Deep learning with stochastic neural networks

It has been recently shown that new insights on deep learning can be obtained by regarding the process of training a deep neural network as a discretization of an optimal control problem involving nonlinear differential equations [5, 4, 8]. One attractive feature of this formulation is that it allows to use tools from dynamical system theory to study deep learning from a rigorous mathematical perspective [12, 9, 14]. For instance, it has been recently shown that by idealizing deep residual networks (ResNet) as continuous-time dynamical systems it is possible to derive sufficient conditions for universal approximation in L^p, which can be understood as an approximation theory built on flow maps generated by dynamical systems [13].

In this note we present a formulation deep neural networks obtained by applying simple probabilistic tools to discrete dynamical systems. Specifically, we consider two types of neural network models:

- Neural networks perturbed by additive random noise;
- Neural networks with random weights and biases.

Modeling neural networks as discrete stochastic dynamical systems. Let us begin by modeling the input-output map of a neural network as a discrete dynamical system (see Figure 1)

$$\boldsymbol{X}_1 = \boldsymbol{F}_0(\boldsymbol{X}_0, \boldsymbol{w}_0) + \boldsymbol{\xi}_0 \qquad \boldsymbol{X}_{n+1} = \boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n) + \boldsymbol{\xi}_n, \tag{1}$$

Here the index n labels a specific layer in the network, $\boldsymbol{X}_0 \in \mathbb{R}^d$ is the input, $\boldsymbol{X}_n \in \mathbb{R}^N$ ($n = 1, \ldots, L$ represents output of the n-th layer, and $\{\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{L-1}\}$ is set of statistically independent random vectors, or more generally a vector-valued Markov process. We allow the initial state $\boldsymbol{X}_0$ to be random as well, which can be directly connected to a data set in a training algorithm. A neural networks of the form (1) is called *recurrent*, to emphasize the fact that the mapping

$$\boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n) = \boldsymbol{\varphi}(\boldsymbol{W}_n \boldsymbol{X}_n + \boldsymbol{b}_n) \qquad \boldsymbol{w}_n = \{\boldsymbol{W}_n, \boldsymbol{b}_n\}, \tag{2}$$

between one layer and the next has the same functional form. In (2) $\boldsymbol{\varphi} : \mathbb{R}^N \mapsto \mathbb{R}^N$ is the *activation function* of the network, $\boldsymbol{W}_n$ is a $N \times N$ weight matrix and $\boldsymbol{b}_n \in \mathbb{R}^N$ is a bias vector.

In a supervised learning setting, the degrees of freedom

$$\boldsymbol{w} = \{\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{L-1}\}, \tag{3}$$

are determined by optimizing a suitable performance metric depending on the network output. For instance, if we are interested in using the network depicted in Figure 1 to approximate a multivariate function $g(\boldsymbol{x}) \in L^2([0,1]^d)$ then we can identify the degrees of freedom (3) by minimizing, e.g., the non-convex functional

$$\{\boldsymbol{\alpha}, \boldsymbol{w}\} = \underset{\boldsymbol{\alpha}, \boldsymbol{w}}{\operatorname{argmin}} \, \| g(\boldsymbol{x}) - \boldsymbol{\alpha} \cdot \mathbb{E}\left[\boldsymbol{X}_L | \boldsymbol{X}_0 = \boldsymbol{x}\right] \|^2_{L^2([0,1]^d)}, \tag{4}$$

where $\boldsymbol{\alpha}$ are the output weights, and $\mathbb{E}\left[\boldsymbol{X}_L | \boldsymbol{X}_0 = \boldsymbol{x}\right]$ conditional expectation of $\boldsymbol{X}_L$ given $\boldsymbol{X}_0 = \boldsymbol{x}$. In this setting, it is clear that the process of training a neural network is basically an optimal control problem (the controls being the weights and biases) of a discrete stochastic differential equation.

