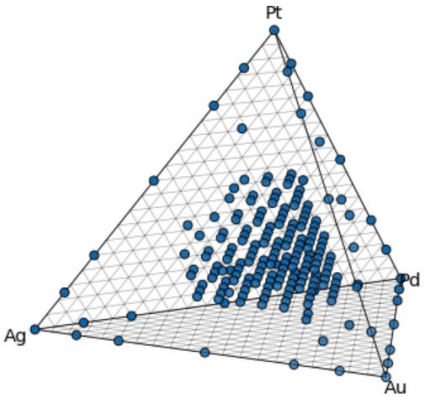
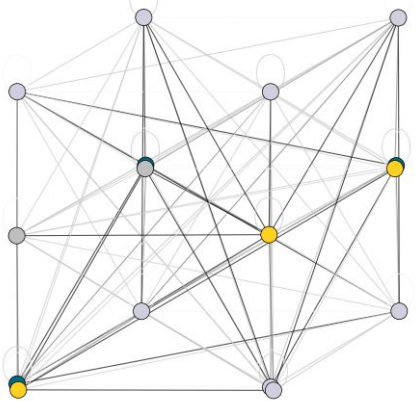
Graph neural networks for materials discovery: Predicting per-atom magnetic moments

Applied Machine Learning 2026 exam

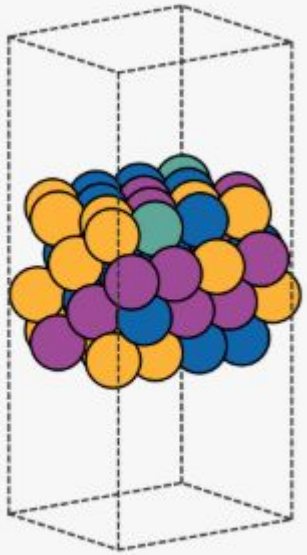
Zacharias Liasi, Kristian Krag Johansen, and Rose Tom-Petersen





Brief background on alloys and spin

- ➔ Alloys are combinations of different metals in an ordered structure or solid solution.
- ➔ Metals with unpaired electrons have spin angular momentum, leading to a magnetic moment.
- ➔ The magnetic moment of one metal in an alloy is influenced by the magnetic moment of its neighbors.
- ➔ We want to use a GNN to predict the magnetic moment of each atom in the alloy.
- ➔ 19,000 structures calculated as our dataset.

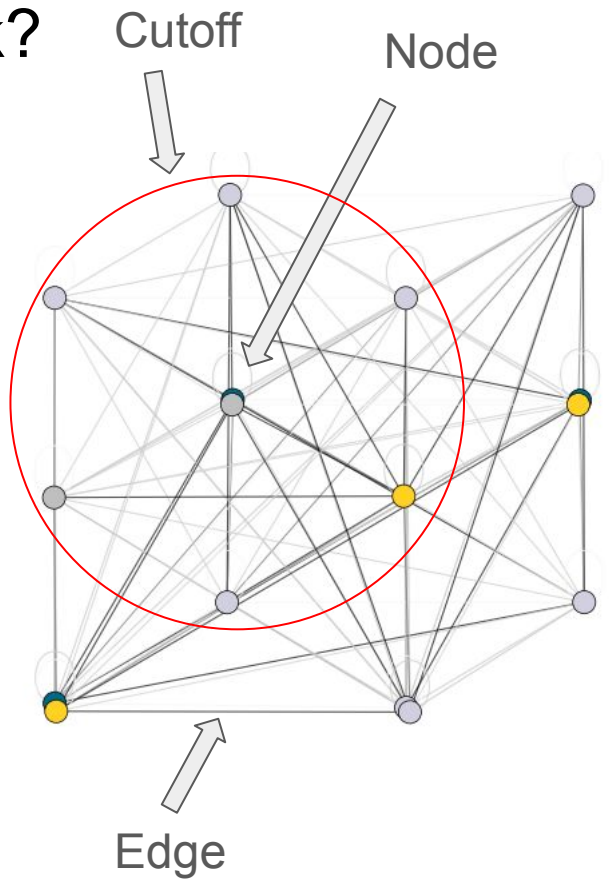


Methodology - why use GNN for this task?

➔ GNNs are ideal for describing materials at atomic scale.

