Supplementary Information: Predicting magnetic edge behaviour in graphene using neural networks

Meriç E. Kucukbas,[1] Seán McCann,[1] and Stephen R. Power[1, 2, *]
[1] *School of Physics, Trinity College Dublin, Dublin 2, Ireland*
[2] *School of Physical Sciences, Dublin City University, Glasnevin, Dublin 9, Ireland*

(Dated: August 15, 2022)

This supplementary information contains further details about

- the tight-binding and mean-field Hubbard models
- the different types of geometries considered in this work
- the training of the neural network
- the accuracy of the model on different types of sites
- spin-polarised transport in pristine and edge-disordered ZGNRs
- using the *MLene* tool to predict moments for arbitrary geometries

S1. TIGHT-BINDING AND SELF-CONSISTENT CALCULATIONS

A nearest-neighbour tight-binding model is used to describe the electronic structure of the graphene flakes, with an onsite Hubbard term capturing electron-electron interactions. Within the mean-field approximation, the many-body term reduces to a spin-dependent onsite potential $\epsilon_{i\sigma}$ at each site i. The Hamiltonian is given by

$$\mathcal{H} = \sum_{i,j,\sigma} t_{ij}\, \hat{c}^{\dagger}_{j\sigma}\, \hat{c}_{i\sigma} + \sum_{i,\sigma} \epsilon_{i\sigma}\, \hat{c}^{\dagger}_{i\sigma}\, \hat{c}_{i\sigma} \,, \tag{1}$$

where $\hat{c}^{\dagger}_i$ ($\hat{c}_i$) creates (annihilates) an electron with spin σ in site i, the hopping parameter t_{ij} takes the value $t = -2.7$ eV when sites i and j are nearest neighbours and is zero otherwise. The magnetic moments at each site, m_i, enter through $\epsilon_{i\sigma} = -\sigma \frac{U}{2} m_i$. A Hubbard parameter $U = 1.33|t|$ gives results in good agreement with DFT calculations [1]. The values of m_i are calculated self-consistently by taking an initial guess for the moment profile and using exact diagonalization to calculate the energy levels. The individual occupations of the ↑ and ↓ states at each site are then calculated, generating a new estimate for the moment profile. This procedure is iterated until the input and output moments at each site are converged to within $10^{-5}\mu_B$. We consider only the energetically-favourable antiferromagnetic (AFM) solutions, which are found by choosing an AFM initial guess. We also focus exclusively on undoped graphene, which imposes a half-filling constraint on the occupation.

Transmission through disordered ribbons is calculated using the TBtrans code [2], via the expression $T(E) = \mathrm{Tr}\left[G^R(E)\Gamma_R(E)G^A(E)\Gamma_L(E)\right]$. Here $G^{R/A}$ are retarded and advanced Green's functions, and $\Gamma_{L/R}$ are the broadening functions associated with pristine semi-infinite leads. The moment profiles for pristine periodic systems are used in the leads, with only the moments in the device region calculated self-consistently. Pristine buffer regions (10 unit cells wide) at either side of the disordered region are included in the self-consistent calculation to ensure accurate profiles where the disordered and pristine regions meet.

S2. GEOMETRIES

The training, validation and test sets include flake geometries created using three different methods.