A different stochastic neural network model can be defined by randomizing weights and biases [6, 21]. In this setting we have

$$\boldsymbol{X}_{n+1} = \boldsymbol{\varphi}(\boldsymbol{W}_n(\omega) \boldsymbol{X}_n + \boldsymbol{b}_n(\omega)), \tag{5}$$

where $\boldsymbol{W}_n(\omega)$ are random weight matrices and $\boldsymbol{b}_n(\omega)$ are random bias vectors. We shall assume that $\boldsymbol{W}_n$ and $\boldsymbol{b}_n$ corresponding to different layers are statistically independent.

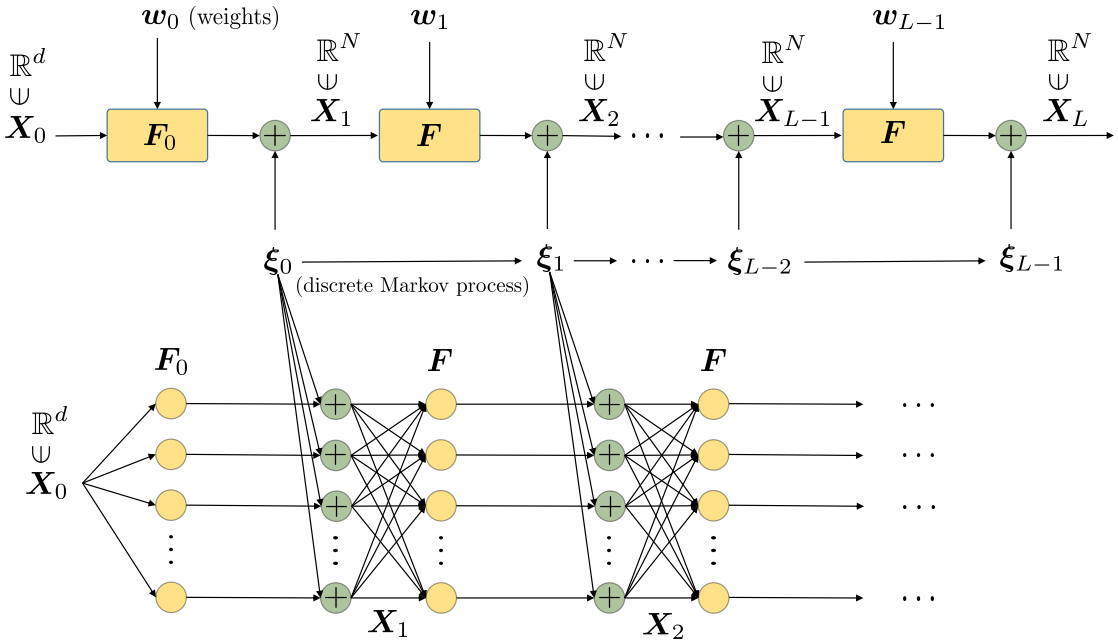

Figure 1: Sketch of the stochastic neural network model (1). Note that the transfer function $\boldsymbol{F}$ is the same in every layer (except the first one). This implies that the random vectors $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_L\}$ all have the same dimension.

By adding random noise to the output of each neural network layer, or by randomizing weights and biases, we are essentially adding an infinite number of degrees of freedom to our system. This allows us to rethink the process of training the neural network from a probabilistic perspective. For instance, random noise allows us to approximately encode secret messages in fully trained deterministic neural networks by selecting an appropriate transition probability for the noise process.

