Machine Learning of Combinatorial Rules in Mechanical Metamaterials

Ryan van Mastrigt,1,2 Marjolein Dijkstra,3 Martin van Hecke,2,4 and Corentin Coulais1
1 Institute of Physics, Universiteit van Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands
2 AMOLF, Science Park 104; 1098 XG Amsterdam, The Netherlands
3 Soft Condensed Matter, Debye Institute for Nanomaterials Science, Department of Physics, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
4 Huygens-Kamerling Onnes Lab, Universiteit Leiden, Postbus 9504, 2300 RA Leiden, The Netherlands

(Dated: February 8, 2022)

Combinatorial problems arising in puzzles, origami, and (meta)material design have rare sets of solutions, which define complex and sharply delineated boundaries in configuration space. These boundaries are difficult to capture with conventional statistical and numerical methods. Here we show that convolutional neural networks can learn to recognize these boundaries for combinatorial mechanical metamaterials, down to finest detail, despite using heavily undersampled training sets, and can successfully generalize. This suggests that the network infers the underlying combinatorial rules from the sparse training set, opening up new possibilities for complex design of (meta)materials.

From proteins and magnets to metamaterials, all around us systems with emergent properties are made from collections of interacting building blocks. Determining the relation between structure and property is often straightforward in principle, yet computationally expensive in practice, e.g., requiring the diagonalization of large matrices. Machine learning algorithms such as neural networks (NNs) forgo the need for such calculations by “learning” the classification of structures. In the context of mechanics and soft matter, machine learning has been used to find patterns in crumpling, active matter and hydrodynamics, predict structural defects and plasticity, and design metamaterials. More broadly in materials science, machine learning has been successfully applied to determine order parameters, identify phase transitions, coarse grain complex molecular compounds, and predict protein structure. In these examples, the relevant property typically varies smoothly and boundaries separating classes in configuration space are hence blurred. The key behind the success of NNs lies in that they are able to interpolate these blurred boundaries although the configuration space is heavily under-sampled.

Here we ask how suitable such techniques are for problems that are combinatorial in nature. Examples of such problems occur in combinatorial chemistry, protein folding and metamaterial design; changing a single amino acid can alter the folding of proteins, and changing a single crease in an origami folding pattern can be sufficient to make the pattern unfoldable. For such combinatorial problems, the configuration space features rare compatible (C) and frequent incompatible (I) combinations of building blocks (Fig. 1(a)). Specifically, starting from a C configuration, only a subset of point mutations remains of class C. Hence, C consists of connected low-dimensional filaments, leading to a ‘Mikado’ structure (Fig. 1(a)) which is distinct from both smooth and random classification spaces (Fig. 1(b)-(c)).

The curse of dimensionality and rarity of C impede naive brute-force exploration of this Mikado structure. Random sampling of design space yields very few points close to the complex boundary between C and I. NNs can interpolate any—smooth or non-smooth—function successfully, so they are in principle able to learn highly complex combinatorial problems. Yet whether they manage to do so when the boundary between I and C is sharp, complex and undersampled remains unclear. Crucially, the C-subspace is defined by a set of (potentially unknown) combinatorial rules. Hence, one may wonder if a network is able to hone in on those rules and in this manner reproduce the sparsity and complex Mikado structure of C.

Here we show that convolutional neural networks (CNNs) are able to successfully approximate the sharp class boundary of a combinatorial metamaterial design problem, which serves as a proxy for more general combinatorial problems. We show that CNNs can not only accurately predict the volume of the rare class C, but also capture its dimensionality and highly complex boundary. This suggests that NNs are capable of learning the underlying rule based structure of combinatorial problems and generalize to never-before-seen configurations, despite being trained on a sparse subset of the configuration space.

Combinatorial Metamaterial—The unit cells of our two-dimensional metamaterial consist of $k \times k$ building blocks, each of which can be oriented in one of four directions, leading to a design space that grows exponentially with unit cell size $k$ as $4^k$ (Fig. 2(a)). Each building block can deform in two distinct ways that do not stretch any of the bonds (Fig. 2(b)) — such zero-energy deformations are referred to as zero modes. Crucially, the number...
of zero modes $M_k(n)$ of a metamaterial consisting of $n \times n$ unit cells depends on the design of the $k \times k$ unit cell: when the linear size of our metamaterial $n$ is increased, the number of zero modes $M_k(n)$ either grows linearly with $n$ or saturates at a non-zero value (Fig. 2(c)).

We note that the zero modes that give rise to linearly increasing $M_k(n)$ are organized along strips, and require specific combinations of building block orientations in these strips (Fig. 2(d), see also Supplemental Materials (SM) [49])—we refer to these modes as combinatorial strip modes. As the linear size $n$ of a metamaterial with strip modes is increased, the number of zero modes $M_k(n)$ grows linearly (Fig. 2(d)). It can be shown that a set of non-trivial combinatorial rules gives the precise conditions for such strip modes to occur, and that the presence of strip modes is a necessary and sufficient condition for class C (see SM [50]). Accordingly, we call the class with such combinatorial strip modes “compatible” (C) and the other class with no combinatorial strip modes “incompatible” (I). Crucially, a single rotation of one building block in the unit cell can be sufficient to change class (Fig. 2(c)). Hence, the boundary between classes C and I is sharp, sensitive to minimal perturbations, and, as we will show, class C becomes increasingly rare with unit cell size $k$. Thus the design space is of Mikado-type (Fig. 1(a)).

Classification—Brute force classification of our metamaterials as C or I requires the determination of $M_k(n)$—via rank-revealing QR factorization [51]—as function of the number of unit cells $n$, which is computationally demanding. As our designs are spatially structured and translation invariant, we ask whether convolutional neural networks (CNNs) are able to distinguish between class C and I, and if so, how complex the neural network needs to be to capture the small volume and complex boundary of class C. The input of our CNNs are pixelated representations of our design (Fig. 3(a)). We use CNNs with a single convolution layer of $n_f \times 2 \times 2$ filters, which are spatially offset with respect to the unit cell (Fig. 3(a)) and subsequently flattened. This approach facilitates the identification of neighboring building blocks that are capable of compatible deformations (see SM [52]). The flattened feature maps are fully-connected to a layer of $n_h$ hidden neurons, which itself is fully-connected to two output neurons that correspond to class C and I. The CNNs are systematically trained using 10-fold stratified cross-validation for varying number of filters $n_f$ and hidden neurons $n_h$. Crucially, we use a balanced training set, where the proportion of class I has been randomly undersampled such that classes C and I are equally represented (see SM [53]).

