrized*** by a neural network (= ***neural quantum state***)
  - Parametrization denser, therefore less parameters needed ✚
- Conventional neural network backpropagation is used to improve the network's performance ***iteratively***
  - Step by step refinement allows for gradual improvement instead of needing to calculate the full solution in one step ✚
- As this is an approximation, only a subset of the input space needs to be sampled ✚
- Different update strategies can be motivated and then used
  - Variational Monte Carlo
  - Diffusion Monte Carlo

New Idea: Why Graphs should help

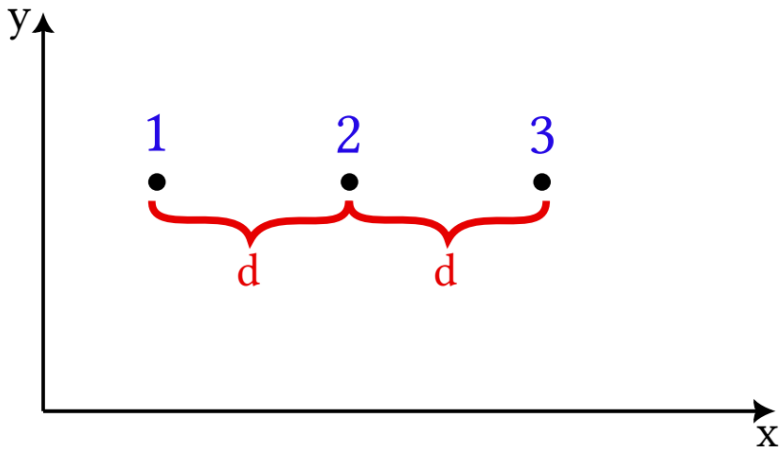
Physical Theory

- Some crystal structures are not able to be directly encoded into a square tensor format
  - Even if they can, it requires specialized encoding
  - Pure graph networks are encoding independent ✚



- Ising model interactions (nearest neighbor only) reflect the graph symmetry
  - Non-Interacting spin sites are already masked out, according to the graph definition ✚
  - Learning this structure is no longer necessary ✚

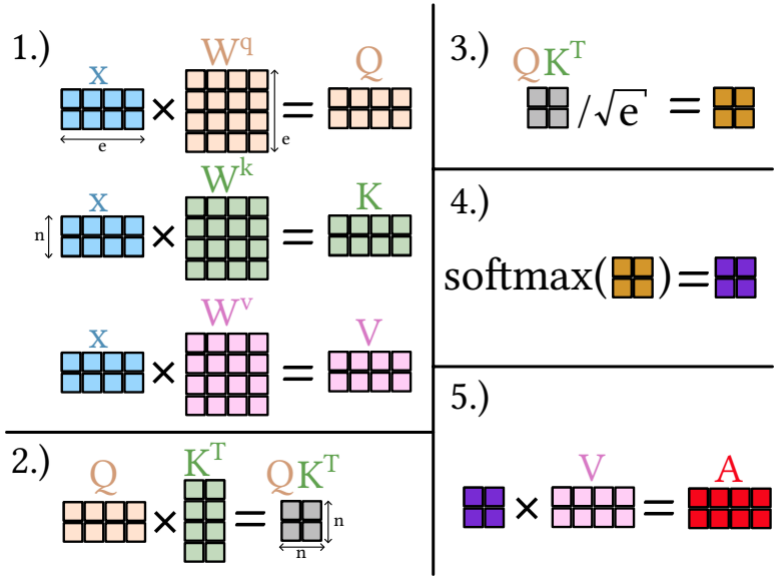
$$\mathcal{H} = J \cdot \sum_{\langle i,j \rangle} \hat{s}_i^z \hat{s}_j^z + h \cdot \sum_i \hat{s}_i^x$$



Transformers: short overview

Machine Learning Architectures

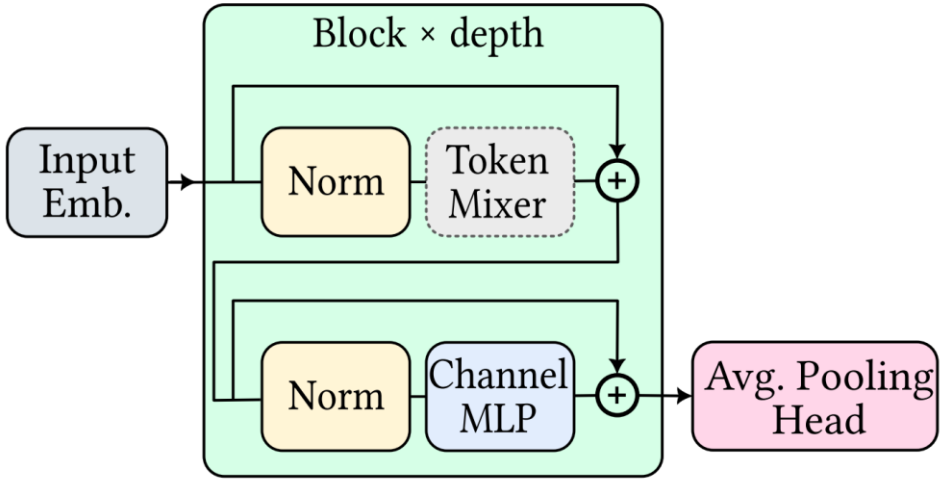
- Has origins/motivated by natural language translation
- Traditionally uses the ***attention*** algorithm for computing inter-token interactions
- Alternating use of ***attention*** and ***multi layer perceptrons*** to utilize the strengths of both operations
- Expensive [$O(n^2)$], but highly effective
  - Good in representing relations among objects with arbitrary relative positions
  - Trainable and weights get calculated dynamically

