# Towards a theory of machine learning

**Vitaly Vanchurin**

Department of Physics, University of Minnesota, Duluth, Minnesota, 55812
Duluth Institute for Advanced Study, Duluth, Minnesota, 55804

E-mail: vvanchur@d.umn.edu

**Abstract.** We define a neural network as a septuple consisting of (1) a state vector, (2) an input projection, (3) an output projection, (4) a weight matrix, (5) a bias vector, (6) an activation map and (7) a loss function. We argue that the loss function can be imposed either on the boundary (i.e. input and/or output neurons) or in the bulk (i.e. hidden neurons) for both supervised and unsupervised systems. We apply the principle of maximum entropy to derive a canonical ensemble of the state vectors subject to a constraint imposed on the bulk loss function by a Lagrange multiplier (or an inverse temperature parameter). We show that in an equilibrium the canonical partition function must be a product of two factors: a function of the temperature and a function of the bias vector and weight matrix. Consequently, the total Shannon entropy consists of two terms which represent respectively a thermodynamic entropy and a complexity of the neural network. We derive the first and second laws of learning: during learning the total entropy must decrease until the system reaches an equilibrium (i.e. the second law), and the increment in the loss function must be proportional to the increment in the thermodynamic entropy plus the increment in the complexity (i.e. the first law). We calculate the entropy destruction to show that the efficiency of learning is given by the Laplacian of the total free energy which is to be maximized in an optimal neural architecture, and explain why the optimization condition is better satisfied in a deep network with a large number of hidden layers. The key properties of the model are verified numerically by training a supervised feedforward neural network using the method of stochastic gradient descent. We also discuss a possibility that the entire universe on its most fundamental level is a neural network.

**Keywords:** machine learning, statistical mechanics, thermodynamics of learning, quantum mechanics, emergent gravity

## Contents



## 1 Introduction

Despite of many attempts [1–6] the effectiveness of deep learning has so far no clear explanation. This is rather surprising given that a neural network is a very simple and a well-defined mathematical object [7–9]. What makes it difficult to analyze is that the deep neural networks are typically described with a very large number of parameters, e.g. weight matrix, bias vector, training data, etc. For such systems most of the analytical techniques are not very useful and one must rely on numerics. The situation is very similar to what happens in physics. Physical systems (both classical and quantum) can often be solved exactly when the number of degrees of freedom is small, but the problem becomes intractable when the number of degrees of freedom is large. Fortunately, there is a set of ideas which proved very useful for analyzing physical systems with many degrees of freedom. It is statistical mechanics. The main point of the present paper is to apply the methods of statistical mechanics to machine learning. In the remainder of this section, we will summarize the main results as it might help the reader to navigate through the paper.

In Sec. 2 we set the stage by defining a neural septuple (i.e. state vector, input/output projection operators, weight matrix, bias vector, activation map and loss function). The septuple is not equivalent to the standard neural architecture used in machine learning, but it does include such systems as a special limit. There are three main motivations to define these more general structures. First of all, we want to develop a unified treatment of different types of learning algorithms, i.e. supervised, unsupervised, etc. Secondly, we want the very

structure of hidden layers to be a dynamical variable in addition to the weight matrices and bias vectors. And finally, we want to have a theoretical framework which is suitable for a statistical description.

In Sec. 3 we address the main problem of unsupervised learning, namely, what should be an appropriate loss function if the training dataset contains only input, but no output data. We claim that an answer can be obtained by defining a local error and a local objective for hidden neurons (or in the bulk) instead of a more conventional error for output neurons (or on the boundary). The boundary loss is usually given by a sum over errors on the boundary (i.e. over input/output neurons), but the bulk loss could be a sum over both local errors and local objectives in the bulk (i.e. over hidden neurons). A simple example of a local objective for a neuron is a binary classification of an incoming signal, and then an outgoing signal with values closer to lower- and upper-bounds are rewarded and values in-between are penalized.

In Sec. 4 we consider two statistical ensembles over state vectors: a micro-canonical-type ensemble and a canonical-type ensemble. We expect that in the limit of a large number of neurons the two ensembles are equivalent, as is usually the case in statistical physics, but the latter ensemble (i.e. canonical ensemble) is a lot easier to handle analytically. Moreover, we show that the canonical ensemble can be derived from the Jaynes' maximum entropy principle [10, 11]. The principle states that the probability distribution (in our case statistical ensemble) which best represents the current state of knowledge is the one with the largest Shannon entropy.

In Sec. 5 we define perhaps the most important object in statistical mechanics - a partition function. For a bulk loss with (or without) a quadratic local objective the canonical ensemble can be approximated by a Gaussian integral and then the (canonical) partition function can be calculated analytically. A minor complication is that the range of integration (which is set by the range of an activation function) is finite in contrast to the Gaussian integral whose range is infinite. Nevertheless, the problem can be solved by replacing the sharp cut-off on the boundaries of integration with a smooth Gaussian window function. In this section we also define an operator $\hat{G}$ whose spectrum determines the canonical partition function and plays a central role in everything that follows.

In Sec. 6 we define a time-invariant state of equilibrium (or what we call a learning equilibrium) and show that in such state the partition function must factorize into product of two factors: a function of the temperature and a function of the bias vector and weight matrix. Among other things it implies that the total free energy is a difference of two terms: a familiar thermodynamic free energy and an unfamiliar product of the temperature and a complexity function. While the thermodynamic free energy is a function of only temperature and usually decreases, the total free energy is expected to increase with learning due to a decrease in the complexity function. This might sound odd, but, in fact, is a mare consequence of an openness of the learning system where the entropy is flowing out of the system.

In Sec. 7 we calculate the total entropy of the canonical ensemble and argue that in an equilibrium it must be a sum of a familiar thermodynamic entropy and of an unfamiliar complexity function which is directly related to a dynamical dimensional reduction of the state space. We then argue that the total entropy must decrease until the systems reaches an equilibrium (i.e. the second law of learning (7.6)) and the increment in the loss function must be proportional to the increment in the thermodynamic entropy plus the increment in the complexity (i.e. the first law of learning (7.11)).

In Sec. 8 we calculate a non-equilibrium production of the Shannon entropy (of a probability distribution function over weight matrices and bias vectors) and argue that in an

optimal neural architecture the entropy production must be maximized (or, more precisely, the entropy destruction should be minimized). This is in a complete agreement with the so-called stationary entropy production principle that was used in [14] to derive an approximate Schrödinger equation from a highly constrained stochastic process and in [15] to derive an approximate Einstein equation from non-equilibrium thermodynamics of the metric tensor. We then show that the rate of the entropy production is proportional to the Laplacian of the free energy in the configuration space of the weight matrices and bias vectors.

In Sec. 9 we used the criteria of minimizing the entropy destruction (or minimizing the negative Laplacian of the free energy) to derive an expected dynamics of a spectrum of operator $\hat{G}$ in an optimal neural architecture. In particular, we show that most of the eigenvalues of the operator $\log \hat{G}$ should remain near zero, a small fraction of the largest eigenvalues should move to positive values $\gtrsim 0$ and a very small fraction of eigenvalues should move to very small values $\ll 0$. This implies that the effectiveness of a neural network can be translate into a skewness of the distribution of eigenvalues of $\log \hat{G}$, i.e. the larger the skewness the better a neural network is expected to perform. Then we show that the skewness in a deep architecture is much larger than in a shallow architecture which demonstrates why the deep architecture is preferred.

In Sec. 10 we discuss the main results of numerical experiments conducted using the TensorFlow Python library [16] and MNIST database of handwritten images [17]. Two different neural networks (a deep network with two hidden layers and a shallow network with a single hidden layer) were trained using the method of stochastic gradient descent and then the numerical data were analyzed in context of the analytical calculations carried out in the paper. More specifically, the training evolution of the bulk and boundary loss functions progressed as expected, the predicted dynamics of the spectrum of operator $\hat{G}$ was established, and the anticipated relaxation of the complexity function towards equilibrium was confirmed.

And finally in Sec. 11 and Sec. 12 we discuss a possibility that the entire universe on its most fundamental level is a neural network. For such an ambitious proposal to actually work we claim that the three components: quantum mechanics, general relativity and macroscopic observers must all emerge from a microscopic neural network. For the time being, we leave aside the problem of observers (see, however, [33]) and study a possible emergence of quantum mechanics and general relativity. In particular, we show that approximate Schrödinger equation (see Sec. 11) and Einstein equations (see Sec. 12) can indeed emerge from a network with a large number of neurons not too far from a learning equilibrium.

## 2 Neural septuple

We start by introducing all of the essential ingredients of a neural network or, what we shall call, a neural septuple consisting of:

(1) $\mathbf{x}$, a state vector which describes the state of all neurons,
(2) $\hat{P}_{in}$, an input projection which describes a subspace spanned by input neurons,
(3) $\hat{P}_{out}$, an output projection which describes a subspace spanned by output neurons,
(4) $\hat{w}$, a weight matrix which describes directed connections between all pairs of neurons,
(5) $\mathbf{b}$, a bias vector which describes biases in the inputs of all neurons,
(6) $\mathbf{f}$ , an activation map which describes a non-linear transformation, and

(7) $H$, a loss function which describes a learning objective of the entire network.

Consider a collection of $N$ neurons described by a column[1] *state vector*, $\mathbf{x} \in \mathbb{R}^N$, whose components are real numbers, $x_i \in \mathbb{R}$, but one can also generalize the construction to complex numbers or other fields. There are three types of neurons: input neurons, hidden neurons and output neurons, and so it is convenient to define three subspaces of the state space: input subspace $\mathcal{V}_{in}$, output subspace $\mathcal{V}_{out}$ and hidden subspace $\mathcal{V}_{hid}$. We shall also refer to the direct sum of the input and output subspaces, $\mathcal{V}_{in} \oplus \mathcal{V}_{out}$, as a boundary and to the hidden subspace, $\mathcal{V}_{hid}$, as a bulk. A neural network is trained by specifying the components of $\mathbf{x}$ in the boundary subspace which represent only the input and output neurons. The boundary components can be described with two projection operators (or matrices): an *input projection* $\hat{P}_{in}$ and an *output projection* $\hat{P}_{out}$. These operators can be used to project a state vector $\mathbf{x}$ to either input subspace, i.e. $\hat{P}_{in}\mathbf{x} \in \mathcal{V}_{in}$, output subspace, i.e. $\hat{P}_{out}\mathbf{x} \in \mathcal{V}_{out}$, boundary subspace, i.e. $(\hat{P}_{in} + \hat{P}_{out})\mathbf{x} \in \mathcal{V}_{in} \oplus \mathcal{V}_{out}$, or bulk subspace, i.e. $(\hat{I} - \hat{P}_{in} - \hat{P}_{out})\mathbf{x} \in \mathcal{V}_{hid}$.

The neurons are connected into a neural network with connections described by a *weight matrix*, $\hat{w}$, which is also an adjacency matrix of a weighted directed graph with individual neurons representing the nodes of the graph. For the neural networks considered here the components of the weight matrix are assumed to be real numbers, $w_{ij} \in \mathbb{R}$, but one can also generalize the construction to complex numbers or other fields. In addition, for the so-called feedforward neural network with $L$ layers (i.e. one input layer, one output layer and $L-2$ hidden layers) the weight matrix is taken to be nilpotent, i.e.

$$\hat{w}^n = \left(\hat{w}^T\right)^n = 0 \quad \forall n \geq L, \tag{2.1}$$

there are no incoming connections to the input neurons, i.e.

$$\hat{P}_{in}\hat{w} = 0, \tag{2.2}$$

and there are no outgoing connections from the output neurons, i.e.

$$\hat{P}_{out}\hat{w}^T = 0. \tag{2.3}$$

The state vector can only change when either a new training data $\hat{P}_{in}\mathbf{x}_\partial \in \mathcal{V}_{in}$ are entered,

$$\mathbf{x}(0) = \hat{P}_{in}\mathbf{x}_\partial \tag{2.4}$$

or the new data propagate through the network

$$\mathbf{x}(t+1) = \mathbf{f}\left(\hat{w}\mathbf{x}(t) + \mathbf{b}\right). \tag{2.5}$$

The vector $\mathbf{b} \in \mathbb{R}^N$ is a *bias vector* and $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^N$ is an *activation map* which acts separately on each component, i.e.

$$f_i(\mathbf{y}) = f_i\left(y_i\right). \tag{2.6}$$

where $f_i(y)$'s are the activation functions (e.g. $f_i(y) = \tanh(y)$, $f_i(y) = \max(0, y)$).

For the input neurons with no incoming connections to remain fixed, i.e.

$$\hat{P}_{in}\mathbf{x}(t) = \hat{P}_{in}\mathbf{x}(0) = \mathbf{x}(0) = \hat{P}_{in}\mathbf{x}_\partial, \tag{2.7}$$

[1] We adopt the physicists' notations where the state vector is a column vector and not a row vector which is usually used in the machine learning literature.

an additional condition must also be imposed on the input bias,

$$\mathbf{f}\left(\hat{P}_{in}\mathbf{b}\right) = \hat{P}_{in}\mathbf{x}_{\partial}. \tag{2.8}$$

This condition is satisfied for a feedforward neural network, but need not be satisfied for more general learning systems. After a finite number of steps $t$ the state vector $\mathbf{x}(t)$ may converge to a fixed state $\mathbf{x}(t) = \bar{\mathbf{x}}$ defined by a fixed point equation

$$\bar{\mathbf{x}} = \mathbf{f}\left(\hat{w}\bar{\mathbf{x}} + \mathbf{b}\right). \tag{2.9}$$

For example, in a deep feedforward neural network with $L$ layers the fixed state would be reached after $L-1$ steps, i.e. $\mathbf{x}(L-1) = \bar{\mathbf{x}}$, given that the condition on the input bias (2.8) is satisfied. For more general systems the state may or may not converge to a fixed point depending on the activation transformation (2.5) and initial conditions (2.4).