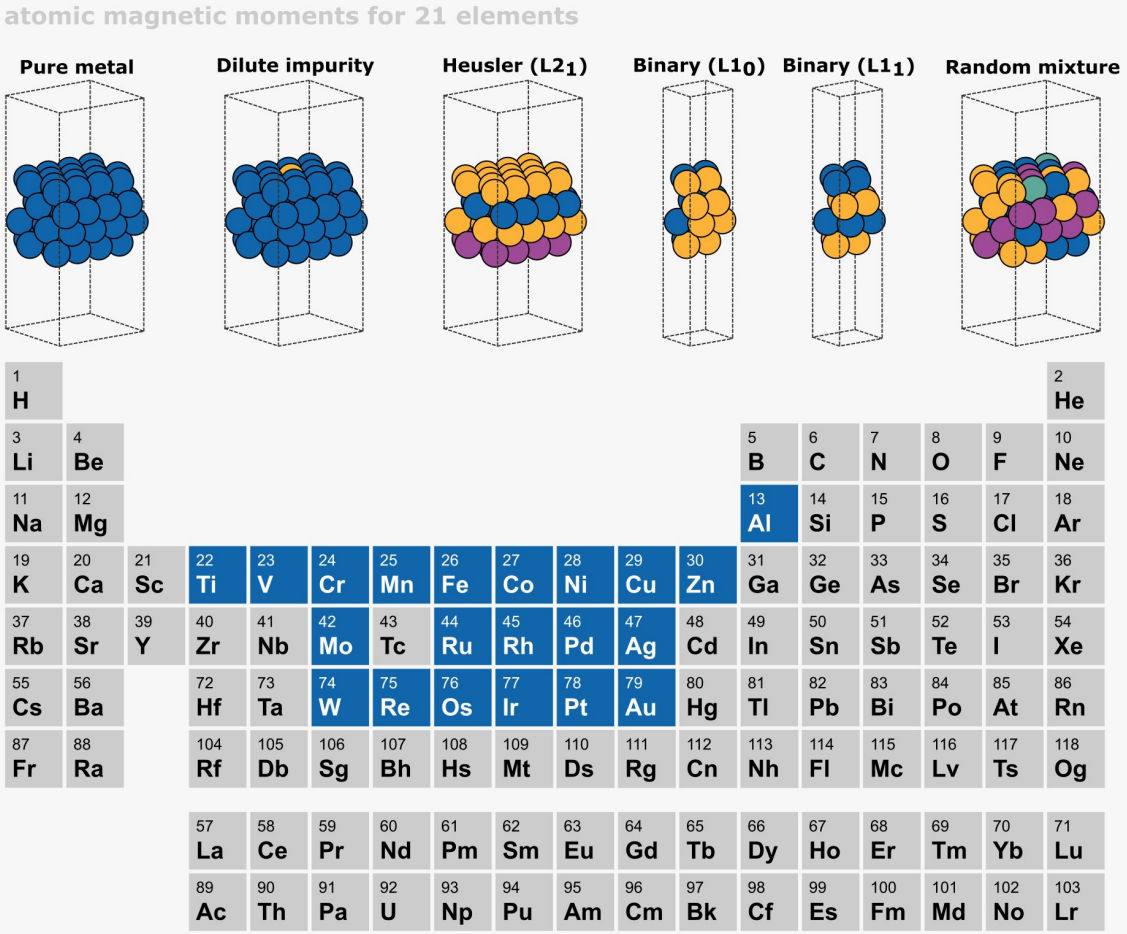
➔ Nodes correspond to atoms, and edges are pairs of atoms within the cutoff radius.

➔ Speedup for spin polarized DFT calculations using ML:
- Better initial guess for magnetic moments
- Faster convergence speeds up calculations

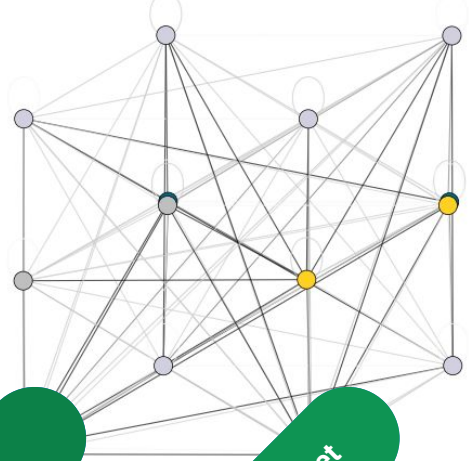


The dataset

- 21 metals
- 3 lattice structures, 6 composition types
- Magnetic moments calculated using DFT
- 19,000 structures -> over 100,000 magnetic moments