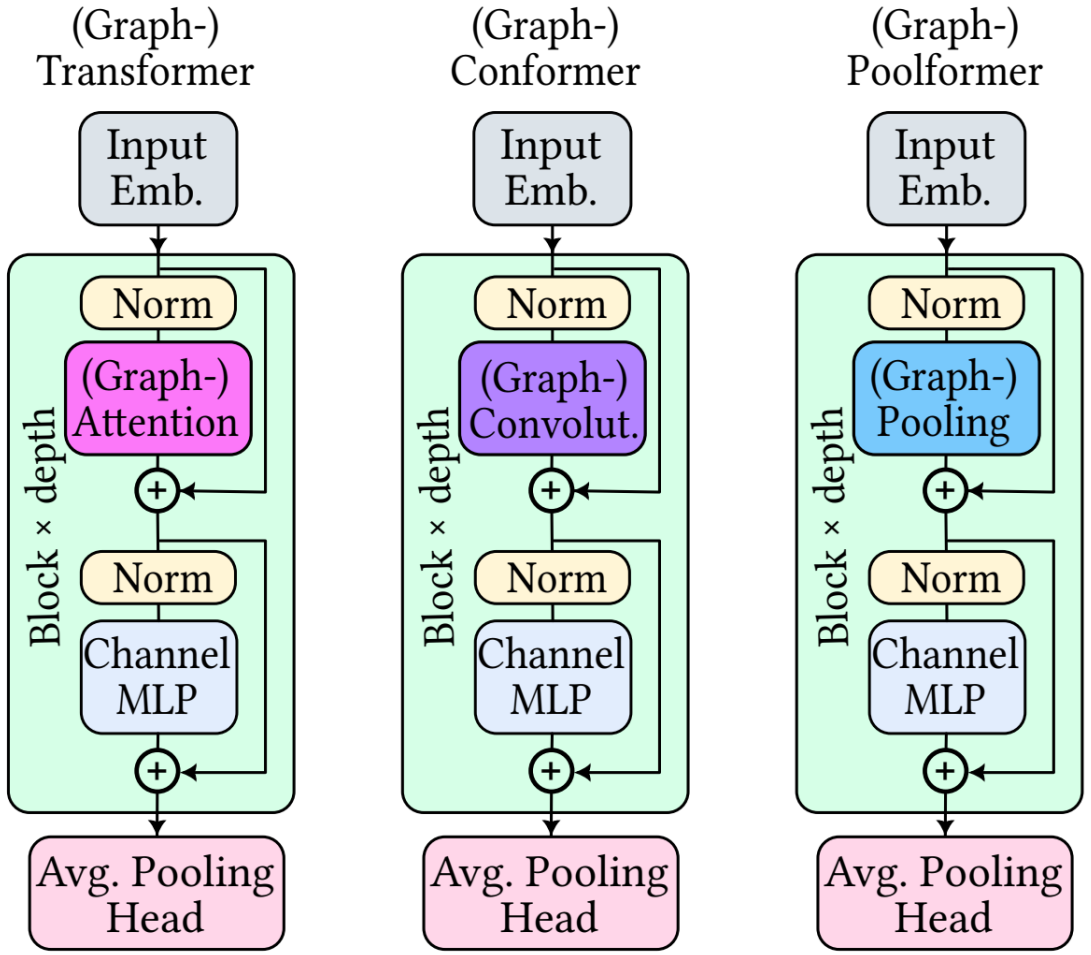


From the Transformer to the Metaformer

Machine Learning Architectures

- Switch out ***Token-Mixer***, keep the rest

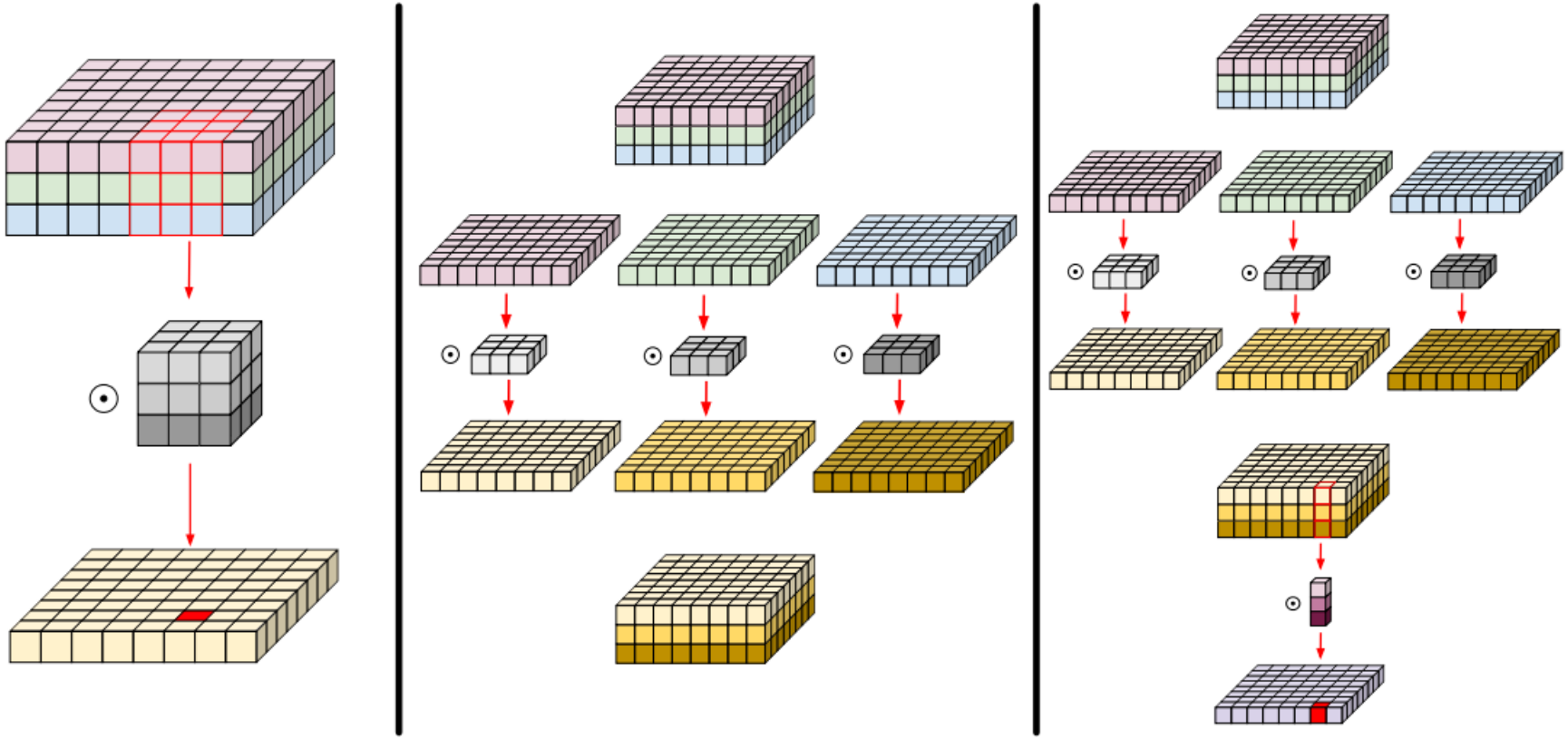




Depthwise Convolutions and Separable Convolutions

Machine Learning Architectures

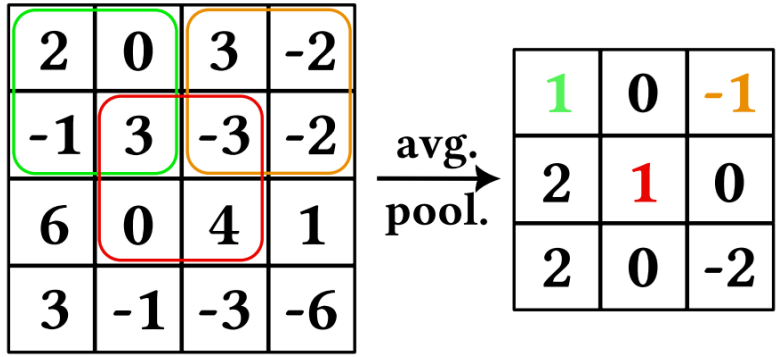
- Full Convolution
- Depthwise Convolution
- Depthwise Separable Convolution



Now where exactly do the Graphs come into play?

Machine Learning Architectures

- Extending established operators to make them applicable to non-tensor like data
  - Convolutions → Graph Convolutions
  - Pooling → Graph Pooling



- Augmenting already compatible operators to provide a supplementary inductive bias
  - Attention → Graph Attention

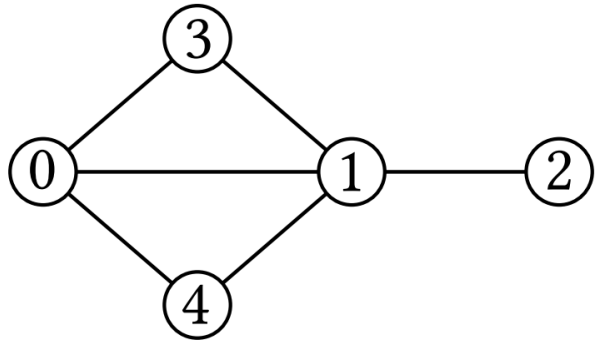
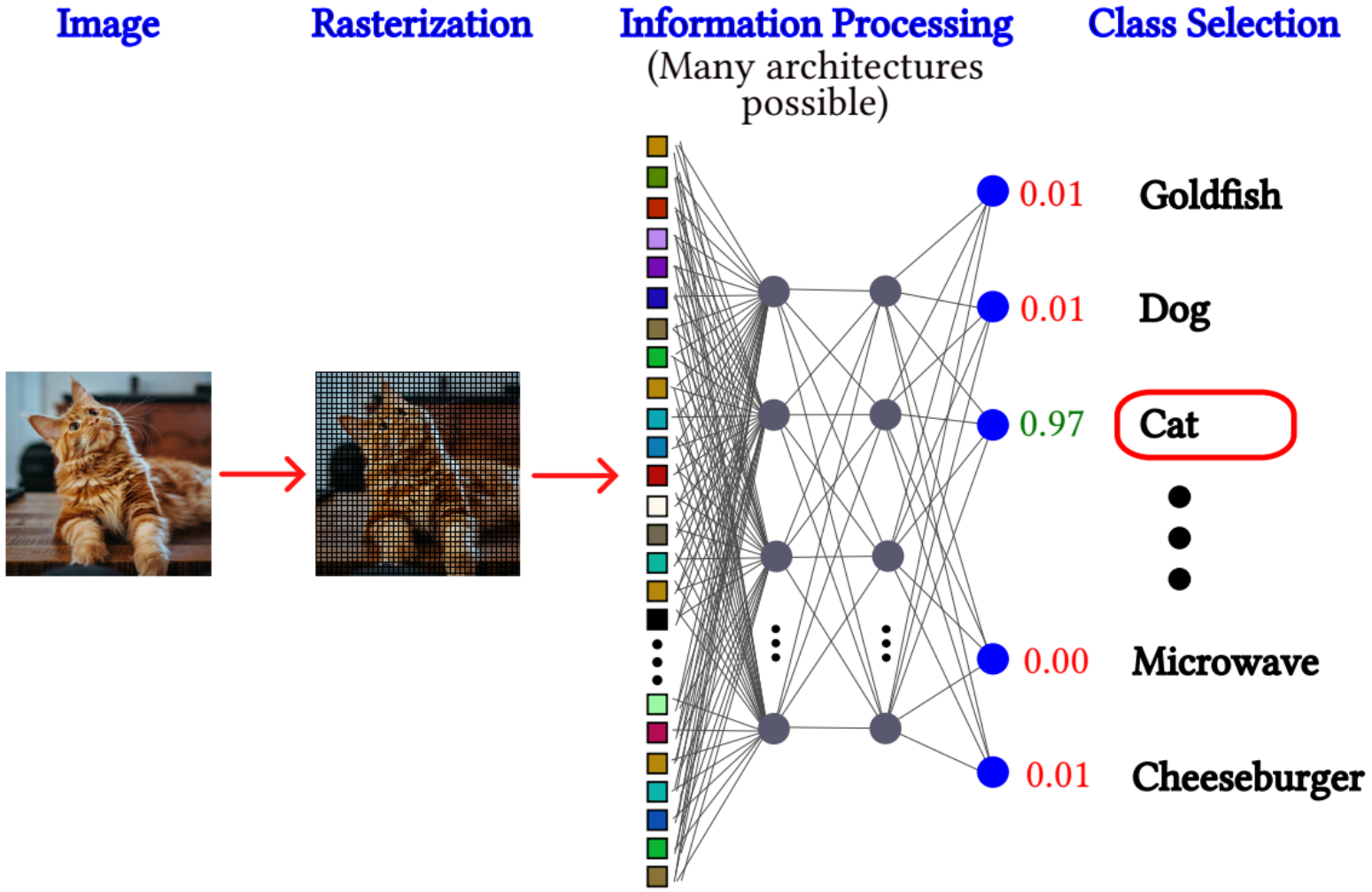


