Emergent Behavior in Neural Networks

Abid Khan

May 13, 2018

Abstract

The main objective of machine learning is to extract important features from data. Similarly, the main goal of studying emergent phenomenon is to take a high-dimensional system and explain it on a higher level with fewer dimensions. In this report, we show that neural networks exhibit emergent behavior and can be explained with the tools of condensed matter physics. We also show how neural networks can be used to extract emergent phenomenon in physical systems.

1 Introduction

Machine learning has been proving itself to be a very powerful tool in every field that has applied it. The prime example is its use is in image recognition, where we start with an $N \times N \times 3$ array of pixels, and end with a label of what that picture is. This is performed by a convolution neural network (CNN), where each layer in the network extracts features in the image that are useful for classification (see figure 1).

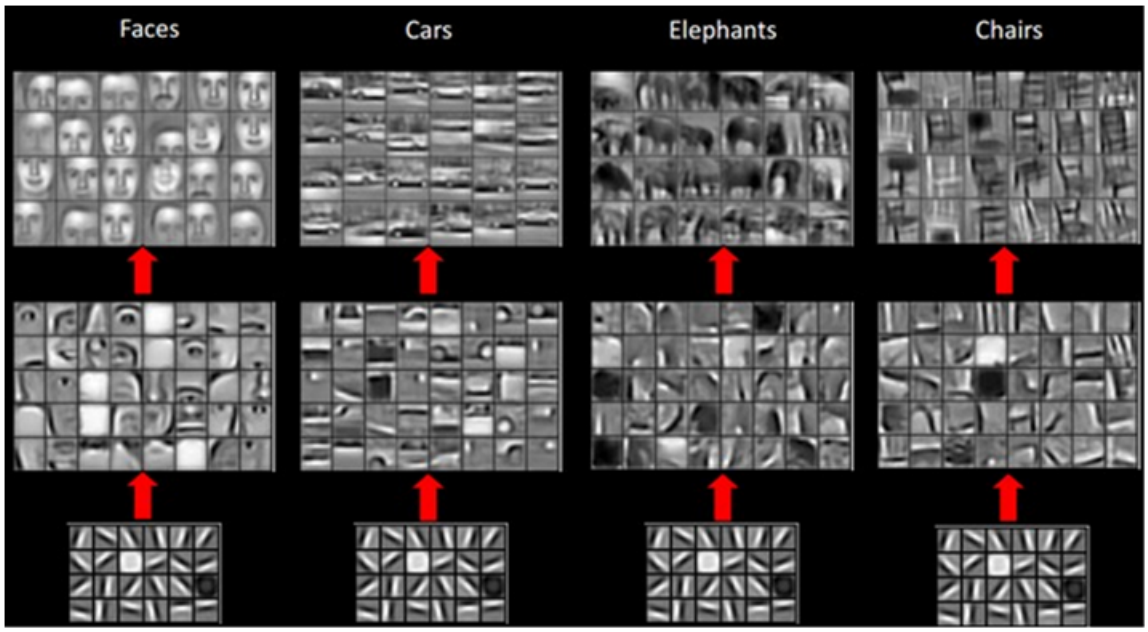

Figure 1: In a convolution neural network, an image is processed through layers. Each layer extracts key features that classify the image (face, car, elephant, chair, etc.). Image taken from [1].

In studying emerging phenomenon in matter, we take a low level system of high dimensionality, and observe features at a high-level of low dimensionality. It is in this way that neural networks work. Each neuron collects its own input and produces its own output which is meaningless by itself, but when observing the collection of neurons as a whole, they collectively produce a meaningful output. We begin by first describing a Neural Network. Then we analyze the emergent properties of neural networks. Finally, we study applications of neural networks on physical systems with emergent properties.

2 What is a Neural Network?

A neural network is described as a collection of nodes that are connected to each other as shown in figure 2. These nodes are usually constructed in layers with an input layer, a hidden layer, and an output layer. A deep neural network (DNN) is simply a neural network with many more hidden layers. The goal of neural networks is to feed the input layer with data so that it produces the correct results in the output layer.