The final ingredient of a neural septuple is a loss function. In a feedforward neural network the loss function is usually defined by projecting the fixed state $\bar{\mathbf{x}}$ to the output subspace $\hat{P}_{out}\bar{\mathbf{x}} \in \mathcal{V}_{out}$ and then by comparing the result with a desired output state $\hat{P}_{out}\mathbf{x}_{\partial} \in \mathcal{V}_{out}$. For example, one can define a loss function as a squared error of the output neurons,

$$\begin{aligned} H_{\partial}(\bar{\mathbf{x}}, \mathbf{b}, \hat{w}) &= \left(\hat{P}_{out}\bar{\mathbf{x}} - \hat{P}_{out}\mathbf{x}_{\partial}\right)^{T}\left(\hat{P}_{out}\bar{\mathbf{x}} - \hat{P}_{out}\mathbf{x}_{\partial}\right) \\ &= \left(\bar{\mathbf{x}} - \mathbf{x}_{\partial}\right)^{T}\hat{P}_{out}^{T}\hat{P}_{out}\left(\bar{\mathbf{x}} - \mathbf{x}_{\partial}\right) \\ &= \left(\bar{\mathbf{x}} - \mathbf{x}_{\partial}\right)^{T}\hat{P}_{out}\left(\bar{\mathbf{x}} - \mathbf{x}_{\partial}\right). \end{aligned} \tag{2.10}$$

Since there is no error on the input neurons (2.7) we can also rewrite it as a squared error on all boundary (i.e. input and output) neurons

$$H_{\partial}(\bar{\mathbf{x}}, \mathbf{b}, \hat{w}) = \frac{1}{2}\left(\bar{\mathbf{x}} - \mathbf{x}_{\partial}\right)^{T}\left(\hat{P}_{in} + \hat{P}_{out}\right)\left(\bar{\mathbf{x}} - \mathbf{x}_{\partial}\right). \tag{2.11}$$

For this reason, we shall refer to $H_{\partial}$ as a boundary loss function.

## 3 Supervised vs. unsupervised

In the pervious section we defined a neural network as a neural septuple $\left(\mathbf{x}, \hat{P}_{in}, \hat{P}_{out}, \hat{w}, \mathbf{b}, \mathbf{f}, H\right)$ where $\mathbf{x}$ is a state vector of all (input, output and hidden) neurons, $\hat{P}_{in}\mathbf{x}$ is a state of only input neurons, $\hat{P}_{out}\mathbf{x}$ is a state of only output neurons, $\hat{w}$ is a weight matrix between all pairs of neurons, $\mathbf{b}$ is a bias vector for all neurons, $\mathbf{f}(\mathbf{y})$ is an activation map and $H(\mathbf{x}, \mathbf{b}, \hat{w})$ is a loss function. A simple example of a loss function is the boundary loss (2.11) which is known to work very well in a supervised learning. Unfortunately, the boundary loss cannot be used in unsupervised systems where the output subspace is empty, $\mathcal{V}_{out} = \emptyset$, and thus the boundary loss is always zero, $H = H_{\partial} = 0$.[2] For this reason, in unsupervised systems (beyond auto-encoders) we must consider other loss functions which are, perhaps, more general than the boundary loss.

A key observation is that in equation (2.11) the boundary loss was due to a mismatch in the output conditions or (together with input conditions) in the boundary conditions, i.e.

[2] In our description an auto-encoder is viewed as a supervised system with periodic boundary conditions, i.e. the input and output states are set equal to each other.

$(\hat{P}_{in} + \hat{P}_{out})\bar{\mathbf{x}} \neq \mathbf{x}_\partial$, but the fixed point equation (2.9) was satisfied exactly. Alternatively, we can assume that the boundary conditions are satisfied exactly $(\hat{P}_{in} + \hat{P}_{out})\mathbf{x} = \mathbf{x}_\partial$, but the fixed point equation is only approximate. Then we can define a bulk loss (as opposed to the boundary loss) as a sum of squares of errors in the fixed point equation, i.e.

$$H(\bar{\mathbf{x}}, \mathbf{b}, \hat{w}) = \frac{1}{2} \left(\bar{\mathbf{x}} - \mathbf{f}\left(\hat{w}\bar{\mathbf{x}} + \mathbf{b}\right)\right)^T \left(\bar{\mathbf{x}} - \mathbf{f}\left(\hat{w}\bar{\mathbf{x}} + \mathbf{b}\right)\right), \tag{3.1}$$

where $\bar{\mathbf{x}}$ is the value of $\mathbf{x}$ at a minimum of $H(\mathbf{x}, \mathbf{b}, \hat{w})$ subject to boundary conditions $(\hat{P}_{in} + \hat{P}_{out})\mathbf{x} = \mathbf{x}_\partial$, i.e.

$$H(\bar{\mathbf{x}}, \mathbf{b}, \hat{w}) = \min_{(\hat{P}_{in}+\hat{P}_{out})\mathbf{x}=\mathbf{x}_\partial} H(\mathbf{x}, \mathbf{b}, \hat{w}). \tag{3.2}$$

This is the simplest bulk loss[3] which is still zero for unsupervised feedforward neural networks, but can be easily generalized to functions which can be used in both supervised and unsupervised learning.

The main idea is that, from the point of view of an individual neuron, a (more general) learning objective can be modeled as a minimization of a local error and at the same time a maximization of a local objective. It is convenient to think of the local error as a supervised quantity (e.g. $(\bar{x}_i - f_i(\sum_j w_{ij}\bar{x}_j + b_i))^2$ in the bulk loss function (3.1)) and of the local objective as a (yet to be defined) unsupervised quantity. Then even if the local error is already at its minimum (as is always the case for unsupervised feedforward neural networks) there is still another quantity which needs to be extremized, i.e. the local objective. This does not mean that the inclusion of the local objective would only benefit an unsupervised learning. Once an appropriate local objective is identified it can be incorporated into a (bulk or boundary) loss function to improve the convergence of a learning algorithm.

For example, the local objective might be a binary classification of an incoming signal $\sum_j w_{ij}\bar{x}_j + p_j$ and then the values of $\bar{x}_i$ closer to lower- and upper-bounds should be rewarded and values in-between should be penalized. Such a classification objective can always be modeled with an appropriately chosen "potential" term for each neuron. For example if

$$V(\bar{\mathbf{x}}) = \sum_i V_i(\bar{x}_i) = -\sum_i \frac{m}{2}\bar{x}_i^2 \tag{3.3}$$

then the bulk loss function can be defined as

$$\begin{aligned} H(\bar{\mathbf{x}}, \mathbf{b}, \hat{w}) &= \frac{1}{2} \left(\bar{\mathbf{x}} - \mathbf{f}\left(\hat{w}\bar{\mathbf{x}} + \mathbf{b}\right)\right)^T \left(\bar{\mathbf{x}} - \mathbf{f}\left(\hat{w}\bar{\mathbf{x}} + \mathbf{b}\right)\right) + V(\bar{\mathbf{x}}) \\ &= \frac{1}{2} \sum_i \left[ \left( \bar{x}_i - f_i\left( \sum_j w_{ij}\bar{x}_j + p_j \right) \right)^2 - m\bar{x}_i^2 \right]. \end{aligned} \tag{3.4}$$

By minimizing this loss function we accomplish both tasks: the minimization of the local error and maximization of the local objective (in this case the binary classification objective). Note that the "tachyonic potential" (3.3) does not lead to any runaway solutions if the range of $\bar{x}_i$ is bounded by the range of the activation function. For example, if the activation function is $f(y) = \tanh(y)$ then $x_i \in (-1, 1)$.

More generally, any two (or more) neurons might have a common objective and then the potential term must also include "interactions", i.e. $V(\bar{\mathbf{x}}) = \sum_i V_i(\bar{x}_i) + \sum_{ij} g^{ij}\bar{x}_i\bar{x}_j \ldots$ .

[3] This bulk loss function is similar in spirit (but not the same) to the error calculated in the back-propagation algorithm where the error on the output neurons is back-propagated to the hidden neurons.

In either case, according to (3.2), the corresponding learning objective remains the same, i.e. we must adjust $\hat{w}$ and $\mathbf{b}$ in such a way that for a given set of boundary condition $(\hat{P}_{in} + \hat{P}_{out})\bar{\mathbf{x}} = \mathbf{x}_\partial$ the bulk loss function is minimized. What is, however, different is that the bulk loss function $H(\mathbf{x}, \mathbf{b}, \hat{w})$ is now given by (3.4) which contains both a local error (a supervised or a kinetic term) and a local objective (an unsupervised or a potential term). As a result, the corresponding bulk loss function is well-defined and (generically) non-zero for both supervised and unsupervised systems. Unfortunately, a solution of equation (3.2) is difficult to obtain exactly and so the statistical methods must be employed.

## 4 Statistical ensembles

There are basically two ways to proceed: experimental (based on numerics) or theoretical (based on statistics). We will start with statistical approach as it might assist us in numerical searches. For starters, consider a statistical ensemble of boundary conditions or, more precisely, a probability distribution $p_\partial(\mathbf{x}_\partial)$ over components of the state vector in the boundary subspace $\mathbf{x}_\partial = (\hat{P}_{in} + \hat{P}_{out})\mathbf{x}$. Such a distribution can, for example, be extracted from a training dataset. Then, instead of minimizing a loss function for individual boundary data, the learning objective could be to minimize an ensemble-averaged loss function, i.e.

$$U_0(\mathbf{b}, \hat{w}) \equiv \int d^{N_\partial} x_\partial \min_{(\hat{P}_{in}+\hat{P}_{out})\mathbf{x}=\mathbf{x}_\partial} H(\mathbf{x}, \mathbf{b}, \hat{w}) p_\partial(\mathbf{x}_\partial) \tag{4.1}$$

where $N_\partial \leq N$ is the dimensionality of the boundary subspace. If we now extend the probability distribution into the bulk by defining

$$p_0(\mathbf{x}) = p_\partial((\hat{P}_{in} + \hat{P}_{out})\mathbf{x})\delta(\bar{\mathbf{x}} - \mathbf{x}), \tag{4.2}$$

where $\bar{\mathbf{x}}$ is given by (3.2), then the ensemble-averaged loss function is simply

$$U_0(\mathbf{b}, \hat{w}) = \int d^N x H(\mathbf{x}, \mathbf{b}, \hat{w}) p_0(\mathbf{x}). \tag{4.3}$$

Of course, all that we did is moved the difficulty of calculating $\bar{\mathbf{x}}$ into $p_0(\mathbf{x})$, but that does not solve the main problem. It is still a computationally intensive task to calculate $U_0(\mathbf{b}, \hat{w})$ exactly and this is where statistical mechanics comes to rescue. The key idea is to replace the micro-canonical-type ensemble (4.2) with a canonical-type ensemble (4.10). Note that for sufficiently large systems (in our case large neural networks) one can often show that the two ensembles are equivalent (i.e. predictions are identical) but the canonical ensemble is much easier to handle analytically.

Consider a statistical ensemble of neural networks, or a probability distribution $p(\mathbf{x})$, over state vectors $\mathbf{x}$. Let's say we do not know how the network was trained (i.e. which algorithm was used), but we do know that the ensemble-averaged bulk loss $H(\mathbf{x}, \mathbf{b}, \hat{w})$ was reduced to some fixed value,

$$U(\mathbf{b}, \hat{w}) = \int d^N x H(\mathbf{x}, \mathbf{b}, \hat{w}) p(\mathbf{x}). \tag{4.4}$$

Then according to the principle of maximum entropy [10, 11], the most reasonable distribution $p(\mathbf{x})$ is a distribution which has the largest Shannon entropy

$$S(p) \equiv -\int d^N x \; p(\mathbf{x}) \log p(\mathbf{x}) = -\langle \log p(\mathbf{x}) \rangle. \tag{4.5}$$

subject to constraint (4.4). The maximization problem can be solved using the method of Lagrange multipliers. If we define a "Lagrangian"

$$\begin{aligned} L(p,\beta,\nu) &= S(p) + \beta\left(U(\mathbf{b},\hat{w}) - \int d^N x\; p(\mathbf{x}) H(\mathbf{x},\mathbf{b},\hat{w})\right) + \nu\left(1 - \int d^N x\; p(\mathbf{x})\right), \\ &= \int d^N x\; p(\mathbf{x})\left(-\log p(\mathbf{x}) - \beta H(\mathbf{x},\mathbf{b},\hat{w}) - \nu\right) + \beta U(\mathbf{b},\hat{w}) + \nu \end{aligned} \tag{4.6}$$

then at a maximum of $L(p,\beta,\nu)$ the variations with respect to $p(\mathbf{x})$, $\beta$ and $\nu$ must vanish

$$0 = \frac{\delta L(p,\beta,\nu)}{\delta p(\mathbf{x})} = -\beta H(\mathbf{x},\mathbf{b},\hat{w}) - \log p(\mathbf{x}) - 1 - \nu \tag{4.7}$$

$$0 = \frac{\partial L(p,\beta,\nu)}{\partial \beta} = U(\mathbf{b},\hat{w}) - \int d^N x\; p(\mathbf{x}) H(\mathbf{x},\mathbf{b},\hat{w}) \tag{4.8}$$

$$0 = \frac{\partial L(p,\beta,\nu)}{\partial \nu} = 1 - \int d^N x\; p(\mathbf{x}). \tag{4.9}$$

Therefore, the maximum entropy distribution must be given by

$$p(\mathbf{x}) = \exp\left(-\beta H(\mathbf{x},\mathbf{b},\hat{w}) - 1 - \nu\right) \tag{4.10}$$

with Lagrange multipliers $\beta$ and $\nu$ determined from the constraint (4.8) and normalization condition (4.9). In what follows we shall refer to the distribution (4.10) as a canonical ensemble.