Image Classification

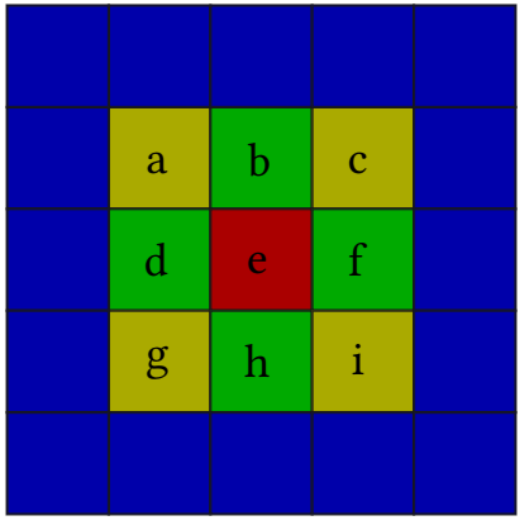
Experiments & Verification: Image Classification

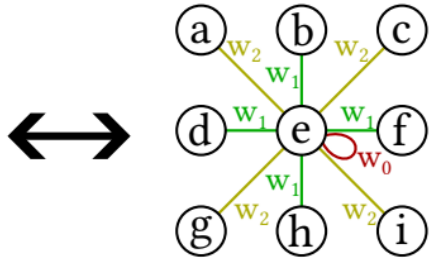


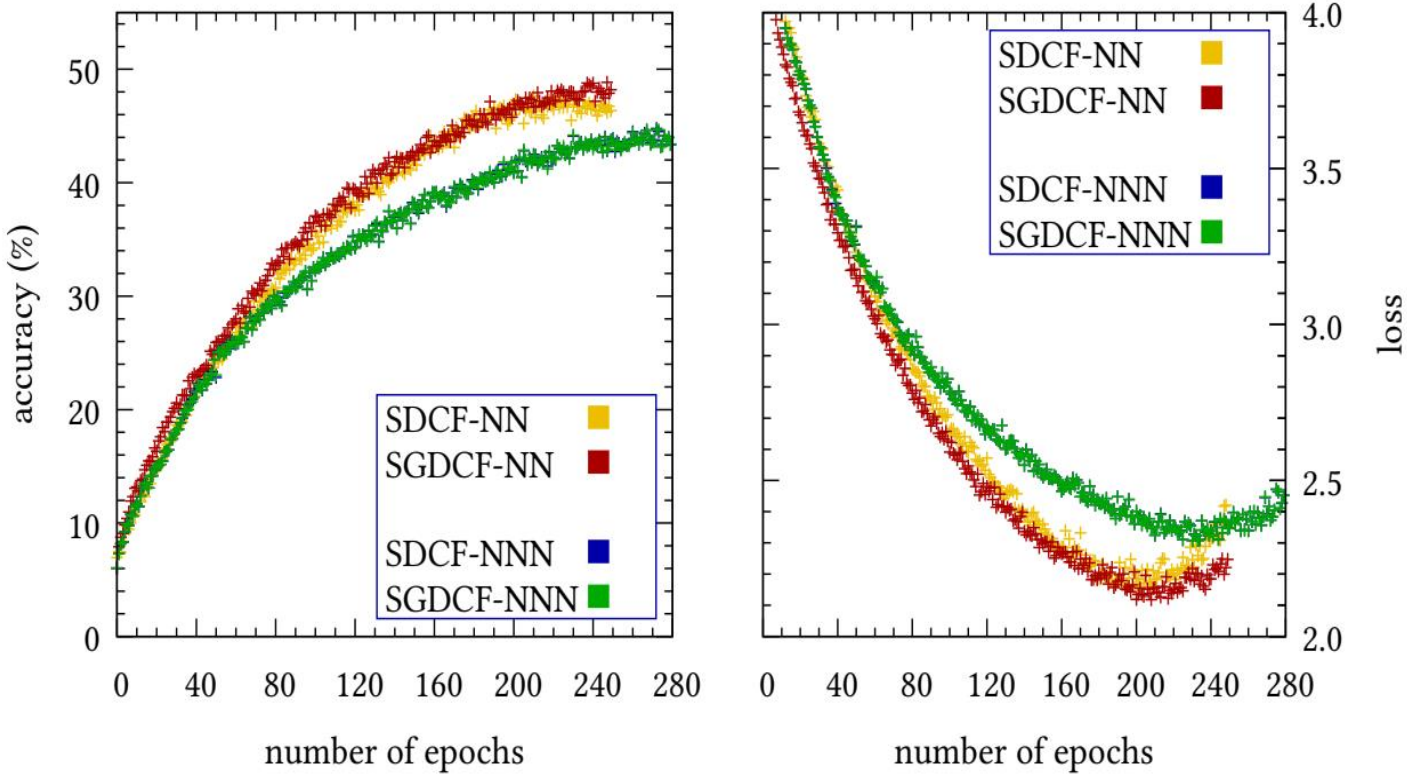
Conventional Convolutions and Graph Convolutions

Experiments & Verification: Image Classification

- For image data, symmetric- and graph-symmetric-convolutions are equivalent



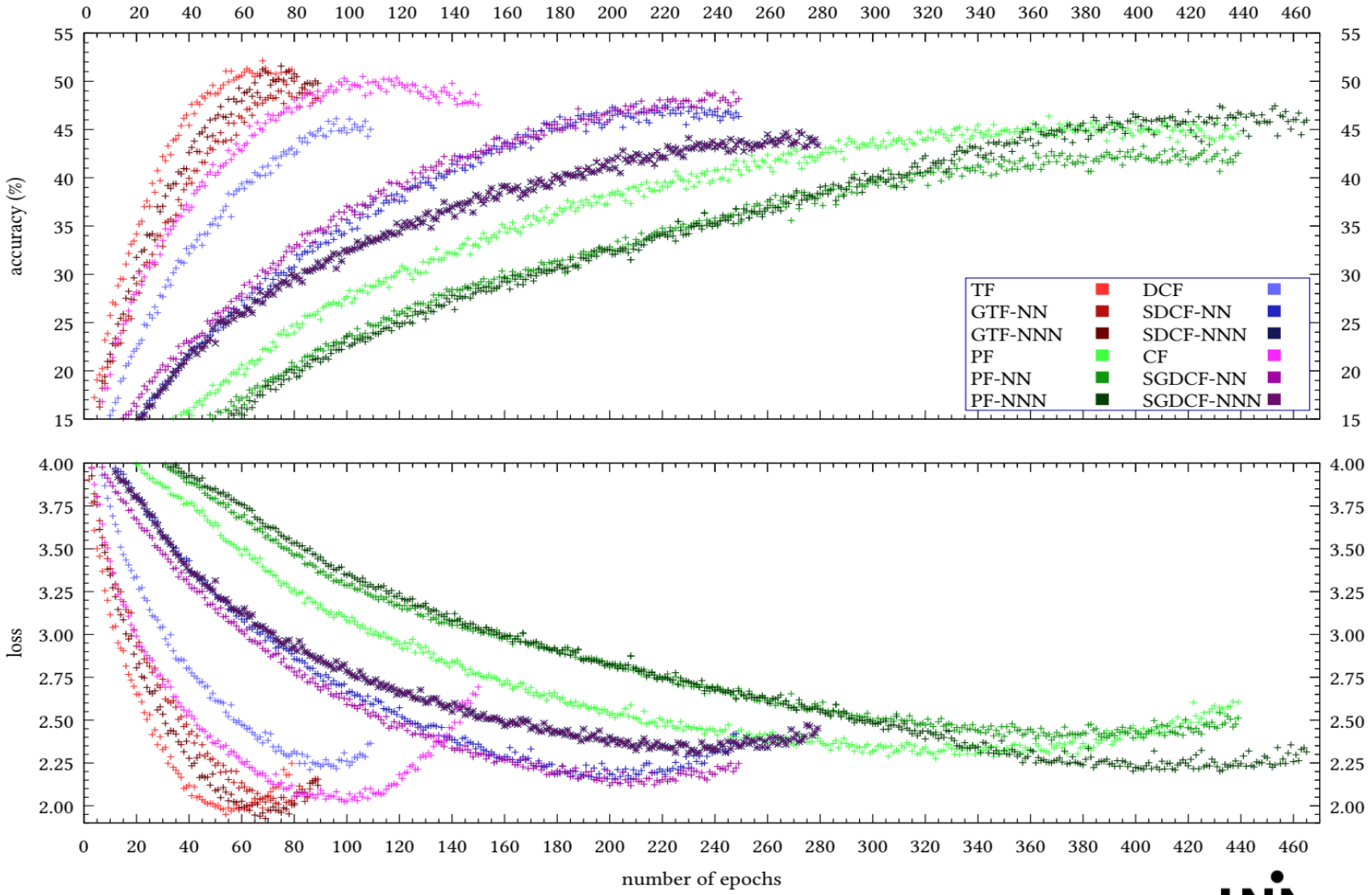




Comparison of different Token Mixers: Visualized

Experiments & Verification: Image Classification

- Transformer
  - High accuracy
  - Converges fast
- (Symmetric-Graph-) Conformer
  - Medium accuracy
  - Converges slower
- Poolformer
  - Worst accuracy
  - Slowest rate of convergence