Graph Neural Network pipeline

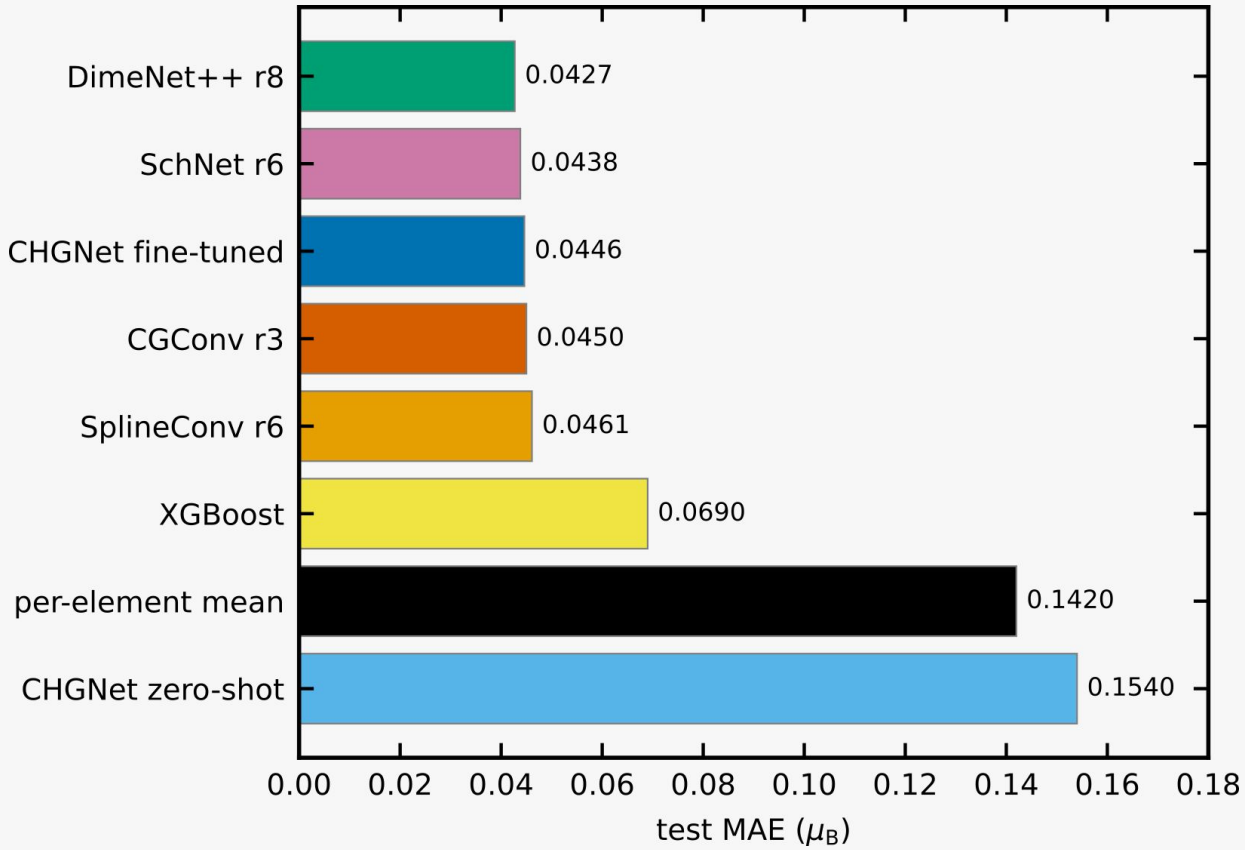


1. Create dataset of DFT calculated structures
2. Split data (avoid leakage)
3. Turn each structure into a graph
4. Define message passing model
5. Train model, evaluating on test set

Building the GNN - message passing model

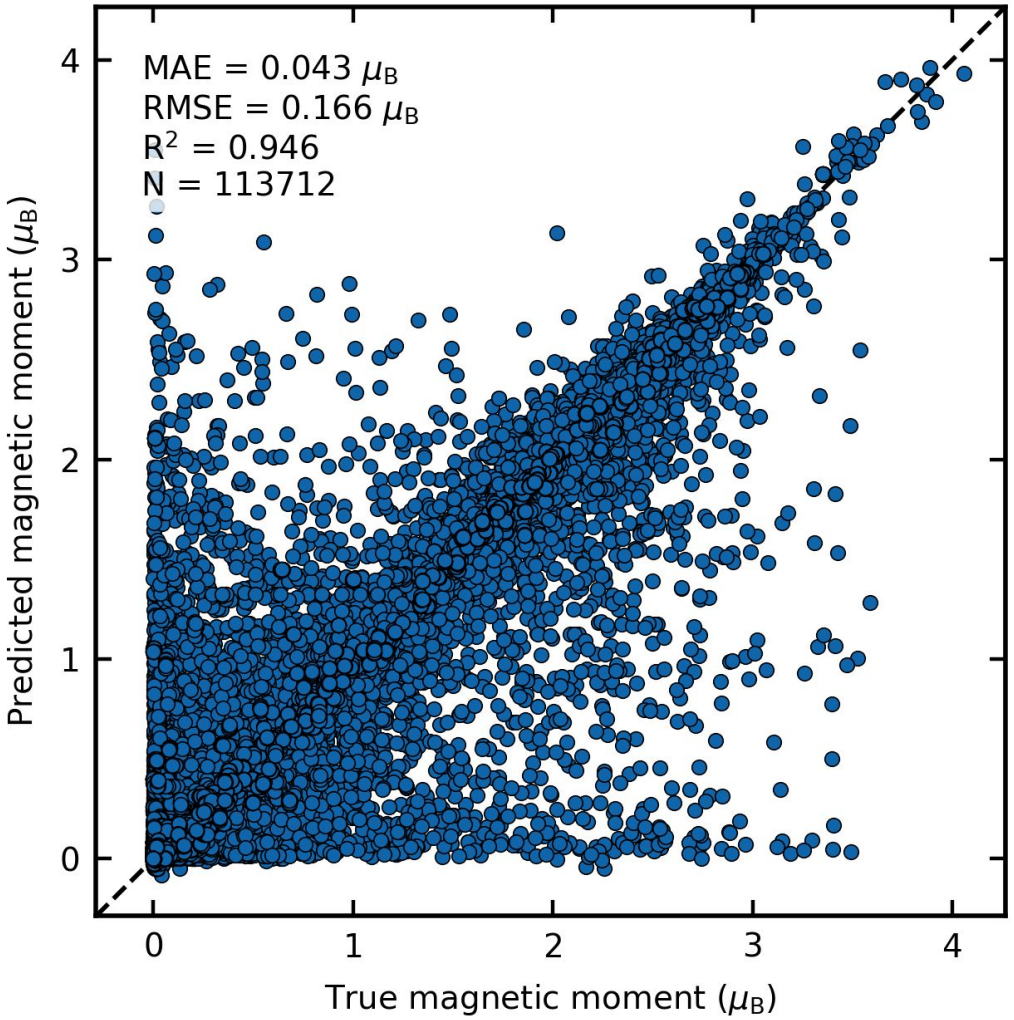
1. Replace each atom's atomic number with a learnable vector ("fingerprint")
2. Each atom's vector is updated by messages from its neighbours, weighted by interatomic distance (× 3 layers)
3. A radius cutoff sets the neighbourhood (we sweep 3-8 Å)
4. Nodes input the atomic number and output one magnetic moment per atom; edges carry distance (+angles for the directional model, DimeNet++)
5. Split dataset 80(train)/10(validation)/10(test), leakage-free