## 5 Partition function

The partition function for the canonical ensemble (4.10) is defined as $\mathcal{Z} \equiv \exp(1+\nu)$ and can be expressed as an integral over the state space

$$\mathcal{Z}(\beta,\mathbf{b},\hat{w},...) = \int d^N x\; e^{-\beta H(\mathbf{x},\mathbf{b},\hat{w})} \tag{5.1}$$

where ... should remind us of any additional variables (e.g. $m$) which could determine the functional form of $H$. To calculate the partition function (5.1) for a bulk loss (3.4) we can approximate the integral with a Gaussian. This can be done by first expanding the activation function

$$\mathbf{f}\left(\hat{w}\mathbf{x} + \mathbf{b}\right) = \mathbf{f}(\hat{w}\langle\mathbf{x}\rangle + \mathbf{b}) + \hat{f}'\hat{w}(\mathbf{x} - \langle\mathbf{x}\rangle) + \mathcal{O}((\mathbf{x} - \langle\mathbf{x}\rangle)^2) \tag{5.2}$$

where the ensemble-averaged state vector is

$$\langle\mathbf{x}\rangle \equiv \int d^N x\; \mathbf{x}\; p(\mathbf{x}) \tag{5.3}$$

and a diagonal matrix of first derivatives of the activation function is

$$f'_{ii} \equiv \left(\frac{df(y_i)}{dy_i}\right)_{y_i = \sum_j w_{ij}\langle x_j\rangle + b_i}. \tag{5.4}$$

Then to the first order in perturbation theory the bulk loss function is

$$H(\mathbf{x},\mathbf{b},\hat{w}) \approx \frac{1}{2}(\mathbf{x} - \langle\mathbf{x}\rangle)^T \hat{G}(\mathbf{x} - \langle\mathbf{x}\rangle) - \frac{m}{2}\mathbf{x}^T\mathbf{x} \tag{5.5}$$

where

$$\hat{G} \equiv \left(\hat{I} - \hat{f}'\hat{w}\right)^T \left(\hat{I} - \hat{f}'\hat{w}\right). \tag{5.6}$$

Next, we note that the domain of $\mathbf{x}$ is bounded by the range of the activation function. For example, if the activation function is $f(x) = \tanh(x) \in (-1, 1)$, then the partition function is

$$\mathcal{Z}(\beta, \mathbf{b}, \hat{w}) = \int_{\mathbf{x}\in(-1,1)^N} d^N x\, e^{-\beta H(\mathbf{x},\mathbf{b},\hat{w})}. \tag{5.7}$$

The sharp boundaries can be approximated with a smooth Gaussian window function, i.e.

$$\mathcal{Z}(\beta, \mathbf{b}, \hat{w}) \approx \int_{\mathbf{x}\in(-\infty,\infty)^N} d^N x\, e^{-\beta H(\mathbf{x},\mathbf{b},\hat{w})} e^{-\frac{1}{2}\mathbf{x}^T\mathbf{x}}. \tag{5.8}$$

and then the overall partition function is a Gaussian and can be easily evaluated,

$$\begin{aligned}\mathcal{Z}(\beta, \mathbf{b}, \hat{w}) &\approx \int d^N x\, e^{-\beta H(\mathbf{x},\mathbf{b},\hat{w})-\frac{1}{2}\mathbf{x}^T\mathbf{x}} \approx \int d^N x\, e^{-\frac{\beta}{2}(\mathbf{x}-\langle\mathbf{x}\rangle)^T\hat{G}(\mathbf{x}-\langle\mathbf{x}\rangle)-\frac{1-\beta m}{2}\mathbf{x}^T\mathbf{x}} \\ &\approx (2\pi)^{N/2} \det\left(\hat{I}(1-\beta m) + \beta\hat{G}\right)^{-1/2} \exp\left(-\frac{1}{2}\langle\mathbf{x}\rangle^T \left(\frac{(1-\beta m)\beta\hat{G}}{\hat{I}(1-\beta m) + \beta\hat{G}}\right)\langle\mathbf{x}\rangle\right).\end{aligned} \tag{5.9}$$

The spectrum of $\hat{G}$ is defined by an eigenvalue equation

$$\hat{G}\mathbf{v}_i = \hat{G}\lambda_i \tag{5.10}$$

where $\lambda_i$ are the real eigenvalues and $\mathbf{v}_i$ are the respected eigenvectors. Then the log of partition function is

$$\log \mathcal{Z}(\beta, \mathbf{b}, \hat{w}) \approx -\frac{1}{2}\sum_i \log\left(1 - \beta m + \beta\lambda_i\right) - \frac{1}{2}\sum_i \frac{(1-\beta m)\beta\lambda_i a_i^2}{1-\beta m+\beta\lambda_i} + \frac{N}{2}\log(2\pi) \tag{5.11}$$

where

$$\langle\mathbf{x}\rangle = a_i\mathbf{v}_i. \tag{5.12}$$

In the limit of a large number of neurons $N$, the average components are small $a_i^2 \ll 1$, the second term in (5.11) is subdominant in comparison to the first term and can be dropped. Then the partition function is given by (5.9) but without an exponential factor, i.e.

$$\mathcal{Z}(\beta, \mathbf{b}, \hat{w}) \approx (2\pi)^{N/2} \det\left(\hat{I}(1-\beta m) + \beta\hat{G}\right)^{-1/2} \tag{5.13}$$

and the log of the partition function is

$$\begin{aligned}\log \mathcal{Z}(\beta, \mathbf{b}, \hat{w}) &\approx -\frac{1}{2}\log\det\left(\hat{I}(1-\beta m) + \beta\hat{G}\right) + \frac{N}{2}\log(2\pi) \\ &\approx -\frac{1}{2}Tr\log\left(\hat{I}(1-\beta m) + \beta\hat{G}\right) + \frac{N}{2}\log(2\pi) \\ &\approx -\frac{1}{2}\sum_i \log\left(1-\beta m+\beta\lambda_i\right) + \frac{N}{2}\log(2\pi).\end{aligned} \tag{5.14}$$

Note, however, that this is a very rough estimate of the true partition function borne out of our statistical description, but the hope is that this approximation is rich enough to explain at least some aspects of machine learning.

## 6 Learning equilibrium

Given the partition function (5.1) the average bulk loss can be calculated by simple differentiation,

$$U(\beta, \mathbf{b}, \hat{w}) = \int d^N x H(\mathbf{x}, \mathbf{b}, \hat{w}) p(\mathbf{x}) = -\frac{\partial}{\partial \beta} \log(\mathcal{Z}(\beta, \mathbf{b}, \hat{w})), \tag{6.1}$$

where we have explicitly shown that $U(\beta, \mathbf{b}, \hat{w})$ depends on the Lagrange multiplier $\beta$ (or, what we shall call, an inverse temperature parameter). If the neural network was already trained for a very long time, then the weight matrix and the bias vector must be in a state which minimizes the average loss $U(\beta, \mathbf{b}, \hat{w})$ and then its variations with respect to $\hat{w}$ and $\mathbf{b}$ must vanish,

$$\begin{aligned} \frac{\partial U(\beta, \mathbf{b}, \hat{w})}{\partial w_{ij}} &= \frac{\partial^2}{\partial w_{ij} \partial \beta} \log(\mathcal{Z}(\beta, \mathbf{b}, \hat{w})) = 0 \\ \frac{\partial U(\beta, \mathbf{b}, \hat{w})}{\partial b_i} &= \frac{\partial^2}{\partial b_i \partial \beta} \log(\mathcal{Z}(\beta, \mathbf{b}, \hat{w})) = 0. \end{aligned} \tag{6.2}$$

We shall call this state, a state of the learning equilibrium or just an equilibrium state. For a generic system, the degeneracy of an equilibrium state or the dimensionally of an equilibrium manifold (or the number of "Goldstone" modes) can be quite large, but is still much smaller than $N$.

A very important property of an equilibrium, which follows from (6.2), is that the partition function must be a product of two terms

$$\mathcal{Z}(\beta, \mathbf{b}, \hat{w}) = \exp\left(-\beta A(\beta)\right) \times \exp\left(C(\mathbf{b}, \hat{w})\right) \tag{6.3}$$

or that the total free energy must decompose into a sum of two terms

$$F(\beta, \mathbf{b}, \hat{w}) \equiv -\frac{1}{\beta} \log \mathcal{Z}(\beta, \mathbf{b}, \hat{w}) = A(\beta) - \frac{1}{\beta} C(\mathbf{b}, \hat{w}). \tag{6.4}$$

The first term is a familiar thermodynamic free energy and, as we shall argue in the following section, the second term is related to a complexity of the neural networks. However, the free energy obtained from (5.14) (with the local objectives parameter $m$ set for simplicity to zero) is

$$F(\beta, \mathbf{b}, \hat{w}) = \frac{1}{2\beta} \sum_i \log\left(1 + \beta \lambda_i\right) - \frac{N}{2\beta} \log(2\pi) \tag{6.5}$$

which does not in general decompose into a sum of two terms as in (6.4). This suggest that in an equilibrium some additional restrictions must be imposed on the eigenvalues $\lambda_i$. One possibility (that we shall verify numerically) is that

$$\sum_{\lambda_i \gg \beta^{-1}} \log\left(1 + \beta \lambda_i\right) \approx \sum_{\lambda_i \gg \beta^{-1}} \log(\lambda_i) + N_> \log(\beta) \gg \sum_{\lambda_i \lesssim \beta^{-1}} \beta \lambda_i \approx \sum_{\lambda_i \lesssim \beta^{-1}} \log\left(1 + \beta \lambda_i\right)$$

where $N_>$ is the number of eigenvalues $\lambda_i$ that are much greater than $\beta^{-1}$. Then the free energy can indeed be decomposed as in equation (6.4),

$$F(\beta, \mathbf{b}, \hat{w}) \approx \frac{1}{2\beta} \sum_{\lambda_i \gg \beta^{-1}} \log(\lambda_i) + \frac{N_>}{2\beta} \log(\beta) - \frac{N}{2\beta} \log(2\pi) \tag{6.6}$$

with

$$C(\mathbf{b},\hat{w}) \approx -\frac{1}{2}\sum_{\lambda_i \gg \beta^{-1}} \log(\lambda_i) + \frac{N}{2}\log(2\pi) \tag{6.7}$$

and

$$A(\beta) \approx \frac{N_>}{2\beta}\log(\beta). \tag{6.8}$$

Recall that $\lambda_i$'s are the eigenvectors of $\hat{G} = \left(\hat{I} - \hat{f}'\hat{w}\right)^T \left(\hat{I} - \hat{f}'\hat{w}\right)$ and, thus, $\lambda_i$'s are functions of $\mathbf{b}$ and $\hat{w}$.

## 7 Thermodynamics of learning

The total Shannon entropy of the canonical ensemble can be obtained from the canonical partition function (5.14),

$$S(\beta,\mathbf{b},\hat{w}) = -\langle \log p\rangle = -\beta\frac{\partial}{\partial\beta}\mathcal{Z}(\beta,\mathbf{b},\hat{w}) + \mathcal{Z}(\beta,\mathbf{b},\hat{w}) = \beta^2\frac{\partial}{\partial\beta}F(\beta,\mathbf{b},\hat{w}). \tag{7.1}$$

Just like the free energy, in a learning equilibrium (6.2), the entropy must also decompose into a sum of two terms,

$$S(\beta,\mathbf{b},\hat{w}) = \beta^2\frac{\partial}{\partial\beta}\left(A(\beta) - \frac{1}{\beta}C(\mathbf{b},\hat{w})\right) = \beta^2\frac{\partial A(\beta)}{\partial\beta} + C(\mathbf{b},\hat{w}). \tag{7.2}$$

The first term depends on only the inverse temperature parameter $\beta$ and we shall refer to it as a thermodynamic entropy

$$S_0(\beta) = \beta^2\frac{\partial A(\beta)}{\partial\beta} = \beta(U(\beta) - A(\beta)). \tag{7.3}$$

For the total free energy (6.6) it is given by

$$\begin{aligned} S_0(\beta) &= -\beta A(\beta) + \beta U(\beta) \approx -\frac{N_>}{2}\log(\beta) + \frac{N_>}{2} \\ &\approx \frac{N_>}{2}\log(U) + \frac{N_>}{2}\left(1 - \log\frac{N_>}{2}\right) \end{aligned} \tag{7.4}$$

where

$$U(\beta) = -\frac{\partial}{\partial\beta}\log\mathcal{Z}(\beta,\mathbf{b},\hat{w}) = -\frac{\partial}{\partial\beta}\left(\beta F(\beta,\mathbf{b},\hat{w})\right) = \frac{\partial}{\partial\beta}\left(\beta A(\beta)\right) \approx \frac{N_>}{2\beta}. \tag{7.5}$$

As the learning progresses, the average loss, $U(\beta)$, decreases, the temperature parameter, $\beta^{-1}$, decreases and, thus, according to (7.4) one might expect that the thermodynamic entropy, $S_0$, should also decrease. However, it is not the thermodynamic entropy, $S_0$, but the total Shannon entropy $S$ (whose exponent describes accessible volume of the configuration space for $\mathbf{x}$) should become smaller with learning. We shall call it the second law of learning (or perhaps the minus second law):

**Second Law of Learning:** *the total entropy of a learning system can never increase during learning and is constant in a learning equilibrium,*

$$\frac{d}{dt}S \leq 0. \tag{7.6}$$

In the long run the system is expected to approach an equilibrium state with the smallest possible total entropy $S$ which corresponds to the lowest possible sum of the thermodynamic entropy, $S_0$, and of the complexity function $C(\mathbf{b}, \hat{w})$ that we shall discuss next.

