A Ising Model

The Ising system can be described by its Hamiltonian. The energy of the system is given by the interaction between the neighboring dipoles and the interaction of those dipoles with an external field applied to the system. The Hamiltonian is written as:

$$H = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j - \mu\sum_i h_i\sigma_i \tag{1}$$

where $\langle i, j\rangle$ is the sum over the nearest neighbours, J_{ij} is the coupling force between the i^{th} and j^{th} magnetic dipole, $\sigma \in \{-1, 1\}$ is the magnetic dipole of a given site, μ is the magnetic moment and h is an external field applied to the lattice.

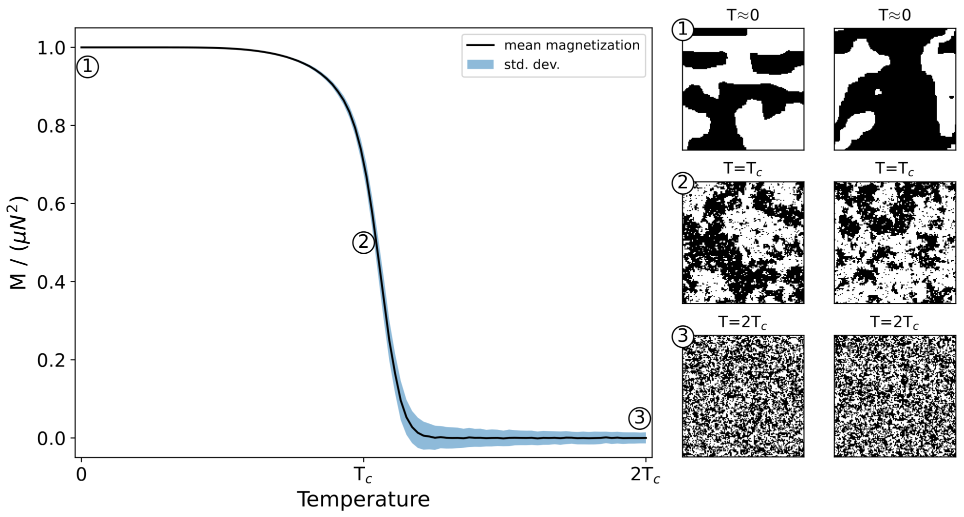


Figure 7: Evolution of the global magnetization of the lattice against the temperature. We can see that the system undertakes a phase transition at the Curie temperature T_c Some examples of the microstructure obtained at different temperatures are shown on the right. The image size is 128×128 pixels.

In our case, we simplify Eq. (1) by setting the external field h to 0 and the coupling force J to 1 for all the magnetic dipole. By fixing $J > 0$, we are in the ferromagnetism regime: the spins in the lattice tend to align in the same direction. The equation becomes:

$$H = -\sum_{\langle i,j \rangle} \sigma_i\sigma_j \tag{2}$$

When neighboring dipoles have the same sign, we can see in Eq. (2) that the energy of the configuration is minimized. One of the specifics of this model is that, in the 2D case and above, the system exhibits a phase transition near the Curie temperature. This phase transition is characterized as going from a large, ordered structure of spins oriented in the same direction, to spins that are randomly oriented without any clear pattern. One way to track this phase transition is with the evolution of the global magnetization $\langle M\rangle$ over a range of temperatures. The global magnetization is the sum of all the dipoles over the number of sites of the lattice. The evolution of the global magnetization and the resulting phase transition can be seen in Fig. 7 with the corresponding example of microstructure for different temperatures. Note, that at exactly $T = 0$, the whole image should contain only positive or only negative dipoles (which additionally may lead to numerical problems). In the figure, the microstructures for the smallest temperature are therefore obtained for a very small positive value.

Simulation set up The Ising system is simulated using the Metropolis Monte Carlo algorithm with periodic boundary conditions. We randomly initialize the directions of the $N \times N$ dipoles in the lattice, where in our case $N = 128$ was chosen. Then, we randomly select one site $s(i, j)$, where s stores the lattice state and $i, j = 1, ..., N$ and flip the corresponding magnetic dipole. The energy of this new configuration is then calculated with Eq. (2). If the energy of the new configuration is smaller than the previous one, we keep the new configuration. If the energy of this configuration is larger than the previous one, we only keep it with a probability $p = e^{-\beta\Delta E}$, $\Delta E = H_p - H_n$, H_p and H_n is the energy of the previous and next configuration respectively and $\beta = \frac{1}{k_B \mathrm{T}}$, where T is the temperature of the system and k_B the Boltzmann constant, set to 1 for simplicity. Those steps are repeated until the system reaches equilibrium or if the number of steps reaches our stopping criterion. We choose N^3 as a stopping criterion for the maximum number of steps for MMC algorithm, where N is the size of the lattice. The idea behind this criteria is that every site of the $N \times N$ lattice is visited approximately N times so that the information has enough time to travel through all the lattices.

