Expectation propagation: a probabilistic view of Deep Feed Forward Networks

Mirco Milletarí
Bambu, Singapore.
milletari@gmail.com

Thiparat Chotibut
Singapore University of Technology and Design (SUTD), Singapore.
thiparatc@gmail.com

Paolo E. Trevisanutto
Graphene Research Centre and CA2DM, National University of Singapore, Singapore.
pe.trevisanutto@nus.edu.sg

Abstract

We present a statistical mechanics model of deep feed forward neural networks (FFN). Our energy-based approach naturally explains several known results and heuristics, providing a solid theoretical framework and new instruments for a systematic development of FFN. We infer that FFN can be understood as performing three basic steps: encoding, representation validation and propagation. We obtain a set of natural activations – such as sigmoid, tanh and ReLu – together with a state-of-the-art one, recently obtained by Ramachandran et al. [1] using an extensive search algorithm. We term this activation ESP (Expected Signal Propagation), explain its probabilistic meaning, and study the eigenvalue spectrum of the associated Hessian on classification tasks. We find that ESP allows for faster training and more consistent performances over a wide range of network architectures.

1 Introduction

Advances in modern computing hardware and availability of massive datasets have empowered multilayer artificial neural networks, or deep learning, with unprecedented capabilities for image and speech recognition tasks. Despite these empirical success, theoretical understanding of why and when multilayer neural networks perform well remains limited. Only recently, theoretical efforts in this direction have been intensively reported. For example, recent works shed light on how FFN attains its expressive power [5,8], what contributes to its generalizability [9,10], and how myriad parameters in the network affect the geometry of the loss function [11,12]. Taken together, these theoretical results have paved the way for a systematic design of robust and explainable FFNs.

Although FFN is one of the most studied and widely used neural network architectures, state-of-the-art models still largely inherit their building blocks from the decades-old multilayer perceptron (MLP). By augmenting MLP with modern optimization techniques such as dropout [15], non-linear activation functions that facilitate optimization protocols, and specialized complex network architectures [16], to name a few, the FFN can be efficiently trained on large-scale datasets such as ImageNet or CIFAR to achieve low training and generalization errors. While these engineering feats improve the performance of FFN, a clear design principle behind these developments is still lacking. This leads to an unsystematic growth of the FFN complexity, which obscures its original exposition as a simple physical system that can perform non-linear classification and regression tasks.

Preprint. Work in progress.
To assist future systematic studies and construction of FFNs, we propose a theoretical framework based on the toolset of statistical mechanics. It allows for the definition of an energy based model in which a hidden unit is regarded as a communication channel, first encoding and then transmitting the result of its computation through a gate with a specific transmission probability; as opposed to a hidden unit in multilayer perceptrons that transmits its sigmoidal firing probability, or in modern FFNs where a standard ad hoc activation function such as ReLu and its variants are used. The transmission probability is obtained via the maximum entropy principle [17, 20] under the biologically inspired constraint that a neuron responds by firing (transmit its signals) or not with a certain probability. The maximum entropy principle provides the least biased probabilistic interpretation of a hidden neuron, and recasts a FFN as an energy-based model. By interpreting a hidden unit as a communication channel, its activation function takes the form of the expected (mean) signal transmission; remarkably, this activation function, which we term Expected Signal Propagation (ESP), agrees with the state-of-the-art activation (Swish), obtained in Ref. [1] through an extensive search algorithm trained with reinforcement learning and shown to best perform on the CIFAR and ImageNet datasets among all candidate activations. Although some of these activations may perform better for the specific datasets they have been built for, they generally fail to generalize. Finally, the standard ReLu activation arises as a limiting case of ESP, corresponding to the noiseless limit of a communication channel. To the best of our knowledge, this provides the first formal derivation of the ReLu activation, typically introduced heuristically to facilitate optimization protocols. Despite restricting our analysis to pure FFNs, most of our conclusions carry on to Convolutional and Recurrent networks, which will be addressed in future works.

The paper is organized as follows: In section 2, we provide a plausible argument based on dimensional analysis to explain why a hidden unit should be regarded as a communication channel transmitting the result of its computation. We then discuss the formulation of a FFN as an energy-based model using the maximum entropy principle over the binary states of a hidden unit, and discuss the advantage of this formulation. In analogy with Refs. [3, 4], we show that each hidden unit acts first as an encoder and then as a filter determining the quality of its input by comparing it with a local bias potential.

In section 3, we explore the geometry of loss surfaces associated with training FFNs with ESP hidden units on simple classification tasks. The index $\alpha$, representing the fraction of negative eigenvalues of the Hessian (a measure of the number of descent directions), as well as the index $\gamma$, representing the fraction of zero eigenvalues (a measure of the number of flat directions), are investigated. We find that FFNs trained with ReLu exhibit $\gamma \neq 0$ while those trained with ESP often show $\gamma \approx 0$, resulting in faster and more flexible training. These two indices seem to determine not only the speed of learning but also whether learning will be successful, although further studies are required to understand the detailed dynamics of gradient descent. Lastly, we conclude by commenting on how our model relates (and incorporates) other recently proposed frameworks such as the Renormalization Group [2] and the information Bottleneck [3, 4] in section 4. Codes and other numerical results, including training on the MNIST dataset, are provided in the supplemental materials (SM).

2 Motivation and Model

In this section, we introduce a statistical mechanics model of an FFN, and highlight its connection to an energy-based model as well as its information theoretic interpretation. To motivate the model, we begin by discussing a (physical) dimensional inconsistency in the standard formulation of forward propagation in FFNs, and how this can be reconciled within the proposed framework.