In a feedforward neural network the weight matrix, $\hat{w}$, is nilpotent (2.1) and, therefore, the eigenvalues of the operator $\hat{w}$ are all zeros. This also implies that the eigenvalues of operator $\hat{I} - \hat{f}'\hat{w}$ are all ones, but that does not tell us much about the eigenvalues of $\hat{G}$. On the other hand, the determinant of $\hat{G}$ is simply related to the determinant of $\hat{I} - \hat{f}'\hat{w}$,

$$\det \hat{G} = \det \left(\hat{I} - \hat{f}'\hat{w}\right)^2 = 1, \tag{7.7}$$

or

$$\sum_i \log(\lambda_i) = 0. \tag{7.8}$$

If we assume that near equilibrium $N_>$ does not change significantly, then a decrease in $C(\mathbf{b}, \hat{w})$ implies that the largest eigenvalues $\sum_{\lambda_i \gg \beta^{-1}} \log(\lambda_i)$ of the operator $\hat{G}$ must increase and at the same time, according to (7.8), the smallest eigenvalues $\sum_{\lambda_i \lesssim \beta^{-1}} \log(\lambda_i)$ must decrease. Therefore, as the learning progresses, the operator $\hat{G}$ becomes better and better approximated by eigenvectors $\mathbf{v}_i$ with only largest eigenvalues, i.e.

$$\hat{G} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^T = \sum_{\lambda_i \gg \beta^{-1}} \lambda_i \mathbf{v}_i \mathbf{v}_i^T + \sum_{\lambda_i \lesssim \beta^{-1}} \lambda_i \mathbf{v}_i \mathbf{v}_i^T \approx \sum_{\lambda_i \gg \beta^{-1}} \lambda_i \mathbf{v}_i \mathbf{v}_i^T \tag{7.9}$$

This is what one might call a dynamical dimensional reduction of the state space (a subspace of dimension $N_> < N$ is sufficient to describe a state vector $\mathbf{x}$), or a reduction in the complexity of interconnections between neurons (a subspace of dimension $N_>^2 < N^2$ is sufficient to describe a weight matrix $\hat{w}$) or a complexity of computations that a given neural networks performs (a subspace of dimension $N_>^2 < N^2$ is sufficient to describe a linearized evolution operator $(\hat{I} - \hat{f}'\hat{w})$). For this reason we shall refer to $C(\mathbf{b}, \hat{w})$ as a measure of complexity or just complexity.

For a system transitioning between equilibrium states at constant temperature $T = 1/\beta$, variations of the free energy must vanish, $dF = 0$, and then equation (6.4) takes the from of the first law,

$$dA - TdC = dU - TdS_0 - TdC = 0. \tag{7.10}$$

or what we shall call the first law of learning (or perhaps the minus first law):

**First Law of Learning:** *the increment in the loss function is proportional to the increment in the thermodynamic entropy plus the increment in the complexity*

$$dU = TdS_0 + TdC. \tag{7.11}$$

This law describes how the learning system behaves when transitioning between equilibrium states, but in order to understand which neural architectures would be the most optimal we must take one step further and consider a non-equilibrium dynamics of the learning system.

## 8 Optimal architecture

Consider a family of bias vectors $\mathbf{b}(\mathbf{Q})$ and weight matrices $\hat{w}(\mathbf{Q})$ parametrized by dynamical parameters $Q_k$'s where $k \in (1, ..., K)$. Typically the number of parameters $K$ is much smaller than $N + N^2$ (i.e. the number of parameters required to describe a generic vector $\mathbf{b}$ and a generic matrix $\hat{w}$) and the art of designing a neural architecture is to come up with functions $\mathbf{b}(\mathbf{Q})$ and $\hat{w}(\mathbf{Q})$ which are most efficient in finding solutions. To make the statement more quantitative, consider an ensemble of neural networks described by a probability distribution $p(\beta, \mathbf{Q})$ which evolves with "time" $\beta$ according to

$$\frac{\partial}{\partial\beta}p(\beta,\mathbf{Q}) = -\sum_k \frac{dQ_k}{d\beta}\frac{\partial}{\partial Q_k}p(\beta,\mathbf{Q}) \tag{8.1}$$

where the parameters $Q_k$'s evolve in the direction which maximizes the free energy

$$\frac{dQ_k}{d\beta} = \alpha\frac{\partial F}{\partial Q_k} = \alpha\sum_i \frac{\partial b_i}{\partial Q_k}\frac{\partial F}{\partial b_i} + \alpha\sum_{i,j}\frac{dw_{ij}}{dQ_k}\frac{\partial F}{\partial w_{ij}}. \tag{8.2}$$

The Shannon entropy of the distribution $p(\beta, \mathbf{Q})$ with continuous variables $\mathbf{Q}$ (not to confuse with entropy $S(\beta, \mathbf{b}, \hat{w})$ defined in the previous section) is,

$$\mathcal{S}(\beta) = -\int d^K Q \;\; p(\beta,\mathbf{Q})\log\left(Mp(\beta,\mathbf{Q})\right) \tag{8.3}$$

where $M$ is a fixed normalization parameter. Large the entropy $\mathcal{S}(\beta)$, larger the accessible volume of the configuration space $\exp(\mathcal{S}(\beta))$, and therefore larger the rate with which new solutions for $\mathbf{b}$ and $\hat{w}$ can be found. Then an optimal architecture (describe by $\mathbf{b}(\mathbf{Q})$ and $\hat{w}(\mathbf{Q})$) is the one for which the entropy destruction is minimized or, equivalently, the entropy production is maximized. We shall call it the principle of the minimum entropy destruction:

**Principle of Minimum Entropy Destruction**: *The path taken by an optimal learning system is the one for which the entropy destruction is minimized (or the entropy production is maximized).*

Note that the principle is the opposite of the minimum entropy production principle [12, 13] that is often used in context of non-equilibrium thermodynamics, but is consistent with the stationary entropy production principle that was recently used in context of emergent quantum mechanics [14] and emergent gravity [15].

In context of the learning systems, a useful expression for the entropy production can be obtained from (8.1), (8.2) and (8.3),

$$\begin{aligned}
\frac{\partial}{\partial\beta}\mathcal{S}(\beta) &= -\frac{\partial}{\partial\beta}\int d^K Q \;\; p\log(Mp) \\
&= \int d^K Q \;\sum_k \frac{dQ_k}{d\beta}\frac{\partial p}{\partial Q_k}\log(Mp) = \alpha\int d^K Q \;\sum_k \frac{\partial F}{\partial Q_k}\frac{\partial p}{\partial Q_k}\log(Mp) \\
&= -\alpha\int d^K Q \;\sum_k \frac{\partial^2 F}{\partial Q_k^2}p\log(Mp) - \alpha\int d^K Q \;\sum_k \frac{\partial F}{\partial Q_k}\frac{\partial p}{\partial Q_k} \\
&= \alpha\int d^K Q \;\sum_k \frac{\partial^2 F}{\partial Q_k^2}p\left(1-\log(Mp)\right)
\end{aligned} \tag{8.4}$$

where we assumed that $p$ vanishes at the boundary and so the integrations by parts can be performed. If we choose the normalization parameter not to be too large $M \ll p^{-1} \sim 2^N$, then $-\log(Mp) \gg 1$ and the entropy production is

$$\frac{\partial}{\partial\beta}\mathcal{S}(\beta) = -\frac{\partial}{\partial\beta}\int d^K Q \;\; p\log(Mp) \approx -\alpha \int d^K Q \;\sum_k \frac{\partial^2 F}{\partial Q_k^2} p\log(Mp). \tag{8.5}$$

This integral equation can be rewritten as a local differential equation

$$\frac{\partial}{\partial\beta}\sigma(\beta,\mathbf{Q}) = \alpha\sum_k \frac{\partial^2 F}{\partial Q_k^2}\;\sigma(\beta,\mathbf{Q}) \tag{8.6}$$

for the entropy density

$$\sigma(\beta,\mathbf{Q}) = -p(\beta,\mathbf{Q})\log\left(Mp(\beta,\mathbf{Q})\right). \tag{8.7}$$

Its solution is given by an exponential

$$\sigma(\beta,\mathbf{Q}) \;=\; \sigma(0,\mathbf{Q})\exp\left(\beta\alpha\Delta F\right) \tag{8.8}$$

where $\sigma(0,\mathbf{Q})$ is determined from initial conditions at some fixed $\beta$ and the Laplacian operator is defined as usual $\Delta \equiv \sum_k \frac{\partial^2}{\partial Q_k^2}$.

To better understand the optimization condition we can choose the parameters $Q_i$'s to be given by eigenvalues of the operator $\hat{G}$, i.e. $Q_i = \lambda_i$. Then the free energy (5.14) (with $m$ set for simplicity to zero) can be approximated as

$$\begin{aligned} F \;&\approx\; \frac{1}{2\beta}\log\det\left(\hat{I}+\beta\hat{G}\right) \\ &\approx\; \frac{1}{2\beta}\sum_i \log\left(1+\beta\lambda_i\right) - \frac{N}{2\beta}\log(2\pi) \end{aligned} \tag{8.9}$$

and its Laplacian as

$$\begin{aligned} \Delta F \;&\approx\; \frac{1}{2\beta}\Delta\log\det\left(\hat{I}+\beta\hat{G}\right) \\ &\approx\; \frac{1}{2\beta}\sum_j \frac{\partial^2}{\partial\lambda_j^2}\sum_i \log\left(1+\beta\lambda_i\right) \\ &\approx\; -\frac{\beta}{2}\sum_i \left(1+\beta\lambda_i\right)^{-2}. \end{aligned} \tag{8.10}$$

Evidently, the Laplacian is always negative and, thus, the entropy density (8.11) must always decrease with learning,

$$\sigma(\beta,\mathbf{Q}) \;=\; \sigma(0,\mathbf{Q})\exp\left(-\frac{\alpha}{2}\sum_i \left(1+\beta\lambda_i\right)^{-2}\right). \tag{8.11}$$

Therefore, to improve learning efficiency we must choose an architecture such that the Laplacian is as close to zero as possible and the entropy destruction is minimized (i.e. the principle of minimum entropy destruction). If we, once again, split all of the eigenvalues into large and

small, then the Laplacian is approximately given by the number $N_<$ of smallest eigenvalues $\lambda_i \ll \beta^{-1}$, i.e.

$$\Delta F \approx -\frac{\beta}{2} \sum_{\lambda_i \ll \beta^{-1}} (1+\beta\lambda_i)^{-2} \approx -\frac{\beta}{2} N_<. \tag{8.12}$$

Note that while the largest eigenvalues $\lambda_i \gg \beta^{-1}$ are responsible for reducing complexity of the already obtained solutions (6.7), the smallest eigenvalues $\lambda_i \ll \beta^{-1}$ are responsible for searching for new solutions.

## 9 Deep vs. shallow

We are now ready to tackle one of the biggest mysteries of machine learning. Why do deep neural networks perform so well? We believe the answer is hidden in the free energy $F$. As we have argued in the previous section (8.11) the Laplacian $\Delta F$ describes the rate with which the entropy density $\sigma$ decays and by minimizing,

$$-\Delta F = \frac{\beta}{2} \sum_i (1+\beta\lambda_i)^{-2} = \frac{\beta}{2} Tr \left(\hat{I} + \beta \hat{G}\right)^{-2}, \tag{9.1}$$

we maximize the efficiently of a neural network to find solutions (i.e. the principle of minimum entropy destruction). To solve the minimization problem, we can consider a neural network with a small fraction $\gamma \ll 1$ of eigenvalues $\lambda_i \sim \lambda$ and a larger fraction $1-\gamma$ of large eigenvalues at $\lambda_i \sim \lambda^{\frac{\gamma}{\gamma-1}}$ so that

$$\det \hat{G} = \prod_i \lambda_i = \lambda^{\gamma N} \lambda^{\frac{\gamma}{\gamma-1}(1-\gamma)N} = 1. \tag{9.2}$$

Then the negative of the Laplacian is

$$-\Delta F \approx \frac{\beta}{2} Tr \left(\hat{I} + \beta \hat{G}\right)^{-2} = \frac{N\beta}{2} \left( \frac{\gamma}{(1+\beta\lambda)^2} + \frac{1-\gamma}{\left(1+\beta\lambda^{\frac{\gamma}{\gamma-1}}\right)^2} \right). \tag{9.3}$$

On Fig. 1 we plotted $-\Delta F(\log\lambda)$ for four different values of the inverse temperature parameter $\beta = 0.25, 0.50, 0.73, 1.00$, $\gamma = 1/3$ and $N = 854$. In the initial phase, $\beta < 1/2$, (e.g. blue line on Fig. 1) there is a stable local minimum at $\log\lambda \approx 3\log\left(-\frac{1}{2} + \sqrt{\frac{1}{4}+\frac{1}{\beta}}\right)$ and an unstable maximum at $\log\lambda = 0$. In this phase, a small number, $N\gamma \ll N$, of eigenvalues is free to move away from a local maximum at $\log\lambda = 0$ to both smaller and larger values, but most of the eigenvalues $N(1-\gamma) \sim N$ should remain near $\log\lambda \sim 0$. In the intermediate phase, $1/2 < \beta < \sqrt{N} - 1$, (e.g. green line on Fig. 1) the two extreme points switch and there an unstable maximum at

$$\log\lambda = 3\log\left(-\frac{1}{2} + \sqrt{\frac{1}{4}+\frac{1}{\beta}}\right). \tag{9.4}$$