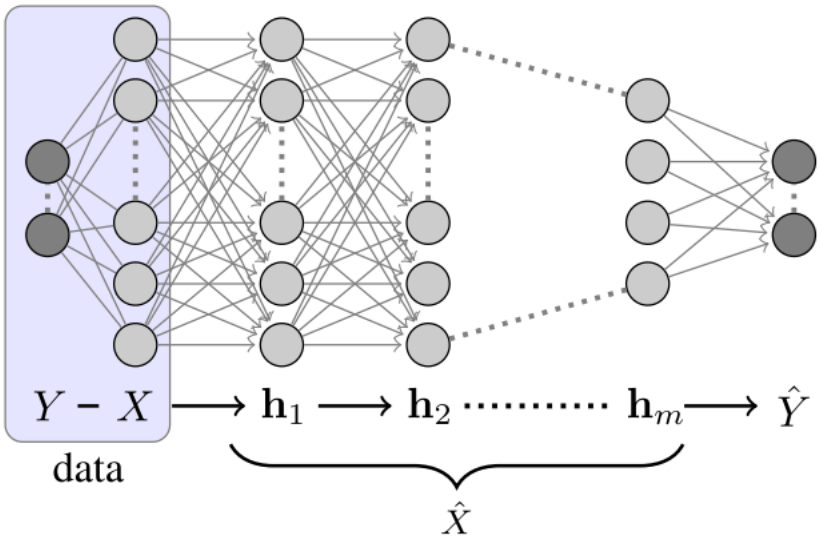

Figure 2: A neural network consists of nodes connected as above. Data is fed through the input, and parameters are tuned to correctly predict the output data. Image taken from [2].

Specifically, a node S_i is computed from the nodes in the previous layer by

$$S_i = f\left(\sum_j T_{ij} S_j + R_i\right),$$

where T_{ij} are weights between nodes S_i and S_j, and R_i is an additional parameter. $f(x)$ is called an activation function that is chosen, usually a sigmoid or rectifier function. This is diagrammatically shown in figure 3.

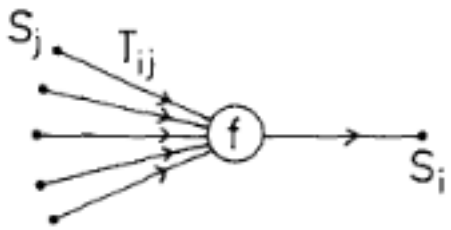

Figure 3: A node S_i is computed by considering the weights connected to it from the the nodes S_j in the previous layer via weights T_{ij}, and an activation function $f(x)$. Image taken from [3].

In supervised learning, the neural network is introduced with a set of inputs $\{\mathbf{x_i}\}$ with respective labels $\{\mathbf{y_i}\}$. The network is trained so that the computed output $\hat{\mathbf{y}}_\mathbf{i}$ matches the labeled data $\mathbf{y}$. This is done by tuning the weights so as to minimize a cost function $E = E(\mathbf{y}, \hat{\mathbf{y}})$.

Convolution neural networks (CNNs) are another type of neural networks usually used for multi-dimensional input data. They consist of convolution layers and pooling layers. In convolution layers, local nodes are convoluted with a matrix (or kernel). This new layer has the same dimensions as the previous layer. Pooling layers take a local region of nodes from one layer to one node in the next layer. As we will see below, CNNs can be more effective and efficient in certain problems.

3 Opening the Black Box of Neural Networks

The big unanswered question relating to neural networks is *how are deep neural networks so successful in extracting important features of data?* This questioned is tackled in [4], where it shown that there is an exact mapping between variational renormalization group (RG) and deep learning.

A restricted Boltzmann machine (RBM) is a stochastic neural network that learns probability distributions over a given input. It consists of input (visible) nodes v_i and hidden nodes h_j which are connected via weights w_{ij} (see figure 4). Additionally, visible nodes are associated with bias weights a_i, and hidden nodes are associated with bias weights b_j. Note that the number of hidden nodes are typically significantly smaller than the number of visible nodes.

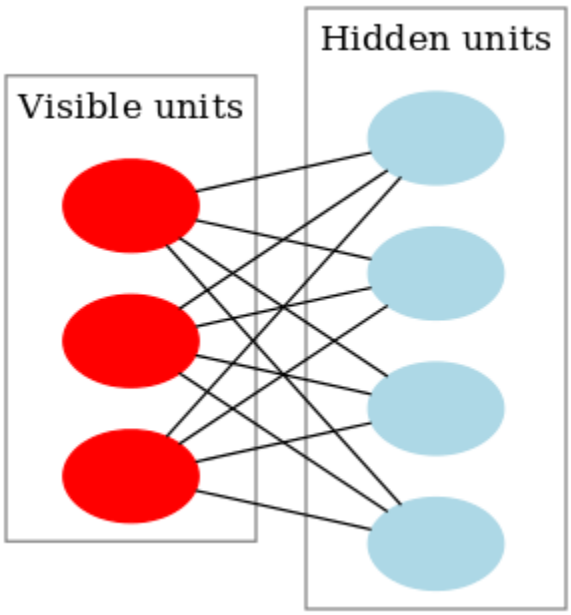

Figure 4: A restricted Boltzmann machine consists of two layers: a visible and a hidden layer. Image taken from Wikipedia.