Consider a general FFN and denote $X^{\mu}$ an input vector, where $i \in [1, n]$ denotes a feature component, and $\mu \in [1, m]$ denotes an example. In analogy with the communication channel scheme in information theory [19, 20], the input vector constitutes the information source entering the processing units (neurons) of the network, while the units constitute the encoders. Quite generally, the encoders can either build a lower (compression) or higher dimensional (redundant) representation of the input data by means of a properly defined transition function. In a FFN, the former corresponds to a compression layer (fewer units) while the latter to an expansion layer (more units). If the encoded information constitutes an informative representation of the output signal (given the input), it is passed over to the next layer for further processing until the output layer is reached. In the biological neuron, this last step is accomplished by the synaptic button, that releases information whether or not the input signal exceeds a local bias potential $b_i$. Both in the brain and in electronic devices, the input information is often conveyed in the form of an electric signal, with the electron charge being the
basic unit of information, the signal has dimension of \textit{Coulomb} in SI. For an image, the information is the brightness level of each pixel, physically proportional to the current passing through it; on a computer, this is encoded in bits\footnote{The input features have often different dimensions. For example, the Boston housing dataset \cite{21} contains 14 features ranging from crime rate to property tax.}. Therefore, it is convenient to measure a general information content in units of \textit{bits} \cite{20}. Clearly, a linear combination of the input signals, together with the bias, has to preserve dimensions:

\[ h_i^\mu = \sum_{j=1}^{n} w_{ij} X_j^\mu + b_i, \quad (1) \]

where \( i \in [1, n_1] \) indices the receiving units in the first layer. For noiseless systems, the input signal is transmitted i.f.f. the overall signal exceeds the bias \( b_i \). However, in the presence of noise, the signal can be transmitted with a certain probability even below the threshold. In so far, we just described the general functioning principle underlying FFNs. Let us first consider the sigmoid non linearity \( \sigma(\beta \mathbf{h}) \), where \( \beta \) has inverse dimension of \( \mathbf{h} \); being \( \sigma \) a \textit{distribution function} defined in \( [0, 1] \) it is intrinsically \textit{dimensionless}. The parameter \( \beta \) defines the spread of the distribution, corresponding to the noise level (the temperature in physics); typically, one sets \( \beta = 1 \) or rescales it inside the weights and bias \cite{17}, but here we keep it general for reasons that will be clear later. Defining \( z = \sigma(\beta \mathbf{h}) \) as the input of the next layer, we can immediately see the dimensional mismatch: a linear combination of \( z \) is dimensionless and when passed through the new non-linearity, \( \sigma(\beta z) \) necessarily becomes dimensionful. In the next section we show how this simple fact is at the root of the well known phenomena of vanishing gradients during back propagation. From a conceptual point of view, one is transmitting the expectation value of the transmission gate rather than the processed signal. This problem persists when using the tanh activation (a translation of the sigmoid), but it is absent when using \textit{ReLu}, that correctly transmits the information itself. In the next subsection we formalise this intuitive picture using the tools of statistical mechanics.

\subsection{Statistical mechanics of Feed Forward networks}

A prototypical statistical mechanics formulation of Neural Networks is the \textit{inverse} Ising problem \cite{17}, where one is interested in inferring the values of the couplings and external fields \( \{ J_{ij}, h_i \} \) of the Ising model, given a set of pairwise connected binary units \( s_i \); in computer science, this problem is solved by the Boltzmann machine. While the Boltzmann machine is an energy-based model, standard FFN is not commonly regarded as such. Here, we propose an energy-based formulation of FFNs to bridge on these two formulations, solely based on the maximum entropy principle \cite{17, 20} to obtain the least biased probabilistic interpretation of hidden neurons. The core task at hand is to determine the state of the channel (open/close). The absence of lateral connections in FFNs means that each unit is only influenced by its receptive signal, and the intralayer correlations are taken – a priori-- to be zero. In other words, each unit within the same layer behaves independently of the others (i.e. no contextual information)\footnote{The contextual information is reintroduced in Convolutional and Recurrent networks – although heuristically--and it accounts for the superior performance of these architectures.}

Given a statistical ensemble of hidden binary units, the most likely (and the least biased) distribution can be obtained by maximising its entropy, subject to constraints imposed by the conserved quantities; in the grand canonical Gibbs ensemble these are the \textit{global} average energy and particle number \cite{18}, while here they correspond to the first and second moments of the data \cite{17, 19}. However, in the absence of lateral connections, the entropy functional of the hidden units \( s_i \) reads

\[ \mathcal{F} = - \sum_s P(s) \log P(s) + \eta \left( \sum_s P(s) - 1 \right) + \sum_i \lambda_i \left( m_i - \sum_s s_i P(s) \right), \quad (2) \]

where \( \lambda_i \) are Lagrange multipliers chosen to reproduce the first moment of the data, while \( \eta \) enforces normalization. Functionally varying with respect to the probability \( P(s) \) and solving for the Lagrange multipliers, one easily obtains \cite{18}:

\[ P(s|\mathbf{h}) = \frac{1}{Z} e^{\sum_i \beta_i s_i h_i}, \quad Z[\mathbf{h}] = \prod_i \sum_{s_i = \{0,1\}} e^{\beta_i s_i h_i} = \prod_i \left( 1 + e^{\beta_i h_i} \right). \quad (3) \]
The parameters $\beta_i$ encode the noise, or statistical uncertainty, concerning our knowledge of $s_i$ ($\lambda_i = \beta_i h_i$). In a physical system, $\beta_i$ is the inverse temperature in units of Boltzmann’s constant and in equilibrium it is the same for each $s_i$; however, here the network is only in a local equilibrium as the units are not allowed to exchange “energy” (information) among themselves due to the lack of pairwise interactions (the lateral connections). We have also introduced the notation $P(s|h)$ to denote the conditional probability of $s$ given the “external field” $h$ – and the partition function $Z$.

We discuss now the central results of this paper, based on the identification of $h_i$, while the discussion here and in Ref. [2] suggests that feedforward propagation in the compression channels is analogous to the coarse graining step of RG, rescaling of the coarse grained variable is observed (empirical) sequence, or assume the form of the distribution from which $X$ is sampled from. The former corresponds to an empirical dataset, whereas the latter is preferred for an analytical inspection and to determine general statistical properties about the system. In this work we consider the former, while the latter will be discussed in a forthcoming manuscript. Consider then the empirical estimator $P(X) = \frac{1}{m} \sum_{\mu=1}^{m} \delta(X, X^{\mu})$, where the Kronecker-delta can be understood as a binning function. Using the empirical estimator together with the explicit form of the transition function in the discrete version of Eq. (4) we find

$$P(h) = \sum_X \delta(h - w^T X - b) \left( \frac{1}{m} \sum_{\mu} \delta(X, X^{\mu}) \right) = \frac{1}{m} \sum_{\mu} \delta(h - w^T X^{\mu} - b),$$

(5)

where the Dirac delta represents the transition function. Eq. (5) is akin to the real space block-spin renormalization developed by Kadanoff, reformulated in the more general language of probability theory [18, 23, 24]. The relation between Deep Learning and the renormalization group (RG) was previously observed in the context of restricted Boltzmann machine [2]. We note in passing that, while the discussion here and in Ref. [2] suggests that feedforward propagation in the compression layer is analogous to the coarse graining step of RG, rescaling of the coarse grained variable is missing. Thus, the analogy is incomplete.