Results – 78 models ranked by MAE

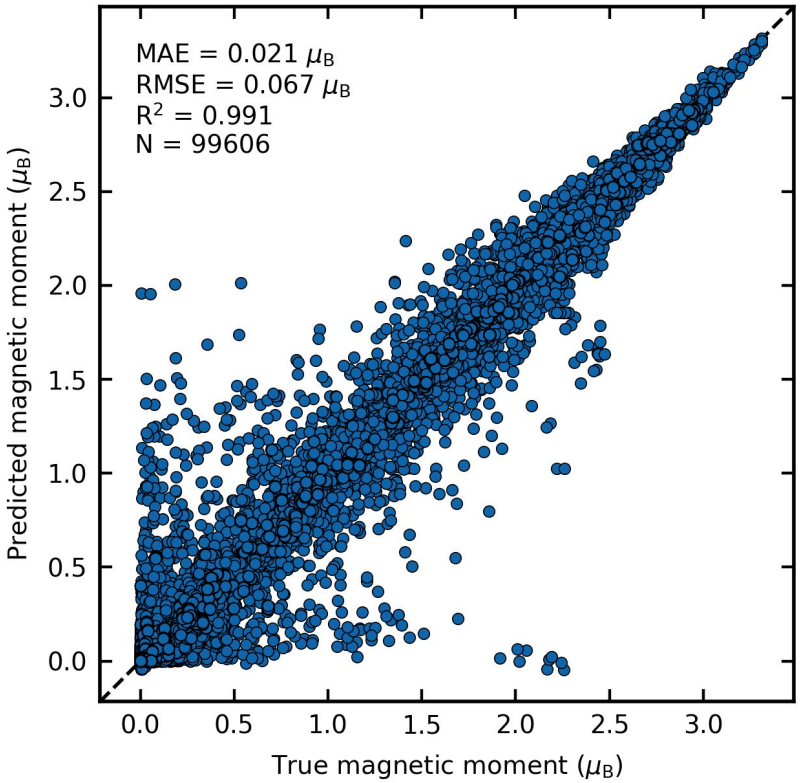


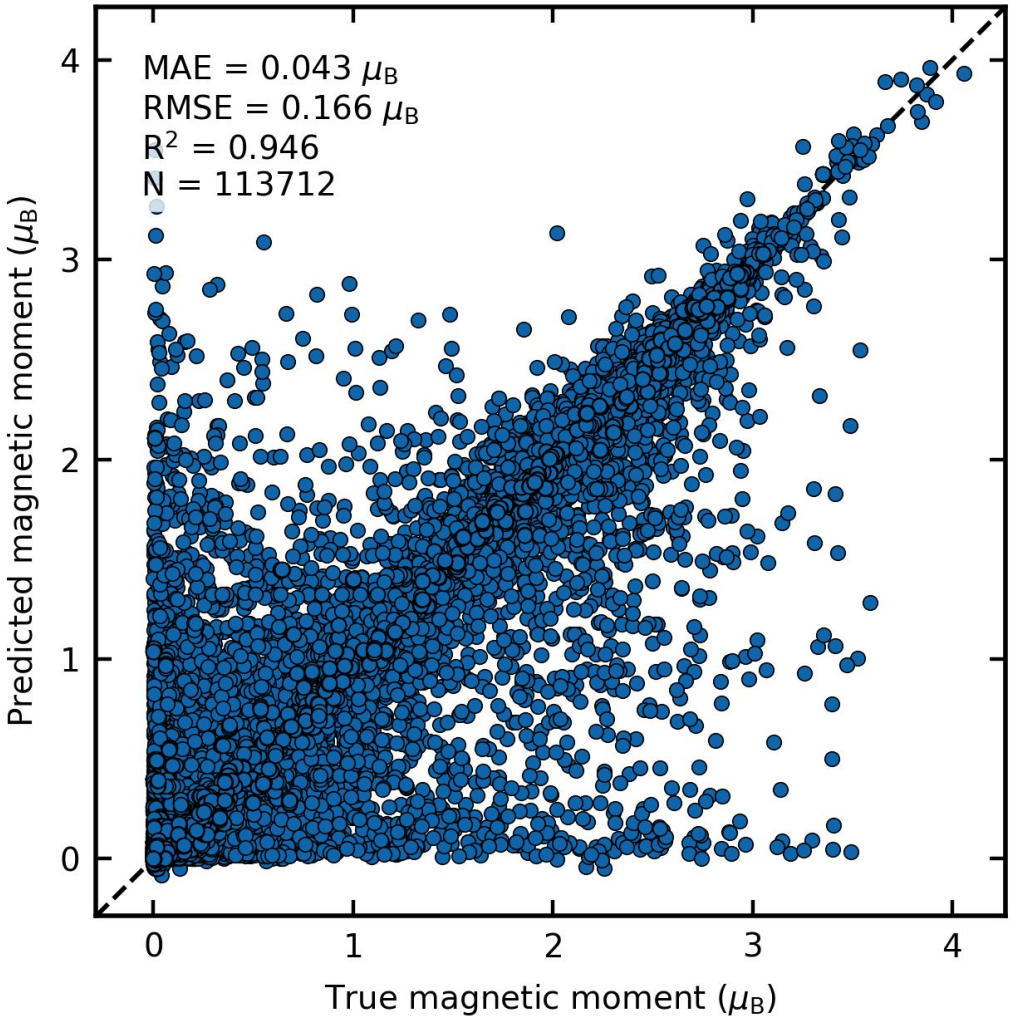
Results (DimeNet++ r8)

- Hard cases: Mn, Cr, V



Results (DimeNet++ r8)





Summary and outlook

- GNN for predicting atomic magnetic moments
- Best in test: DimeNet++ r8 (MAE 0.0427 μ_B)
- ~12% antiferromagnetic atoms (V, Cr, Mn); moment isn't a function of local structure; irreducible label noise, not a model failure
- We beat the “state of the art” model (CHGNet)
- Future: test the usability of the GNN, and improve dataset

Appendix

All code needed for reproduction is provided as a zip file, but the same files are available as a GitHub repository at: https://github.com/zliasi/appliedml-finalproject

Contributions (all contributed to each part of the project, with individual focuses):

Zacharias: Codebase set up

Kristian: Dataset construction and backend selection

Rose: GNN training and analysis

Appendix - Dataset (magmom21)

- 19,651 surface slabs from 8,452 unique compositions; ~1.2M atoms; 21 metals over FCC(111), BCC(110), HCP(0001).
- 64-atom 4x4x4 random alloys + dilute placements; 16-atom 2x2x4 binary intermetallic prototypes; 4x4x4 ternary Heuslers.
- Geometry: bottom 2 layers fixed; 10 Å vacuum; periodic in-plane (pbc [True, True, False]).
- DFT (GPAW): plane-wave 350 eV, RPBE, Monkhorst-Pack k-points 3.5 1/Å, Marzari-Vanderbilt smearing, dipole correction along the surface normal.
- Per slab: UMA-S-1.2 pre-relaxation -> staged GPAW (LCAO warm-up -> plane-wave LBFGS relaxation, fmax 0.2 eV/Å).
- SCF: 4-level retry ladder (Pulay/MSR1 mixing stages 0-2, direct minimisation stage 3); eigenstate criterion loosened 1e-8 -> 1e-5; smearing widened on retries.
- Spin-polarized when a magnetic element is present; initial moments from per-element/per-lattice equation-of-state values. Target = absolute per-atom moment |m| (muB).
- Split: 80/10/10 stratified by composition type, seeded (SPLIT_SEED=42), partitioned by composition so all variants stay together (no leakage). Test = 1,941 slabs = 113,712 atoms; reconstructable for external baselines.