In contrast, our test set is uniformly sampled and
Figure 3. (color online) (a) Unit cell design (top) and its pixelation representation (bottom). Each building block (top blue square) is represented by $2 \times 2$ pixels, one black (1) and three white (0) (bottom blue square). The black pixel is placed in the quadrant without a rigid triangle in the building block. The pixel representation is padded with a single pixel wide layer using periodic boundary conditions. The convolutional layer consists of $n_f$ $2 \times 2$ filters (green square) and stride across the image such that they convolve each area enclosed by a gray square individually. (b) Heatmaps of the balanced accuracy $BA$ over the test set of trained CNNs with $n_f$ filters and $n_h$ hidden neurons. The unit cell size $k$ used for training is indicated at the top of each heatmap.

thus strongly imbalanced—with many more I. Combining the classification given by the CNN with the true classes $C$ and $I$ yields four subspaces, $[(TC), (FC), (TI), (FI)]$ with respective volumes $V_C = N_C/N$ with $N$ and $N_C$ the total number of configurations and configurations in subspace $x$ (Fig. 1(a)). We quantify the CNNs’ accuracy by the balanced accuracy $BA$ which is invariant to class-imbalance and reads: $BA = \langle 1/2 (V_{TC}/(V_{TC} + V_{TI}) + V_{FI}/(V_{FI} + V_{FC})) \rangle$, where $\langle \rangle$ denotes the average over the 10 folds.

Results—Despite the complexity of the classification problem, we find that, for sufficiently large $n_f$ and $n_h$, the balanced accuracy $BA$ approaches its maximum value 1 for every considered unit cell size $k$ (Fig. 3(b)). Strikingly, the number of filters $n_f$ required to achieve large $BA$ does not vary with $k$, and in the remainder we set $n_f = 20$. The number of required hidden neurons $n_h$ increases with $k$, but not dramatically, despite the combinatorial explosion of the design space. To interpret this result, we note that a high $BA$ corresponds to correctly classifying most $C$ unit cells as class $C$, and most $I$ unit cells as class $I$ [54]. Hence, sufficiently large networks yield decision boundaries such that $V_{TC} \gg V_{TI}$ and $V_{FI} \gg V_{FC}$. However, $BA$ is inherently insensitive to cases where $V_{FC} \gg V_{TC}$, so that the predicted class $C$ volume, $\tilde{\beta} = V_{TC} + V_{FC}$ may still be much larger than the true class $C$ volume $\beta = V_{TC} + V_{FI}$ (Fig. 1(b)). Hence, whether our CNNs capture the sparsity of $C$ and its complex boundary cannot be deduced from a coarse measure such as $BA$ alone.

Class volume—We now probe how well our CNNs capture the sparsity of class $C$ by comparing their true and predicted volumes $\beta$ and $\tilde{\beta}$; note that $\beta$ decreases rapidly with $k$ (Fig. 4(a)). The CNNs’ $\beta$ approaches $\beta$ as the number of hidden neurons $n_h$ increases sufficiently, despite their balanced training set (Fig. 4(a)) [55]. Interestingly, $\tilde{\beta}$ approaches $\beta$ as rapidly as $BA$ approaches $1$ (Fig. 4(b)). This implies that, when $n_h$ is large enough, $V_{TC} \gg V_{FC}$, so that the volume of $C$ becomes negligible. Hence, the CNNs closely approximate the volume of the Mikado-structure, (Fig. 1(a)).

Combinatorial structure—In what follows, we ask whether they can also approximate the complex shape of the subspace $C$. To probe the shape of both the true $C$ configurations and the set of classified $C$ configurations, we start from a true or classified $C$ configuration, perform random walks in configuration space $[54]$, and at each step probe the probabilities to be in the set of true or classified $C$ (Fig. 5(a)). We randomly change the orientation of a single random building block at each step $s \rightarrow s+1$ (Fig. 5(a)-inset) and average over 1000 realizations (see SM [57]). The probability to remain in true class $C$, $P_{C\rightarrow C}(s)$, decreases with $s$ and saturates to the class $C$ volume $\beta$ (Fig. 5(a)). We note that we can fit this decay by a simple model, where we assume that space $C$ is highly complex, so that the probabilities to leave it are uncorrelated:

$$p_{C\rightarrow C}(s) = \alpha^s + \beta \left(1 - \alpha^{s-1}\right),$$

where the parameter $\alpha$ is the probability to remain in the $C$ space at each step, i.e., the local dimensionality of space $C$ (Fig. 5(a)). The uncorrelated nature of the steps are consistent with a random Mikado structure (Fig. 1(a)), where the the coefficient $\alpha \times k^2$ corresponds to the dimensionality of each Mikado. We can interpret $\alpha$ as the fraction of building blocks that are outside of the relevant combinatorial strip. Taking the width of these strip modes into account leads to $\alpha \approx 1 - 1/k$ and $\alpha \approx 1 - 2/k$ for even and odd $k$, which are in good agreement with fitted values of $\alpha$ (Fig. 5(b); see SM [57]).

To see whether the CNNs are able to capture these key features of space $C$, we repeat our random walk procedure using the CNNs’ classification and obtain the probability
Figure 5. (color online) (a) Probabilities to remain in true and predicted class C under random walks of s steps, $p_{C→C}(s)$ (red crosses) and $\tilde{p}_{C→C}(s)$ (symbols; color indicates $n_h$), for $k = 6$. The red continuous line is a least-squares fit to $p_{C→C}(s)$ using Eq. (1). Inset: example of a 3-step random walk through design space (red dots) and sketches of the decision boundaries of trained CNNs with low $n_h$ (purple) and high $n_h$ (yellow). (b) True dimensionality $\tilde{\alpha}$ (red) and predicted dimensionality $\hat{\alpha}$ (colorbar) obtained through least-squares fits to data as in panel (a) for all $k$. The estimated $\alpha$ for both odd (dashed line) and even (dashed-dotted line) $k$ agree well with $\alpha$.

$\tilde{p}_{C→C}(s)$ as a function of the number of hidden neurons $n_h$. For small values of $n_h$, the probabilities to remain in class C are overestimated by the CNNs; in other words, CNNs with a small number of hidden neurons blur the complex shape of C (Fig. 5(a)). Remarkably, for large $n_h$, the decay of $\tilde{p}_{C→C}(s)$ closely matches that of the true class C (Fig. 5(a)). By fitting the predicted probability $\tilde{p}_{C→C}(s)$ to Eq. (1), using our previous measurements of the volume $\tilde{\beta}$ to constrain the fit, we obtain the dimensionality $\tilde{\alpha}$. While for small values of $n_h$, $\tilde{\alpha}$ overestimates $\alpha$, $\hat{\alpha}$ closely matches $\alpha$ for large $n_h$ (Fig. 5(b)). Our results thus demonstrate that larger CNNs successfully capture the complex shape of the combinatorial space C.