The dipole configuration from the simulation is then saved as a black and white image (0 for a negative dipole and 255 for a positive one). The images are generated from 0 to $2 \times \mathrm{T}_c$, where T_c is the Curie temperature at which the system undergoes a phase transition as shown in Fig. 7 and is defined as in the general case as

$$\mathrm{T}_c = \frac{2J}{k_B \ln(1+\sqrt{2})} \tag{3}$$

Which can be simplify to $2/\ln(1+\sqrt{2}) \approx 2.269$ by assuming that $k_B = 1$ and $J = 1$.

B Architecture Design

The architectures for CcGAN and BcGAN used the same model, based on the SAGAN architecture [19], and were trained for 100 epochs using a batch size of 200. We used the Adam optimizer [8] with a learning rate of 10^{-4} and $\beta = (0, 0.99)$.

While CcGAN normalizes the conditioning values, using the binary representation removes the need for it as all values will be encoded in the same manner, following defined rules. The embedding network consists of two fully connected layers with a hyperbolic activation function used after each layer. The first fully connected layer is divided into 3 parts, one for each constituting part of the binary representation of the floating point number, as seen in Fig. 8. The embedding network is added inside the generator and the discriminator and is trained alongside the rest of the respective model.

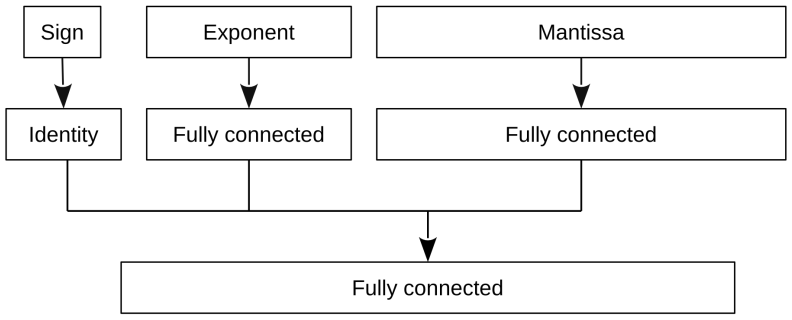


Figure 8: Graphical representation of the embedding network used for the BcGAN architecture

C Neuron Statistics

A boxplot gives a graphical representation of the distribution of the data, offering a visual summary of its key characteristics. The box in the plot spans the interquartile range, with a line inside representing the median. This provides a quick insight into the central tendency and spread of the data. The whiskers extend from the box, indicating the data variability.

In order to visualize the statistics from each embedding, each GAN (i.e., the CGAN, CcGAN, and BcGAN) are first trained on the Ising model. Once the models are trained, we extract their embedding layer. Each embedding has a different “working range”, i.e., the range of numbers they operate with:

The CGAN embedding expects a vector containing information on the classes, ranging from 0 to the number of classes in the dataset. The CcGAN embedding expects a vector containing the normalized labels, ranging from 0 to 1. Finally, the BcGAN embedding expects value in the working range of the float32, thus the label of the dataset can directly be used. For each embedding strategy, we give as an input a vector containing values in their respective working range that are uniformly distributed between the minimum and the maximum. As an output, we get the vector used for the conditioning of the model, on which we can compute and display the statistics.