Appendix - Graph construction (src/graph_builder.py)

- Each slab -> a graph: nodes = atoms, edges = atom pairs within a radius cutoff. Cutoffs cached separately: r3, r4, r6, r8 (Å).
- Neighbours via ASE neighbor_list('ijdDS', cutoff) with FULL periodic boundary conditions; capped at max_neighbors = 50 (nearest by distance).
- Node feature: atomic number Z only (a learned Embedding(119, 128) handles it). No hand-built node features.
- Edge features stored: scalar interatomic distance (edge_attr); periodic minimum-image displacement vector edge_vec = D = r_j + S·cell - r_i; integer cell offsets S; and the cell.
- The displacement vectors/offsets are what let the directional reference (DimeNet++) compute distances and triplet angles under PBC.
- Graphs cached as LMDB (graphs-r{cutoff}-magmom21-v0p1-{split}.lmdb) so the GNNs and baselines train on identical data/splits.

Appendix - Model and message-passing operators

- MagmomGNN: Embedding(Z) -> 3 message-passing layers (width 128) -> per-node readout MLP (1 hidden layer) -> one moment per atom. No pooling. Shared architecture token c128l3h1; only the operator changes.
- 15 node-level operators (d = consumes the distance edge feature):
  - cgconv(d), schnetconv(d), graphconv, sageconv, gatv2conv(d), gcnconv, transformerconv(d), gineconv(d), nnconv(d), genconv, gmmconv(d), resgatedgraphconv(d), generalconv(d), pdnconv(d), splineconv(d).
- schnetconv: GaussianSmearing(0, r_c, 50) radial basis + PyG InteractionBlock with a residual skip and a built-in cosine SMOOTH cutoff (decays to 0 at the boundary).
- splineconv: B-spline kernels over (dist/r_c) clamped to [0,1].
- Directional reference DimeNet++ (periodic): PyG DimeNet++ adapted to per-atom output, fed the periodic edges + displacement vectors (Sec. PBC).
- Residual-target variant (residual: true): a fixed per-element-mean bias buffer; the GNN learns only the deviation. Ran for cgconv/gineconv/gmmconv -> no improvement.

Appendix - Training setup and loss

- Loss: Huber (smooth-L1), delta = 0.1 muB, for every trained model (including the CHGNet fine-tune) so the comparison is not confounded by the objective.
- Why Huber in LINEAR (not log) space: the target has genuine zeros (non-magnetic + quenched atoms) so log is undefined; range is only 0-4 muB with no order-of-magnitude spread; delta de-weights the large V/Cr/Mn outliers without distorting the zeros.
- Optimiser AdamW (lr 1e-3, weight decay 0.01); scheduler ReduceLROnPlateau on val MAE; early stopping (patience 100); max_epochs 1000; gradient clip norm 1.0; batch size 32; checkpoint on best val MAE.
- TF32 matmul/conv on; 7 DataLoader workers (from bench tuning).
- Slurm pipeline (00-run-pipeline.sh): build graphs -> {train array, trees + CHGNet zero-shot, CHGNet fine-tune} -> evaluate. wandb logging optional.
- Evaluation: per-model MAE/RMSE/R2/max-error, per-element breakdown, and a V/Cr/Mn-excluded metric set; parity + signed-error plots.

Appendix - Periodic boundary fix/issues

- Problem: PyG's DimeNet++ (and ViSNet) build their neighbour graph from raw positions via torch_cluster.radius_graph, which has NO cell/pbc argument - so they silently ignore periodicity and discard the periodic edges the graph already carries.
- Measured impact (ASE, 64-atom slab): 30 / 42 / 55% of edges dropped at r4 / r6 / r8. A 'reference' that omits up to half its neighbours is not a fair upper bound.
- Fix (OCP-style): feed DimeNet++ the precomputed periodic edge_index and the minimum-image displacement vectors edge_vec; compute distances as ||edge_vec|| and triplet angles from edge_vec[idx_ji], edge_vec[idx_kj] instead of raw position differences.
- Validated by a translation-invariance test: a correct periodic model gives identical per-atom outputs when atoms are translated and wrapped; the fixed DimeNet passes, confirming the angle/sign bookkeeping.
- Result: periodic DimeNet++ becomes the best model (0.0427), and its cutoff preference reverses (non-periodic peaked at r6 and worsened at r8; periodic improves to r8) - geometry, not loss.
- ViSNet dropped (computes neighbours internally; cannot take precomputed edges without reimplementation). Equivariant periodic references via torchmd-net (ET, TensorNet) were implemented + CPU-validated but removed: their CUDA neighbour search (2.6.x) needs torch.library.triton_op (torch >= 2.6), so they crashed on the GPU nodes.