Associated with an RBM is the energy function

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{ij} v_i w_{ij} h_j.$$

RBMs are trained by maximizing

$$\prod_{v \in V} P(v),$$

where v is an example in the training set V, and $P(v)$ is given by

$$P(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)}.$$

Here, Z is the normalizing constant (partition function).

From this description alone, we can see the resemblance of coarse-graining. In detail, [4] shows that the hidden nodes h_i are the same coarse-grained variables that one would obtain by variational RG. When RBMs are stacked, where the hidden nodes in one layer become the visible nodes in another, a deep neural network is formed. This DNN is then an iterative coarse-graining scheme, where each new high-level layer of the neural network learns increasingly abstract higher level features from the data.

While the ideas from [4] are specific to Deep RBMs, the theory gave inspiration to a more general and abstract theory of how deep neural networks are so successful: the information bottleneck (IB) principle. This theory described in [2] illustrates how information is passed from one layer to another. The central idea stems from the data processing inequality(DPI), which states that information can never be gained when transmitted through a noisy channel. To understand this better, mutual information is used. The mutual information $I(X;Y)$ between two random variables X, Y essentially defines the amount of information obtained about one random variable through the other. The quantity is given by

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right).$$

In the context of neural networks, let X_i represent the random variable at the i-th layer. then the DPI states that

$$I(X_i; X_{i+1}) \geq I(X_i; X_{i+2}).$$

Hence information lost in one layer cannot be recovered in higher layers. Hence a neural network is forced to compress only relevant information as it goes from one layer to the next, and, as argued in [2], compression is necessary for generalization.

The most fascinating consequence of this analysis is the presence of phase transitions while learning. In the training process, the DNN learns in such a way as to maximize the transfer of information from one layer to the next. [2] shows that at critical points, there are bifurcations to simpler representations along an information curve.

4 Using Neural Networks to Describe Emergent Phenomenon

4.1 Autoencoders and Phase Transitions

Autoencoders are neural networks that produce an output nearly identical to the input. They are composed of an input layer, a set of encoder layers, a latent variable layer, a set of

decoder layers, and an output layer (see figure 5). The object of an autoencoder is for the latent variable layer to be able to contain all the information needed to reproduce the input data. In order for autoencoders to be useful, the latent layer should have significantly fewer nodes than the input layer.

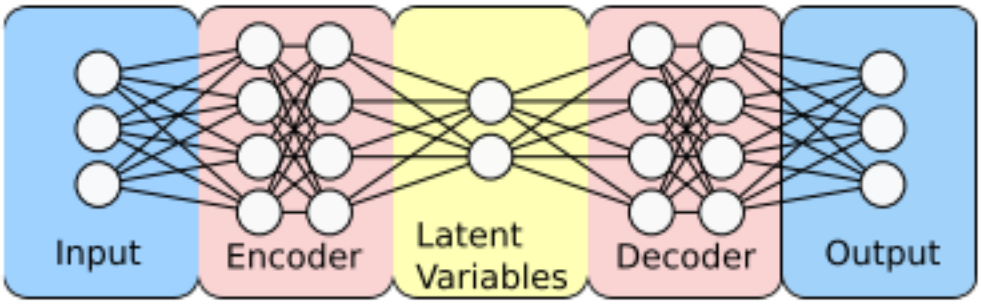

Figure 5: Autoencoder neural network architecture. The encoder takes the input data and translate it into latent variables. The decoder then takes the latent variables and translates it into the output which is an approximation of the input. Figure taken from [5]

Surprisingly, these autoencoders "learn" condensed matter systems the same way humans do: by finding order parameters. This result is shown in [5], where autoencoders are introduced to the two-dimensional Ising model. The autoencoder is trained to reproduce configurations of spins that are fed into it. These configurations are at temperatures above and below the critical temperature, i.e. configurations in the paramagnetic and ferromagnetic phase. The input and output layers are then $N \times N$ arrays of binary spins. The latent layer was set to have just one node, i.e. just one latent variable. After training, the latent variable is found to have a strong linear correlation with the magnetization of the configuration. From this, it is apparent that the autoencoder has found that the best way to describe a configuration is by its magnetization. Just as surprising, the study also found that magnetization is the *only* parameter needed to best describe the configuration. To show this, the authors ran the same training, but with the latent layer having two nodes and thus two latent variables. It was found that adding the second latent variable did not change the results, and contained very little information since it stayed very close to zero.