Finally, given the distribution of the coarse grained inputs and the conditional probability of the channels $P(s|h)$, one needs to evaluate the channel transmission probability

$$P(s) = \int dh \ P(s|h) \ P(h) = \sum_{\mu} e^{-\sum_i \beta_i s_i \mathcal{H}_i[s,X^\mu]} Z / m, \quad Z = \frac{1}{m} \sum_{\mu=1}^{m} \prod_{i=1}^{n_i} \left[ 1 + e^{\beta_i h_i^{\mu}} \right],$$

(6)

where $h_i^{\mu} = \sum_j w_{ij} X_j^{\mu} + b_i$ and, in analogy with the Ising model, we have identified the simple coarse grained Hamiltonian

$$\mathcal{H}_i = - \sum_j s_i w_{ij} X_j^{\mu} - s_i b_i.$$ 

(7)

Eqs. (6) and (7) can be seen as the starting point of an energy based model, where it is the coarse grained probability $P(s)$ that propagates through the network (see e.g. [27]) as opposed to signals in a FFN. The expected value of the channel transmission, $\langle s_i \rangle$, can be easily obtained as

$$\langle s_i \rangle = \frac{1}{\beta_i} \frac{\partial}{\partial h_i} \log Z_i = \prod_{\mu} \frac{1}{1 + e^{-\beta_i h_i^{\mu}}} = \prod_{\mu} \sigma(\beta_i h_i^{\mu}),$$

(8)

3Technically, this corresponds to a generalized Gibbs ensemble, where conserved quantities are local rather than global, see e.g. [23].

4Here we refer to compression layers, but similar conclusions hold for expansion ones.

5Alternatively, one can interpret $h$ as an effective field representing the action of units $s_i$ on unit $s_j$, see e.g. Ref. [25] and [26] for connections to the belief propagation algorithm.
that is the logistic function (or Fermi-Dirac distribution in physics), evaluated for each empirical realization \( \mu \). We stress that this quantity is not the coarse grained input signal transmitted by the channel; it solely determines the expectation of channel transmissions given the coarse grained input signal \( h_i^l \). Note that \( \langle s_i \rangle \) would be the correct quantity for a single perceptron, or equivalently the output layer of an MLP performing a classification task. However, as discussed in the previous section, the hidden layers should not transmit \( \langle s_i \rangle \), as this leads to dimensional mismatch. To ensure dimensional consistency across hidden layers, the output signal of each hidden unit must have the same dimension as its input signal. Therefore, the correct quantity to consider is the expectation value of the coarse grained field \( h_i^l \); this can be easily obtained by summing over all gate states, for each unit and example, or by using the partition function of Eq. (6).

\[
a_i^l \equiv \langle h_i^l \rangle = \frac{\partial}{\partial \beta_i} \log Z_{i,\mu} \equiv h_i^l \sigma(\beta_i h_i^l),
\]

that agrees with the definition of the energy flux in statistical mechanics [29]. Note that contrary to Eq. (8), the noise parameters \( \beta \) cannot be rescaled away now. We call this activation ESP, “Expected Signal Propagation” to stress its meaning and physical analogy, even though here \( \beta \) does not have to be strictly positive. This function was recently obtained in Ref. [1] (there named Swish), through an extensive search algorithm trained with reinforcement learning. In their extensive search, the authors found that activations of the form \( a = xf(x) \) better performed on several benchmark datasets. Our model naturally explains this observation by identifying ESP with the expectation value of the coarse grained input transmitted by each unit. In the next section we will better analyse this activation function and its meaning within backward propagation. Consider now the limit of a noiseless system, i.e. \( \forall i, \beta_i \to \infty \) in the above equation:

\[
\lim_{\beta_i \to \infty} \langle h_i^l \rangle = h_i^l \theta(h_i^l) \equiv \max \{ h_i^l, 0 \} \equiv \text{ReLu},
\]

where \( \theta() \) is the Heaviside step function and in the last equality we have identified the ReLu activation. To the best of our knowledge, this is the first formal derivation of ReLu, usually obtained heuristically. ReLu should be understood as the noiseless limit of the mean transmitted information across the units, thus explaining the empirical observation that it often outperforms \( \sigma \) or tanh in multilayer networks. Both ESP and ReLu pass on the expected value of their computation; the latter does that with probability one only if the coarse grained input signal exceeds a threshold, while the former can pass a signal lower than the threshold with a finite probability. In Fig. 1 we plot the three activation functions for \( \beta = 1 \). Clearly, ESP encompasses the statistical uncertainty conveyed by each processing unit in estimating the pre-activation value \( h \). A positive value of the activation tells the units in the next layer that a certain combination of the previous inputs should be strengthened while a negative value means that it should be weakened, i.e. unlearned; the latter option is absent when using the ReLu function, a feature known to generally slow down learning [30]. In the opposite extreme, the noisy limit \( \beta \ll 1 \), the ESP becomes linear. Therefore, we can consider linear networks as a noisy limit of non linear ones. This process has to be repeated for each and every layer, feeding the post activation distribution \( P(a) \) of the previous layer as the new empirical distribution, see Eqs. (4) and (9). Therefore, we introduce a layer index for the weights, biases and effective fields: \( l \in [1, L] \).

We would like to conclude this section noticing that in spin glasses [31,32], the energy landscape presents an increasing number of degenerate local minima as the temperature is lowered, a feature shared with the Hopfield model of associative memory [33]. The effect of noise in FFNs was recently considered in Ref. [34], where an improved optimization algorithm based on the maximum entropy principle was proposed.