[*] stephen.r.power@dcu.ie

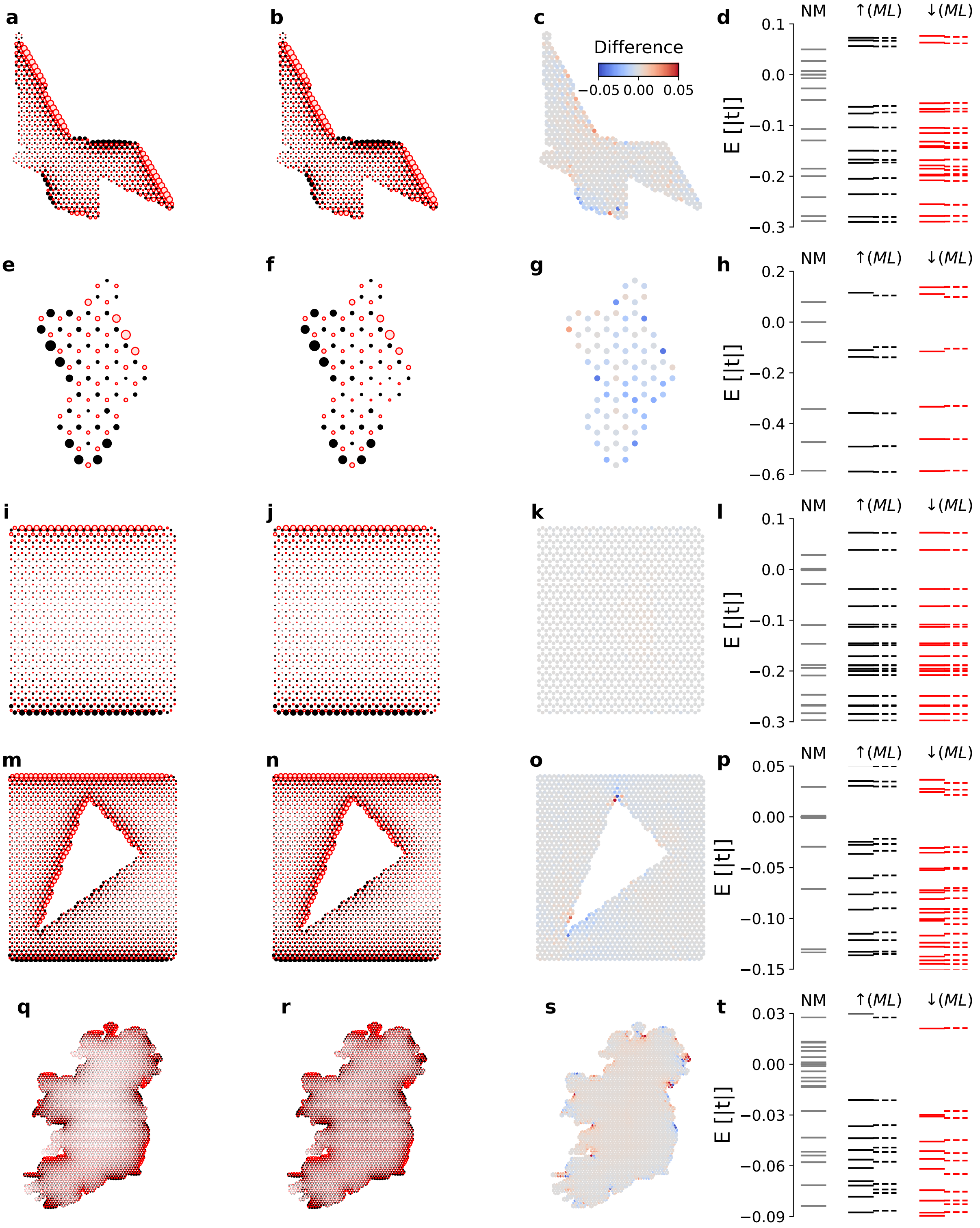


FIG. S1. **Examples of different geometry types and associated neural network performance.** The first row of panels reproduces the results for the polygon geometry shown in Fig. 2 (c,e) of the main paper, showing maps of the **(a)** actual moments, **(b)** predicted moments and **(c)** deviation between the two. **(d)** compares the energy levels calculated with the two methods, together with those of the non-magnetic dot. The subsequent rows show the same analysis for **(e-h)** an edge-disordered dot from the test set; **(i-l)** a finite ZGNR segment from the test set; **(m-p)** the antidot system and **(q-t)** the Irish coastline flake discussed in Fig. 3.

- **Random polygons:** A polygon is created by choosing a random number of vertices in the range $[4, 20]$, and placing them around the origin subject to certain constraints (minimum and maximum distance from origin). This shape is then used to etch a $N \times N$ supercell of graphene (with N chosen randomly), before finally any dangling atoms are removed. This approach yields irregular structures as shown in Fig. 1 and 2 in the main paper and Fig. S1a, which typically have long edge sections with a wide range of geometries (zigzag, armchair and chiral) meeting each other at a wide range of different angles.

- **Edge-disordered dots:** Starting from a diamond-shaped supercell of graphene, etching sweeps are performed along all edges to probabilistically remove edge sites. Dangling atoms are removed after the final sweep. The starting size (N), number of sweeps (p) and removal probability (P) are chosen randomly over a wide parameter space. This approach yields dots of varying size whose edges are highly disordered and do not maintain particular directions for very long. An example is shown in Fig. S1e.