Appendix - Baselines

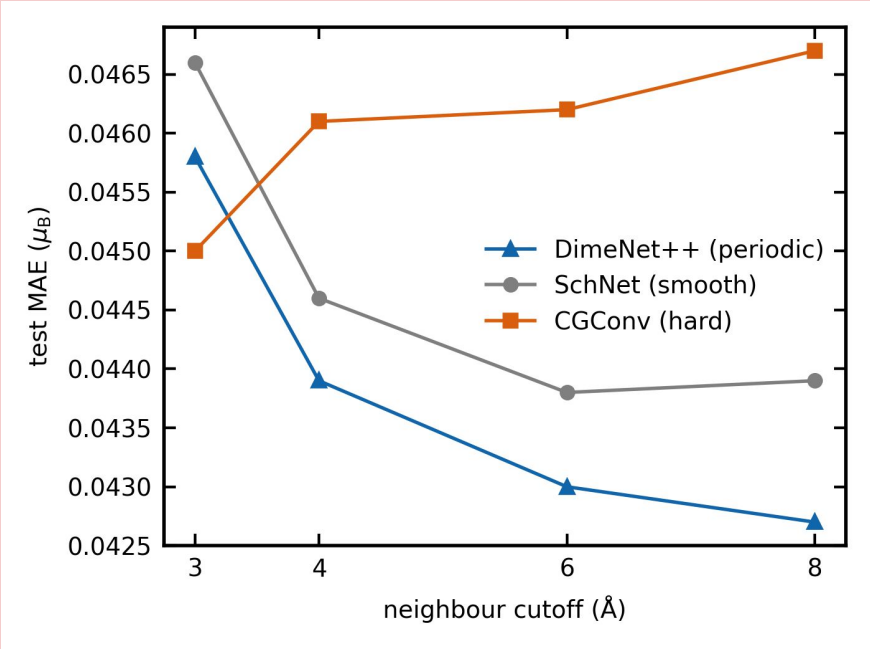
- per-element-mean (the floor, MAE 0.142): predict each atom's moment as the training-set mean for its element. Uses element identity only, no geometry; identical at every cutoff. The bar any real model must beat.
- Linear regression and XGBoost on hand-built per-atom features: self-element one-hot, neighbour-element counts, coordination number, and mean/min/max neighbour distance. XGBoost (500 trees, depth 6, lr 0.1) is the strongest non-GNN at 0.069 - still ~60% worse MAE than the top.
- CHGNet zero-shot (0.154): the pretrained foundation model used as-is. Out-of-domain on alloy slabs -> below the trivial floor (worst R2, 0.76).
- CHGNet fine-tuned (0.0446, best R2 0.9528): magmom-only fine-tune on train+val under the same Huber loss. Rank 6 of 78, within ~4% MAE of the best from-scratch model. Per-element behaviour mirrors the GNNs (Mn 0.377, Cr 0.157, V 0.065) - it hits the same V/Cr/Mn wall.
- Three-tier conclusion: from-scratch domain GNN (0.0427) >= fine-tuned foundation model (0.0446) >> zero-shot foundation / trees.