### 2.2 Back-propagation: a probabilistic point of view

At the heart of any FFN model is the back propagation algorithm [28,35]. Forward propagation returns a first “guess” on the value of the learning parameters, that are subsequently adjusted layer by layer by minimizing a properly defined loss function, i.e. the energy of the system. Consider the output layer “L”, then the gradient of the weights is

\[
\frac{\partial \mathcal{L}}{\partial w_L} = \frac{1}{m} \sum_{n=1}^{m} \left[ e^n_i g(h_i^n_L) a_{L-1} \right],
\]

where \( e^n_i \) is the residual error, depending on the difference between the network output \( \hat{y}^n \) and the ground truth vector \( y^n \). In the optimization phase we look for the stationary point \( \frac{\partial \mathcal{L}}{\partial \theta_L} = 0 \),
where $\theta^a = \{w_i, b_i, \beta_i\}$ is the set of learning parameters. For a linear network, this condition is strictly satisfied if $e^h = 0$. However, in a non-linear network we can also have $g(h^\mu) = 0$. In principle, there may be situations in which $e^h$ is far from zero but $g(h^\mu) \simeq 0$, in which case learning will not be effective. For a $\sigma$ activation, $g(h^\mu) = \sigma(\beta_i h^\mu_i)\sigma(-\beta_i h^\mu_i)$, commonly known in physics as a phase space factor; it appears e.g. in the collision integral of the Boltzmann equation [18], which describes the transition probability from an occupied to an empty state. Ref. [3, 4] show that there are two distinct phases of learning: empirical error minimization (the residual) and representation learning. Following the discussion of the previous section, we can identify the latter with the task of optimizing $g(h^\mu)$. When $h_i \gg 1/\beta_i$, i.e. when the signal greatly exceeds the noise, then $\sigma(\beta_i h_i) \equiv P(s_i = 1|x_i) \simeq 1$ and the back propagated signal is small, being proportional to $g(h) \approx 0$. We then have the paradoxical situation in which, although the lower dimensional representation of the information is considered to be relevant, learning is likely to be inefficient.

Consider now the same limiting behaviour for ESP. If $h_i \gg 1/\beta_i$ we now have $g(h^\mu_i) \approx 1$, i.e. the representation learning phase is completed and learning moves towards minimising the empirical error. In the opposite limit, the signal is much smaller than the noise and learning is impossible, as expected. Finally, in the noiseless case one obtains the ReLu solution $g(h^\mu_i) = (1, 0)$, for $h_i$ respectively greater or smaller than zero. This corresponds to a purely “excitatory” network, in which representation unlearning is not possible. In other words, the fact that ESP can be negative for $\sigma(h)/\beta < h < 0$ allows for greater flexibility and the possibility to avoid plateaus surrounding local minima and saddle-points. In Ref. [14] it was proposed that plateaus of zero (or small) curvatures in the loss are mostly responsible for slowing down or preventing convergence of gradient descent. The following section provides a numerical analysis to support this picture. Details of the the back-propagation algorithm with ESP for a general $L$ layer network can be found in the SM.

### 3 Numerical Analysis

A thorough performance analysis of ESP versus other popular choices of activations was carried out in Ref. [1], where it was shown that the former outperformed all other choices on image classification tasks using a convolutional structure. It was also noted that to take full advantage of ESP one should also reconsider the way convolution is performed; for the moment, we leave this interesting point open and rather focus on understanding the optimization dynamics in pure FFNs. We have considered both artificial and experimental datasets trained with ADAM gradient descent [37]. Given that both datasets lead to the same qualitative features, we focus our analysis on the former and discuss the latter, together with additional details in the SM. In Fig. (2) we show the loss functions for two binary classification tasks: a linear and a non-linear one, each trained with one and two hidden layers. For the linear task with a single, 10 units layer (adding more units does not improve performance), all three activations attain full train/test accuracy but ESP is the faster converging. For the non-linear task, ReLu quickly converges to a suboptimal plateau. To obtain a better understanding, we have evaluated two different indices: the fraction of negative eigenvalues of the Hessian $-\alpha$ (Fig. (3)) and the fraction of zero eigenvalues $-\gamma$. The former measures the ratio of descent to ascent directions on the energy landscape; when $\alpha$ is large, gradient descent can quickly escape a critical point – a saddle point in this case – due to the existence of multiple unstable directions. However, when a critical

---

6 Here $\bar{x}_i$ is the input of the unit, not necessarily the input data of the network.
point exhibits multiple near-zero eigenvalues, roughly captured by $\gamma$, the energy landscape in this neighbourhood consists of several near-flat (to second-order) directions; in this situation, gradient descent will slowly decrease the training loss. In Ref. [11] it was noted that energy plateaus are responsible for slowing down or preventing learning. This can be understood by considering the case in which learning completely fails: the loss does not decrease and all the eigenvalues of the Hessian are zero. In this case the analysis is not conclusive and higher order inspection is needed to reveal possible inflection points. In general, we find that for ReLU networks $\gamma \neq 0$, while this is typically not the case for both ESP and $\sigma$-networks. Taking the two layer case as a representative example, we show that ReLU networks are sensitive to fine tuning of the model: choosing a $10-2$ or an $8-5$ configuration over the $8-2$ considered here, greatly improves learning. In stark contrast, ESP networks exhibit consistent performance over a wider choice of architecture/learning parameters.

Figure 2: Loss functions for linear (left) and non-linear (right) binary classification. (Top) 10 units hidden layer (Bottom) two hidden layer of 8 and 2 units respectively. All cases have a sigmoid activation in the last layer and a learning rate of 0.01.

Figure 3: $\alpha$ index v.s. energy for the single layer network with linear (left) and non-linear (right) decision boundary. Curves are evaluated with the same number of epochs: 1000 (left) and 2300 (right).
Although the performance impact might be fairly small for small networks, it certainly plays an important role for larger datasets, as discussed in Ref. [1]. In Fig. 3 we show the $\alpha$ index, usually smaller for the ReLu network, and a finite value of $\gamma$ that slows down learning. We find $\gamma \approx 0.45 - 0.6$ for ReLu and $\gamma = 0$ for ESP and $\sigma$, both for the linear and non-linear task. We have also evaluated the fraction of residuals $e^\mu \approx (\hat{y}^\mu - y^\mu)/m$ closer to zero and found, surprisingly, that ESP greatly outperforms the other activations in minimizing the empirical error, see Fig. 4.