- **Finite ribbons:** Pristine, finite ZGNR segments with randomly chosen widths and lengths, such as that shown in Fig. S1i, were also included. This allowed the model to learn the expected behaviour near long sections of pristine edges. The performance on these systems was much better than for irregular systems, as is clearly shown in Fig. 2b of the main manuscript. The breakdown of model accuracy by site type in Fig. S3 excludes these systems to give a better indication of the model performance on irregular systems.

The antidot system in Fig. 3c (Fig S1m-p) was generated by applying the random polygon method in reverse to a pristine finite ribbon, and removing the atoms *inside* the shape instead of those outside of it. The highly-irregular Irish coastline geometry in Fig. 3c (Fig S1q-s) was generated by applying a map-based mask to the graphene lattice. A comparison of the real and actual moments, together with a map showing deviations and a comparison of energy levels calculated with both sets of moments, is shown for all five geometry types in Fig. S1.

S3. NEURAL NETWORK TRAINING

Descriptor vectors for each site are generated from the connectivity matrix D for each flake, where D_{ij} is the number of carbon-carbon bonds along the shortest path connecting sites i and j. We generate a reduced connectivity matrix of dimension $2(N_Z + N_C + N_A) + 1$ for each site. This contains the elements of D corresponding to the site of interest, and the N_Z nearest zigzag, N_C nearest corner and N_A nearest armchair edge sites from each sublattice. The classification of edge sites as 'Zigzag', 'Corner', or 'Armchair' is determined from the connectivity of a site and its neighbours [3]. Fig. 1 in the main text shows the sites considered for the simple $(N_Z, N_C, N_A) = (3, 1, 2)$ case, whereas the descriptor used to generate the main results uses $(N_Z, N_C, N_A) = (11, 3, 5)$. The descriptor is subject to a finite cut-off and overall renormalization of $r_{\text{cut}} = 3.7$ nm.

The library of flakes is split in the ratio 7 : 2 : 1 to form training, validation and test sets for the training process. The neural networks are trained using the TensorFlow [4] and keras [5] packages. Each neuron employed rectified linear unit (ReLU) activation with the He normal distribution of initial weights, and the networks were trained using an adaptive learning rate algorithm (ADAM). The mean absolute error (MAE) and mean square error (MSE) were used to compare different networks and descriptors, with better results found by optimizing the MAE during training. The optimal network presented in the main manuscript has 6 hidden layers, with 1000 neurons in all but the final layer, which contains 200.

A variety of descriptor definitions and network hyperparameters were considered, with the best performance found using the model described above. The descriptors for this model were generated as follows:

- The Euclidean distance matrix for a geometry was generated, and used to identify neighbouring sites, and site types (bulk, zigzag, armchair, corner).

- The `shortest_path` routine from the `scipy.sparse` module was used to generate a shortest-path connectivity matrix D.

- A maximum distance $r_{\text{cut}} = 3.7$ nm was chosen, and all distances D_{ij} adjusted such that $D_{ij}^{\text{new}} = \max(D_{ij}^{\text{old}}, r_{\text{cut}})$

- For each site in a flake:

  - the indices of the 11 nearest zigzag edge sites, 3 nearest corner sites and 5 nearest armchair edge sites *from each sublattice* were found and ordered by distance, type and sublattice.

  - Together with the index of the site of interest, these were used to create a reduced 39×39 connectivity matrix $D_{\text{red.}}$.