In this phase only a decreasing fraction, $\gamma < (\beta+1)^{-2}$, of the small eigenvalues, $\log\lambda < 3\log\left(-\frac{1}{2} + \sqrt{\frac{1}{4}+\frac{1}{\beta}}\right)$, can still move to even smaller values, but the motion is terminated when the smallest values, $\log\lambda \ll 3\log\left(-\frac{1}{2} + \sqrt{\frac{1}{4}+\frac{1}{\beta}}\right)$, reach the plateau. However, since

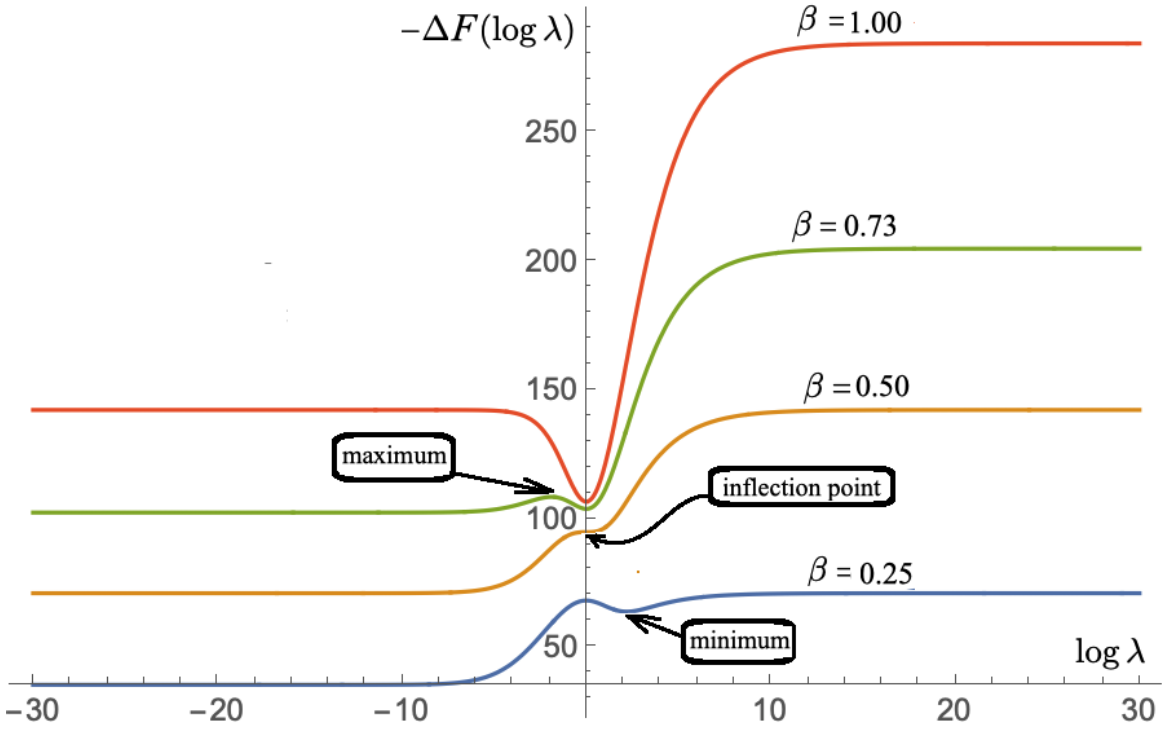


**Figure 1**. $-\Delta F(\log\lambda)$ for four different values of $\beta = 0.25, 0.50, 0.73$ and $1.00$.

$\det\hat{G} = 1$, it is expected that the sum of the largest eigenvalues would continue to grow and according to (6.7) the complexity of a network should continue to decrease. And finally, in the final phase, $\beta > \sqrt{N}-1$, (e.g. red line on Fig. 1) the global minimum is at $\log\lambda = 0$, the individual eigenvalues can no longer move towards $\log\lambda = -\infty$ and the ability of the neural network to learn becomes exponentially suppressed.

This is what happens in an optimal system, however, by enforcing an architecture on a neural network (e.g. deep or shallow) we impose additional constraints on the free energy (and on its Laplacian) which limits the ability of a network to explore the space of solutions. For example, in a feedforward neural network with many input neurons and few output/hidden neurons, most of the eigenvalues are set to $\log\lambda_i = 0$ and only a small fraction of eigenvalues is free to move to smaller and larger values. Clearly, the larger the number of the dynamical eigenvalues a neural architecture has, the better it is for learning. Therefore, in order to compare "apples to apples" we must first fix the number of the dynamical eigenvalues, and then look for an architecture which is flexible enough to support a skewed distribution of $\log\lambda_i$'s. As we have argued in the previous paragraph, want we want is to be able to start with a single peak with all eigenvalues at $\log\lambda_i \sim 0$ and then to gradually grow a second peak with the largest eigenvalues $\log\lambda_i \gtrsim 0 > -\log\beta$ which is to be balanced by the smallest eigenvalues $\log\lambda_i < -\log\beta$ that are dragged to smaller and smaller values. With this respect, a better architecture is the one which supports a larger variance

$$\mu_2 \equiv Tr\left(\log\hat{G}\right)^2 \tag{9.5}$$

and a more skewed distribution or a more negative

$$\mu_3 \equiv Tr\left(\log \hat{G}\right)^3. \tag{9.6}$$

In a feedforward neural network the weight matrix is nilpotent (2.1) as well as a product of the weight matrix, $\hat{w}$ and a diagonal matrix of first derivatives, $\hat{f}'$, i.e.

$$\left(\hat{f}'\hat{w}\right)^n = \left(\hat{w}^T\hat{f}'\right)^n = 0 \quad \forall n \geq L \tag{9.7}$$

where $L$ is the number of layers. For starters, consider a vary shallow network with no hidden layers (i.e. $L = 2$) and thus $\left(\hat{w}^T\hat{f}'\right)^2 = \left(\hat{f}'\hat{w}\right)^2 = 0$. Then there must exist functions $F_1(x)$ and $F_2(x)$ such that

$$\log \hat{G} = F_1(\hat{f}'\hat{w}\hat{w}^T\hat{f}')\,\hat{f}'\hat{w} + F_2(\hat{w}^T\hat{f}'\hat{f}'\hat{w})\hat{w}^T\hat{f}'. \tag{9.8}$$

and, therefore,

$$Tr\left[\log \hat{G}\right] = 0 \tag{9.9}$$

$$Tr\left[\left(\log \hat{G}\right)^2\right] = Tr\left[F_2(\hat{w}^T\hat{f}'\hat{f}'\hat{w})\hat{w}^T\hat{f}'F_1(\hat{f}'\hat{w}\hat{w}^T\hat{f}')\,\hat{f}'\hat{w} + F_1(\hat{f}'\hat{w}\hat{w}^T\hat{f}')\,\hat{f}'\hat{w}F_2(\hat{w}^T\hat{f}'\hat{f}'\hat{w})\hat{w}^T\hat{f}'\right]$$

and

$$\mu_3 = Tr\left[\left(\log \hat{G}\right)^3\right] = 0. \tag{9.10}$$

In fact the traces of all odd powers must also be zero

$$Tr\left[\left(\log \hat{G}\right)^{2n+1}\right] = 0 \tag{9.11}$$

since every term in $\left(\log \hat{G}\right)^{2n+1}$ would have a product of unequal number of $F_1(\hat{f}'\hat{w}\hat{w}^T\hat{f}')\,\hat{f}'\hat{w}$ and $F_2(\hat{w}^T\hat{f}'\hat{f}'\hat{w})\hat{w}^T\hat{f}'$ terms which must be traceless. As we shall see in the next section, even with a single hidden layer (i.e. $L = 3$) the second powers of operators $\hat{f}'\hat{w}$ and $\hat{w}^T\hat{f}'$ are very small and the skewness is still very small $\mu_3 \approx 0$. What this means is that the effective number of dynamical eigenvalues is half of what it would have been if all eigenvalues were free to move without having to respect the symmetry of the distribution. However, as we keep adding more hidden layers the skewness of distribution grows larger, the eigenvalues become less constrained and the efficiency of learning is greatly improved. This might be why the deep learning is so efficient: hidden layers are essential for larger skewness $\mu_3$ and, as a result, for less negative Laplacian $\Delta F$ (and a slower decay of the entropy density $\sigma$) which we claim is necessary for efficient learning.

## 10 Numerical experiments

A direct numerical calculation of the distribution $p(\mathbf{x})$ is a computationally intensive task, but the main advantage of our statistical description is that the canonical ensemble (4.10) can be viewed as purely phenomenological object. Then the main problem should be to come up with a model of the bulk loss function, $H(\mathbf{x}, \mathbf{b}, \hat{w})$, which best describes the canonical ensemble

and, consequently, the canonical partition function, $\mathcal{Z}(\beta, \mathbf{b}, \hat{w})$, and other thermodynamic quantities. On the other hand, the analysis of the preceding sections already suggests certain forms of the bulk loss function and of the partition function which we can easily verify numerically. In this section, we will check to what extent a feedforward neural network can be modeled by the bulk loss function without local objectives (i.e. (3.4) with $m = 0$) or with the corresponding thermodynamic quantities:

(a) average bulk loss (estimated in (7.5)),

$$U(\beta) = \frac{N_>}{2\beta}, \tag{10.1}$$

(b) complexity function (estimated in (6.7)),

$$C(\mathbf{b}, \hat{w}) = -\frac{1}{2} \sum_{\lambda_i \gg \beta^{-1}} \log(\lambda_i) + \frac{N}{2} \log(2\pi) + \text{const}, \tag{10.2}$$

(c) thermodynamic entropy (estimated in (7.4)),

$$S_0(\beta) = -\frac{N_>}{2} \log(\beta) + \text{const} = \frac{N_>}{2} \log(U) - \frac{N_>}{2} \log\left(\frac{N_>}{2}\right) + \text{const}, \tag{10.3}$$

where $N_>$ is the number of eigenvalues $\lambda_i$'s much larger than $\beta^{-1}$. In addition, we will verify the expected dynamics of the eigenvalues and the anticipated dependence of the variance (9.5) and skewness (9.6) parameters on the performance of the neural networks obtained in the previous sections.

All of the numerical experiments were carried out using the TensorFlow Python library [16] and MNIST database of handwritten images [17]. Unfortunately, in the TensorFlow library the hidden layers are not dynamical and must be set prior to training. Nevertheless, we were able to obtain the desired results by running two different programs: the first one with two hidden layers (or what we shall call a "deep" neural network) and one with a single hidden layer (or what we shall call a "shallow" neural network). In the deep network we used an input layer with 784 neurons, the first hidden layer with 40 neurons, the second hidden layer with 20 neurons and the output layer with 10 neurons; and in the shallow network we used the same number of neurons on the input and output layers (i.e. 784 and 10), but only a single hidden layer with 60 neurons. Altogether there are $N = 784 + 40 + 20 + 10 = 784 + 60 + 10 = 854$ neurons in each neural network and so the state vectors $\mathbf{x}$ and the bias vectors $\mathbf{b}$ are 854-dimensional vectors. The weight matrix $\hat{w}$ has $854 \times 854$ components $w_{ij}$, but most of them are zero due to the predetermined architecture of hidden layers. The input layers represent a handwritten image of a number from 0 to 9 which is passed to $28 \times 28 = 784$ input neurons. One of the 10 output neurons is to be activated only if the corresponding number is on the image. The activation function on all neurons is $f(y) = \tanh(y)$ and so the diagonal matrix $\hat{f}'$ has diagonal elements given by

$$f'_{ii}(y_i) = \frac{df(y_i)}{dy_i} = \frac{d\tanh(y_i)}{dy_i} = \text{sech}(y_i)^2 = 4(\exp(y_i) + \exp(-y_i))^{-2}. \tag{10.4}$$

The training was carried out using the method of stochastic gradient descent for $30,000$ epochs with $6,000$ samples in the training dataset.