In addition to finding order parameters, the autoencoder also has the potential in classifying phases and phase transitions. Figure 6 shows how this is done. When plotting the magnetization versus latent parameter, clusters can be seen which can be attributed to a different phases. A clearer indication is in Figure 6b, where a histogram of the data is plotted according to the latent parameter. The sharp peak in the middle indicates the unordered phase, while the side peaks indicate the ordered phase. Finally in Figure 6c, we see a sharp decline in the Reconstruction loss as well as magnetization. This indicates the critical temperature which is near to the exact critical temperature of $T_c = 2.269$.

4.2 Learning Topological Invariants

Topological phases are challenging to learn for three main reasons. (i) phases are characterized by topological properties like topological invariants which are intrinsically non-local. (ii)

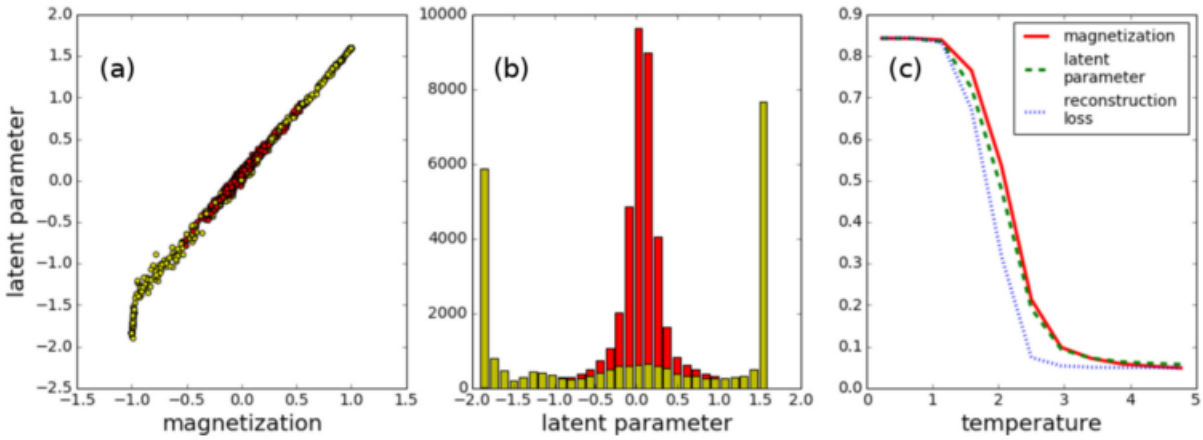

Figure 6: "Ferromagnetic Ising model. (a) The correlation between the latent parameter and magnetization is shown for each spin sample. Red dots indicate points in the unordered phase, while yellow dots correspond to the ordered phase. (b) The histogram counts occurrences of latent parameters. (c) One can see the average values at fixed temperature of the absolute value of magnetization, the absolute value of the latent parameter, and the cross-entropy reconstruction loss. The reconstruction loss is mapped on the $T = 0$ and 5 values of the magnetization and the latent parameter is re-scaled to the magnetization at T $= 0$." Figure and caption taken from [5]

Topological invariants are non-linear with respect to the field configuration. (iii) Topological invariants are intensive instead of extensive compared to conventional order parameters.

For these reasons, neural networks can learn topological invariants successfully for topological band insulators because (i) the input data are completely non-local, (ii) they are not restricted to any specific model, and (iii) neural networks have great generalization power after training.

This has been shown in [6], where they work with one-dimensional topological two-band insulators of AIII symmetry class. Here, the general form of the Hamiltonian is

$$H(k) = h_x(k)\sigma_x + h_y(k)\sigma_y$$

The topological invariant is the winding number, given by

$$w = \frac{-i}{2\pi}\int_0^{2\pi} U^*(k)\partial_k U(k)\, dk,$$

where

$$U(k) = \frac{h_x(k) + ih_y(k)}{\sqrt{h_x^2(k) + h_y^2(k)}}.$$

Hence the winding number is an integer that counts how many times $U(k)$ winds around the origin when k goes from 0 to 2π.