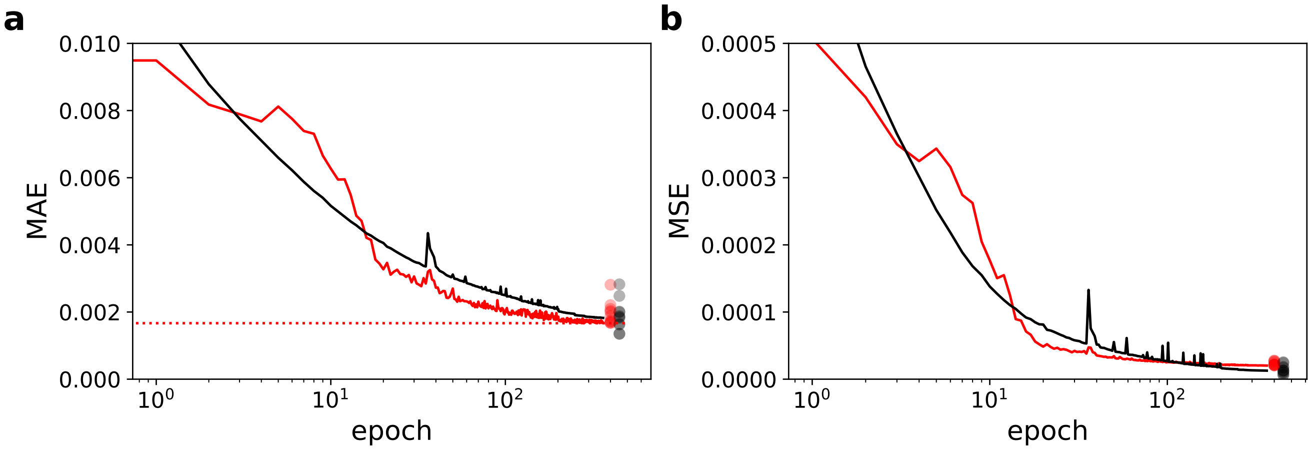


FIG. S2. **Training curves.** **a** Mean absolute error (MAE) and **b** Mean squared error (MSE) on the training (black) and validation (red) sets during each epoch of the training procedure. The network was trained by minimising the MAE. The black and red dots show the best values achieved for other networks trained on the same descriptor, but with different network topologies or hyperparameter choices.

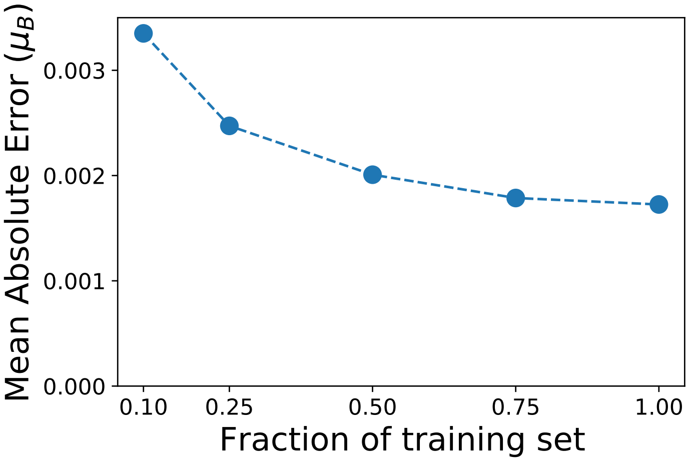


FIG. S3. **Effect of training set size** Mean absolute error for different sized training sets, expressed as a fraction of the training set flakes used in the main manuscript.

– The super-diagonal elements of $\frac{1}{r_{\mathrm{cut}}} D_{\mathrm{red.}}$ were placed in a one-dimensional array to act as the descriptor for that site.

Variations on this descriptor: (i) with different numbers of edge sites, (ii) taking just first row of $D_{\mathrm{red.}}$ instead of all super-diagonal elements, (iii) using a Euclidean distance metric, and/or (iv) using different values for r_{cut} were also tested, but gave less accurate predictions after training.

The tensorflow[4] and keras[5] packages were used to train the neural networks, with a wide range of network topologies and hyperparameter values tested. The best accuracy was obtained using the following setup:

- a 6-layer network, with 1000 neurons in each of the first 5 layers, and 200 in the final layer.
- Rectified linear unit (ReLU) activation for each neuron, with a He normal distribution of initial weights.
- The adaptive learning rate algorithm (ADAM) was employed to minimise the mean absolute error (MAE), with a batch size of 2048, and an initial learning rate of 10^{-4}.
- The learning rate was reduced during training using the `ReduceLROnPlateau` callback in keras.

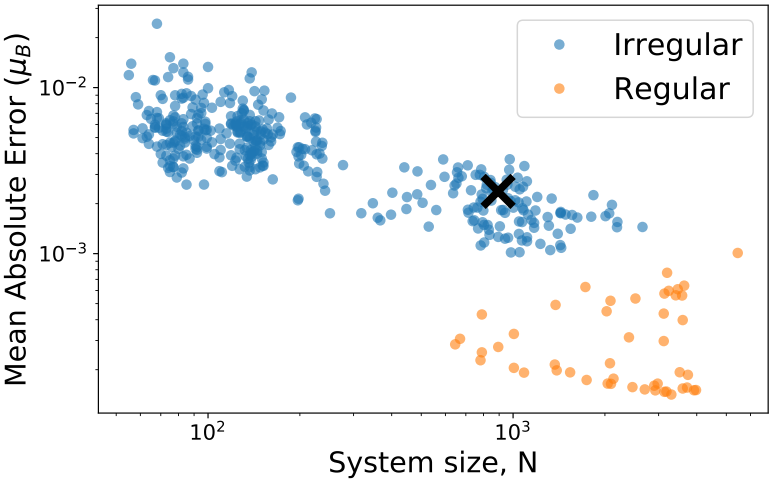


FIG. S4. **Effect of flake size on accuracy** Mean absolute error (MAE) of every system in the test set, according to flake size and etching method. Irregular flakes includes polygon and edge-disordered flakes, whereas regular systems consist of finite, pristine ZGNR segments. The 'X' marks the particular system shown in the bottom panels of Fig. 2 in the main manuscript.