Finally, we observe that the predicted class C volume $\tilde{\beta}$ more quickly reaches its asymptotic value than the dimensionality $\tilde{\alpha}$. To see this, we plot $\tilde{\beta}(n_h) - \beta$ versus $\tilde{\alpha}(n_h) - \alpha$, which demonstrates that after $\tilde{\beta}$ reaches $\beta$, the number of hidden neurons improves $\tilde{\alpha}(n_h)$ towards its asymptote $\alpha$ (Fig. 5). Thus, to correctly capture the low dimensionality and features of the Mikado-like shape of C requires more neurons than to capture its volume, but nevertheless is achieved. As during learning the algorithm ‘sees’ very few class I unit cells that are close to the class boundary, we conclude that as the decision boundary captures the sparsity and the fine structure of the class C subset for large $n_h$, the CNNs infer the combinatorial rules, rather than fitting the shape in high dimensional design space.

Discussion—NNs are known to be universal approximators [53] and efficient classifiers. They often generalize well when the training data samples representative portions of the input space sufficiently, even for non-smooth [44] or noisy data [59]. As combinatorial problems are sharply delineated and severely classimbanced, one expects that the fine details of an undersampled complex boundary would be blurred by NNs. However, we have shown that CNNs will closely approximate such a complex combinatorial structure, despite being trained on a sparse training set. We attribute this to the underlying set of rules which govern the complex space of compatible configurations—in simple terms, the CNN learns the combinatorial rules, rather than the geometry of design space, which is the complex result of those rules.

NNs’ ability to learn these rules from a sparse representation of the design space opens new strategies for design. For instance, our CNNs could be readily used as surrogate models within a design algorithm to save computational time. Alternatively, one could instead devise a design algorithm based on generative adversarial NNs [60]. It is an open question whether and how such generative adversarial NNs could successfully leverage the learning of combinatorial rules [61].

Our work shows that metamaterials provide a compelling avenue for machine learning combinatorial problems, as they are straightforward to simulate yet exhibit complex combinatorial structure (Fig. 1(a)). More broadly, applying neural networks to combinatorial problems opens many exciting questions. What is the relation between the complexity of the combinatorial rules and that of the networks? Can unsolved combinatorial problems be solved by neural networks? Conversely, can these problems help us understand why neural networks work so well [62]? We believe combinatorial metamaterials are well suited to answer such questions.

Data availability statement.—The code supporting the findings reported in this paper is publicly available on GitHub [63][64]—the data on Zenodo [65][66].

Acknowledgements.—We thank David Dykstra, Marc Serra-Garcia, Jan-Willem van de Meent, and Tristan Breau for discussions. This work was carried out on the Dutch national e-infrastructure with the support of
SURF Cooperative. C.C. acknowledges funding from the European Research Council under Grant Agreement 852587.

[1] J. Hoffmann, Y. Bar-Sinai, L. M. Lee, J. Andrejevic, S. Mishra, S. M. Rubinstein, and C. H. Rycroft, Machine learning in a data-limited regime: Augmenting experiments with synthetic data uncovers order in crumpled sheets, Science advances 5, eaau6792 (2019).

[2] J. Colen, M. Han, R. Zhang, L. M. Lemma, L. Morgan, P. V. Rujigroek, R. Adkins, Z. Bryant, Z. Dogic, et al., Machine learning active-nematic hydrodynamics, Proceedings of the National Academy of Sciences 118, 10 (2021).

[3] M. J. Falk, V. Alizadeh Yazdi, H. Jaeger, and A. Murugan, Learning to control active matter, arXiv preprint arXiv:2105.04641 (2021).

[4] A. R. Dulaney and J. F. Brady, Machine learning for phase behavior in active matter systems, Soft Matter 17, 6808 (2021).

[5] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner, Learning data-driven discretizations for partial differential equations, Proceedings of the National Academy of Sciences 116, 15344 (2019).

[6] M. Harrington, A. J. Liu, and D. J. Durian, Machine learning characterization of structural defects in amorphous packings of dimers and ellipses, Physical Review E 99, 022903 (2019).

[7] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, and M. Bessa, Deep learning predicts path-dependent plasticity, Proceedings of the National Academy of Sciences 116, 26414 (2019).

[8] M. A. Bessa, P. Glowacki, and M. Houlder, Bayesian machine learning in metamaterial design: Fragile becomes supercompressible, Advanced Materials 31, 1904845 (2019).

[9] G. X. Gu, C.-T. Chen, and M. J. Buehler, De novo composite design based on machine learning algorithm, Extreme Mechanics Letters 18, 19 (2018).

[10] G. M. Coli, E. Boattini, L. Filion, and M. Dijkstra, Inverse design of soft materials via a deep learning-based evolutionary strategy, Science Advances 8, eabj6731 (2022).

[11] J.-H. Bastek, S. Kumar, B. Telgen, R. N. Glaesener, and D. M. Kochmann, Inverting the structure–property map of truss metamaterials by deep learning, Proceedings of the National Academy of Sciences 119, 1 (2022).

[12] D. Shin, A. Cupertino, M. H. de Jong, P. G. Steeneken, M. A. Bessa, and R. A. Norte, Spiderweb nanomechanical resonators via bayesian optimization: inspired by nature and guided by machine learning, Advanced Materials 2106248 (2021).

[13] E. D. Cubuk, S. S. Schoenholz, J. M. Rieser, D. B. Malone, J. Rottler, D. J. Durian, E. Kaxiras, and A. J. Liu, Identifying structural flow defects in disordered solids using machine-learning methods, Physical Review Letters 114, 108001 (2015).

[14] V. Bapst, T. Keck, A. Grabbska-Barwińska, C. Donner, E. D. Cubuk, S. S. Schoenholz, A. Obika, A. W. Nelson, T. Back, D. Hassabis, et al., Unveiling the predictive power of static structure in glassy systems, Nature Physics 16, 448 (2020).

[15] S. S. Schoenholz, E. D. Cubuk, D. M. Sussman, E. Kaxiras, and A. J. Liu, A structural approach to relaxation in glassy liquids, Nature Physics 12, 469 (2016).

[16] Y.-T. Hsu, X. Li, D.-L. Deng, and S. D. Sarma, Machine learning many-body localization: Search for the elusive nonergodic metal, Physical Review Letters 121, 245701 (2018).

[17] J. Venderley, V. Khemani, and E.-A. Kim, Machine learning out-of-equilibrium phases of matter, Physical Review Letters 120, 257204 (2018).

[18] K. Swanson, S. Trivedi, J. Lequieu, K. Swanson, and R. Kondor, Deep learning for automated classification and characterization of amorphous materials, Soft matter 16, 435 (2020).

[19] R. van Danime, G. M. Coli, R. van Roij, and M. Dijkstra, Classifying crystals of rounded tetrahedra and determining their order parameters using dimensionality reduction, ACS Nano 14, 15144 (2020).