Comparison of different Token Mixers: Quantized

Experiments & Verification: Image Classification

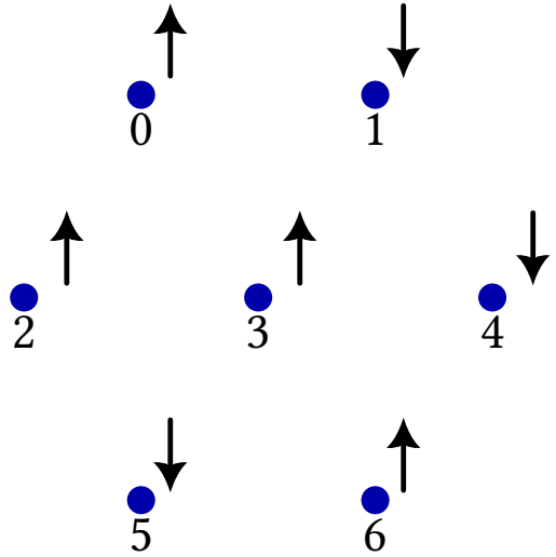
- Transformer
  - Biggest number of weights
  - Longest evaluation time
- (Symmetric-Graph-) Conformer
  - Significantly less parameters
  - Faster evaluation
  - Almost comparable accuracy
- Poolformer
  - Extremely fast and efficient evaluation
  - Smallest model
  - Performance not far behind others

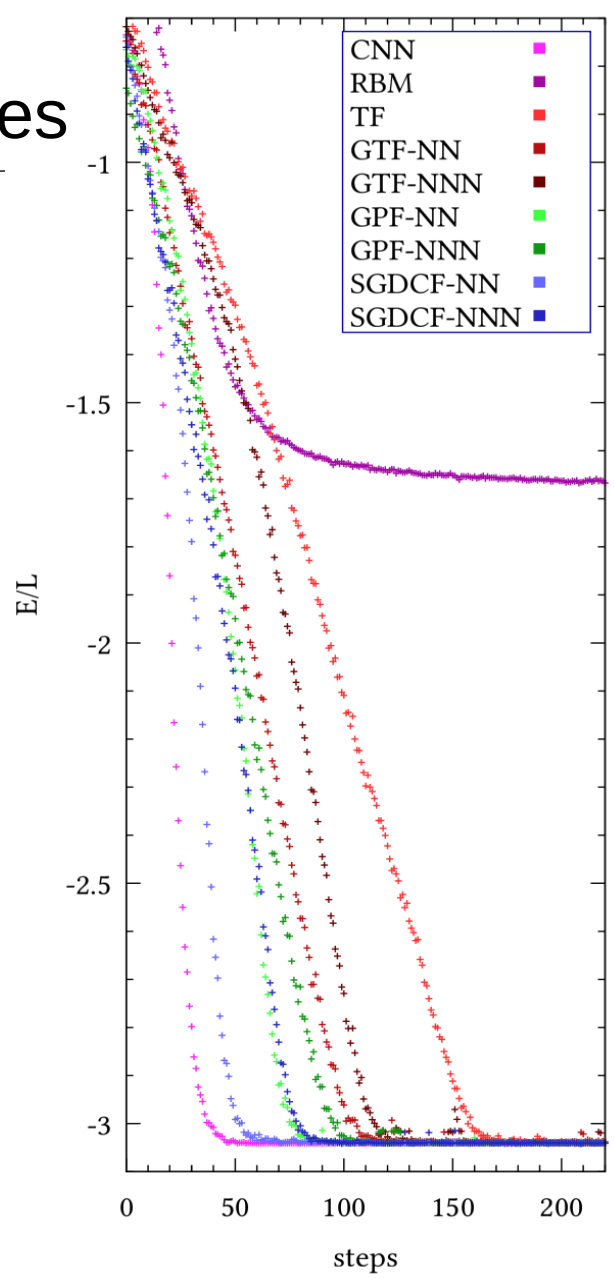
name	acc_{max}	ep_{max}	#params	t / ep	t_{max}	$\frac{t}{\text{param}}$	$\frac{t_{\text{max}}}{\text{acc}}$	$\frac{\text{param}}{\text{acc}}$
TF (learned)	51.30 %	62	5 543 332	391.01 s	6.73 h	1.00	1.00	1.00
TF (sinus)	51.03 %	66	5 505 700	391.32 s	7.17 h	1.01	1.07	1.00
TF (none)	49.11 %	63	5 505 700	391.22 s	6.85 h	1.01	1.06	1.04
GTF-NN (learned)	50.01 %	80	5 543 368	435.25 s	9.67 h	1.11	1.47	1.03
GTF-NN (sinus)	49.38 %	68	5 505 736	435.53 s	8.23 h	1.12	1.27	1.03
GTF-NN (none)	48.42 %	80	5 505 736	435.73 s	9.68 h	1.12	1.52	1.05
GTF-NNN (learned)	51.58 %	75	5 543 368	435.27 s	9.07 h	1.11	1.34	0.99
GTF-NNN (sinus)	49.87 %	70	5 505 736	435.47 s	8.47 h	1.12	1.29	1.02
GTF-NNN (none)	50.38 %	79	5 505 736	434.84 s	9.54 h	1.12	1.44	1.01
PF	46.39 %	367	3 727 012	217.14 s	22.14 h	0.83	3.64	0.74
GPF-NN	43.26 %	403	3 727 012	213.39 s	23.89 h	0.81	4.21	0.80
GPF-NNN	47.43 %	432	3 727 012	213.40 s	25.61 h	0.81	4.11	0.73
DCF	46.13 %	96	3 747 748	317.41 s	8.46 h	1.20	1.40	0.75
SDCF-NN	47.66 %	210	3 731 620	319.98 s	18.67 h	1.22	2.98	0.72
SDCF-NNN	44.49 %	260	3 733 924	319.72 s	23.09 h	1.21	3.95	0.78
SGDCF-NN	48.78 %	237	3 731 620	250.64 s	16.50 h	0.95	2.58	0.71
SGDCF-NNN	44.39 %	260	3 733 924	280.66 s	20.27 h	1.07	3.48	0.78
CF	50.50 %	106	7 710 628	386.45 s	11.38 h	0.71	1.72	1.41