- The training was performed using the CPU implementation of tensorflow and keras, using data generators to avoid memory issues associated with loading the entire training data set at once.
- A dropout rate of 12% was used throughout the network to prevent overfitting.

The training curves obtained for this model are shown in Fig. S2, which show the (a) MAE and (b) MSE as a function of epoch number. Similar validation set (red curve) accuracies could be obtained for different network topologies and hyperparameter (initial learning rate, dropout rate, etc.) values. We note that other training regimes gave lower errors on the training set alone, suggesting that any improvement here was as a result of overfitting. The best accuracies achieved by these other models are shown by black (training) and red (validation) dots in Fig. S2.

We also considered how the number of flakes in the training set affected the accuracy of the trained network. To do so, we kept the same validation and test sets as in the main manuscript, but restricted the training set to a fraction of the flakes in the full set. For each fraction of the training set, we considered three different values of the dropout parameter to account for possible differences in the regularization required for different training set sizes. Fig. S3 shows the minimum MAE achieved as the training set size is increased. While significant improvements are noted as the size of the training set initially increases, the MAE curve is quite flat near 1.00 (e.g. the complete training set). This suggests that the training set size used in the study is of an appropriate size, and any enlargement is unlikely to yield significant improvement in prediction accuracy.

S4. ACCURACY BY FLAKE SIZE AND SITE TYPE

While the mean absolute error (MAE) is a useful metric to compare networks trained on the same set, as in Fig. 2(b,c) of the main manuscript, its magnitude depends on the specific flakes in the test set. Fig. S4 shows that the MAE for a single flake depends on its size. Larger flakes have more bulk sites, whose moments (and associated errors) tend to be very small. Smaller systems have, proportionally, more edge sites with larger moments and prediction errors. The MAE decays as $\sim 1/\sqrt{N}$ for irregular flakes, where N is the number of sites in the flake, confirming that the MAE follows the relative edge length of the flake. Regular geometries have a much smaller MAE due to their uniform moment profiles.

The overall MAE of the test set therefore gives limited absolute information, as it depends sensitively on the distribution of flake sizes. However, relative error is also not a particularly useful overall metric, as sites deep in the bulk have huge relative errors, despite the moments and absolute errors being so small as to have no meaningful effect on physical quantities. To give more detail of the accuracy of the final model, Fig. S5 shows a complete breakdown of errors by site type. This analysis is restricted to the irregular geometries, *i.e.* does not include the pristine ribbon segments on which the model performs excellently due to their uniformity. The top row panels compare the model performance on bulk and edge sites. Bulk sites far outnumber edge sites, particularly in larger flakes, whereas edge sites tend to have larger moments and thus a stronger influence on low-energy electronic properties. The bottom row