[20] C. Miles, A. Bohrdt, R. Wu, C. Chiu, M. Xu, G. Ji, M. Greiner, K. Q. Weinberger, E. Demler, and E.-A. Kim, Correlator convolutional neural networks as an interpretable architecture for image-like quantum matter data, Nature Communications 12, 1 (2021).

[21] X. Andrejevic, J. Andrejevic, C. H. Rycroft, and M. Li, Machine learning spectral indicators of topology, arXiv preprint arXiv:2003.00994 (2020).

[22] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nature Physics 13, 431 (2017).

[23] E. P. Van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Learning phase transitions by confusion, Nature Physics 13, 435 (2017).

[24] D.-L. Deng, X. Li, and S. D. Sarma, Machine learning topological states, Physical Review B 96, 195145 (2017).

[25] Y. Zhang and E.-A. Kim, Quantum loop topography for machine learning, Physical Review Letters 118, 216401 (2017).

[26] P. Zhang, H. Shen, and H. Zhai, Machine learning topological invariants with neural networks, Physical Review Letters 120, 066401 (2018).

[27] Y. Zhang, A. Mesaros, K. Fujita, S. Edkins, M. Hamidian, K. Ch’ng, H. Eisaki, S. Uchida, J. S. Davis, E. Katamani, et al., Machine learning in electronic-quantum-matter imaging experiments, Nature 570, 484 (2019).

[28] K. Ch’Ng, J. Carrasquilla, R. G. Melko, and E. Katamani, Machine learning phases of strongly correlated fermions, Physical Review X 7, 031038 (2017).

[29] B. S. Rem, N. Käming, M. Tarnowski, L. Astria, N. Fläschner, C. Becker, K. Sengstock, and C. Weitenberg, Identifying quantum phase transitions using artificial neural networks on experimental data, Nature Physics 15, 917 (2019).

[30] A. Bohrdt, C. S. Chiu, G. Ji, M. Xu, D. Greif, M. Greiner, E. Demler, F. Grusdt, and M. Knap, Classifying snapshots of the doped hubbard model with machine learning, Nature Physics 15, 921 (2019).

[31] J. Carrasquilla, Machine learning for quantum matter, Advances in Physics: X 5, 1797528 (2020).

[32] E. van Nieuwenburg, E. Bairey, and G. Refael, Learning phase transitions from dynamics, Physical Review B 98, 060301 (2018).
A. Bohrdt, S. Kim, A. Lukin, M. Rispoli, R. Schittko, M. Knap, M. Greiner, and J. Léonard, Analyzing nonequilibrium quantum states through snapshots with artificial neural networks, Physical Review Letters 127, 150504 (2021).

H. Y. Sigaki, E. K. Lenz, R. S. Zola, M. Perc, and H. V. Ribeiro, Learning physical properties of liquid crystals with deep convolutional neural networks, Scientific Reports 10, 1 (2020).

P. Geiger and C. Dellago, Neural networks for local structure detection in polymorphic systems, The Journal of Chemical Physics 139, 164105 (2013).

C. Dietz, T. Kretz, and M. Thoma, Machine-learning approach for local classification of crystalline structures in multiphase systems, Physical Review E 96, 011301 (2017).

L.-F. Zhang, L.-Z. Tang, Z.-H. Huang, G.-Q. Zhang, W. Huang, and D.-W. Zhang, Machine learning topological invariants of non-hermitian systems, Physical Review A 103, 012419 (2021).

G. M. Celi and M. Dijkstra, An artificial neural network reveals the nucleation mechanism of a binary colloidal ab15 crystal, ACS Nano 15, 4335 (2021).

C. Scherer, R. Scheid, D. Andrienko, and T. Bereau, Kernel-based machine learning for efficient simulations of molecular liquids, Journal of Chemical Theory and Computation 16, 3194 (2020).

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Zděk, A. Potapenko, et al., Highly accurate protein structure prediction with alphafold, Nature 596, 583 (2021).

T. A. Hopf, J. B. Ingraham, F. J. Poelwijk, C. P. P. Dieleman, N. Vasmel, S. Waitukaitis, and M. van Hecke, Jigsaw puzzle design of pluripotent origami, Nature Physics 35, 128 (2017).

P. Dieden, N. Vasnel, S. Waïtuikaitis, and M. van Hecke, Jigsaw puzzle design of pluripotent origami, Nature Physics 16, 63 (2020).

Generalization for random classification spaces is not possible, only memorization [67].

M. Imaizumi and K. Fukumizu, Deep neural networks learn non-smooth functions effectively, in Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, Vol. 89, edited by K. Chaudhuri and M. Sugiyama (PMLR, 2019) pp. 869–878.

Machine learning algorithms have addressed the combinatorial nature of protein folding by targeting pairs of amino-acids that frequently enter in contact and thus that define the boundary between different folding pathways through co-evolution data [65]. This corresponds to adding a prior to the input in order to bias the sampling close to the boundary between classes.

C. Coulais, E. Teomy, K. De Reus, Y. Shokef, and M. Van Hecke, Combinatorial design of textured mechanical metamaterials, Nature 535, 529 (2016).

A. S. Meeussen, E. C. Oğuz, Y. Shokef, and M. van Hecke, Topological defects produce exotic mechanics in complex metamaterials, Nature Physics 16, 307 (2020).

A. Bossart, D. M. Dykstra, J. van der Laan, and C. Coulais, Oligomodal metamaterials with multifunctional mechanics, Proceedings of the National Academy of Sciences 118, 21 (2021).

See Supplemental Material at [URL will be inserted by publisher] for a more detailed description of strip modes.

See Supplemental Material at [URL will be inserted by publisher] for a set of rules for strip modes and numerical evidence to support the rules.

Y. P. Hong and C.-T. Pan, Rank-revealing qr factorizations and the singular value decomposition, Mathematics of Computation 58, 213 (1992).

See Supplemental Material at [URL will be inserted by publisher] for a more detailed description of the pixel representation.

See Supplemental Material at [URL will be inserted by publisher] for more details about the training and test sets.

The balanced accuracy BA is the arithmetic mean of the true class P rate and true class O rate, these rates are insensitive to the class-imbalance. Individually, these rates increase nearly identically with CNN size (Supplemental Materials), hence BA is a good combined measure for the accuracy.

We note that for larger unit cells (7 × 7 and 8 × 8) our CNNs overestimate β, even when the BA is high. This is due to the sparsity of class C, and not a fundamental problem: significantly increasing the number of training samples would decrease β. See Supplemental Material at [URL will be inserted by publisher] for the effect of increasing the size of the training set.

M. Briesch, D. Sobania, and F. Rothlauf, The randomness of input data spaces is an a priori predictor for generalization, arXiv preprint arXiv:2106.04181 (2021).