Appendix - Full ranking

	model	MAE	RMSE	R2
1	dimenet r8	0.0427	0.1658	0.9456
2	dimenet r6	0.0430	0.1699	0.9428
3	schnetconv r6	0.0438	0.1633	0.9472
4	schnetconv r8	0.0439	0.1673	0.9446
5	dimenet r4	0.0439	0.1727	0.9410
6	CHGNet fine-tuned	0.0446	0.1545	0.9528
7	schnetconv r4	0.0446	0.1674	0.9445
8	cgconv r3	0.0450	0.1755	0.9390
9	dimenet r3	0.0458	0.1777	0.9375
10	splineconv r6	0.0461	0.1757	0.9389
11	cgconv r4	0.0461	0.1768	0.9381
12	cgconv r6	0.0462	0.1810	0.9352
13	cgconv r4 res	0.0463	0.1757	0.9389
14	schnetconv r3	0.0466	0.1743	0.9399
15	cgconv r8	0.0467	0.1808	0.9353
16	splineconv r4	0.0473	0.1776	0.9375
17	gineconv r3	0.0476	0.1802	0.9358
18	splineconv r8	0.0484	0.1770	0.9380
19	gineconv r4	0.0494	0.1829	0.9338
20	splineconv r3	0.0498	0.1818	0.9346
21	resgatedgraph r4	0.0502	0.1871	0.9308
22	resgatedgraph r3	0.0505	0.1894	0.9290
23	gineconv r6	0.0511	0.1838	0.9332
24	gmmconv r4 res	0.0515	0.1867	0.9310
25	resgatedgraph r8	0.0518	0.1841	0.9329
26	resgatedgraph r6	0.0521	0.1870	0.9308
27	gmmconv r4	0.0522	0.1862	0.9314
28	gmmconv r3	0.0527	0.1855	0.9319
29	gmmconv r6	0.0539	0.1881	0.9300
30	gmmconv r8	0.0542	0.1914	0.9275
31	transformer r3	0.0546	0.1894	0.9290
32	transformer r8	0.0547	0.1882	0.9299
33	graphconv r3	0.0548	0.1896	0.9289
34	transformer r6	0.0555	0.1911	0.9277
35	transformer r4	0.0555	0.1899	0.9286
36	sageconv r3	0.0566	0.1963	0.9237
37	generalconv r3	0.0570	0.1890	0.9293
38	genconv r3	0.0577	0.2002	0.9207
39	graphconv r4	0.0578	0.1980	0.9224
40	sageconv r4	0.0581	0.1949	0.9248
41	pdnconv r3	0.0583	0.1941	0.9254
42	pdnconv r4	0.0594	0.1998	0.9210
43	genconv r4	0.0601	0.2069	0.9153
44	gatv2conv r6	0.0619	0.2014	0.9197
45	gatv2conv r4	0.0620	0.2016	0.9196
46	generalconv r4	0.0631	0.2015	0.9196
47	gatv2conv r3	0.0636	0.1981	0.9224
48	sageconv r6	0.0639	0.2085	0.9139
49	gatv2conv r8	0.0646	0.2121	0.9109
50	genconv r8	0.0649	0.2112	0.9117
51	genconv r6	0.0652	0.2114	0.9115
52	sageconv r8	0.0658	0.2154	0.9082
53	pdnconv r8	0.0689	0.2156	0.9080
54	xgboost r4	0.0690	0.1862	0.9314
55	pdnconv r6	0.0709	0.2385	0.8874
56	xgboost r8	0.0719	0.1851	0.9322
57	xgboost r6	0.0721	0.1881	0.9300
58	gineconv r6 res	0.0916	0.2438	0.8824
59	per-elem-mean r6	0.1420	0.2908	0.8327
60	per-elem-mean r8	0.1420	0.2908	0.8327
61	per-elem-mean r4	0.1420	0.2908	0.8327
62	linear r6	0.1474	0.2647	0.8613
63	linear r4	0.1480	0.2649	0.8611
64	linear r8	0.1483	0.2659	0.8600
65	CHGNet zero-shot	0.1539	0.3463	0.7627
66	gcnconv r3	0.1712	0.4879	0.5289
67	gcnconv r4	0.1747	0.4870	0.5306
68	gcnconv r8	0.1783	0.4843	0.5358
69	gcnconv r6	0.1797	0.4807	0.5427
70	gineconv r8	0.2367	0.5622	0.3745
71	nnconv r3	0.3426	0.7725	-0.1810
72	graphconv r6	0.3429	0.7721	-0.1797
73	generalconv r8	0.3432	0.7716	-0.1783
74	nnconv r4	0.3433	0.7715	-0.1780
75	generalconv r6	0.3434	0.7713	-0.1774
76	nnconv r6	0.3434	0.7714	-0.1776
77	nnconv r8	0.3436	0.7710	-0.1765
78	graphconv r8	0.3437	0.7709	-0.1763

Appendix - Cut-off sweep

- SchNet (smooth cosine cutoff): flat across cutoffs (r3->r8: 0.0466, 0.0446, 0.0438, 0.0439) - neighbour contributions decay to zero at the boundary.
- CGConv (hard cutoff): best at the smallest radius (r3 0.0450), degrades with cutoff - boundary discontinuities as neighbours cross the radius.
- DimeNet++ (periodic, directional): best at the largest radius (r8 0.0427) - now sees the true neighbours.



Appendix - Per-element error

- Mn, Cr, V carry essentially all the error. Removing them (12.4% of atoms): full 0.0427 / R2 0.946 -> 0.0209 / R2 0.991 (RMSE -60%).
- Cause: their moment is not a function of local structure - the same environment yields a quenched or moment-bearing atom depending on the global spin state DFT converged to (an unobserved variable).
- One-to-many map -> the regressor predicts the conditional median (~0 for V/Cr) -> moment-bearing minority lands at predicted ~0 (the parity stripe). Irreducible label noise for any structure-only model.

element	MAE	R2	n_atoms
Mn	0.406	0.447	4703
Cr	0.153	0.591	2667
V	0.068	0.529	6736
Fe	0.079	0.883	7424
Co	0.065	0.899	7374
Ni	0.043	0.951	6300
Pt	0.011	0.952	5739
Cu	0.004	0.880	6313
Al	0.003	0.869	3835

Appendix - Deployment cost (single 64 atom slab)

- DimeNet++ is 3.3x the params, ~27x the inference latency, ~230x the load time of CGConv (the 13.6 s is DimeNet's spherical-basis construction, paid once per process, so batched jobs would nullify the time difference).
- But all of it is negligible next to the DFT being warm-started: a spin-polarized GPAW slab run is minutes-to-hours, so a surrogate at 4-99 ms/slab (plus a one-time load) is orders of magnitude cheaper, any backend.
- So cost is not a constraint here; accuracy drives the choice

backend	params	load	infer	test MAE
DimeNet++ r8 (best)	764 k	13.6 s	98.8 ms	0.0427
SchNet r6	249 k	64 ms	5.6 ms	0.0438
CGConv r3	231 k	58 ms	3.6 ms	0.0450