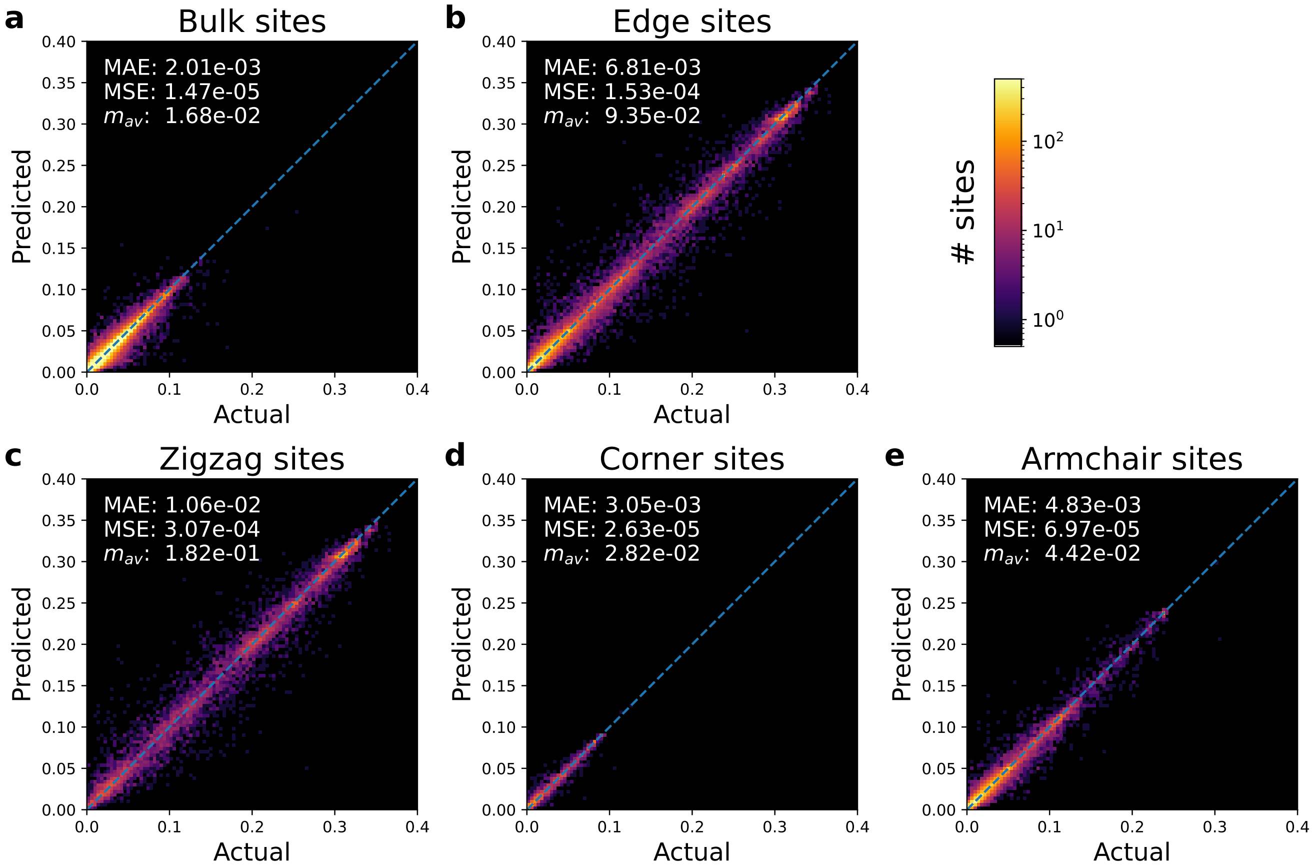


FIG. S5. **Comparison of actual and predicted moments by site type.** Similar to Fig. 2a in the main text, but restricted to sites in the irregular geometries in the test set and broken down by type of site. For each set of sites, the mean absolute error (MAE), mean square error (MSE) and average moment value (m_{av}) are also given. **(a, b)** Comparison between the performance of the model on bulk and edge sites. **(c, d, e)** Breakdown of edge sites by type: zigzag, corner and armchair. All moment magnitudes in units of μ_B.

panels compare the three different types of edge site considered in our model: zigzag, corner and armchair. Although edge magnetism is most closely associated with zigzag edges, we note that sizeable moments can appear on corner and armchair sites also. In all cases, the distributions shown here peak along the diagonal, corresponding to exact agreement between predicted and actual moments, and decay rapidly away from it. In each case, the MAE and average magnetic moment are also shown, and together give an estimate of the relative error for each type of site. For comparison, the magnetic moment at the centre of an extended zigzag edge is $\sim 0.31\mu_B$.

S5. TRANSPORT

The main manuscript discusses the effect of a long-ranged smooth edge disorder on the electronic and spin transport properties of ZGNRs. The inset of Fig 4b shows the current flow and spin polarisation at a single energy, highlighted by the arrow in both the main paper and Fig. S6a. Clear spin-polarised channels, with opposite orientations, are visible at the top and bottom edges of the ribbon. The persistence of these channels in the presence of edge disorder is discussed in the manuscript, and Fig. S6 shows additional current maps allowing the flows in pristine and disordered ribbons, and at higher energies, to be visualised. Fig. S6b considers the same energy shown in the paper (black arrow in Fig. S6a), but for a pristine ribbon. The disordered case from the manuscript is reproduced in Fig. S6c for comparison. The spin channels are not diminished significantly by the introduction of edge-disorder. This is due to a lack of back-scattering possibilities for current in the spin-polarised edge channels. In contrast, Fig. S6d and e compare the pristine and disordered ribbons for a higher energy, marked by the blue arrow in Fig. S6a. In this case, the presence of higher-order spin-degenerate modes with weight near the edges diminishes the overall spin-polarisation of current in the edge regions, even for the pristine case shown in Fig. S6d. These modes also provide additional