A convolution neural network (CNN) is created to take in normalized Hamiltonians as input and output the winding number. The Hamiltonian is discretized, so that $k = 2\pi n/L$,

where $n = 0, \ldots, L$. The input data is then a $2 \times (L+1)$ array of the form

$$\begin{pmatrix} h_x(0) & h_x(2\pi/L) & h_x(4\pi/L) & \ldots & h_x(2\pi) \\ h_y(0) & h_y(2\pi/L) & h_y(4\pi/L) & \ldots & h_y(2\pi) \end{pmatrix}$$

The authors begin training with the Su-Schrieffer-Heeger (SSH) model, where the Hamiltonian is

$$H_{\text{SSH}}(k) = (t + t' \cos k)\sigma_x + (t' \sin k)\sigma_y.$$

This model has two topologically distinct gapped phases with

$$w = \begin{cases} 0, & t > t' \\ 1, & t < t' \end{cases}$$

When trained using this model, the CNN achieves 100% accuracy, but when tested against more general Hamiltonians with $w > 1$, it drops to 50%, which is essentially just guessing. Hence the trained CNN does not have generalization power. This is because the CNN took advantage of the additional inversion symmetry in the SSH model: $H_{\text{SSH}}(k) = \sigma_x H_{\text{SSH}}(-k)\sigma_x$. The CNN can then just read out the winding number directly from $\mathbf{h}(\pi)$, where

$$w = \begin{cases} 0, & \mathbf{h}(\pi) = (1, 0) \\ 1, & \mathbf{h}(\pi) = (-1, 0) \end{cases}$$

The moral here is that restricting the training data to a certain model causes the neural network to exploit the features of the model instead of the universal ones.

Instead of training the CNN with just SSH Hamiltonians, the most general Hamiltonians $H(k) = h_x(k)\sigma_x + h_y(k)\sigma_y$ were trained, where h_x, h_y are of the form

$$h_i(k) = \sum_{n=0}^{c} \left(a_{i,n} \cos nk + b_{i,k} \sin nk\right).$$

c is the cutoff that determines the highest possible winding number of H The CNN is trained with these generalized Hamiltonians all with winding numbers $|w| \leq 2$, and is then tested with Hamiltonians with winding numbers $|w| \leq 4$, producing nearly 100% accuracy. Hence this CNN has learned the universal features and has generalization power. A further study by [6] shows that the CNN actually does learn to perform the discretized integral for the winding number.

The CNN was trained with 310 tunable parameters. A fully connected neural network with one hidden layer was also trained to perform the same task using 4061 tunable parameters. This network had about 90% accuracy for Hamiltonians with $|w| \leq 2$, but about 10% accuracy for Hamiltonians with $2 < |w| \leq 4$. This fully connected network does not learn the universal features and fails to have generalization power despite the order of magnitude increase of tunable parameters. This leads to a powerful conclusion about how to set up a proper neural network architecture: the neural network should compliment the symmetry of system. In this case, a fully connected neural network has to rediscover the translational symmetry of the system, while the CNN respects translational symmetry explicitly. Hence, a CNN was the optimal choice.

5 Conclusion

In this report we have discussed the relationship between neural networks and emergent phenomenon. Deep neural networks themselves exhibit emergent behavior as discussed by [4] and [2]. They are also successful in extracting important features in condensed matter systems, such as order parameters [5], phase transitions [5, 7], and topological invariants [6]. In just the recent years, there has been an explosion of publications on neural networks and its application in condensed matter systems. There is no reason to believe that this exciting field will stop growing in the future.

References

[1] Charles Siegel, Jeff Daily, and Abhinav Vishnu. Adaptive neuron apoptosis for accelerating deep learning on large scale systems. *CoRR*, abs/1610.00790, 2016.

[2] Naftali Tishby and Noga Zaslavsky. Deep Learning and the Information Bottleneck Principle. mar 2015.

[3] Carsten Peterson. Track finding with neural networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 279(3):537 – 545, 1989.

[4] Pankaj Mehta and David J. Schwab. An exact mapping between the Variational Renormalization Group and Deep Learning. 2014.

[5] Sebastian J Wetzel. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Physical Review E*, 96(2), 2017.

[6] Pengfei Zhang, Huitao Shen, and Hui Zhai. Machine Learning Topological Invariants with Neural Networks. *Physical Review Letters*, 120(6), aug 2018.

[7] Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. pages 1–18, 2016.