On Fig. 2 we plot (log of the ensemble-averaged) bulk loss $U = \langle H \rangle$ (blue line) and

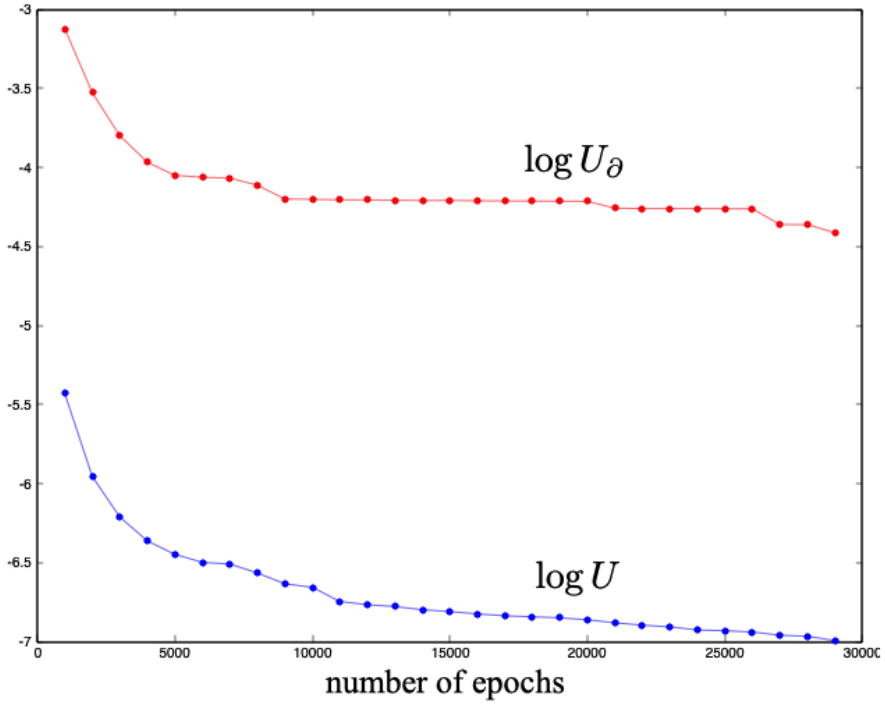


**Figure 2**. Bulk loss (blue line) and boundary loss (red line) for 30,000 training epochs.

(log of the ensemble-averaged) boundary loss $U_\partial = \langle H_\partial \rangle$ from the deep neural network. As expected, the bulk loss remains few orders in magnitude smaller than the boundary loss, but both functions decrease with time. For training the neural network we used the (more familiar, but less general) boundary loss function $H_\partial$, but a similar result is expected even if the (less familiar, but more general) bulk loss function $H$ would have been used instead. On Fig. 3 we plot the bulk loss $\log U$ vs. the boundary loss $\log U_\partial$ from the same deep network. Note that at late times the bulk loss keeps decreasing while the boundary loss remains almost constant (inside of red oval on Fig. 3). This behaviors continue for about 10,000 (!) training epochs until the network finally finds a better solution and the boundary loss jumps to a smaller value (inside of green oval on Fig. 3). And then essentially the same behavior continues, i.e. bulk loss decreases monotonically, but boundary loss makes sudden jumps. There is a simple explanation of the phenomena. The boundary loss is stuck in a saddle point with a large number of nearly flat directions for a very long time before it finds a way out. As the learning progress the system keeps moving along the flat directions and that does not reduce the boundary loss considerably, but the bulk loss and, as we shall see shortly, complexity keep decreasing with roughly the same pace. This shows that the bulk loss function has a lot fewer flat directions and with this respect a much better loss function. In addition, as we have argued in Sec. 3, it can be defined beyond supervised systems, e.g. for unsupervised learning.

On Figs. 4 and 5 we plot histograms of the dynamical eigenvalues of operator $\log \hat{G}$ from the, respectively, shallow and deep networks. As expected, most of the eigenvalues remain near origin and only a fraction of eigenvalues is displaced significantly from $\log \lambda_i \sim 0$. The distribution of the eigenvalues in the shallow network is almost completely symmetric, the

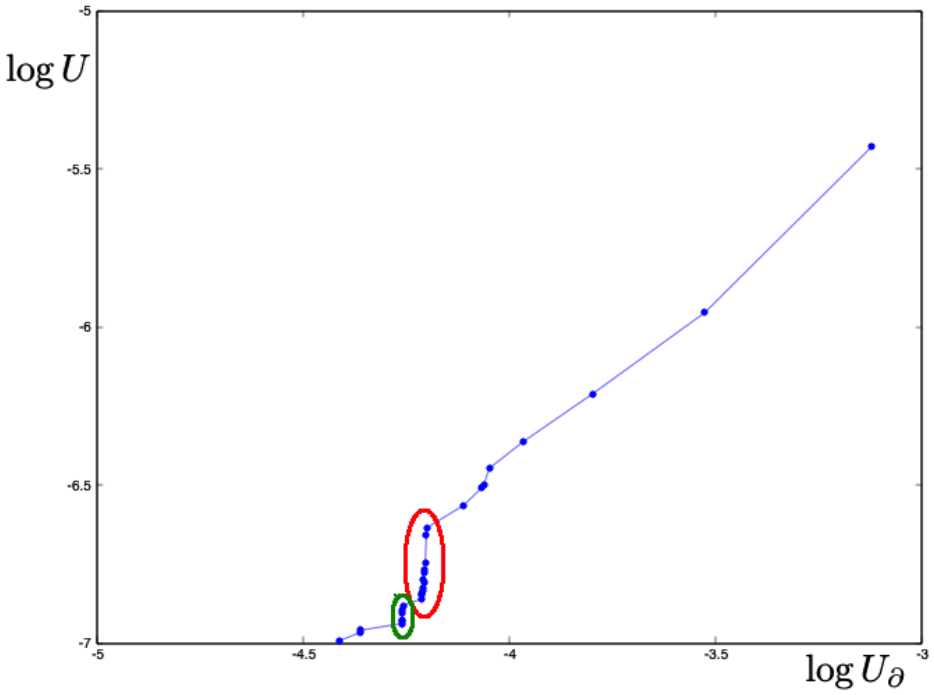


**Figure 3**. Bulk loss vs. boundary loss for 30,000 training epochs.

skewness after 10000 epochs is $\mu_3 \approx -2.7 \times 10^{-10}$, the learning efficiency is suppressed and, as a result, the variance remains small $\mu_2 \approx 1.84$. In contrast, the distribution of eigenvalues in the deep network is not symmetric, the skewness after 10000 epochs is more negative $\mu_3 \approx -1.7$, the learning efficiency is enhanced and the variance grows larger $\mu_2 \approx 2.44$. There is a clear gap between the smallest and larger eigenvalues (marked by red arrows on Fig. 5) which can be seen after 10000 epochs at $\log \lambda \approx -3.0$, after 1000 epochs at $\log \lambda \approx -2.5$ and may be even after 100 epochs at $\log \lambda \approx -2.3$. This gap is expected to be at unstable maximum defined by equation (9.4) which implies that after 100 epochs $\beta \approx 1.37$, after 1000 epochs $\beta \approx 1.40$ and after 10000 epochs $\beta \approx 1.47$. In the shallow network the gap cannot be clearly identified since it is closer to the origin and the inverse temperature parameter $\beta$ is smaller. Also note that while the smallest eigenvalues move to smaller values, to satisfy (7.8) the largest eigenvalues must move to larger values. Recall, that the largest eigenvalues describe the complexity of the network (10.2) and the increase of the largest eigenvalues represents a decrease in the complexity of the network.

In the previous paragraph, we estimated the values of $\beta$ by identifying a gap (marked by red arrows on Figs. 5) between the smallest eigenvalues and the rest. However, as one can see from Fig. 5 the smallest eigenvalues $\log \lambda_i < -\log \beta$ keep moving to smaller values together with $-\log \beta$. This suggests that (instead of using equation (10.2)) we can try to define an approximate complexity by a sum of a fixed number of the largest eigenvalues,

$$C_n(\mathbf{b}, \hat{w}) = -\frac{1}{2} \sum_{i=1}^{n} \log(\lambda_i) + \frac{N}{2} \log(2\pi), \tag{10.5}$$

where it is assumed that the eigenvalues are ordered $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N$. On Fig. 6 we plot

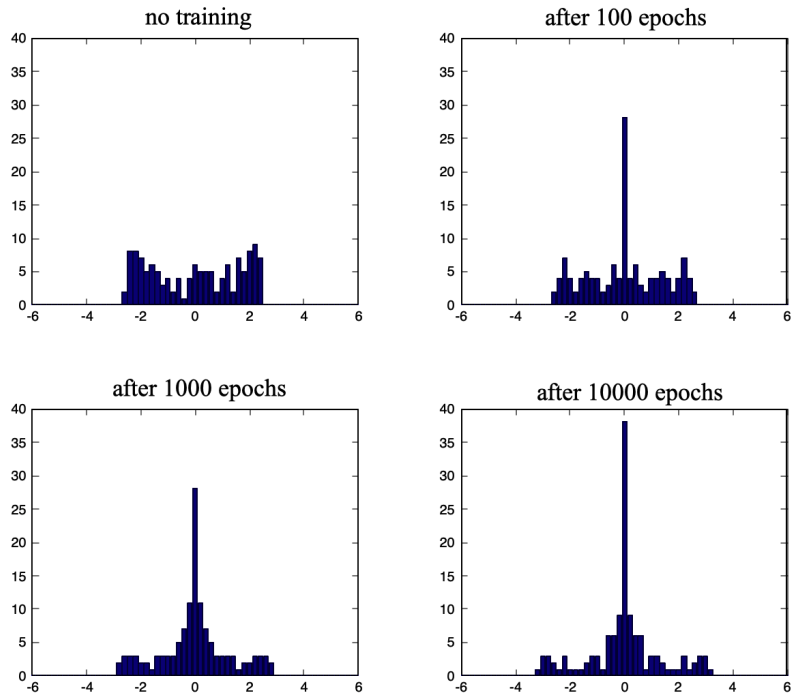


**Figure 4**. Histogram of eigenvalues of operator $\log \hat{G}$ from a shallow network.

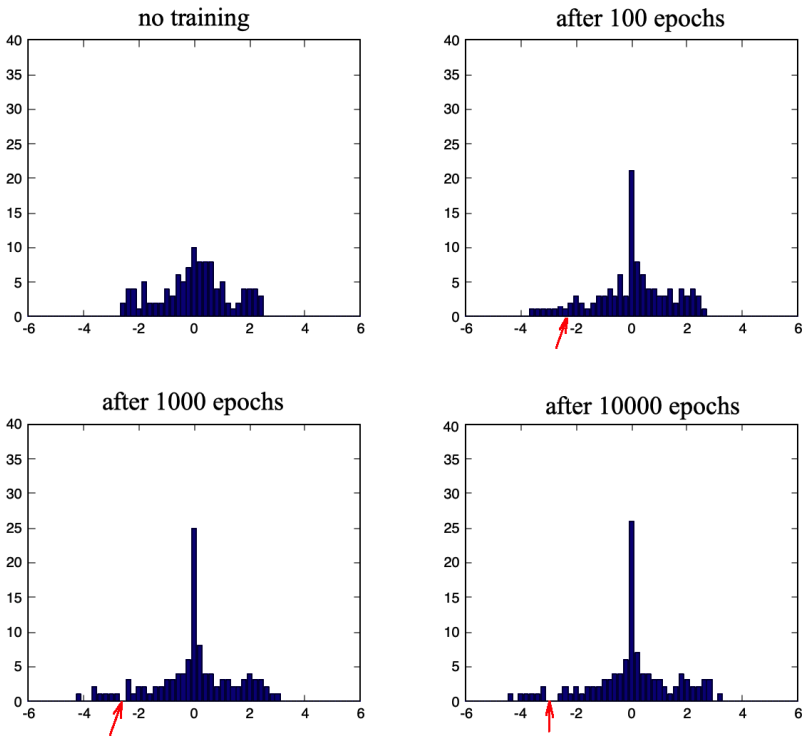


**Figure 5**. Histogram of eigenvalues of operator $\log \hat{G}$ from a deep network.

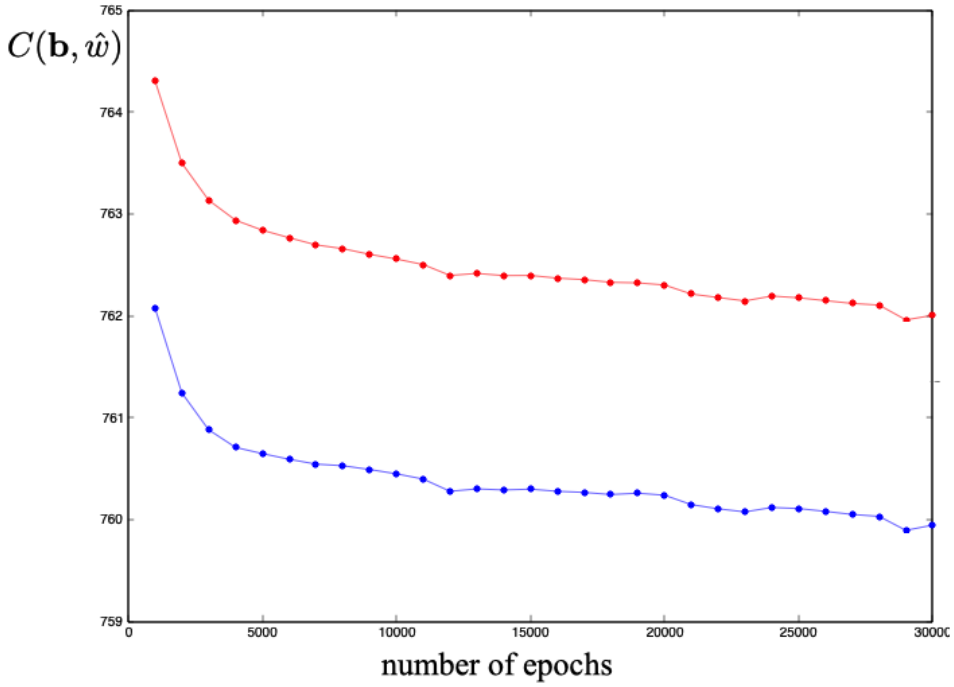


**Figure 6**. Complexity $C(\mathbf{b}, \hat{w})$ of a deep neural network as a function of the number of training epochs.