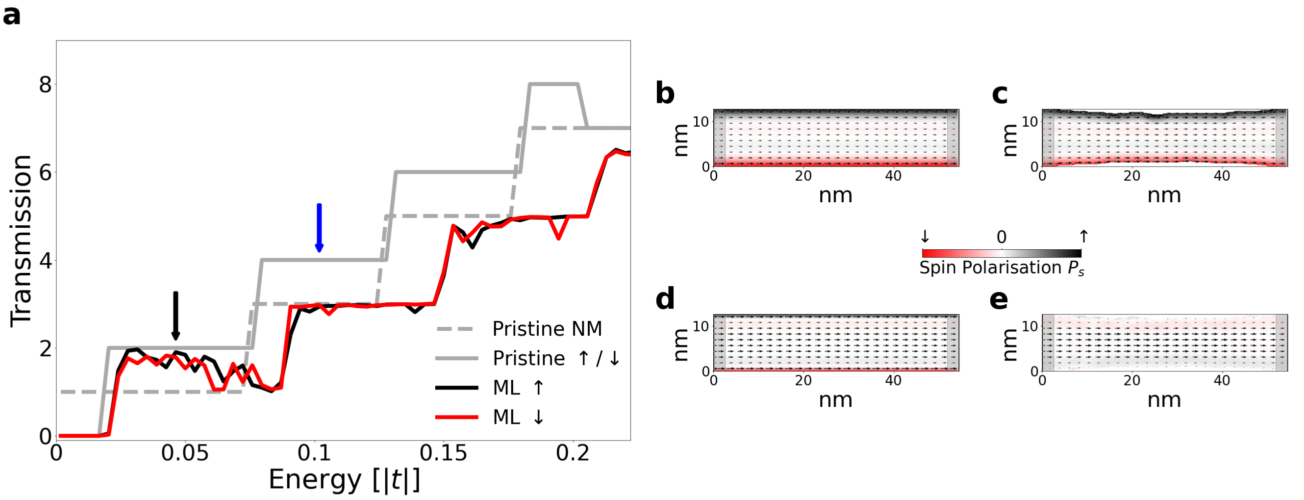


FIG. S6. **Transport in pristine and disordered nanoribbons.** **(a)** Reproduction of Fig. 4b from the main text, showing the spin- polarised transmissions (black for spin up, red for spin down) in a large, edge-disordered ZGNR. The light grey curves show transmission for the corresponding pristine ribbon, both for spin-polarised (solid) and non-magnetic (dashed) calculations. **(b,c)** Comparison of current flow and polarisation in the pristine and disordered case at the energy shown by the black arrow in panel a. (Panel c reproduces the inset of Fig. 4b from the main manuscript.) **(d,e)** A similar comparison for the higher-energy blue arrow.

back-scattering mechanisms in the presence of edge-disorder, so that the spin-polarised channels are notably absent in Fig. S6e.

S6. USING THE *MLENE* TOOL

See `https://github.com/ml-ene/ml-ene` to download a working version of the predictive model used in the manuscript. The code provided requires python 3, basic scientific packages (numpy, scipy) and the tensorflow package. This repository also includes interactive python notebooks to allow predictions to be made for user-defined geometries. It is straightforward to convert a coordinate file into a suitable format, create a descriptor, predict moments and calculate energy levels for arbitrary structures.

The complete repository, including sample geometries, can be cloned with the git version control system using the command: `git clone https://github.com/ml-ene/ml-ene`. Fig. S7 shows an example of MLene being used to predicted moments and compare the model performance on an example system.