Composition and transfer operators

Let us now derive the composition and transfer operators associated with the neural network models (1) and (5), which map, respectively, the conditional expectation $\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_n = \boldsymbol{x}\}$ and $p_n(\boldsymbol{x})$ (the probability density of $\boldsymbol{X}_n$) forward and backward across the network. To this end, we assume that $\{\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{L-1}\}$ in (1) are independent random vectors. Similarly, we assume that the random matrices $\{\boldsymbol{W}_0, \ldots, \boldsymbol{W}_{L-1}\}$ and the random vectors $\{\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{L-1}\}$ in (5) are statistically independent. These assumptions imply that the sequences of vectors $\{\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_L\}$ generated by either (1) or (5) are *discrete Markov processes*[1]. Therefore, the joint probability density function (PDF) of the random vectors $\{\boldsymbol{X}_0, \ldots, \boldsymbol{X}_L\}$, i.e., joint PDF of the state of the entire neural network, can be factored[2] as

$$p(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_L) = p_{L|L-1}(\boldsymbol{x}_L|\boldsymbol{x}_{L-1}) p_{L-1|L-2}(\boldsymbol{x}_{L-1}|\boldsymbol{x}_{L-2}) \cdots p_{1|0}(\boldsymbol{x}_1|\boldsymbol{x}_0) p_0(\boldsymbol{x}_0). \tag{6}$$

By using the identity

$$p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) = p_{k+1|k}(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) p_k(\boldsymbol{x}_k) = p_{k|k+1}(\boldsymbol{x}_k|\boldsymbol{x}_{k+1}) p_{k+1}(\boldsymbol{x}_{k+1}) \tag{7}$$

[1]The independence assumption assumption of the random noise vector $\{\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{L-1}\}$ or the random weights and biases $\{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_1\}$ and $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m\}$ is not necessary for the process $\{\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_L\}$ to be Markov. Such assumption just simplifies the expression of the transition density $p(\boldsymbol{x}_{i+1}|\boldsymbol{x}_i)$.

[2]In equation (6) we used the shorthand notation $p_{i|j}(\boldsymbol{x}|\boldsymbol{y})$ to denote the conditional probability density function of the random vector $\boldsymbol{X}_i$ given $\boldsymbol{X}_j = \boldsymbol{x}_j$. With this notation we have that the conditional probability density of $\boldsymbol{X}_i$ given $\boldsymbol{X}_i = \boldsymbol{y}$ is $p_{i|i}(\boldsymbol{x}|\boldsymbol{y}) = \delta(\boldsymbol{x} - \boldsymbol{y})$, where $\delta(\cdot)$ is the Dirac delta function.

we see that the chain of transition probabilities (6) can be reverted, yielding

$$p(\boldsymbol{x}_0,\ldots,\boldsymbol{x}_L) = p_{0|1}(\boldsymbol{x}_0|\boldsymbol{x}_1)p_{1|2}(\boldsymbol{x}_1|\boldsymbol{x}_2)\cdots p_{L-1|L}(\boldsymbol{x}_{L-1}|\boldsymbol{x}_L)p_L(\boldsymbol{x}_L). \tag{8}$$

From these expression, it follows that

$$p_{n|q}(\boldsymbol{x}|\boldsymbol{y}) = \int p_{n|j}(\boldsymbol{x}|\boldsymbol{z})p_{j|q}(\boldsymbol{z}|\boldsymbol{y})d\boldsymbol{z}, \tag{9}$$

for all indices n, j and q in $\{0,\ldots,L\}$, excluding $n=j=q$ (see footnote 2). The transition probability equation (9) is known as *discrete Chapman-Kolmogorov equation* and it allows us to define the transfer operator mapping the PDF $p_n(\boldsymbol{x}_n)$ into $p_{n+1}(\boldsymbol{x}_{n+1})$, together with the composition operator for the conditional expectation $\mathbb{E}\{\boldsymbol{u}(\boldsymbol{x}_L)|\boldsymbol{X}_n=\boldsymbol{x}_n\}$. As we shall will see hereafter, the discrete composition and transfer operators are adjoint to one another.