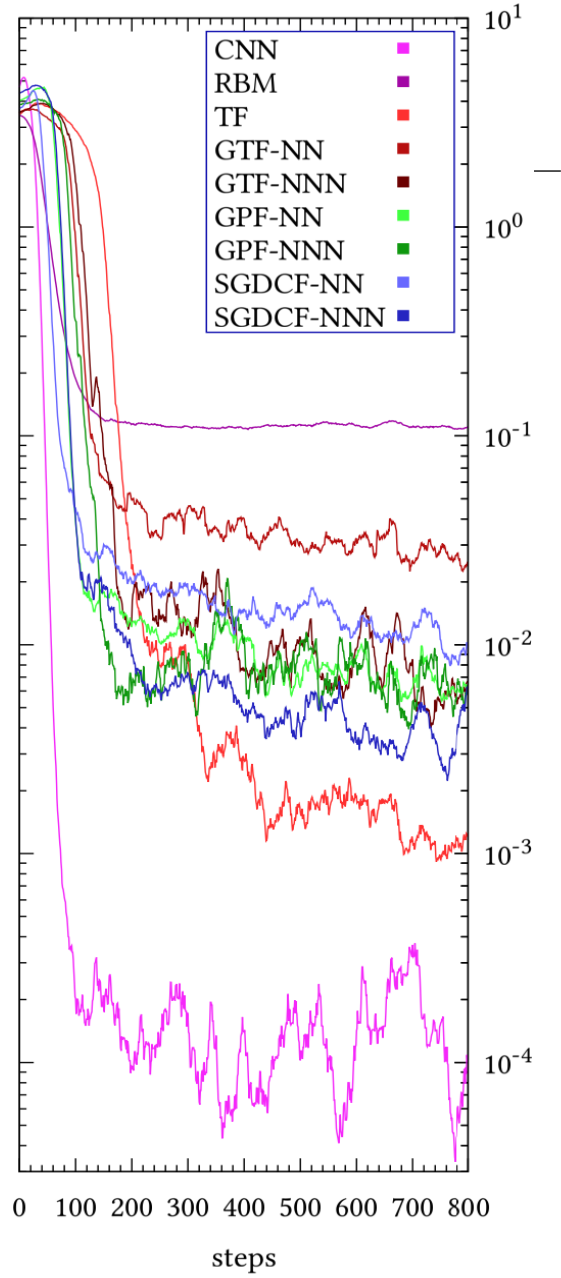
Comparison to Established Architectures

Experiments & Verification: Ground State Search

- Comparing predefined and custom models on the ground state search task

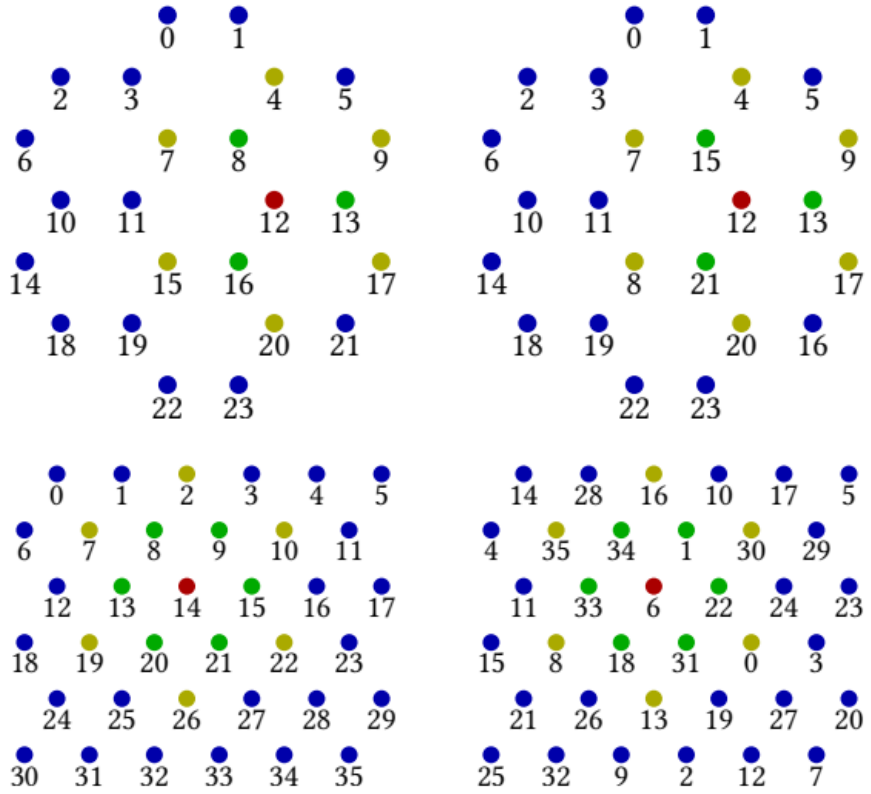


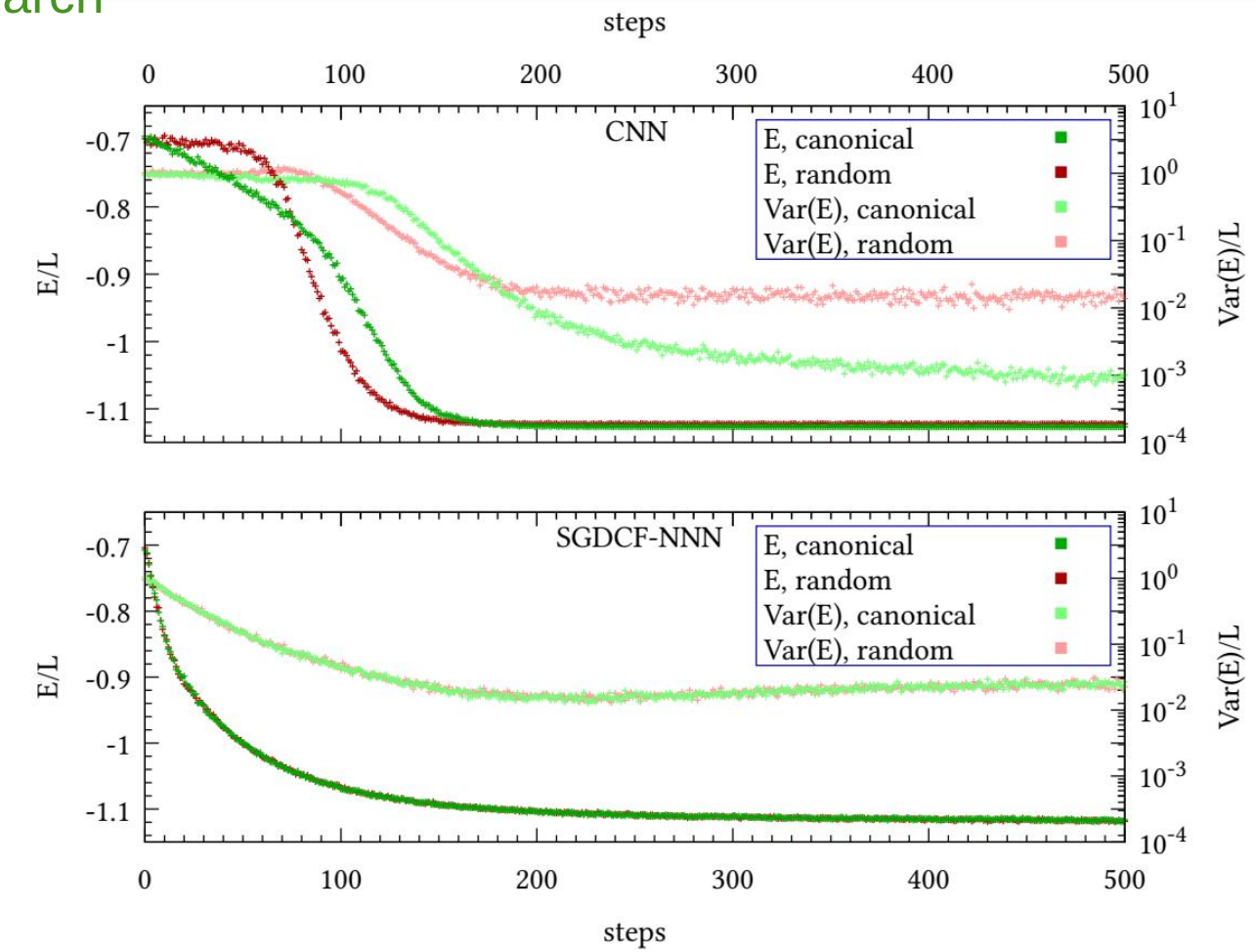




Resiliency to the Choice of Lattice Encoding

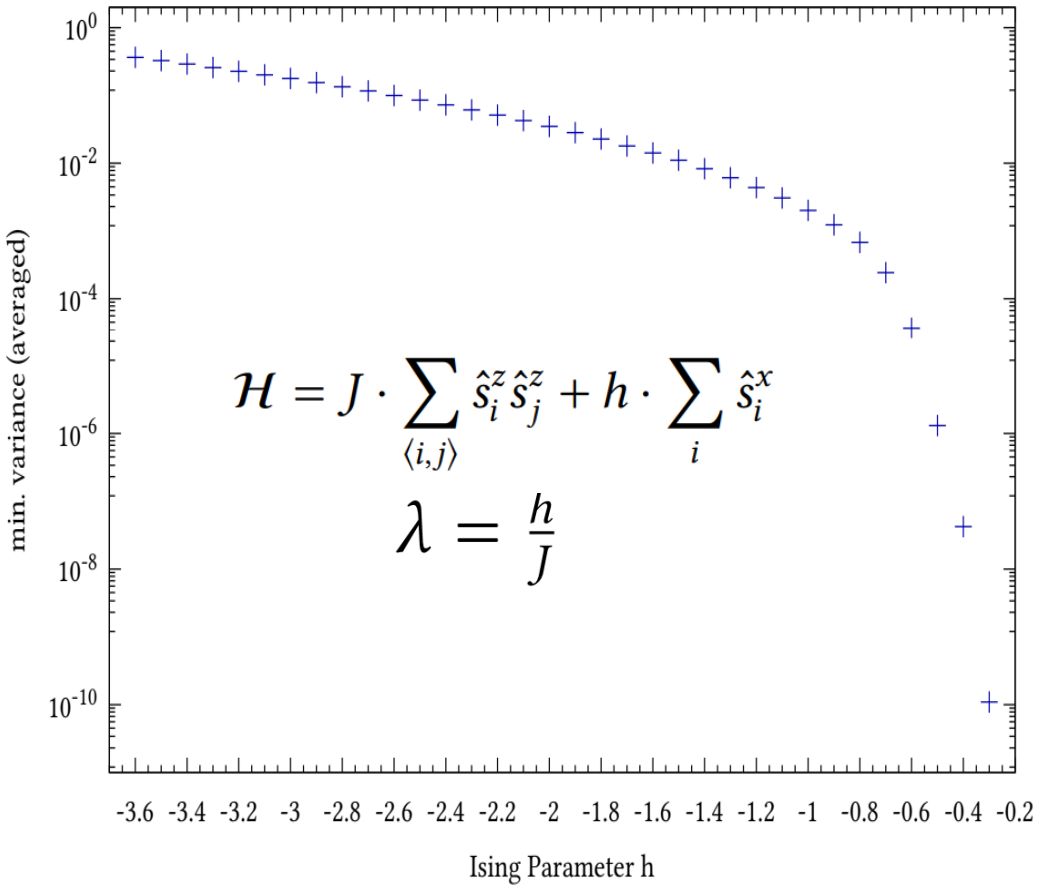
Experiments & Verification: Ground State Search