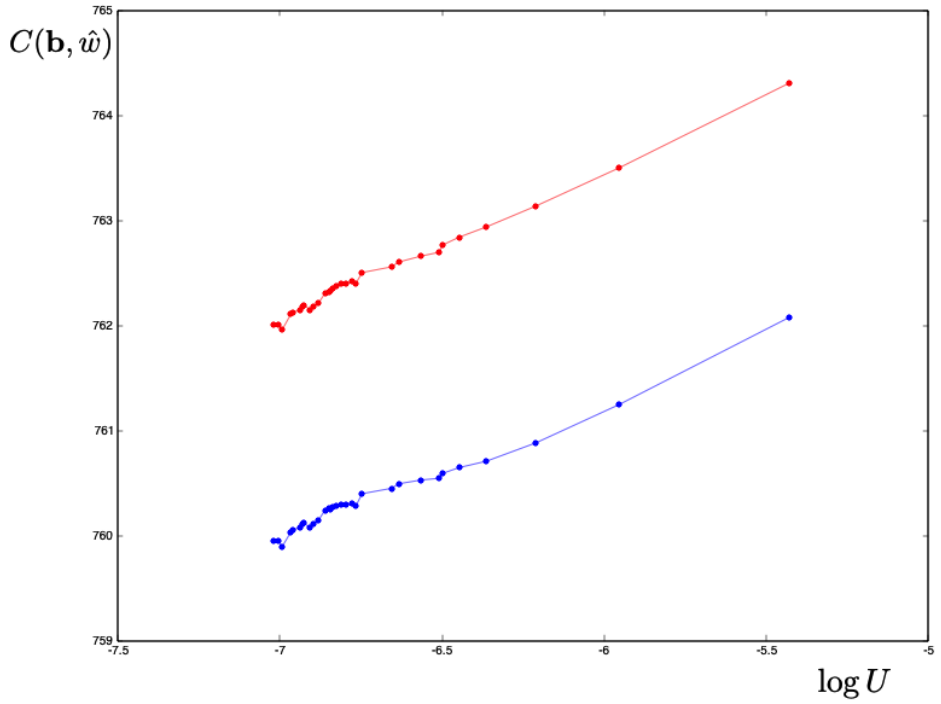


**Figure 7**. Complexity $C(\mathbf{b}, \hat{w})$ of a deep neural network as a function of the bulk loss $\log U$.

two (limiting) complexities by summing over twenty largest eigenvalues, $C_{20}(\mathbf{b}, \hat{w})$, (blue line) and by summing over all but twenty smallest eigenvalues, $C_{834}(\mathbf{b}, \hat{w})$, (red line). Evidently, up to an additive constant, the behavior of both curves is similar and so either one (or anyone in-between) can be used to study a relaxation of the system towards equilibrium. On Fig. 7 we plot the same complexities, but as a function of the bulk loss $\log U$. Both functions are nearly linear with slopes of order one: $C_{20}(\mathbf{b}, \hat{w}) \approx 769 + 1.25 \log U$ for the blue line and $C_{834}(\mathbf{b}, \hat{w}) \approx 772 + 1.41 \log U$ for the red line. In fact the linear dependance is in agreement with the second law of learning (7.6) which states that the total entropy must decay with learning. Recall the total entropy is a sum of the complexity (10.2) and thermodynamic entropy (10.3) (which scales linearly with $\log U$ whenever $N_>$ remains constant) and so it is expected that both quantities would decay with roughly the same rate.

There are certainly many other numerical experiments that we could have done, but it should already be evident that the statistical description developed in the paper might actually shed light on what is happening behind scenes in machine learning. We now switch to the most speculative part of the paper by asking if the entire universe on its most fundamental level could be described by a neural network.

## 11 Entropic mechanics

Quantum mechanics is a remarkably successful paradigm for modeling physical phenomena observed on a wide range of scales ranging from $10^{-19}$ meters (i.e. high-energy experiments) to $10^{+26}$ meters (i.e. cosmological observations.) The paradigm is so successful that it is widely believed that on the most fundamental level the entire universe is governed by the rules of quantum mechanics and even gravity should somehow emerge from it. This is known as the problem of quantum gravity that so far has not been solved, but some progress has been made in context of AdS/CFT correspondence [21–23], emergent gravity [15, 24, 25], quantum entanglement [26–28] and holographic complexity [18–20].[4] Although extremely important, the problem of quantum gravity is not the only problem with quantum mechanics. The quantum framework also starts to fall apart with introduction of observers. Everything seems to work very well when observers are kept outside of a quantum system, but it is far less clear how to described macroscopic observers in a quantum system such as the universe itself. The realization of the problem triggered an ongoing debate on the interpretations of quantum mechanics, which remains unsettled to this day. On one side of the debate, there is an increasing number of proponents of the many-worlds interpretation claiming that everything

[4] There seems to be an interesting connection between thermodynamics of learning systems (see Sec. 7) and thermodynamics of holographic complexity. In Ref. [19] the authors showed that the quantum computational complexity of a holographic states on the anti-de Sitter boundary is dual to an action over a Wheeler-de Witt patch in the bulk. In the learning systems, the complexity function $C(\mathbf{b}, \hat{w})$ is the quantity which best describes the complexity of a boundary state (for example, in a feedforward network $C(\mathbf{b}, \hat{w})$ is the complexity of a neural network which maps the input boundary data to output boundary data with the smallest error) and a thermodynamic free energy $A(\beta)$ is the quantity which best describes the state of the local degrees of freedom in the bulk. According to the First Law of learning (7.11) the two quantities are in fact related if not in an absolute sense, then at least in a relative sense, i.e.

$$dA = dU - TdS_0 = TdC. \tag{11.1}$$

Also note that even the Second Law of learning (7.6) is somewhat related to the recently proposed Second Law of complexity of quantum states [20] with the main difference that during learning the complexity $C$ decreases not on its own, but together with the thermodynamic entropy $S_0$. This suggests that there might be a deep connection between learning systems and holographic systems that we still have not figured out.

in the universe (including observers) must be governed by the Schrödinger equation [29], but then it is not clear how classical probabilities would emerge. One the other side of the debate, there are proponents of the hidden variables theories [30], but there it is also unclear what is the role of the wave-function in a purely statistical system. It is important to emphasize that a working definition of observers is necessary not only for settling some philosophical debates, but for understanding the results of real physical experiments and cosmological observations. In particular, a self-content, paradoxes-free definition of observers would allow us to understand the significance of Bell's inequalities [31] and to make probabilistic prediction in cosmology [32].

To resolve the apparent inconsistency (or incompleteness) in our description of the physical world, we shall entertain a (not so new) idea of having a more fundamental theory than quantum mechanics. A working hypothesis is that on the most fundamental level the dynamics of the entire universe is described by a microscopic neural network. If correct, then not only macroscopic observers should emerge from the microscopic neural network (see, for example, [33]), but, more importantly, the equations of quantum mechanics and general relativity should correctly describe an emergent dynamics of the corresponding learning system. Our main goal in this section is to show that quantum mechanics (or, more precisely, Schrödinger equation) indeed provides a good description of an optimal neural network not too far from an equilibrium and we postpone the discussion of general relativity until the next section. Note that most of the results in the remainder of this section were originally obtained in Ref. [14], but in a slightly different context and with slightly different assumptions.

Recall that equation (8.1) describes evolution of a probability distribution $p(\beta, \mathbf{Q})$ where $Q_k$'s (for $k \in (1, ..., K)$) parametrize the weight matrix $\hat{w}(\mathbf{Q})$ and the bias vector $\mathbf{b}(\mathbf{Q})$. The equation works well away from an equilibrium, but at an equilibrium the first derivatives of the free energy vanish

$$\frac{dQ_k}{d\beta} = \alpha \frac{\partial F}{\partial Q_k} \sim 0 \tag{11.2}$$

and the dominant contribution to the entropy production comes from "diffusion". Then we can study evolution of the system along the equilibrium manifold of dimension $K - \tilde{K} \ll K$ (i.e. the number of "Goldstone" modes is $K - \tilde{K}$) defined by a (degenerate) maximum of the free energy $F(\mathbf{Q})$. More formally, the equilibrium manifold can be defined by a set of equations

$$\Theta_{\tilde{k}}(\mathbf{Q}) = 0 \tag{11.3}$$

where $\tilde{k} \in (1, ..., \tilde{K})$. These equations are satisfied only along the maxima of the free energy, but in our statistical description we shall only insist that they are satisfied on average, i.e.

$$\int d^K Q \; p(t, \mathbf{Q}) \Theta_{\tilde{k}}(\mathbf{Q}) = 0. \tag{11.4}$$

Note that instead of studying the dynamics in $\beta$ we switched to a new parameter $t$ (e.g. the number of training epochs) for which the dynamics can be described by a Fokker-Planck equation

$$\frac{\partial p(t, \mathbf{Q})}{\partial t} = \frac{D}{2} \sum_k \frac{\partial^2 p(t, \mathbf{Q})}{\partial Q_k^2} = \frac{D}{2} \Delta p(t, \mathbf{Q}) \tag{11.5}$$

with a time-independent diffusions coefficient $D$. For simplicity, we also assume that the diffusion coefficient $D$ does not depend on $\mathbf{Q}$ and so no factor ordering problems arise. Then the main problem is to solve for $p(t, \mathbf{Q})$ subject to constraints (11.4) which is exactly the

type of problems considered in Ref. [14]. There it was shown that the constrained dynamics can be described by an approximate Schrödinger equation, with Lagrange multipliers playing the role of phases, if the number of the constraints is large (i.e. $\tilde{K} \sim K$) and the so-called principle of stationary entropy production is satisfied:

**Principle of Stationary Entropy Production**: *The path taken by the system is the one for which the entropy production is stationary.*

However, in Sec. 8 we argued that in an optimal architecture a closely related principle (of minimum entropy destruction) should be satisfied and, therefore, all that we need to assume is that the microscopic neural network has an optimal architecture.

The optimization problem can then be solved by combining into a single functional $\mathcal{S}(p, \mathbf{\Phi})$ two terms: the total entropy production from time $t = 0$ to time $t = T$ and constraints (11.4) imposed by time-dependent Lagrange multipliers $\Phi_{\tilde{k}}(t)$'s, i.e.

$$\begin{aligned}\mathcal{S}(p, \mathbf{\Phi}) &= -\int_0^T dt \frac{d}{dt} \int d^K Q \;\; p(t, \mathbf{Q}) \log(p(t, \mathbf{Q})) + \int_0^T dt \int d^K Q \sum_{\tilde{k}} \frac{d\Phi_{\tilde{k}}(t)}{dt} \Theta_{\tilde{k}}(\mathbf{Q}) p(t, \mathbf{Q}) \\ &= \int_0^T dt \int d^K Q \;\; \left( -\log(p(t, \mathbf{Q})) \frac{D}{2} \sum_k \frac{\partial^2 p(t, \mathbf{Q})}{\partial Q_k^2} + \sum_{\tilde{k}} \frac{d\Phi_{\tilde{k}}(t)}{dt} \Theta_{\tilde{k}}(\mathbf{Q}) p(t, \mathbf{Q}) \right). \end{aligned} \tag{11.6}$$

After integrating by parts and ignoring the boundary terms (assuming that $p$ vanishes at the boundaries of integration) we obtain

$$\begin{aligned}\mathcal{S}(p, \mathbf{\Phi}) &= \int_0^T dt \int d^K Q \;\; \left( 2D \sum_k \left( \frac{\partial \sqrt{p(t, \mathbf{Q})}}{\partial Q_k} \right)^2 + \sum_{\tilde{k}} \frac{d\Phi_{\tilde{k}}(t)}{dt} \Theta_{\tilde{k}}(\mathbf{Q}) p(t, \mathbf{Q}) \right) \\ &= \int_0^T dt \int d^K Q \;\; \sqrt{p(t, \mathbf{Q})} \left( -2D \sum_k \frac{\partial^2}{\partial Q_k^2} + \sum_{\tilde{k}} \frac{d\Phi_{\tilde{k}}(t)}{dt} \Theta_{\tilde{k}}(\mathbf{Q}) \right) \sqrt{p(t, \mathbf{Q})} \\ &= -4 \int_0^T dt \int d^K Q \;\; \Psi^*(t, \mathbf{Q}) \left( \frac{D}{2} \Delta + i \frac{d}{dt} \right) \Psi(t, \mathbf{Q}) \end{aligned} \tag{11.7}$$

where the wave-function is defined as

$$\Psi(t, \mathbf{Q}) \equiv \sqrt{p(t, \mathbf{Q})} \exp\left( i \frac{1}{4} \sum_{\tilde{k}} \Theta_{\tilde{k}}(\mathbf{Q}) \Phi_{\tilde{k}}(t) \right). \tag{11.8}$$

Evidently, upon varying (11.7) we arrive at a Schrödinger equation

$$-i \frac{d}{dt} \Psi(t, \mathbf{Q}) = \frac{D}{2} \Delta \Psi(t, \mathbf{Q}) \tag{11.9}$$

whose solutions extremize the functional $\mathcal{S}(p, \mathbf{\Phi})$ or, in other words, describe a trajectory in the configuration space which minimizes entropy destruction. (See Ref. [14] for details). Therefore, we conclude that quantum mechanics (or at least Schrödinger equation) can in fact emerge from a microscopic neural network with an optimal architecture near equilibrium.

## 12 Emergent gravity

Now we turn to gravity.[5] If the microscopic neural network has an optimal architecture then it is still the case that the principle of minimum entropy destruction (or, the more general, principle of stationary entropy production) should be satisfied and so the relevant quantity to extremize is still (11.6), which contains both entropy production (first term) and constraints (second term). What is, however, different is that we must allow for the larger system to be further away from a learning equilibrium and so the number of constraints $\tilde{K}$ can be much smaller than the number of parameters $K$. In other words, the dimensionality of the equilibrium manifold (or, if you wish, the number of symmetries) remains very high. This implies that the probability distribution $p(t, \mathbf{Q})$ should have a higher degree of symmetry and thus can be parametrized $p(\hat{g}, \mathbf{Q})$ with a (relatively) small number of auxiliary parameters $\hat{g}(t)$. For example, if the probability distribution is parametrized by Gaussian distributions, $p(\hat{g}(t), \mathbf{Q}) \propto \exp\left(-\sum_{k,k'} g_{kk'}(t) Q_k Q_{k'}\right)$, then the optimization problem is to find $\hat{g}(t)$ and $\mathbf{\Phi}(t)$ which extremize (11.6),

$$\mathcal{S}(\hat{g}, \mathbf{\Phi}) = \int_0^T dt \sum_k \left\langle -\frac{D}{2} \frac{\partial^2 \log p(\hat{g}, \mathbf{Q})}{\partial Q_k^2} \right\rangle + \int_0^T dt \sum_{\tilde{k}} \left\langle \frac{d\Phi_{\tilde{k}}}{dt} \Theta_{\tilde{k}}(\mathbf{Q}) \right\rangle . \tag{12.1}$$

The first term represents the entropy production and depends only on $\hat{g}$ and the second term represents constraints and depends on both $\hat{g}$ and Lagrange multipliers $\mathbf{\Phi}$.