D Sensitivity with respect to the label

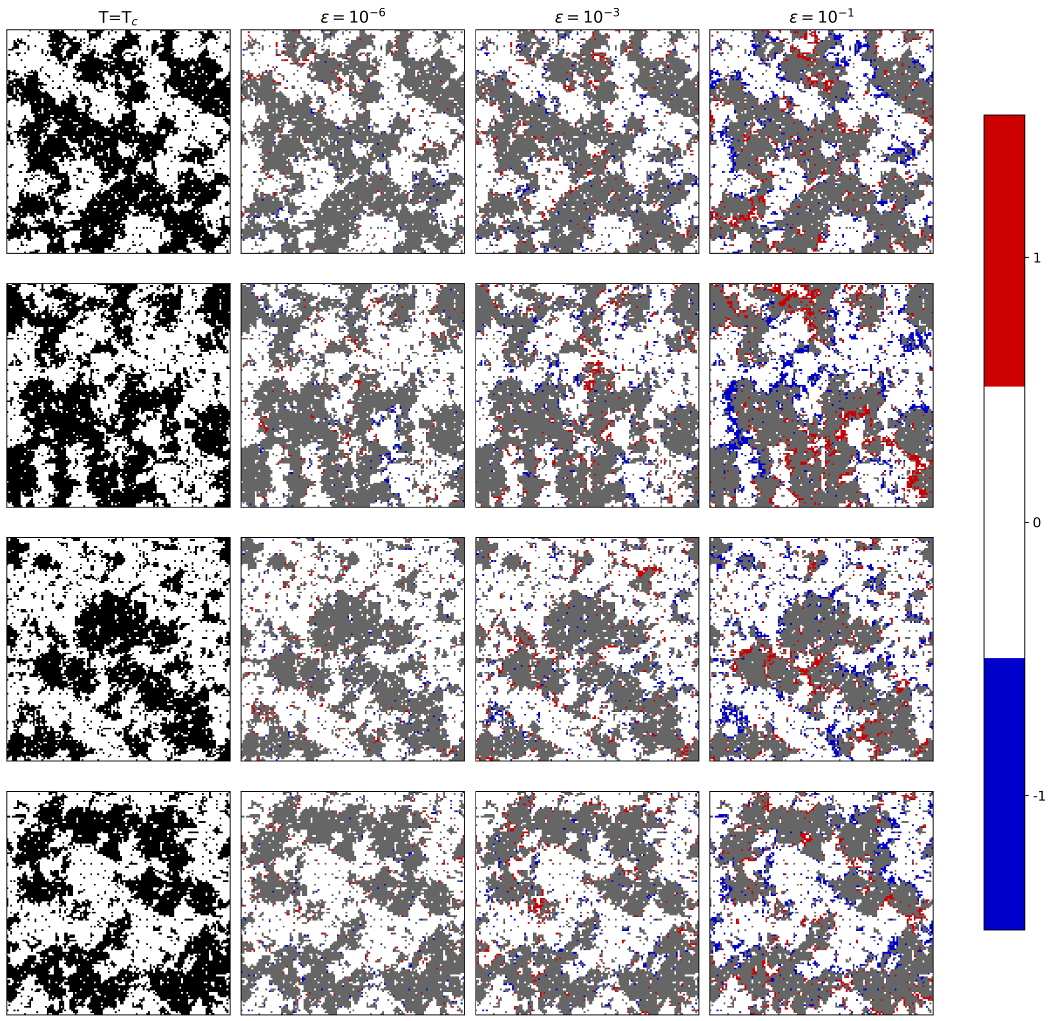