In addition, we find that the eigenvalue distribution obtained with ReLu shrinks with increasing training epochs, giving rise to the singular distribution reported in Ref. [13, 38]. In contrast, ESP trained networks show a significantly wider spread in the eigenvalue distribution at the end of training, see SM for details and discussions, together with results for the MNIST dataset [?].

4 Conclusions and perspectives

In this work we have introduced an energy-based model that allows systematic construction and analysis of FFNs. It provides a coherent interpretation of the computational process of each and every unit (neuron). Furthermore, it presents a method that overcomes the dimensional mismatch that arises from using heuristic activations. Enforcing dimensional consistency naturally leads to a class of activations with the prime focus of propagating the expectation value of processed signals, hence Expected Signal Propagation. Our results provide a theoretical justification of the activation found in Ref. [1]. In addition, we demonstrate the superiority of this ESP activation through numerical experiments that reveal the geometry of its loss manifold. ReLu networks, unlike $\sigma$ and tanh, do not suffer from dimensional mismatch, yet their restricted phase space results in a finite fraction of null directions of gradient descent that can slow down or, in some cases, completely prevent learning. To overcome this problem, one needs a detailed fine tuning of the network’s topology that likely increases the complexity of the learning task. In stark contrast, ESP trained networks are less prone to fine tuning and outperform other activations on minimizing the empirical error. We hope this statistical mechanics view can facilitate future studies of FFNs, e.g. to explore the scaling relation between the ESP Hessian’s eigenvalue distributions and the parameters that describe its loss landscape, analogous to those studied in Ref. [13] for standard activations. As previously discussed, the Renormalization Group (RG) approach could hint at design principles of FFNs, as it delineates how complex behaviour emerges from the elementary constituents of a system. Although this framework has been considered in Ref. [2, 39], the complete mathematical formalism to correctly use it in FFNs is still missing. We believe this could answer some of the most pressing questions in the construction of FFNs, namely: given a dataset showing specific strength and extent of input/output correlations, what is the optimal network topology? Finally, extensions of the methods discussed here to Convolutional and Recurrent
architectures could lead to substantial performance improvement and, potentially, faster training algorithms.

Acknowledgements

M.M. would like to thank Ned Phillips, Bill Phillips, Sarah Chan and Roberto Raimondi for support and discussions. A special thanks to Fábio Hipólito and Aki Ranin for carefully reading and commenting the manuscript. T.C. would like thank Shaowei Lin for useful discussions and for financial support from the startup research grant SRES15111, and the SUTD-ZJU collaboration research grant ZJURP1600103. P.E.T would like to thank M. D. Costa for help with HPC. Some of the calculations were carried out at the HPC facilities of the NUS Centre for Advanced 2D materials. Finally, M.M. And T.C. would like to thank the AI Saturday meetup initiative organised by Nurture.ai, that sparkled some of the ideas presented in this work.

References

[1] Prajit Ramachandran, Barret Zoph and Quoc V. Le. Searching for Activation Functions. arXiv:1710.05941, 2017.

[2] Pankaj Mehta and David. J. Schwab. An exact mapping between the Variational Renormalization Group and Deep Learning. arXiv:1410.3837, 2014.

[3] Navtali Tishby and Noga Zaslavsky. Deep Learning and the Information Bottleneck Principle. Invited paper to ITW 2015; 2015 IEEE Information Theory Workshop, arXiv:1503.02406v1, 2015.

[4] Ravid Schwartz-Ziv abd Navtali Tishby. Opening the black box of Deep Neural Networks via Information. arXiv:1703.00810, 2017.

[5] Tomaso Pogio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review International Journal of Automation and Computing 14, pp. 503-519,2017.

[6] Henry W. Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? Journal of Statistical Physics 168, pp. 1223-1247, 2017

[7] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. Proceedings of the 34th International Conference on Machine Learning pp. 2847-2854, 2017.

[8] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. Advances in Neural Information Processing Systems 29 pp. 3360–3368, 2016.

[9] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals. Understanding deep learning requires rethinking generalization.

[10] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Joshua Bengio. Sharp minima can generalize for deep nets. Proceedings of the 34th International Conference on Machine Learning

[11] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. Advances in Neural Information Processing Systems 27, pp. 2933-2941, 2014.

[12] Anna Choromanska, Mikael Henaff, Michael Mathieu, Arous Michael, Ben Gerard, and Yann LeCun. The loss surface of multilayer networks. JMLR, 38, 2015.

[13] Jeffrey Pennington, Yasaman Bahri. Geometry of Neural Network Loss Surfaces via Random Matrix Theory. Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.
[14] Jeffrey Pennington, Pratik Worah. Nonlinear random matrix theory for deep learning. *31st Conference on Neural Information Processing Systems* (NIPS 2017).

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting *Journal of Machine Learning Research* 15 1929-1958, 2014.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[17] H. Chau Nguyen, Riccardo Zecchina, and Johannes Berg. Inverse statistical problems: from the inverse Ising problem to data science. *Advances in Physics*, 66:3, pp. 197-261, 2017.

[18] Carlo Di Castro and Roberto Raimondi. Statistical mechanics and applications in condensed matter. *Cambridge University press*, 2015.

[19] David J.C. MacKay. Information Theory, Inference, and Learning Algorithms. *Cambridge University press*, 2003.

[20] E.T. Jaynes. Probability Theory, the logic of science. *Cambridge University Press*, 2003.

[21] See for example the related Kaggle page for a description of the problem and the dataset https://www.kaggle.com/c/boston-housing.

[22] Pasquale Calabrese and John Cardy. Quantum quenches in extended systems. *J. Stat. Mech.*, P06008, 2007.

[23] Shang-Keng Ma. Modern Theory of Critical Phenomena. *Advanced Book Program*, Westview press, 1976.

[24] Marzio Cassandro and Giovanni Jona-Lasinio. Critical Point Behaviour and Probability Theory. *Advances in Physics*, 27:6, pp. 913-941, 1978.

[25] Marc Mézard and Giorgio Parisi. The Bethe lattice spin glass revisited. *European Physical Journal B* 20, pp. 217-233, 2001.