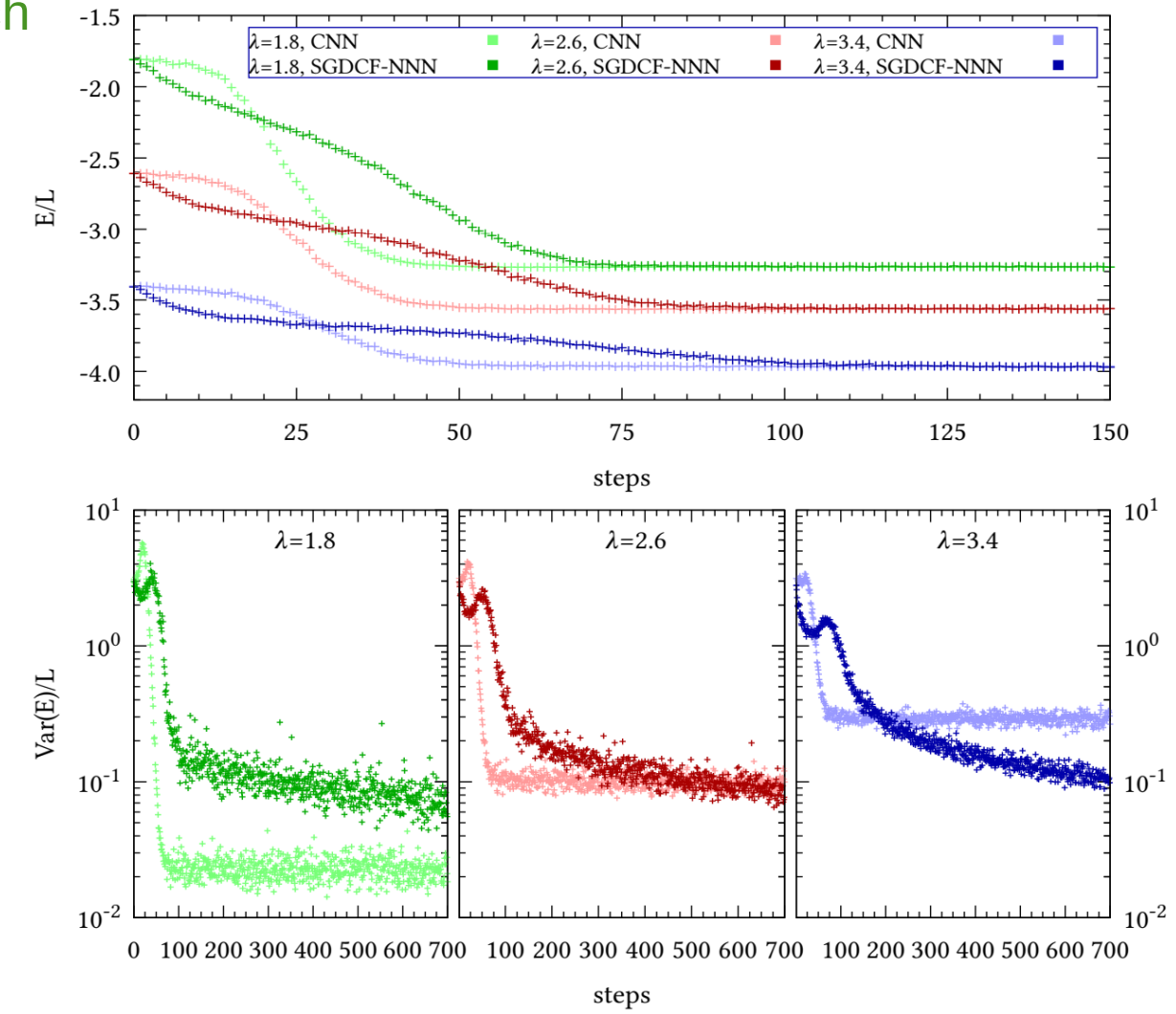




Differences across the Phase Diagram

Experiments & Verification: Ground State Search





Conclusion

Conclusion & Future Work

- Method provides:
  - Compatibility with graph data
    - While also providing augmentations to conventional methods
  - Resiliency against choice of input encoding
  - High customizability
    - Many independent hyperparameters
  - Vast computational potential
    - Transformer architecture
  - Stability in the phase transition region

Future Work

Conclusion & Future Work

- Employ the flexible graph-metaformers in related technical fields
  - There are several datasets available, that label themselves as “graph-dataset”. Whether they can be evaluated more efficiently/precisely with graph networks remains to be found out.

- Better implementation of the graph algorithm lowers computational cost
  - Theoretically graph-masked calculation is less computationally demanding (not yet in my implementation)
  - Further improve by implementing directly in GPU microcode
  - Even further improve by solving the computation with dedicated hardware

- Research the application on even more complex problems
  - Harder to express wavefunctions, closer to critical regimes
    - Could prove to make even better use of the transformer’s computational power
  - Calculations on more irregular lattices (with defects like vacancies, phase or grain boundaries)
    - Could prove to make the graph operations even more important for efficient calculations

Thank you for your kind attention

Universität Augsburg
Fakultät für Angewandte
Informatik

Jonas Kell
Augsburg University
jonas.kell@uni-augsburg.de
www.uni-augsburg.de

References

Text and Image References

- Main Thesis Document and data: https://github.com/jonas-kell/bachelor-thesis-documents
- Code Implementation Computer Science: https://github.com/jonas-kell/bachelor-thesis-experiments
- Code Implementation Physics: https://github.com/jonas-kell/bachelor-thesis-code

All graphs and visualizations shown are extracted from the thesis itself and were created only for use in the thesis.

If they were inspired by a pre-existing visualization, the reference is given in the thesis.

All data graphs were plotted from data, specifically recorded for the thesis. No foreign data is used in the plots.

Imaginary Time Evolution

Supplementary Slides

$$\mathcal{H}\,|\Psi(t)\rangle = i\hbar\frac{\partial}{\partial t}\,|\Psi(t)\rangle$$

$$|\Psi(t)\rangle = e^{-i\mathcal{H}t/\hbar}\,|\Psi(0)\rangle$$

$$\begin{aligned} i\hbar\frac{\partial}{\partial t}\,|\Psi(t)\rangle &= i\hbar\frac{\partial}{\partial t}e^{-i\mathcal{H}t/\hbar}\,|\Psi(0)\rangle \\ &= \frac{-i\cdot i\mathcal{H}\hbar}{\hbar}e^{-i\mathcal{H}t/\hbar}\,|\Psi(0)\rangle \\ &= \mathcal{H}\,|\Psi(t)\rangle \end{aligned}$$

$$\begin{aligned} |\Psi(t)\rangle &= e^{-i\mathcal{H}t/\hbar}\,|\Psi(0)\rangle \\ &= \sum_n e^{-i\mathcal{H}t/\hbar}\,\langle n|\Psi(0)\rangle\,|n\rangle \\ &= \sum_n e^{-iE_nt/\hbar}\,\langle n|\Psi(0)\rangle\,|n\rangle \end{aligned}$$

$$\tau = i\cdot t \qquad |\Psi(\tau)\rangle = \sum_n e^{-E_n\tau/\hbar}\,\langle n|\Psi(0)\rangle\,|n\rangle$$

$$\lim_{\tau\to\infty}\frac{|\Psi(\tau)\rangle}{e^{-E_0\tau/\hbar}} = \langle 0|\Psi(0)\rangle\,|0\rangle$$