See Supplemental Material at [URL will be inserted by publisher] for a more detailed description of the random walks.

K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2, 359 (1989).

D. Rolnick, A. Veit, S. Belongie, and N. Shavit, Deep learning is robust to massive label noise, arXiv preprint arXiv:1705.10694 (2017).

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, Advances in Neural Information Processing Systems 27 (2014).

Y. Bengio, A. Lodi, and A. Prouvost, Machine learning for combinatorial optimization: a methodological tour d’horizon, European Journal of Operational Research 290, 405 (2021).

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, Understanding deep learning (still) requires rethinking generalization, Communications of the ACM 64, 107 (2021).

See https://github.com/Metadude1996/CombiMetaMaterial for code to calculate zero modes.

See https://github.com/Metadude1996/CNN_MetaCombi for code to train and evaluate convolutional neural networks.

R. van Mastrigt, M. Dijkstra, M. van Hecke, and C. Coulais, Zero Modes and Classification of a Combinatorial Metamaterial, 10.5281/zenodo.5879125 (2022).

R. van Mastrigt, M. Dijkstra, M. van Hecke, and C. Coulais, Convolutional Neural Networks for Classifying Combinatorial Metamaterials, 10.5281/zenodo.5992648 (2022).

D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Beng-
gio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al., A closer look at memorization in deep networks, in *International Conference on Machine Learning* (PMLR, 2017) pp. 233–242.

[68] F. Noé, G. De Fabritiis, and C. Clementi, Machine learning for protein folding and dynamics, *Current Opinion in Structural Biology* 60, 77 (2020).

[69] J. C. Maxwell, L. on the calculation of the equilibrium and stiffness of frames, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 27, 294 (1864).

[70] J. N. Grima, A. Alderson, and K. Evans, Auxetic behaviour from rotating rigid units, *Physica Status Solidi* (b) 242, 561 (2005).

[71] C. Coulais, C. Kettenis, and M. van Hecke, A characteristic length scale causes anomalous size effects and boundary programmability in mechanical metamaterials, *Nature Physics* 14, 40 (2018).

[72] There is a small and exponentially decreasing portion of unit cells that requires to calculate $M_k(n)$ with $n = 5$ and 6 to determine whether they belong to class I or C. We leave these out of consideration in the training data to save computational time.

[73] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[74] M. Hossin and M. N. Sulaiman, A review on evaluation metrics for data classification evaluations, *International Journal of Data Mining & Knowledge Management Process* 5, 1 (2015).

**Zero modes in combinatorial metamaterials**

In this section, we present theoretical and numerical results at the root of the classification of zero modes in the combinatorial metamaterial of Fig. 2(a). We first derive the zero modes of the building block, then we postulate a set of rules for classification of unit cells. Finally, we provide numerical proof of those rules.

**Zero Modes of the Building Block**

The fundamental building block is shown schematically in Fig. A1. Each black line represents a rigid bar, while vertices can be thought of as hinges; the 11 bars are free to rotate about the 8 hinges in 2 dimensions. The colored triangles form rigid structures, i.e. they will not deform. From the Maxwell counting (69) we obtain $N_{zm} = 2 \cdot 8 - 11 - 3 = 2$, where the 3 trivial zero modes in 2 dimensions, translation and rotation, are subtracted such that $N_{zm}$ is the number of zero modes of the building block. The precise deformation of these two zero modes can be derived from the geometric constraints of the building block.

$A + B + C + D + E = 3\pi$. 

(A1)

Next, we expand the angles from their rest position to linear order:

\[
A = \frac{\pi}{2} + \alpha, \quad B = \frac{3\pi}{4} + \beta, \quad C = \frac{\pi}{2} + \gamma, \\
D = \frac{\pi}{2} + \delta, \quad E = \frac{3\pi}{4} + \epsilon.
\]

(A2)

Then, from the condition that the bars cannot change length, we obtain

\[
1 - \cos(A) = 3 - 2\cos(C) - 2\cos(D) + 2\cos(C + D),
\]

(A3)
and
\[
\sin(D) - \frac{\sin(D + E)}{\sqrt{2}} = \sin(C) - \frac{\sin(C + B)}{\sqrt{2}}. \tag{A4}
\]

Up to first order in \(\alpha, \beta, \gamma, \delta, \epsilon\), equations (A3) and (A4) can be rewritten as:
\[
\alpha = 2\gamma + 2\delta, \tag{A5}
\]
\[
\delta + \epsilon = \beta + \gamma. \tag{A6}
\]

Together with the loop condition (A1), we obtain a set of three equations which express \(\alpha, \beta, \gamma, \delta, \epsilon\) in \(\beta, \gamma\):
\[
\begin{bmatrix}
\alpha \\
\delta \\
\epsilon
\end{bmatrix} =
\begin{bmatrix}
-2 & -2 \\
-1 & -2 \\
2 & 3
\end{bmatrix}
\begin{bmatrix}
\beta \\
\gamma
\end{bmatrix}. \tag{A7}
\]

This demonstrates that we can choose the two parameters \(\beta, \gamma\) arbitrarily, while still satisfying equations (A1), (A5) and (A6), consistent with the presence of two zero modes.

We now choose the basis of the zero modes such that the first zero mode is the deformation of the square BCDE, such that \(\alpha = 0\). This leads to the well-known counter-rotating squares (CRS) mode when tiling building blocks together. Thus we choose the basis
\[
\begin{bmatrix}
\alpha \\
\beta \\
\gamma
\end{bmatrix} = M_{CRS} \begin{bmatrix}
-1 \\
1 \\
1
\end{bmatrix} + M_D \begin{bmatrix}
3 \\
0 \\
0
\end{bmatrix}. \tag{A8}
\]

\(M_{CRS}\) is the amplitude for the counter-rotating squares mode, while \(M_D\) is the amplitude of the mode that does change corner \(A\). We refer to this mode as the diagonal mode.

By tiling together the building block in different orientations, we can create \(4^k\) size \(k \times k\) unit cells. These unit cells — and metamaterials built from them — may have more or less zero modes than the constituent building blocks, depending on the number of states of self-stress. Previous work on \(2 \times 2\) unit cells showed that each unit cell could be classified based on the number of zero modes \[48\]. Here, we consider the previously unexplored cases of \(3 \times 3\) up to \(8 \times 8\) square unit cells.

**Rule-based classification of unit cells**

Unit cells are classified based on the number of zero modes \(M_k(n)\) for \(n \geq 2\) as either class I or class C as described in the main text. Here we formulate a set of empirical rules that distinguishes class I unit cells from class C unit cells.