[26] Jack Raymond, Andre Manoel, Manfred Opper. Expectation Propagation. Lecture notes from the autumn school: ?Statistical Physics, Optimization, Inference, and Message-Passing Algorithms?, Les Houches, 2013. [arXiv:1409.6179v1].

[27] Connie Kou, Hwee Kuan Lee and Teck Khim Ng. Distribution Regression Network. [arXiv:1804.04775], 2018.

[28] John Hertz, Anders Krogh and Richard G. Palmer. Introduction to the Theory of Neural Computation. *Addison-Wesley Publishing Company*, 1991.

[29] Michel Le Bellac, Fabrice Mortessagne and G. George Batrouni. Equilibrium and non-Equilibrium Statistical Thermodynamics. *Cambridge University press*, 2004.

[30] A. Engel and C. Van den Broeck. Statistical mechanics of learning. *Cambridge University press*, 2004.

[31] Marc Mezard, Giorgio Parisi and Miguel Angel Virasoro. Spin Glass Theory and Beyond. *World Scientific Lecture Notes in Physics*, Volume 9, 1987.

[32] Cirano De Dominicis and Irene Giardina. Random Fields and Spin Glasses A Field Theory Approach. *Cambridge University Press*, 2016.

[33] Daniel J. Amit and Hanoch Gutfreund. Spin Glass model of Neural Networks. *Physical Review A*, 32 , pp. 1007, 1985.

[34] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, Riccardo Zecchina. Entropy-SGD: Biasing Gradient descent into wide valleys. [arXiv:1611.01838], 2017.
All the code and data produced in this work can be found on [https://github.com/WessZumino/expectation_propagation](https://github.com/WessZumino/expectation_propagation). This includes a pure Python and a Tensorflow implementation, together with the data used in the numerical experiments.

Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference for Learning Representations*, 2015.

Levent Sagun, Leon Bottou and Yann LeCun. Eigenvalues of the Hessian in Deep Learning: Singularity and Beyond. *arXiv:1611.07476v2*, 2017.

Maciej Koch-Janusz and Zohar Ringel. Mutual information, neural networks and the renormalization group. *Nature Physics*, doi: 10.1038/s41567-018-0081-4, 2018.
Supplemental Material

In this supplemental material we provide details on the back-propagation algorithm for ESP, the empirical analysis over the MNIST dataset and the evolution of eigenvalue distributions. The code and data used to perform the numerical experiments, together with the videos of the time evolution of the eigenvalue distribution, can be found in Ref. [36].

5 Back-propagation for ESP

Multi-layer Neural Networks operate – broadly speaking – in two steps: Forward and Backward propagation. The former performs operations on the input information, that then propagates like a signal through the network until it reaches the output layer. Here the processed signal is compared with the provided answer by means of a properly defined error (loss) function, measuring the alignment between the two. Let us introduce first the layer index \( l \in [0, L] \) where \( l = 0 \) denotes the input layer. For algorithmic implementation it is convenient to work directly with the transpose of the weight matrices, i.e. \( \mathbf{w} \rightarrow \mathbf{w}^T \); then FW propagation is easily defined as:

\[
\textbf{Algorithm 1: Forward Pass}
\]

\[
\text{Data: } \mathbf{X}, \mathbf{w}_l, \mathbf{b}_{l}, \beta \\
\text{Result: } \mathbf{a}_l, \mathbf{h}_l \\
\text{for } l \in [1, L] \text{ do } \\
\quad \mathbf{h}_l = \mathbf{w}_l \cdot \mathbf{a}_{l-1} + \mathbf{b}_l \\
\quad \mathbf{a}_l = \mathbf{h}_l \sigma (\mathbf{h}_l, \beta) \\
\text{end}
\]

In the above snippet, the entries are \( \mathbf{X} \) – the input data – of dimension (\( m= \) number of training samples, \( f= \) number of features), the weights matrices \( \mathbf{w}_l \), the biases \( \mathbf{b}_l \) and the noise terms \( \beta \). Given a loss function \( \mathcal{L} \) and a set of parameters \( \theta_l = \{ \mathbf{w}_l, \mathbf{b}_l, \beta_l \} \), we need to find the optimal configuration \( \theta_l^* \) via optimization \( \partial \mathcal{L} / \partial \theta_l = 0 \forall l \). For the last layer one easily finds:

\[
\frac{\partial \mathcal{L}}{\partial \mathbf{w}_L} = \frac{1}{m} \sum_{\mu=1}^{m} \mathbf{e}_\mu \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{w}_L} = \frac{1}{m} \sum_{\mu=1}^{m} \mathbf{e}_\mu \cdot \left( \frac{\partial \mathbf{h}_L}{\partial \mathbf{w}_L} \sigma (\mathbf{h}_L, \beta) + \mathbf{h}_L \frac{\partial \sigma (\mathbf{h}_L, \beta)}{\partial \mathbf{w}_L} \right) \\
= \frac{1}{m} \sum_{\mu=1}^{m} \{ \mathbf{e}_\mu \sigma (\mathbf{h}_L, \beta) + \mathbf{h}_L \beta \sigma (\mathbf{h}_L, \beta) (1 - \sigma (\mathbf{h}_L, \beta)) \} \cdot \mathbf{a}_{L-1} = \frac{1}{m} \sum_{\mu=1}^{m} \mathbf{e}_\mu g (\mathbf{h}_L, \beta) \cdot \mathbf{a}_{L-1}.
\]

We use \( \cdot \) to denote the scalar product, otherwise products are element-wise. We have also introduced the “residual” \( \mathbf{e}_\mu = \mathbf{e}_\mu^T \), whose explicit expression depends on the form of the Loss function. For the regression task, we use the minimum square error:

\[
\mathcal{L} = \frac{1}{2m} \sum_{\mu=1}^{m} (\mathbf{y}_\mu - \mathbf{y}_\mu) = \frac{1}{2m} \sum_{\mu=1}^{m} (\mathbf{e}_\mu)^2.
\]