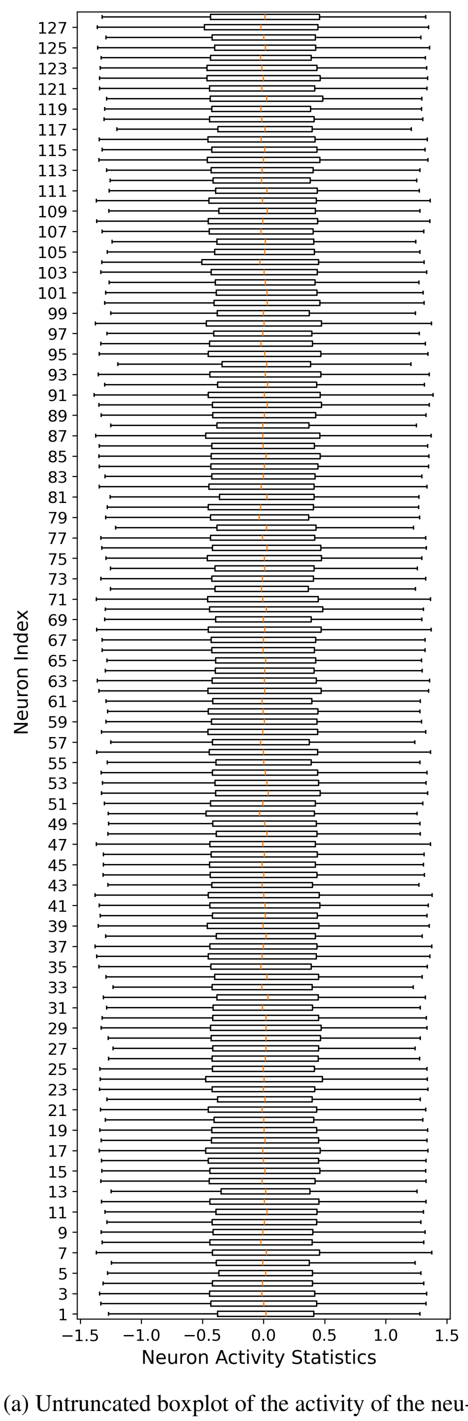
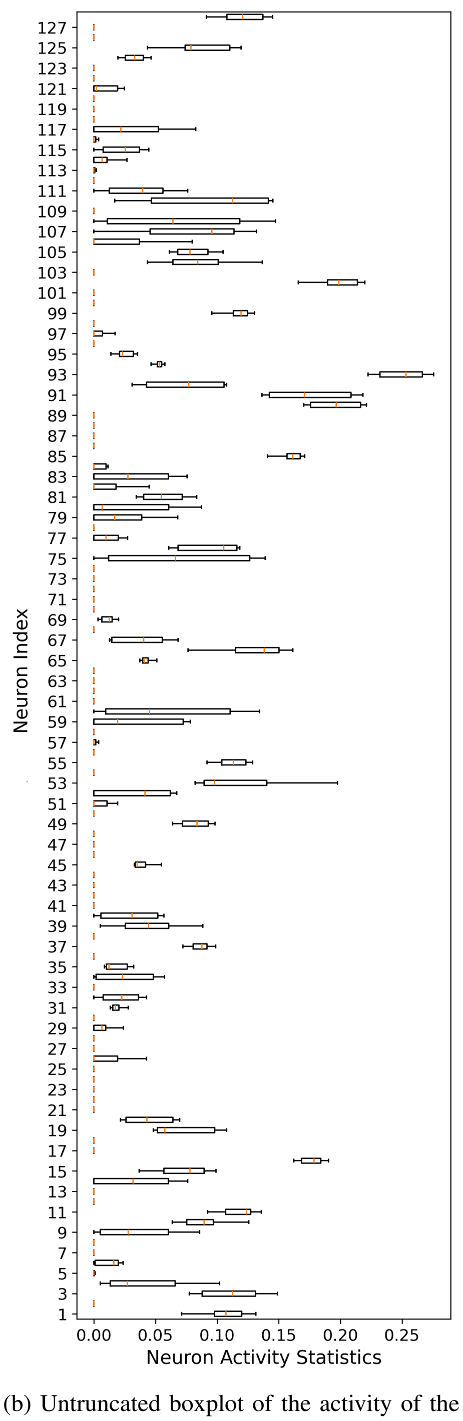