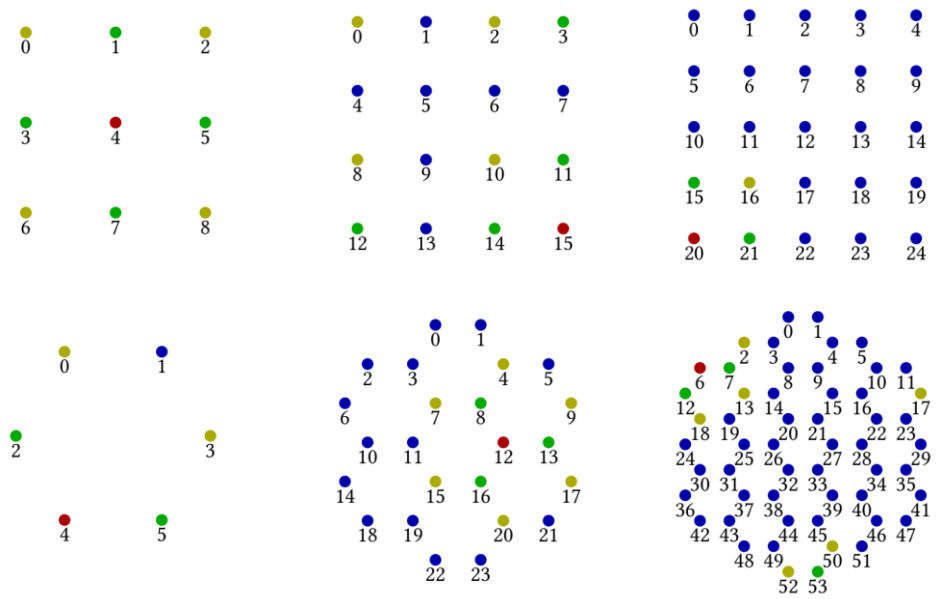
General Lattice Data

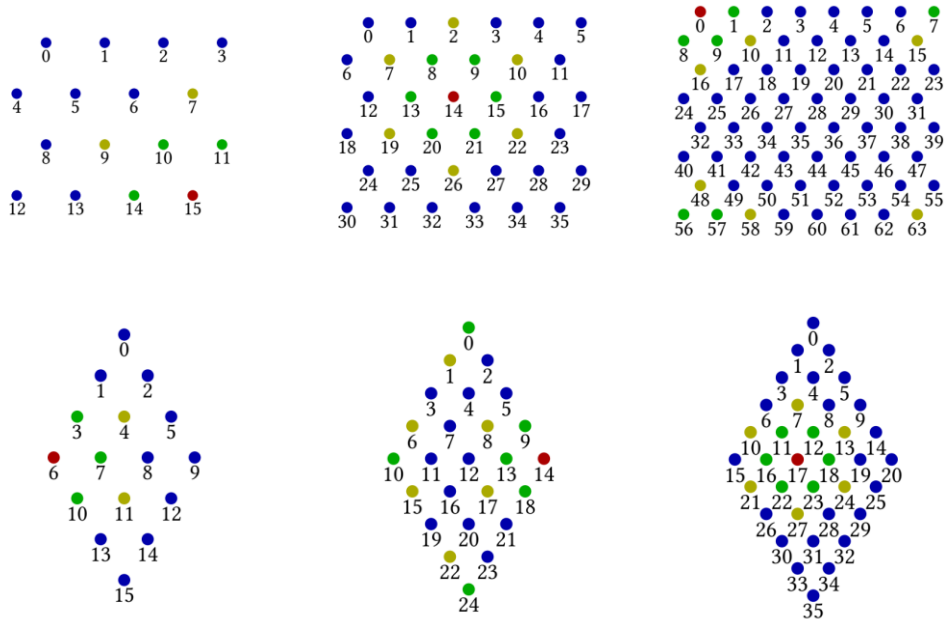
Supplementary Slides

Lattice name	#nn	#nnn	$L(1)$	$L(2)$	$L(3)$	$L(4)$	$L(5)$	$L(6)$	$L(7)$
linear	2	2	1	2	3	4	5	6	7
square	4	4	4	9	16	25	36	49	64
trigonal_square	6	6	4	16	36	64	100	144	196
trigonal_diamond	6	6	4	9	16	25	36	49	64
trigonal_hexagonal	6	6	7	19	37	61	91	127	169
hexagonal	3	6	6	24	54	96	150	216	294