This is what might be happening on a microscopic level, but our task in this section is to only develop a phenomenological model gravity based on what we already know about general relativity. In gravitational theories the dynamical degrees of freedom are described by a metric tensor $g_{\mu\nu}(x)$ and other fields $\Phi(x)$ all of which are functions of four space-time coordinates $x = (x^0, x^1, x^2, x^3)$. From that perspective a better parametrization of the probability distribution is given by $g_{\mu\nu}(x)$ and of the Lagrange multipliers by $\Phi(x)$. Then (12.1) can be expressed phenomenologically as

$$\mathcal{S}(g_{\mu\nu}, \Phi) = \int d^{D+1}x \, \sqrt{|g|} \, \left( -\frac{1}{2\kappa} R(g_{\mu\nu}(x)) + \Lambda \right) + \int d^{D+1}x \, \sqrt{|g|} \, \mathcal{L}(g_{\mu\nu}(x), \Phi(x)) \tag{12.2}$$

where, as before, the first term represents the entropy production and the second term represents the constraints. Several comments are in order. First of all, in equation (12.1) the parameter $\hat{g}$ was a finite dimensional matrix, but in equation (12.2) the parameter $g_{\mu\nu}(x)$ is a continuous function and so at best it is an approximate mapping which should break down at some UV scale (e.g. Planck scale). Secondly, even if the metric tensor $g_{\mu\nu}(x)$ is defined only on some very fine-grained lattice, there is a sense of distance between gravitational degrees of freedom which is not present in a neural network. This would be true for a general learning system, but we expect that for a clever choice of local objectives the weight matrix $\hat{w}$ (which is also an adjacency matrix describing the strength of connections between neurons) could be attracted towards a three-dimensional lattice (see [35] for a possible mechanism) and then the space-time locality would emerge. Thirdly, any lattice-like structure would break a general covariance which is known to be a very precise symmetry of nature. Therefore, we must

[5] As far as we know the only attempt to describe gravity in terms of quantum neural networks was made in Ref. [34]. However, the main difference with our approach is that the microscopic neural network considered here is not quantum, but statistical. On the other hand, as we have argued in Sec. 11, the quantum behavior of the microscopic neural network is expected and so it is possible that the two systems are equivalent.

also assume that the local objectives of neurons are such that the general covariance would emerge on large scales (see [36] for a possible mechanism), but exactly how this might work is presently unknown.

In the remainder of this section we shall follow closely a phenomenological procedure outline in Ref. [15]. We first expand the entropy production term in (12.2) around equilibrium, i.e.

$$\frac{1}{2\kappa}R = g_{\alpha\beta,\mu}J^{\mu\alpha\beta} \tag{12.3}$$

where the fluxes are denoted by $J^{k\alpha\beta}$ and the generalized forces are taken to be[6]

$$g_{\alpha\beta,\mu} \equiv \frac{\partial g_{\alpha\beta}}{\partial x^\mu}. \tag{12.4}$$

Then we can expand fluxes around local equilibrium to the linear order in generalized forces

$$J^{\mu\alpha\beta} = L^{\mu\nu\ \alpha\beta\ \gamma\delta}g_{\gamma\delta,\nu}. \tag{12.5}$$

to obtain

$$\frac{1}{2\kappa}R = L^{\mu\nu\ \alpha\beta\ \gamma\delta}g_{\alpha\beta,\mu}g_{\gamma\delta,\nu}. \tag{12.6}$$

One can think of (12.6) as a defining equation for the Onsager tensor, but then we are forced to only consider Onsager tensors $L^{\mu\nu\ \alpha\beta\ \gamma\delta}$ that are symmetric under interchanges $(\mu,\alpha,\beta) \leftrightarrow (\nu,\gamma,\delta)$, i.e.

$$L^{\mu\nu\ \alpha\beta\ \gamma\delta} = L^{\nu\mu\ \beta\alpha\ \delta\gamma}. \tag{12.7}$$

These are the Onsager reciprocity relations [37] for our learning system, but there are also other (trivial) symmetries that one should impose $(\alpha) \leftrightarrow (\beta)$, $(\gamma) \leftrightarrow (\delta)$, due to symmetries of the metric, i.e.

$$L^{\mu\nu\ \alpha\beta\ \gamma\delta} = L^{\mu\nu\ (\alpha\beta)\ (\gamma\delta)}. \tag{12.8}$$

The overall space of such tensors is still pretty large, but it turns out that a very simple choice leads to general relativity:

$$L^{\mu\nu\ \alpha\beta\ \gamma\delta} = \frac{1}{8\kappa}\left(g^{\alpha\nu}g^{\beta\delta}g^{\mu\gamma} + g^{\alpha\gamma}g^{\beta\nu}g^{\mu\delta} - g^{\alpha\gamma}g^{\beta\delta}g^{\mu\nu} - g^{\alpha\beta}g^{\gamma\delta}g^{\mu\nu}\right). \tag{12.9}$$

After integrating by parts, neglecting boundary terms and collecting all other terms we get

$$\begin{aligned}\int d^{D+1}x\ \sqrt{|g|}\ \frac{1}{2\kappa}R &= \int d^{D+1}x\sqrt{|g|}g^{\mu\nu}\ \ \frac{1}{\kappa}\left(\Gamma^\alpha{}_{\nu[\mu,\alpha]} + \Gamma^\beta{}_{\nu[\mu}\Gamma^\alpha{}_{\alpha]\beta}\right) = \\ &= \int d^{D+1}x\sqrt{|g|}\ \ \frac{1}{8\kappa}\left(g^{\alpha\nu}g^{\beta\delta}g^{\mu\gamma} + g^{\alpha\gamma}g^{\beta\nu}g^{\mu\delta} - g^{\alpha\gamma}g^{\beta\delta}g^{\mu\nu} - g^{\alpha\beta}g^{\gamma\delta}g^{\mu\nu}\right)g_{\alpha\beta,\mu}g_{\gamma\delta,\nu}\end{aligned} \tag{12.10}$$

where

$$\Gamma^\mu{}_{\gamma\delta} \equiv \frac{1}{2}g^{\mu\nu}\left(g_{\nu\gamma,\delta} + g_{\nu\delta,\gamma} - g_{\gamma\delta,\nu}\right) \qquad \text{and} \qquad \Gamma^\alpha{}_{\mu\nu,\beta} \equiv \frac{\partial}{\partial x^\beta}\Gamma^\alpha{}_{\mu\nu}. \tag{12.11}$$

Therefore, upon varying (12.2) with respect to the metric $g^{\mu\nu}$ we get the Einstein equations

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} + \Lambda g_{\mu\nu} = \kappa T_{\mu\nu} \tag{12.12}$$

[6] Summations over repeated indices are implied everywhere in this section.

where the Ricci tensor is

$$R_{\mu\nu} \equiv 2\left(\Gamma^{\alpha}{}_{\nu[\mu,\alpha]} + \Gamma^{\beta}{}_{\nu[\mu}\Gamma^{\alpha}{}_{\alpha]\beta}\right) \tag{12.13}$$

and the energy-momentum tensor is

$$T_{\mu\nu} \equiv -\frac{2}{\sqrt{|g|}}\frac{\delta(\sqrt{|g|}\mathcal{L})}{\delta g^{\mu\nu}}. \tag{12.14}$$

(See Ref. [15] for details). Of course, the expectations are that this result would only hold near equilibrium, and there should be deviations from general relativity when some of the symmetries in the Onsager tensor (12.9) are broken.

*Acknowledgments.* I would like to express my sincere gratitude to my former teacher, Walter Johnson, who introduced me to the subject of artificial neural networks. This work was supported in part by the Foundational Questions Institute (FQXi).

## References

[1] A. M. Saxe, J. L. McClelland, S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," In the International Conference on Learning Representations, (2014)

[2] A. Choromanska, M. Henaff, M. Mathieu, G. Arous, Y. LeCun, "The Loss Surfaces of Multilayer Networks," In Proceedings of the 18th International Conference on Artificial Intelligence, volume 38, (2015)

[3] J. Kadmon, H. Sompolinsky, "Optimal Architectures in a Solvable Model of Deep Networks," In Advances in Neural Information Processing Systems, (2016)

[4] R. Shwartz-Ziv, N. Tishby, "Opening the black box of deep neural networks via information," arXiv:1703.00810 [cs.LG], (2017)

[5] M. S. Advani, A. M. Saxe. "High-dimensional dynamics of generalization error in neural networks," arXiv preprint arXiv:1710.03667 (2017)

[6] H. W. Lin, M. Tegmark, D. Rolnick, "Why Does Deep and Cheap Learning Work So Well?", Journal of Statistical Physics, Volume 168, Issue 6, pp.1223-1247 (2017)

[7] A.I. Galushkin, "Neural Networks Theory," Springer, 396 p., (2007)

[8] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," Neural Networks. 61: 85-117. (2015)

[9] Haykin, Simon S. "Neural Networks: A Comprehensive Foundation," Prentice Hall. (1999)

[10] E. T. Jaynes, "Information Theory and Statistical Mechanics," Physical Review. Series II. 106 (4): 620-630, (1957)

[11] E. T. Jaynes, "Information Theory and Statistical Mechanics II," Physical Review. Series II. 108 (2): 171-190, (1957)

[12] Prigogine, I. "Etude Thermodynamique des phénoménes irréversibles". Desoer, Liége, (1947)

[13] M. J. Klein, P. H. E. Meijer, "Principle of minimum entropy production." Phys. Rev. 96: 250-255, (1954)

[14] V. Vanchurin, "Entropic Mechanics: towards a stochastic description of quantum mechanics," Found. Phys. **50**, no. 1, 40 (2019)

[15] V. Vanchurin, "Covariant Information Theory and Emergent Gravity," Int. J. Mod. Phys. A **33**, no. 34, 1845019 (2018)

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., "Tensorflow: A system for large- scale machine learning," in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265-283.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 86(11):2278-2324, 1998.

[18] A. R. Brown, D. A. Roberts, L. Susskind, B. Swingle and Y. Zhao, "Holographic Complexity Equals Bulk Action?," Phys. Rev. Lett. **116**, no. 19, 191301 (2016)

[19] A. R. Brown, D. A. Roberts, L. Susskind, B. Swingle and Y. Zhao, "Complexity, action, and black holes," Phys. Rev. D **93**, no. 8, 086006 (2016)

[20] A. R. Brown and L. Susskind, "Second law of quantum complexity," Phys. Rev. D **97**, no. 8, 086015 (2018)

[21] J. M. Maldacena, "The Large N limit of superconformal field theories and supergravity," Int. J. Theor. Phys. **38**, 1113 (1999)

[22] E. Witten, "Anti-de Sitter space and holography," Adv. Theor. Math. Phys. **2**, 253 (1998)

[23] L. Susskind, "The World as a hologram," J. Math. Phys. **36**, 6377 (1995)

[24] T. Jacobson, "Thermodynamics of space-time: The Einstein equation of state," Phys. Rev. Lett. **75**, 1260 (1995)

[25] E. P. Verlinde, "On the Origin of Gravity and the Laws of Newton," JHEP **1104**, 029 (2011)

[26] S. Ryu and T. Takayanagi, "Holographic derivation of entanglement entropy from AdS/CFT," Phys. Rev. Lett. **96**, 181602 (2006)

[27] B. Swingle, "Entanglement Renormalization and Holography," Phys. Rev. D **86**, 065007 (2012)

[28] A. Almheiri, X. Dong and D. Harlow, "Bulk Locality and Quantum Error Correction in AdS/CFT," JHEP **1504**, 163 (2015)

[29] H. Everett, "Relative State Formulation of Quantum Mechanics," Reviews of Modern Physics. 29 (3): 454-462, (1957)

[30] D. Bohm, "A Suggested Interpretation of the Quantum Theory in Terms of 'Hidden Variables' I," Physical Review. 85 (2): 166-179, (1952)

[31] J. Bell, "On the Einstein Podolsky Rosen Paradox," Physics. 1 (3): 195-200, (1964)

[32] V. Vanchurin, A. Vilenkin and S. Winitzki, "Predictability crisis in inflationary cosmology and its resolution," Phys. Rev. D **61**, 083507 (2000)

[33] G. Tononi, "Consciousness as integrated information: a provisional manifesto," Biol Bull 215: 216-242, (2008)

[34] G. Dvali, "Black Holes as Brains: Neural Networks with Area Law Entropy," Fortsch. Phys. **66**, no. 4, 1800007 (2018)

[35] V. Vanchurin, "Information Graph Flow: a geometric approximation of quantum and statistical systems," Found. Phys. **48**, no. 6, 636 (2018)

[36] V. Vanchurin, "A quantum-classical duality and emergent space-time," arXiv:1903.06083 [hep-th].

[37] Onsager, L. "Reciprocal relations in irreversible processes, I". Physical Review. 37 (4) 405-426 (1931)