Figure 9: Generated images from BcGAN with different starting noise for various values of ϵ added at $\mathrm{T} = \mathrm{T}_C$. Each row represents a different noise input for the generator.

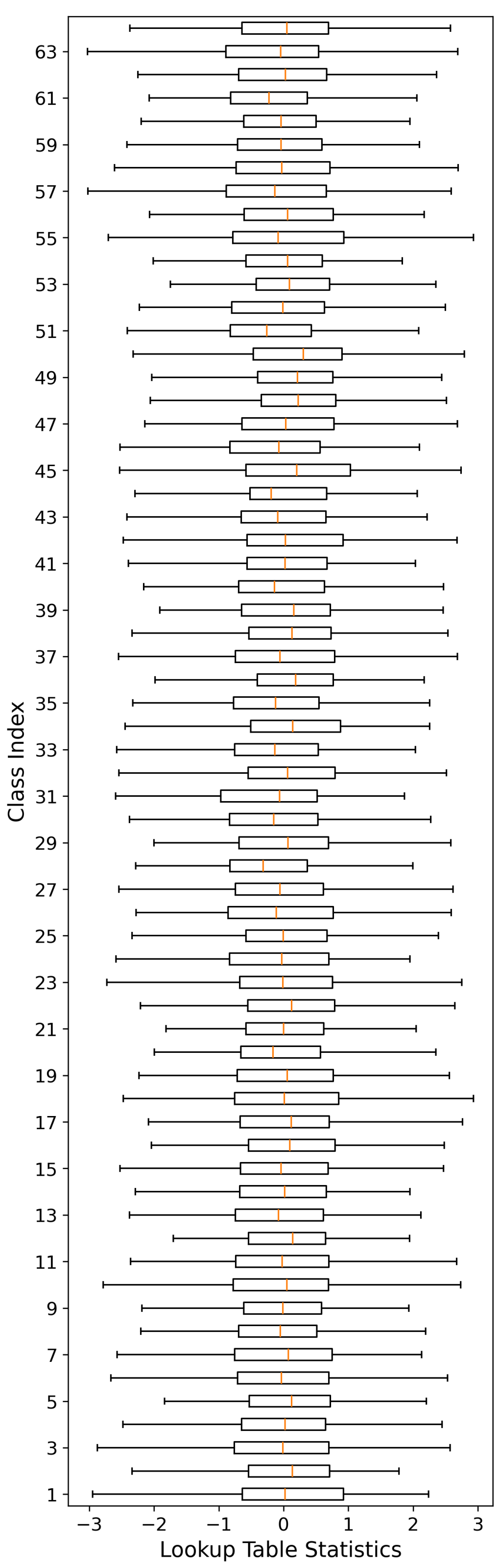


(a) Untruncated boxplot of the activity of the neurons in BcGAN embedding space

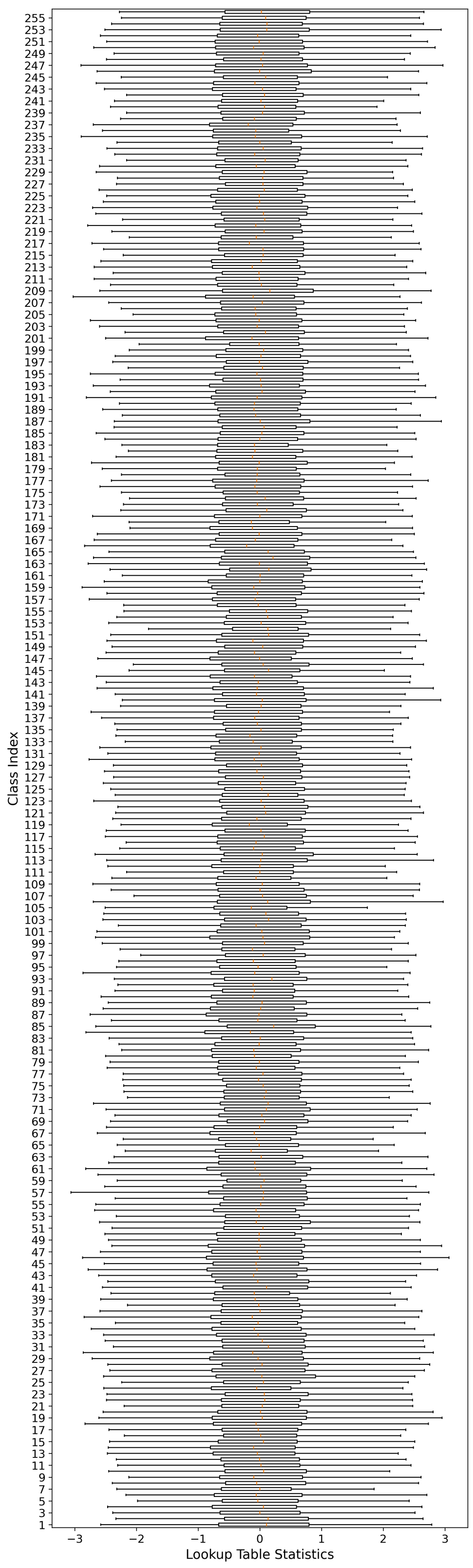


(b) Untruncated boxplot of the activity of the neurons in CcGAN embedding space

Figure 10: Comparison of the untruncated neurons activity for both continuous embedding strategies.



(a) Untruncated boxplot of the lookup table entry for CGAN with 64 classes.



(b) Untruncated boxplot of the lookup table entry for CGAN with 256 classes.

Figure 11: Comparison of the untruncated lookup table of the embedding from the generator of CGAN for two different classes counts. We can see that the statistics stay similar regardless of the number of classes.