Exemplary Selection of Lattices

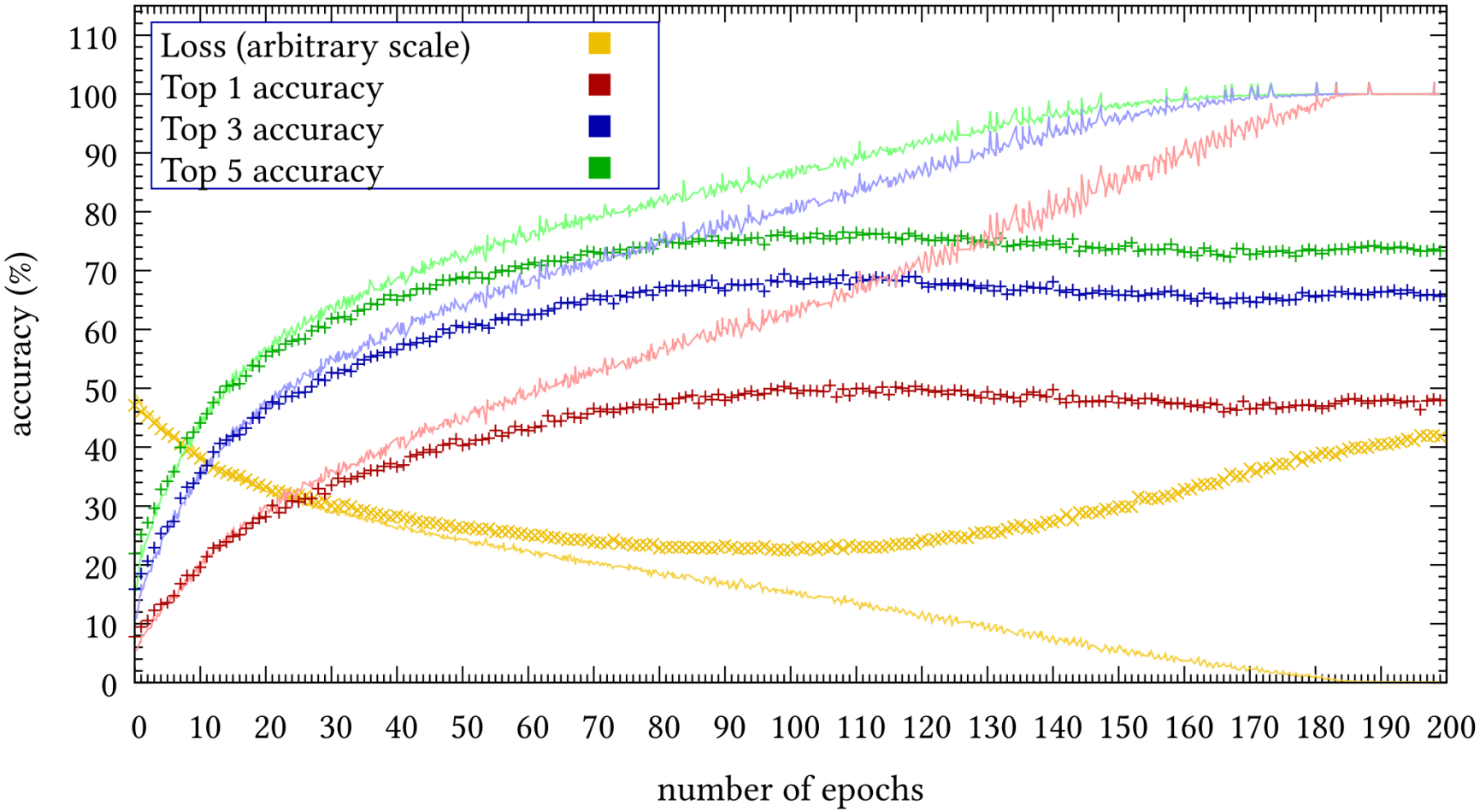
Supplementary Slides





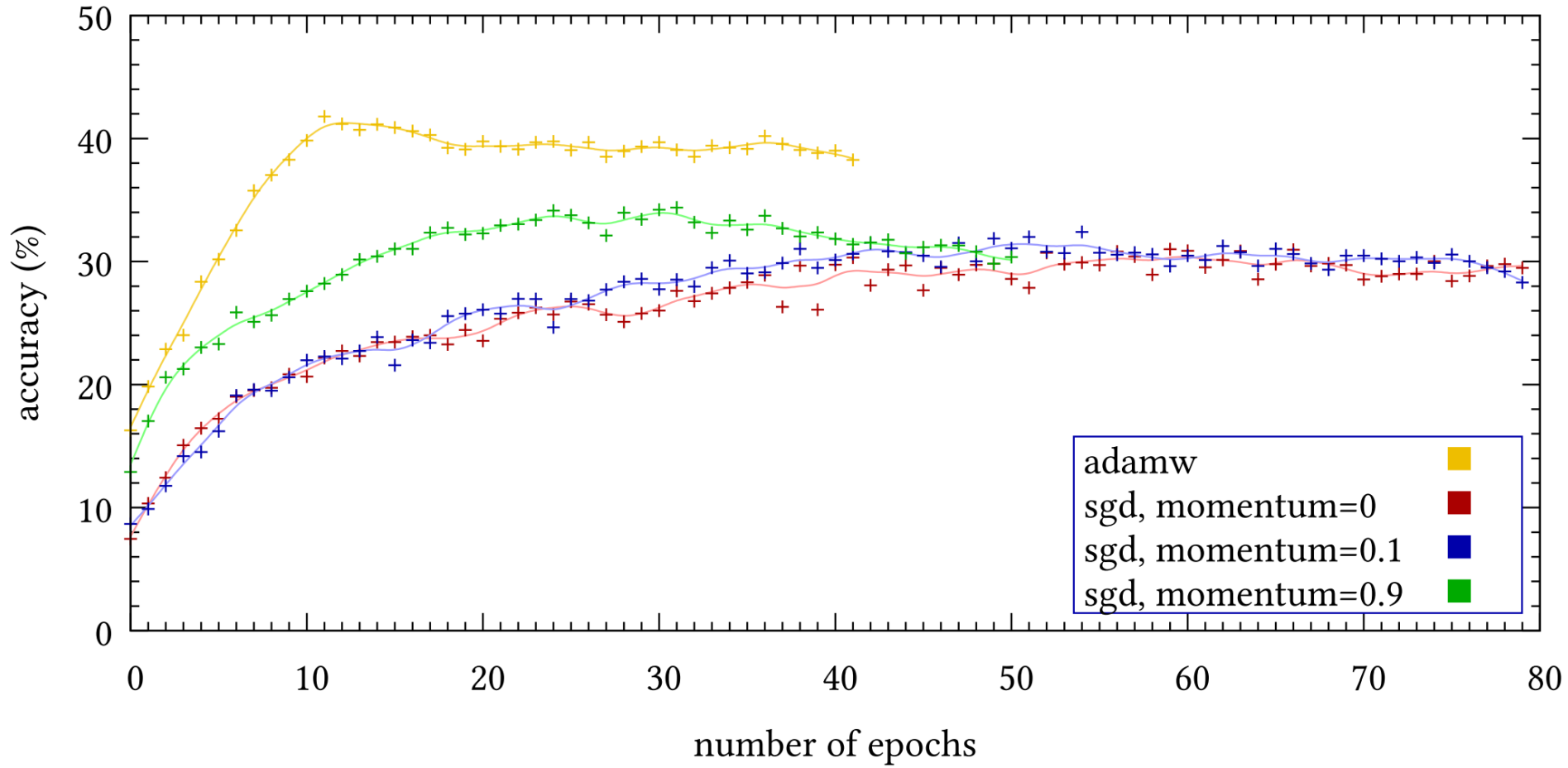
Explanation of recorded Parameters

Supplementary Slides



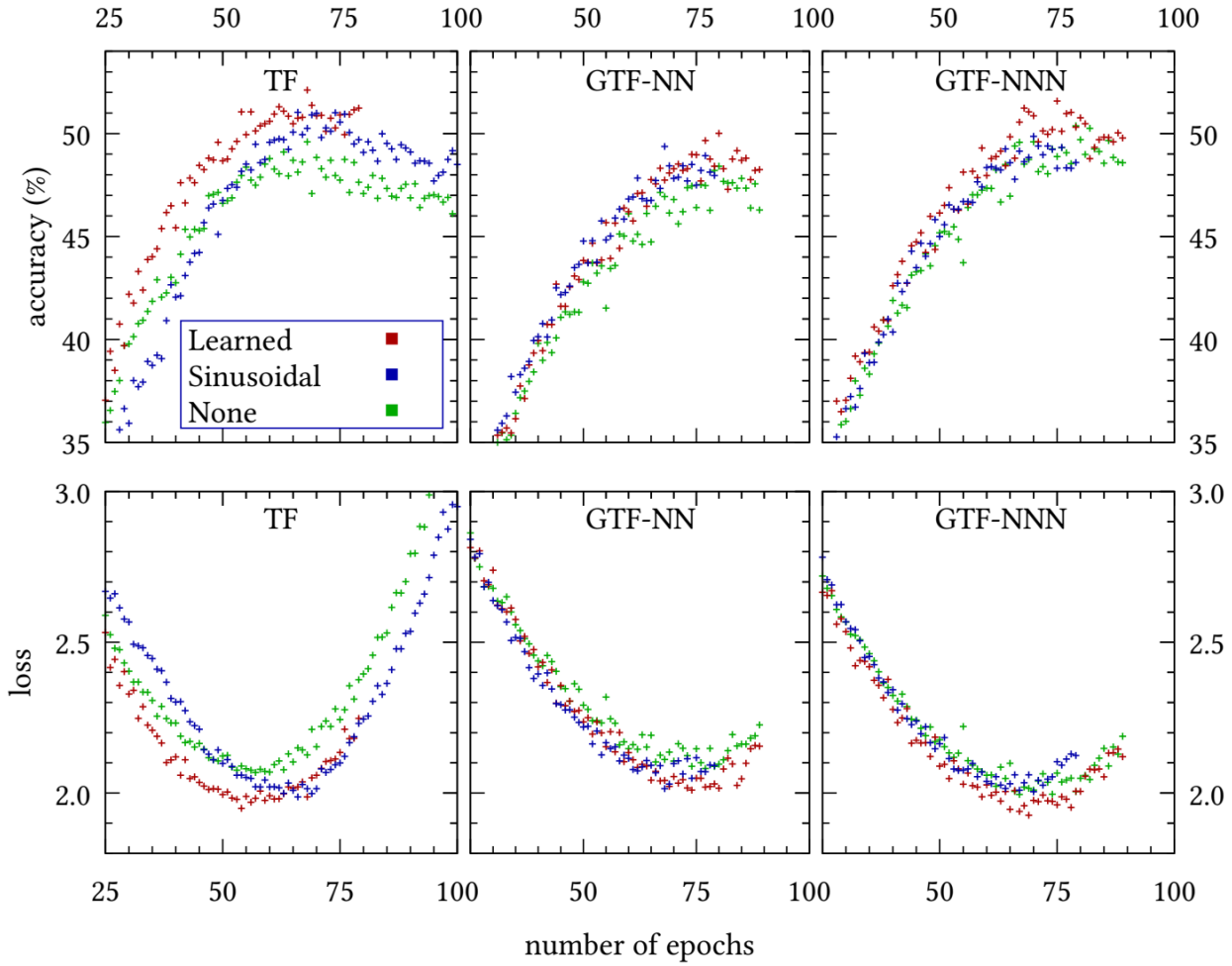
Choosing the optimizer

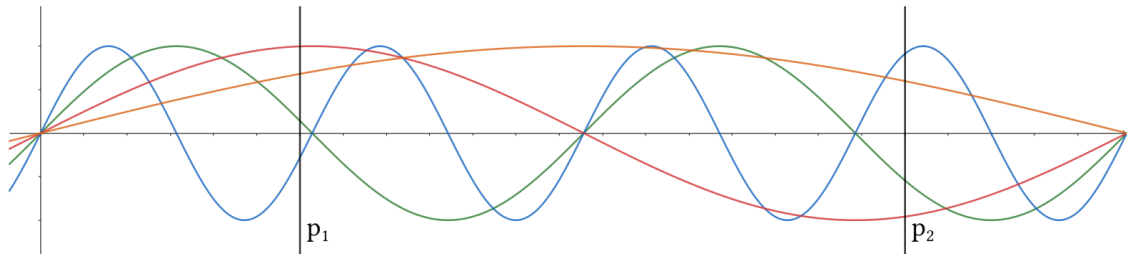
Supplementary Slides



Positional Encoding

Supplementary Slides

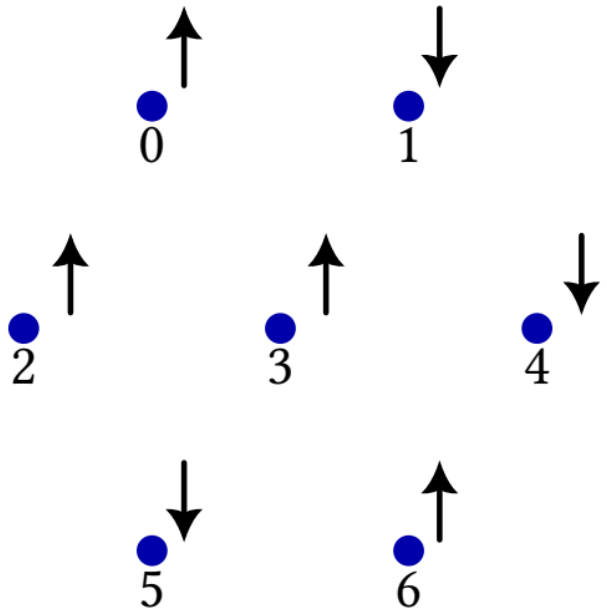




$$e(p_1) = \begin{pmatrix} -0.279 \\ 0.141 \\ 0.997 \\ 0.682 \end{pmatrix} \qquad e(p_2) = \begin{pmatrix} 0.913 \\ -0.544 \\ -0.959 \\ 0.598 \end{pmatrix}$$

Graph Patching

Supplementary Slides



$$s_{input} = (s_0, s_1, s_2, s_3, s_4, s_5, s_6) = (1, 0, 1, 1, 0, 0, 1)$$

$$s'_{duplicate\_spread} = \begin{pmatrix} (s_0), \\ (s_1), \\ (s_2), \\ (s_3), \\ (s_4), \\ (s_5), \\ (s_6) \end{pmatrix}$$

$$s'_{duplicate\_nn} = \begin{pmatrix} (s_0, s_1, s_2, s_3, -, -, -), \\ (s_1, s_0, s_3, s_4, -, -, -), \\ (s_2, s_0, s_3, s_5, -, -, -), \\ (s_3, s_0, s_1, s_2, s_4, s_5, s_6), \\ (s_4, s_1, s_3, s_6, -, -, -), \\ (s_5, s_2, s_3, s_6, -, -, -), \\ (s_6, s_3, s_4, s_5, -, -, -) \end{pmatrix}$$

$$s'_{duplicate\_nnn} = \begin{pmatrix} (s_0, s_1, s_2, s_3, -, -, -, s_4, s_5), \\ (s_1, s_0, s_3, s_4, -, -, -, s_2, s_6), \\ (s_2, s_0, s_3, s_5, -, -, -, s_1, s_6), \\ (s_3, s_0, s_1, s_2, s_4, s_5, s_6, -, -), \\ (s_4, s_1, s_3, s_6, -, -, -, s_0, s_5), \\ (s_5, s_2, s_3, s_6, -, -, -, s_0, s_4), \\ (s_6, s_3, s_4, s_5, -, -, -, s_1, s_2) \end{pmatrix}$$

Choice of Ansatz

Supplementary Slides