Any finite configuration of building blocks, no matter the orientation of each block, supports the counter-rotating squares (CRS) mode with open boundary conditions, where all building blocks will deform with \(M_{CRS} \neq 0\) and \(M_D = 0\). They must all have equal magnitude \(|M_{CRS}|\), but alternate in sign from building block to building block in a checkerboard pattern, similar to the ground state of the anti-ferromagnetic Ising model on a square lattice. An arbitrary configuration in the real space representation, and the CRS mode of that configuration in the directed graph representation, are shown in Fig. A2(a).

However, precisely because the building block supports another mode, there could in principle be other collective modes than the CRS mode in any given configuration. We have observed that class C unit cells have a specific structure, which we refer to as a strip mode. A strip mode spans the unit cell periodically in one direction, such that the total number of zero modes for a configuration of \(n \times n\) tiled unit cells grows linearly with \(n\).

The pattern of deformations for these modes consists of two rectangular patches of building blocks with CRS modes (where \(M_D = 0\) for every building block) — potentially of different amplitude — separated by a strip of building blocks (the *strip*) that connects these patches, where \(M_D \neq 0\). A unit cell configuration with a strip mode, which consists of building blocks in a strip of block-width \(W = 2\) that deform with \(M_D \neq 0\), and building blocks in the two areas outside of the strip, U & V, that do not deform, is shown in Fig. A2(a). Note that the CRS mode can always be freely added or subtracted from the total configuration.

We conjecture that the presence of a strip mode is

---

**Figure A2.** Schematic and pixel representation of modes in a \(4 \times 4\) unit cell. (a) Schematic deformation of counter-rotating squares mode (top unit cell, blue) and a strip mode (bottom unit cell, pink). The strip mode spans the entire area of the strip (white) of width \(W = 2\), while the areas U and V do not deform. (b) Respective pixel representations of the left unit cells. Paired unit cells are highlighted through red dots connected by orange lines. Note that the top unit cell does not contain a strip that meets the strip mode rules, while the bottom unit cell does.
a necessary and sufficient condition for a unit cell to be of class C.

We verify (i) below. Moreover, we now conjecture a set of necessary and sufficient conditions on the configuration of the strip that lead to a strip mode. Underlying this set of conditions is the notion of paired building blocks: neighboring blocks that connect with their respective A corners, or equivalently, blocks that have their black pixels in the same plaquette in the pixel representation, see Fig. A2(b). Depending on the orientation of the paired building blocks, pairs of these blocks are referred to as horizontal, vertical or diagonal pairs. The set of conditions to be met within the strip to have a horizontal (vertical) strip mode can be stated as follows:

ii Each building block in the strip is paired with a single other neighboring building block in the strip.

iii Apart from horizontal (vertical) pairs, there can be either vertical (horizontal) or diagonal pairs within two adjacent rows (columns) in the strip, never both.

Consider the unit cells of Fig. A2 the top unit cell has multiple paired building blocks, but contains no horizontal (or vertical) strip where every block is paired. Conversely, the bottom unit cell does contain a strip of width \( W = 2 \) blocks where every block is paired to another block in the strip. Consequently, the bottom unit cell obeys the rules and supports a strip mode, while the top unit cell does not.

Each indivisible strip of building blocks for which these conditions hold, supports a strip mode. For example, if a unit cell contains a strip of width \( W = 2 \) which obeys the rules, but this strip can be divided into two strips of width \( W = 1 \) that each obey the rules, then the width \( W = 2 \) strip supports two strip modes, not one.

We refer to (i) as the strip mode conjecture, and (ii) and (iii) as the strip mode rules. We now present numerical evidence that supports these rules.

Numerical evidence for strip mode rules

The conjecture and rules (i)-(iii) stated in the previous section can be substantiated through numerical simulation. To do so, we determine the class of randomly picked unit cells.

To assess the rules, a large number of square unit cells are randomly generated over a range of sizes \( k \in \{3, 4, 5, 6, 7, 8\} \). For each unit cell configuration, \( n_x \times n_y \) metamaterials, composed by tiling of the unit cells, are generated over a range of \( n_x = n_y = n \in \{1, 2, 3, 4\} \) for \( k \leq 4 \). From \( k \geq 5 \) onward, the \( 1 \times 1 \) configuration is generated, as well as \( n_x \times 2 \) and \( 2 \times n_y \) configurations with \( n_x, n_y \in \{2, 3, 4\} \) to save computation time.

The rigidity, or compatibility, matrix \( R \) is constructed for each of these configurations, subsequently rank-revealing QR factorization is used to determine the dimension of the kernel of \( R \). This dimension is equivalent to the number of zero modes of the configuration, \( M_k(n) \) is then equal to this number minus the number of trivial zero modes: two translations and one rotation.

From the behavior of \( M_k(n) \) as a function of \( n \), we define the two classes: I and C. In Class I \( M_k(n) \) saturates to a constant for \( n \geq 2 \), thus class I unit cells do not contain any strip modes. Note that they could still contain additional zero modes besides the CRS mode. In Class C \( M_k(n) \) grows linearly with \( n \) for \( n \geq 2 \), therefore class C unit cells could support a strip mode \[2]. Moreover, if conjecture (i) is true, the number of strip modes supported in the class C configuration should be equivalent to the slope of \( M_k(n) \) from \( n \geq 2 \) onward.

In class I, \( M_k(n) \) is constant for sufficiently large \( n \), thus class I unit cells do not contain any strip modes. Note that they could still contain additional zero modes besides the CRS mode. In class C \( M_k(n) \) grows linearly with \( n \) for sufficiently large \( n \), therefore class C unit cells could support a strip mode. Moreover, if conjecture (i) is true, the number of strip modes supported in the class C configuration should be equivalent to the slope of \( M_k(n) \) for sufficiently large \( n \).

To test conjecture (i) and the strip mode rules (ii) and (iii), we check for each generated unit cell if it contains a strip that obeys the strip mode rules. If (ii)-(iii) are correct, the number of indivisible strips that obey the rules within the unit cell should be equal to the slope of \( M_k(n) \) for class C unit cells, and there should be no strips that obey the rules in class I unit cells. Simulations of all possible \( k = 3 \) unit cells, one million \( k = 4, 5, 6 \) unit cells, two million \( k = 7 \) unit cells, and 1.52 million \( k = 8 \) unit cells show perfect agreement with the strip mode rules for unit cells belonging to either class I or C, see Fig. A3. Consequently, numerical simulations provide strong evidence that the strip mode rules as stated are correct.

Constructing and Training Convolutional Neural Networks for metamaterials

In this section, we describe in detail how we construct and train our convolutional neural networks (CNNs) for classifying unit cells into class I and C. We first transform our unit cells to a CNN input, secondly we establish the architecture of our CNNs. Next, we obtain the training set, and finally we train our CNNs.
This representation naturally divides the building blocks into $2 \times 2$ plaquettes in which paired building blocks are easily identified, see Fig. 3(a). Building blocks sharing their black pixel in the same plaquette are necessarily paired, and thus allow for deformations beyond the counter-rotating squares mode. Note that this includes diagonally paired building blocks as well. This way, the filters do not convolve on the building blocks, which do not contain any extra information for classification.