Transfer operator. Let us denote by $p_q(\boldsymbol{x})$ the PDF of $\boldsymbol{X}_q$, i.e., the output of the q-th neural network layer. We first define the operator that maps $p_q(\boldsymbol{x})$ into $p_n(\boldsymbol{x})$. By integrating the joint probability density of $\boldsymbol{X}_n$ and $\boldsymbol{X}_q$, i.e., $p_{n|q}(\boldsymbol{x}|\boldsymbol{y})p_q(\boldsymbol{y})$ with respect to $\boldsymbol{y}$ we immediately obtain

$$p_n(\boldsymbol{x}) = \int p_{n|q}(\boldsymbol{x}|\boldsymbol{y})p_q(\boldsymbol{y})d\boldsymbol{y}. \tag{10}$$

At this point, it is convenient to define the operator

$$\mathcal{N}(n,q)f(\boldsymbol{x}) = \int p_{n|q}(\boldsymbol{x}|\boldsymbol{y})f(\boldsymbol{y})d\boldsymbol{y}. \tag{11}$$

$\mathcal{N}(n,q)$ is known as *transfer operator* [3]. From a mathematical viewpoint $\mathcal{N}(n,q)$ is an integral operator with kernel $p_{n|q}(\boldsymbol{x},\boldsymbol{y})$, i.e., the transition density integrated "from the right". It follows from the Chapman-Kolmogorov identity (9) that the set of integral operators $\{\mathcal{N}(n,q)\}$ forms a group. Namely,

$$\mathcal{N}(n,q) = \mathcal{N}(n,j)\mathcal{N}(j,q), \qquad \mathcal{N}(j,j) = \mathcal{I}, \qquad \forall n,j,q \in \{0,\ldots,L\}. \tag{12}$$

The operator $\mathcal{N}$ allows us to map the one-layer PDF, e.g., the PDF of $\boldsymbol{X}_q$, either forward or backward across the neural network (see Figure 2). As an example, consider a network with four layers with states $\boldsymbol{X}_0$ (input), $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{X}_3$, and $\boldsymbol{X}_4$ (output). Then Eq. (11) implies that,

$$p_2(\boldsymbol{x}) = \underbrace{\mathcal{N}(2,1)\mathcal{N}(1,0)}_{\mathcal{N}(2,0)}p_0(\boldsymbol{x}) = \underbrace{\mathcal{N}(2,3)\mathcal{N}(3,4)}_{\mathcal{N}(2,4)}p_4(\boldsymbol{x}).$$

In summary, we have

$$p_n(\boldsymbol{x}) = \mathcal{N}(n,q)p_q(\boldsymbol{x}) \qquad \forall n,q \in \{0,\ldots,L\}, \tag{13}$$

where

$$\mathcal{N}(n,q)p_q(\boldsymbol{x}) = \int p_{n|q}(\boldsymbol{x}|\boldsymbol{y})p_q(\boldsymbol{y})d\boldsymbol{y}. \tag{14}$$

We emphasize that modeling PDF dynamics via neural networks has been studied extensively in machine learning, e.g., in the theory of normalizing flows for density estimation or variational inference [17, 10, 18].

Composition operator For any measurable deterministic function $\boldsymbol{u}(\boldsymbol{x})$, the conditional expectation of $\boldsymbol{u}(\boldsymbol{X}_j)$ given $\boldsymbol{X}_n=\boldsymbol{x}$ is defined as

$$\mathbb{E}\left\{\boldsymbol{u}(\boldsymbol{X}_j)|\boldsymbol{X}_n=\boldsymbol{x}\right\} = \int \boldsymbol{u}(\boldsymbol{y})p_{j|n}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}. \tag{15}$$

A substitution of (9) into (15) yields

$$\mathbb{E}\left\{\boldsymbol{u}(\boldsymbol{X}_j)|\boldsymbol{X}_n=\boldsymbol{x}\right\}=\int\mathbb{E}\left\{\boldsymbol{u}(\boldsymbol{X}_j)|\boldsymbol{X}_q=\boldsymbol{y}\right\}p_{q|n}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}, \tag{16}$$

which holds for all $j,n,q\in\{0,\ldots,L-1\}$. At this point we define the integral operator

$$\mathcal{M}(n,q)f(\boldsymbol{x})=\int f(\boldsymbol{y})p_{q|n}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}, \tag{17}$$

which is known as *composition* [3] or "stochastic Koopman" [19, 23] operator. Thanks to the Chapman-Kolmogorov identity (9), the set of operators $\{\mathcal{M}(q,j)\}$ forms a group, i.e.,

$$\mathcal{M}(n,q)=\mathcal{M}(n,j)\mathcal{M}(j,q),\qquad \mathcal{M}(j,j)=\mathcal{I},\qquad \forall n,j,q\in\{0,\ldots,L\}. \tag{18}$$