For classification tasks we used instead the so called “cross entropy” loss. For a binary classification task, \( y_\mu = 0, 1 \forall \mu \in [1, m] \) and the loss function reads:

\[
\mathcal{L} = - \frac{1}{m} \sum_{\mu=1}^{m} \{ y_\mu \log (\hat{y}_\mu \mid \theta, \mathbf{X}^\mu) + (1 - y_\mu) \log (1 - \hat{y}_\mu \mid \theta, \mathbf{X}^\mu)) \},
\]

where the output of the network is now the probability \( \hat{y}_\mu = q_\mu \) of the underlying binomial process. Note that the argument of the logarithm is correctly a dimensionless quantity. In this case the residue reads:

\[
ed_\mu = - \left( \frac{y_\mu}{\hat{y}_\mu} - \frac{1 - y_\mu}{1 - \hat{y}_\mu} \right) = \frac{\hat{y}_\mu - y_\mu}{\hat{y}_\mu (1 - \hat{y}_\mu)}.
\]

In equation (1) we have also defined

\[
g (\mathbf{h}_L^\mu) = \sigma (\beta \cdot \mathbf{h}_L^\mu) + \mathbf{h}_L^\mu \beta \cdot \sigma (\beta \cdot \mathbf{h}_L^\mu)(1 - \sigma (\beta \cdot \mathbf{h}_L^\mu)) \equiv \sigma (\beta \cdot \mathbf{h}_L^\mu) [1 + \mathbf{h}_L^\mu \beta \cdot \sigma (\beta \cdot \mathbf{h}_L^\mu)].
\]

12
Likewise, for the other parameters we find:

$$\frac{\partial L}{\partial b_L} = \frac{1}{m} \sum_{\mu=1}^{m} e^\mu g(h^\mu_L)$$

(6)

$$\frac{\partial L}{\partial \beta_L} = \frac{1}{m} \sum_{\mu=1}^{m} e^\mu f(h^\mu_L)$$

(7)

$$f(h^\mu_L) = |h^\mu_L|^2 \sigma(h^\mu_L) \sigma(-h^\mu_L).$$

(8)

In general, one finds for the l-th hidden layer:

$$\frac{\partial L}{\partial w_l} = \frac{1}{m} \sum_{\mu=1}^{m} \{e^\mu g(h^\mu_L)\} \cdot \frac{\partial h_L}{\partial w_l} = \frac{1}{m} \left[ \prod_{i=l+1}^{L} w_i \right] \cdot \left[ e \prod_{i=L}^{l+1} g(h_i) \right] \cdot a_{l-1}. \quad (9)$$

$$\frac{\partial L}{\partial b_l} = \frac{1}{m} \sum_{\mu=1}^{m} \{e^\mu g(h^\mu_L)\} \cdot \frac{\partial h_L}{\partial b_l} = \frac{1}{m} \left[ \prod_{i=l+1}^{L} w_i \right] \cdot \left[ e \prod_{i=L}^{l+1} g(h_i) \right]$$

$$\frac{\partial L}{\partial \beta_l} = \frac{1}{m} \sum_{\mu=1}^{m} \{e^\mu g(h^\mu_L)\} \cdot \frac{\partial h_L}{\partial \beta_l} = \frac{1}{m} \left[ \prod_{i=l+1}^{L} w_i \right] \cdot \left[ e \prod_{i=L}^{l+1} g(h_i) \right] f(h^\mu_L).$$

Note that for each of the gradients above, the last equality is fully vectorized, i.e. the sum over $\mu$ is taken care of in the scalar product. The algorithmic implementation of the above equations can be found in the pure python code provided in Ref. [36].

6 Numerical analysis

In this section we present additional details on the numerical analysis performed in the main text. The two datasets used to study the binary classification problem are shown in Fig. (1) and provided in Ref. [36]. We also present the analysis of the multi-class classification task performed on the MNIST dataset. In Ref. [1] the authors considered the CIFAR-10 and CIFAR-100 datasets as benchmarks to test ESP against ReLu and many of the heuristic activations used in the literature. A detailed analysis using different Convolutional based architectures showed how ESP consistently outperformed all other activations. Here we carry out our analysis using a simple FFN, in which the original MNIST $8 \times 8$ image is flattened to a one dimensional array and used as an input to a 25-units, single layer FFN. The problem with this handwaving method is that it destroys the two dimensional spatial correlations of the original image, hence the poor performance compared to convolutional networks. Within this approach, the difference between ESP and ReLu is less marked in terms of overall performance, but it is qualitatively similar to the binary classification task considered in the main text. Also in this case, training with ESP leads to a faster decay of the loss function, an higher value of $\alpha$ and a lower (but here finite) of $\gamma$. Finally, the fraction of residues close to zero is, also in this case, larger for the ESP trained FFN. A collection of results for the MNIST dataset is shown in Fig. (2). For completeness, in Fig. (3) we show plots of the $\gamma$-index for the binary classification tasks considered in the main text.
6.1 Empirical Eigenvalue Distributions

In this section, we report numerical experiments on the time evolution of the empirical eigenvalue distributions of the Hessian as we train the FFN with different activations on different classification tasks. We then compare and contrast the geometry of loss surfaces associated with ESP to that associated with ReLu and with Sigmoid activations.

Recall that, at a critical point, the eigenvector corresponding to a negative (positive) eigenvalue of the Hessian specifies the descent (ascent) direction of the loss surface near the critical point. A critical point is a local minima if all the eigenvalues are positive. On the other hand, the larger the relative
fraction of the negative eigenvalues, measured by an index $\alpha$ in the main text, the larger the ratio of the descent directions to the ascent and null directions. As a result, when $\alpha$ is large, gradient descent can quickly escape a critical point – a saddle point in this case – due to the existence of multiple unstable directions. However, when a critical point exhibits multiple near-zero eigenvalues, roughly captured by the index $\gamma$ in the main text, the geometry near the critical point consists of several near-flat (to second-order) directions; in this situation, gradient descent will slowly decrease the training loss, a typical situation when training is almost completed or when the training get stuck in a spurious local minima.