CNN architecture details

To classify the unit cells into class I and C, we use a small convolutional neural network (CNN) architecture. First, the input image is periodically padded with a pixel-wide layer, such that a $2k \times 2k$ image becomes a $2k + 2 \times 2k + 2$ image. This image is then fed to a convolutional layer, consisting of $n_f \times 2 \times 2$ filters with bias and ReLu activation function. The filters move across the input image with stride $(2, 2)$, such that the filters always look at the parts of the image showing the interactions between four building blocks (Fig. 3(a)). Subsequently the $n_f \times 1 \times 1$ feature maps are flattened and fully-connected to a hidden layer of $n_h$ neurons with bias and ReLu activation function. This layer is then fully-connected to 2 output neurons corresponding to the two classes with bias and softmax activation function. The total number of parameters for a network with $n_f$ filters and $n_h$ neurons is thus

$$
(4 + 1)n_f + ((k + 1)^2 n_f + 1)n_h + (2 + 1)n_h \quad (A9)
$$

The networks are trained using a cross-entropy loss function. This loss function is minimized using the Adam optimization algorithm [1]. This algorithm introduces additional parameters to set before training compared to stochastic gradient descent. We keep all algorithm-specific parameters as standard ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-07$), and only vary the learning rate $\eta$ from run to run.

Training set details

The networks are trained on training sets of size $|D_t|$ as shown in Table A1. For all these unit cells, the class is determined through the total number of modes $M_k(n)$ as described in the subsection Numerical evidence for strip mode rules. Since there is a strong class-imbalance in the design space, for the network to learn to distinguish between class I and C, the training set needs to be class-balanced. If the training set is not class-balanced, the networks will learn to always predict the majority class. The training set is class-balanced using random undersampling of the class I designs, and the number of class C

\[ \text{Pixel Representation} \]

To feed our design to a neural network, we need to choose a representation a neural network can understand. Since we aim to use convolutional neural networks, this representation needs to be a two-dimensional image. For our classification problem, the presence or absence of a zero mode ultimately depends on compatible deformations between neighboring building blocks. As such, the representation we choose should allow for an easy identification of the interaction between neighbors.

In addition to being translation invariant, the classification is rotation invariant. While we do not hard code this symmetry in the convolutional neural network, we do choose a representation where rotating the unit cell should still yield a correct classification. For example, this excludes a representation where each building block is simply labeled by a number corresponding to its orientation. For such a representation, rotating the design without changing the numbers results in a different interaction. For such a representation, rotating the unit cell

$$
\text{mode scaling in comparison to classification based on rules (i)-(ii). The } k \times k \text{ unit cell size is indicated on top of each matrix.}
$$

Figure A3. Confusion matrices for classification based on mode scaling in comparison to classification based on rules (i)-(ii). The $k \times k$ unit cell size is indicated on top of each matrix.
designs is artificially increased using translation and rotation of class C designs from unit cell size \( k = 6 \) onward. We then use stratified cross-validation over 10 folds, thus for each fold 90% is used for training and 10% for validation. The division of the set changes from fold to fold.

To illustrate how sparse the training set is, we divide the number of training unit cells per class, \( |D_{i}(\text{Class})| \) over the estimated total number of unit cells of that class, \( |\Omega_D(\text{Class})| \). We estimate this number for class C through multiplying the ratio of class C to class I unit cells with the total number of possible unit cells \( |\Omega_D| \): \( |\Omega_D(C)| = 4^k \times |\Omega_D| \). Likewise, we determine the ratio for class I. The resulting ratio for class C and I is shown in Fig. A4(a). Clearly, for increasing unit cell size \( k \), the class sparsity in the training set increases exponentially. Consequently, the neural networks get relatively fewer unit cells to learn the design rules bisecting the design space for increasing unit cell size.

Moreover, the training set unit cells of different class are, on average, farther removed from one another for increasing unit cell size \( k \). The distance between two unit cells \( |\Delta X| \) is defined as the number of building blocks with a different orientation compared to their corresponding building block at the same spatial location in the other unit cell. So two \( k \times k \) unit cells can at most be \( k^2 \) building blocks removed from one another, if every single building block has a different orientation compared to its corresponding building block at the same spatial location in the other unit cell. Note that we only consider different orientations in this definition, we do not define an additional notion of distance between orientations of building blocks.

By measuring the distance in number of different building block orientations \( |\Delta X| \) between every class C to every class I unit cell, we obtain the probability density function of distance in number of different building blocks between two unit cells of different class in the training set, see Fig. A4(b). Consequently, if \( k \) increases, the networks are shown fewer examples of unit cells similar to each other, but of different class. Thus the boundary between C and I is undersampled in the training set, with few I designs close to the boundary.

**CNN hyperparameter grid search details**

To see if and how well a convolutional neural network (CNN) is able to classify unit cell designs into class I or C, a hyperparameter grid search is performed. This search varied three hyperparameters: the number of filters \( n_f \), the number of hidden neurons \( n_h \), and the learning rate \( \eta \). The number of filters \( n_f \) runs from 2 to 20 in steps of 2, the number of hidden neurons \( n_h \) first runs from 2 to 20 in steps of 2, then from 20 to 100 in steps of 10. The learning rate ranges from \( \eta \in \{0.0001, 0.001, 0.002, 0.003, 0.004, 0.005\} \). For each possible hyperparameter combination, a 10-fold stratified validation is performed on a class-balanced training set. Early stopping using the validation loss is used to prevent overfitting.

To create the fold-averaged Balanced Accuracy BA heatmaps shown in Fig. 3(b), each \( (n_f, n_h) \)-iteration was picked by selecting the \( \eta \) with the highest fold-averaged BA over the validation set. Whenever this paper refers to the CNN with a given \( (n_f, n_h) \), we refer to the hyperparameter combination with learning rate with the highest fold-averaged BA over the validation set, unless stated otherwise.

**Assessing the performances of CNNs**

In this section, we describe in detail how we assess the performance of our trained convolutional neural networks (CNNs). We first quantify performance over the test set, then we define our sensitivity measure. Finally, we apply this sensitivity measure to the CNNs.

**Test set results**

After training the CNNs on the class-balanced training sets, we test their performance over the test set. The test set consists of unit cells the networks have not seen during training, and it is not class-balanced. Instead, it is highly class-imbalanced, since the set is obtained from uniformly sampling the design space. In this way, the performance of the network to new, uniformly generated designs is fairly assessed. The size of the test set \( D_{test} \) for each unit cell size \( k \) is shown in Tab. A1.