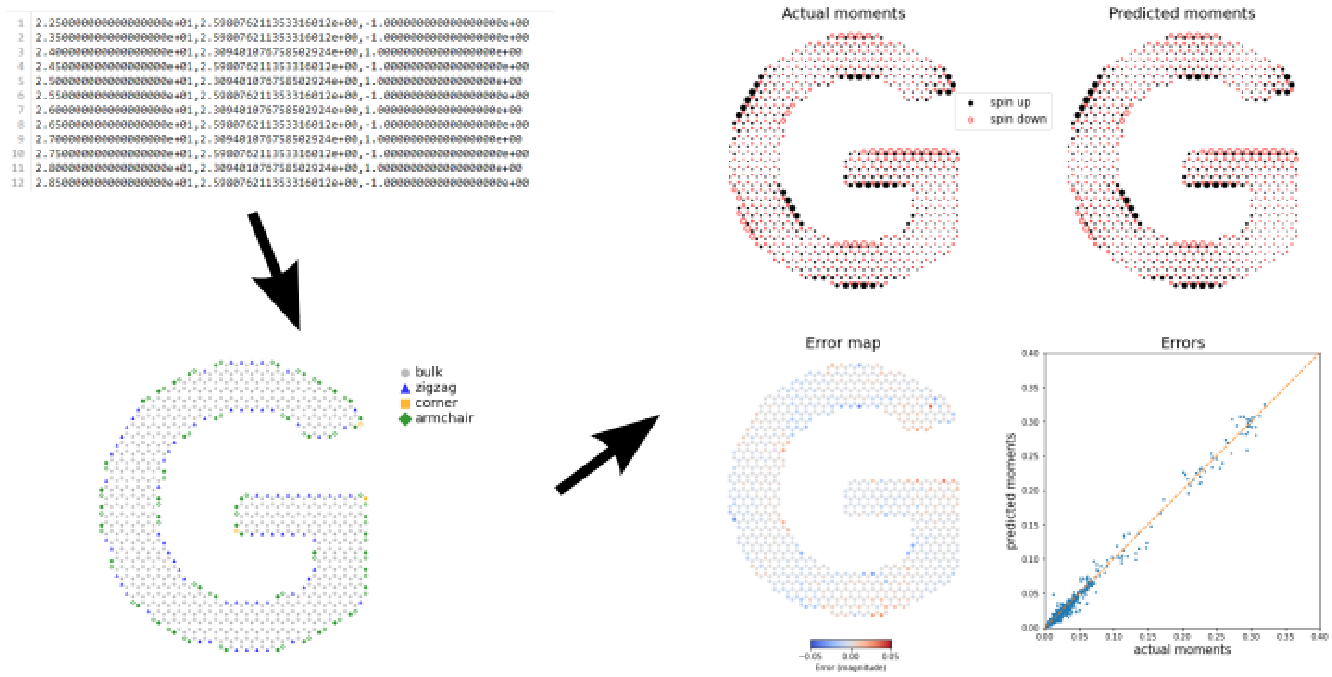
[1] O. V. Yazyev, Emergence of magnetism in graphene materials and nanostructures, Reports on Progress in Physics **73**, 056501 (2010).

[2] N. Papior, N. Lorente, T. Frederiksen, A. García, and M. Brandbyge, Improvements on non-equilibrium and transport Green function techniques: The next-generation transiesta, Computer Physics Communications **212**, 8 (2017).

[3] A. K. Singh, E. S. Penev, and B. I. Yakobson, Armchair or Zigzag? A tool for characterizing graphene edge, Computer Physics Communications **182**, 804 (2011).

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.

[5] F. Chollet *et al.*, Keras, `https://keras.io` (2015).



```
2.250000000000000000e+01,2.598076211353316012e+00,-1.000000000000000000e+00
2.350000000000000000e+01,2.598076211353316012e+00,-1.000000000000000000e+00
2.400000000000000000e+01,2.309401076758502924e+00,1.000000000000000000e+00
2.450000000000000000e+01,2.598076211353316012e+00,-1.000000000000000000e+00
2.500000000000000000e+01,2.309401076758502924e+00,1.000000000000000000e+00
2.550000000000000000e+01,2.598076211353316012e+00,-1.000000000000000000e+00
2.600000000000000000e+01,2.309401076758502924e+00,1.000000000000000000e+00
2.650000000000000000e+01,2.598076211353316012e+00,-1.000000000000000000e+00
2.700000000000000000e+01,2.309401076758502924e+00,1.000000000000000000e+00
2.750000000000000000e+01,2.598076211353316012e+00,-1.000000000000000000e+00
2.800000000000000000e+01,2.309401076758502924e+00,1.000000000000000000e+00
2.850000000000000000e+01,2.598076211353316012e+00,-1.000000000000000000e+00
```

FIG. S7. **MLene.** Using the trained model and sample code provided with the MLene tool, available from `https://github.com/ml-ene/ml-ene`, users can easily convert any graphene geometry file to a descriptor, and use this to predict moments and analyse the model performance.