Defining the parameter vector for layer $l \in \{1, L\}$ as $\theta^{[l]}_\alpha = (w^{[l]}_\alpha, b^{[l]}_\alpha, \beta^{[l]}_\alpha), \alpha = 1, 2, 3,$ and for a generic loss function $L$, the Hessian matrix is defined as $H^{L}_{\alpha, \beta} = \partial^2 L / \partial \theta^{[l]}_\alpha \partial \theta^{[l]}_\beta$. For a single hidden layer ESP network, it reads:

$$
\hat{H} = \begin{pmatrix}
\frac{\partial^2 L}{\partial w^{[1]}_1 \partial w^{[1]}_2} & \frac{\partial^2 L}{\partial w^{[1]}_1 \partial b^{[1]}_1} & \frac{\partial^2 L}{\partial w^{[1]}_1 \partial \beta^{[1]}_2} & \frac{\partial^2 L}{\partial w^{[1]}_1 \partial \beta^{[1]}_3} \\
\frac{\partial^2 L}{\partial w^{[1]}_2 \partial w^{[1]}_1} & \frac{\partial^2 L}{\partial w^{[1]}_2 \partial b^{[1]}_1} & \frac{\partial^2 L}{\partial w^{[1]}_2 \partial \beta^{[1]}_2} & \frac{\partial^2 L}{\partial w^{[1]}_2 \partial \beta^{[1]}_3} \\
\frac{\partial^2 L}{\partial b^{[1]}_1 \partial w^{[1]}_1} & \frac{\partial^2 L}{\partial b^{[1]}_1 \partial b^{[1]}_1} & \frac{\partial^2 L}{\partial b^{[1]}_1 \partial \beta^{[1]}_2} & \frac{\partial^2 L}{\partial b^{[1]}_1 \partial \beta^{[1]}_3} \\
\frac{\partial^2 L}{\partial \beta^{[1]}_2 \partial w^{[1]}_1} & \frac{\partial^2 L}{\partial \beta^{[1]}_2 \partial b^{[1]}_1} & \frac{\partial^2 L}{\partial \beta^{[1]}_2 \partial \beta^{[1]}_2} & \frac{\partial^2 L}{\partial \beta^{[1]}_2 \partial \beta^{[1]}_3} \\
\frac{\partial^2 L}{\partial \beta^{[1]}_3 \partial w^{[1]}_1} & \frac{\partial^2 L}{\partial \beta^{[1]}_3 \partial b^{[1]}_1} & \frac{\partial^2 L}{\partial \beta^{[1]}_3 \partial \beta^{[1]}_2} & \frac{\partial^2 L}{\partial \beta^{[1]}_3 \partial \beta^{[1]}_3}
\end{pmatrix} = \begin{pmatrix}
H_{11} & H_{12} \\
H_{21} & H_{22}
\end{pmatrix}.
$$

(10)

In the last equality we have introduced a block matrix representation for the Hessian. Clearly, in order to capture correlations between different layers, one needs to evaluate the full Hessian matrix.

In the following figures, we show our empirical observations of different classification tasks. The eigenvalue distributions from training a FFN consisting of a single hidden layer with 10 units and training a FFN consisting of two hidden layers with 8 and 2 hidden units for the non-linear binary classification task are shown in Fig. (4) and Fig. 5 respectively. The results for the same FFN architectures trained on the linear binary classification task are shown in Fig. 6 and Fig. 7, while those trained on MNIST with a single hidden layer of 25 units is shown in Fig. 8. In these figures, we zoom in near zero, where almost all of the eigenvalues are located, while a few significantly larger eigenvalues (by many order of magnitudes) are not displayed in the plot. The percentage of eigenvalues displayed within the visualization domain is reported in the label of each plot. Separate plots that show the magnitude of the top eigenvalues are shown separately in Figs. (9) and (10).

In all the cases, we observe that the time evolution of the distributions proceeds through two phases (see the videos in [36] for detailed illustrations). In the first phase, the distribution tends to widely spread out around zero, with the index $\alpha$ roughly equal to 0.5, signifying that gradient descent behaviour would descend quickly in the beginning of training due to the existence of multiple unstable directions in the loss surface. In the second phase, the width of the distribution begins to shrink and, at long-times, the negative eigenvalues tend to disappear (or relatively a few exist, which is typically the case when training reaches a very stable critical point) while the distribution of the positive eigenvalues clusters near zero with a large deviation tail, whose largest eigenvalue is shown in Figs. (9) and (10). These behaviours are qualitatively consistent with the predictions of Refs [13][14] (albeit the results there require additional assumptions), and with observations discussed in Ref. [38]. Whether these two phases are related to the empirical error minimisation (ERM) and the representation compression phase of Ref. [4] is worth further investigation.

In linear and nonlinear binary classification tasks, the eigenvalue distributions of ReLu are mostly located near zero during training, suggesting that the ReLu loss landscape consists of critical points with many second-order flat directions; thus, gradient descent could get stuck in these nearly flat critical points. As seen in Fig.2 of the main text, the fact that training errors of ReLu plateaux out at high loss value supports this picture. For ESP and Sigmoid, the distributions spread further from 0 at the beginning, with may negative eigenvalues (large $\alpha$ index), and gradient descent can conveniently move towards lower loss regime due to the existence of multiple unstable directions. When the training is stopped after many epochs have passed, the ESP eigenvalue distribution appears non-negative, whereas ReLu eigenvalue distribution still exhibits negative values; thus, training FFN
with ESP on binary classification tasks allow gradient descent to find local minima faster than training with Sigmoid, and of course faster than ReLu. For training on MNIST datasets, ESP also reaches the non-negative eigenvalue distributions faster than the other two activation functions do, as shown in the last training epoch of Fig. (8). Again, this demonstrates the superiority of ESP in allowing gradient descent to reach a local minima in its loss landscape faster than the other two activation functions do.

Figure 4: Empirical eigenvalue distributions from training with nonlinear binary classification artificial dataset with FFN of structure (10,1) as epochs grow.

Figure 5: Empirical eigenvalue distributions from training with nonlinear binary classification artificial dataset with FFN of structure (8,2,1) as epochs grow.
Figure 6: Empirical eigenvalue distributions from training with linear binary classification artificial dataset with FFN of structure (10,1) as epochs grow.

Figure 7: Empirical eigenvalue distributions from training with linear binary classification artificial dataset with FFN of structure (8,2,1) as epochs grow.
Figure 8: Empirical eigenvalue distributions from training with MNIST dataset as epochs grow.

Figure 9: Top eigenvalues for the Binary classification task. (Top left/right) linear/non-linear dataset with a single, 10 units hidden layer. (Bottom left/right) linear/non-linear dataset with two hidden layers of 8 and 2 units respectively.

Figure 10: Top eigenvalues for MNIST.