Precisely because the test set is imbalanced, standard performance measures, such as the accuracy, may not be good indicators of the actual performance of the network. There is a wide plethora of measures to choose from [74].
Since we are initially equally interested in the classification accuracy over class I as class C designs, we use the Balanced Accuracy:

\[
BA = \frac{1}{2} \left( \frac{V_{TC}}{V_{TC} + V_{FI}} + \frac{V_{TI}}{V_{TI} + V_{FC}} \right)
\]

(A10)

\[
= \left\langle \frac{1}{2} (TCR + TIR) \right\rangle,
\]

(A11)

where \(V_{TC}, V_{TI}, V_{FC}, \) and \(V_{FI}\) are the volumes of the subspaces true class C TC, true class I TI, false class C FC, and false class I FI (Fig. A1(a)). We do not consider other commonly used performance measures for class-imbalanced classification, such as the \(F_I\) score, since they are sensitive to the class-balance.

The BA can be understood as the arithmetic mean between the true class C rate TCR (sensitivity), and true class I rate TIR (specificity). As such, it considers the performance over all class C designs and all class I designs separately, giving them equal weight in the final score. Class-imbalance therefore has no impact on this score.

The usage of BA to show trends between neural network performance and hyperparameters is warranted, since no significant difference between the true class C rate TCR and true class I rate TIR appears to exist, see Fig. A5. Evidently TCR and TIR depend similarly on the number of filters \(n_f\) and number of hidden neurons \(n_h\). This is to be expected, since the networks are trained on a class-balanced training set.

The effect of class-imbalance on CNN performance can be further illustrated through constructing the confusion matrices (Fig. A6(b)). Though all CNNs show high true C and I rates, the sheer number of falsely classified C unit cells can overtake the number of correctly classified C unit cells if the class-imbalance is sufficiently strong. This effect is captured by the predicted volume \(\bar{\beta}\), which is orders of magnitude removed from the true volume \(\beta\) for CNNs trained on \(7 \times 7\) and \(8 \times 8\) unit cells (Fig. A1(a)). However, \(\bar{\beta}\) can be improved through increasing the training set size.

**Increasing the size of the training set**

To illustrate how the size of the training set \(D_t\) influences the predicted volume \(\bar{\beta}\), we compare CNNs trained on two training sets of different size consisting of \(7 \times 7\) unit cells. The training sets are obtained from \(1M\) and \(2M\) uniformly sampled unit cells respectively, and the number of class C unit cells is artificially increased using translation and rotation to create class-balanced training sets. The best predicted volume \(\bar{\beta}\) is a factor 2 smaller for CNNs trained on the larger training set, compared to the smaller training set (Fig. A7). Thus, the overestimation of \(\bar{\beta}\) is due to the sparsity of the training set, and can be improved through increasing the number of training samples.

**Random walk near the class boundary**

To better understand the complexity of the classification problem, we probe the design space near test set unit cell designs. Starting from a test set design \(X_0\) with true class C, we rotate a randomly selected unit cell to create a new unit cell design \(X_1\). We do this iteratively up to a given number of steps \(s\) to create a chain of designs. For each generated design, we assess the new true class using

### Table A1. Details of the hyperparameter grid search.

| \(k\) | size of \(D_t\) | size of \(D_{test}\) |
|---|---|---|
| 3 | 31180 | 39321 |
| 4 | 397914 | 150000 |
| 5 | 793200 | 149980 |
| 6 | 1620584 | 150000 |
| 7 | 292432 | 600000 |
| 8 | 1619240 | 144000 |

**Figure A5.** True class rates. (a) Heatmaps of the fold-averaged true class C rate \(\langle TCR \rangle\) for CNNs with \(n_f\) filters and \(n_h\) hidden neurons trained on the unit cell size \(k \times k\) indicated on top of each heatmap. (b) Heatmaps of the fold-averaged true class I rate \(\langle TIR \rangle\).
the design rules.

For each unit cell size \(k\), we take \(s = k^2\) steps in design space. The probability to transition from an initial design \(X_0\) of class C to another design \(X_s\) of class C as a function of \(s\) random walk steps in design space \(\rho_{C\rightarrow C}(s)\), is shown in Fig. A8. A clear difference between the different unit cell sizes is visible. Both the rate at which the probability decreases initially, and the value to which it saturates differs per unit cell size.

For even unit cell size, the dominant strip mode width is \(W = 1\) and each class C design is most likely to just have a single strip mode. Thus, the probability to transition from C to I relies on the probability to rotate a unit cell inside the strip of the strip mode, which is \(1/k\), so \(\alpha_t \approx 1/k\). For odd unit cell sizes, the dominant strip mode width is \(W = 2\), such that \(\alpha_t \approx 2/k\).

To understand the asymptotic behavior, we note that for large \(s\) the unit cells are uncorrelated to their original designs. Thus, the set of unit cells are akin to a uniformly sampled set of unit cells. Consequently, the probability to transition from C to C for large \(s\) is approximately equal to the true class C volume \(\beta\).

**Random walk near the decision boundary**

In addition to the true class, we can assess the predicted class by a given network for each unit cell in the random walk. This allows us to probe the decision boundary, which is the boundary between unit cells that a given network will classify as C and those it will classify as I. By comparing the transition probabilities for given networks to the true transition probability we get an indication of how close the decision boundary is to the true class boundary.

To quantitatively compare the true class boundary

![Figure A7](image-url)  
**Figure A7.** Predicted volume \(\tilde{\beta}(n_h)\) minus the true volume \(\beta\) for CNNs with \(n_f = 20\) trained on a smaller training set (circles) and larger training set (squares). The size of the training set is indicated in the legend.

![Figure A8](image-url)  
**Figure A8.** Probability \(\rho_{C\rightarrow C}\) (polygons) to transition from initial design \(X_0\) of class C to another design \(X_s\) of class C as a function of \(s\) random walk steps in design space starting from the initial design. The legend indicates the polygon and color for each unit cell size \(k\). The continuous lines are obtained from a least-squares fit using Eq. (1) of the Main Text.

![Figure A6](image-url)  
**Figure A6.** Confusion matrices over the test set for trained CNNs with the highest accuracy over the class-balanced validation set. The \(k \times k\) unit cell size is indicated on top of each matrix.
with the decision boundaries, we fit the measured transition probability for each network to Eq. (1) of the Main Text with $\bar{\alpha}$ as fitting parameter. We start from designs with true and predicted class C, and track the predicted class for the random walk designs. We set the asymptotic value to the predicted class C volume $\bar{\beta}$ for each network. We do this for CNNs with fixed number of filters $n_f = 20$ and varying number of hidden neurons $n_h$. We select the networks with the best-performing learning rate over the validation set, and obtain a 10-fold averaged estimate of $\bar{\alpha}$ for each $n_h$. The results are shown in Fig. 5.