Equation (18) allows us to map the conditional expectation (15) of any measurable phase space function $\boldsymbol{u}(\boldsymbol{X}_j)$ forward or backward through the network. As an example, consider again a neural network with four layers and states $\{\boldsymbol{X}_0,\ldots,\boldsymbol{X}_4\}$. We have

$$\begin{aligned}\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_j)|\boldsymbol{X}_2=\boldsymbol{x}\}=&\mathcal{M}(2,3)\mathcal{M}(3,4)\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_j)|\boldsymbol{X}_4=\boldsymbol{x}\}\\ =&\mathcal{M}(2,1)\mathcal{M}(1,0)\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_j)|\boldsymbol{X}_0=\boldsymbol{x}\}.\end{aligned} \tag{19}$$

Equation (19) holds for every $j\in\{0,..,4\}$. Of particular interest in machine-learning context is the conditional expectation of $\boldsymbol{u}(\boldsymbol{X}_L)$ (network output) given $\boldsymbol{X}_0=\boldsymbol{x}$, which can be computed as

$$\begin{aligned}\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_0=\boldsymbol{x}\}&=\mathcal{M}(0,L)\boldsymbol{u}(\boldsymbol{x}),\\ &=\mathcal{M}(0,1)\mathcal{M}(1,2)\cdots\mathcal{M}(L-1,L)\boldsymbol{u}(\boldsymbol{x}),\end{aligned} \tag{20}$$

i.e., by propagating $\boldsymbol{u}(\boldsymbol{x})=\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_L=\boldsymbol{x}\}$ *backward* through the neural network using single layer operators $\mathcal{M}(i-1,i)$. Similarly, we can compute, e.g., $\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_0)|\boldsymbol{X}_L=\boldsymbol{x}\}$ as

$$\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_0)|\boldsymbol{X}_L=\boldsymbol{x}\}=\mathcal{M}(L,0)\boldsymbol{u}(\boldsymbol{x}). \tag{21}$$

For subsequent analysis, it is convenient to define

$$\boldsymbol{q}_n(\boldsymbol{x})=\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_{L-n}=\boldsymbol{x}\}. \tag{22}$$

In this way, if $\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_n=\boldsymbol{x}\}$ is propagated *backward* through the network by $\mathcal{M}(n-1,n)$, then $\boldsymbol{q}_n(x)$ is propagated *forward* by the operator

$$\mathcal{G}(n,q)=\mathcal{M}(L-n,L-q). \tag{23}$$

In fact, equations (22)-(23) allow us to write (20) in the equivalent form

$$\begin{aligned}\boldsymbol{q}_L(\boldsymbol{x})=&\mathcal{G}(L,L-1)\boldsymbol{q}_{L-1}(\boldsymbol{x})\\ =&\mathcal{G}(L,L-1)\cdots\mathcal{G}(1,0)\boldsymbol{q}_0(\boldsymbol{x}),\end{aligned} \tag{24}$$

i.e., as a forward propagation problem (see Figure 2). Note that we can write (24) (or (20)) explicitly in terms of iterated integrals involving single layer transition densities as

$$\begin{aligned}\boldsymbol{q}_L(\boldsymbol{x})&=\int\boldsymbol{u}(\boldsymbol{y})p_{0|L}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}\\ &=\int\boldsymbol{u}(\boldsymbol{y})\left(\int\cdots\int p_{L|L-1}(\boldsymbol{y}|\boldsymbol{x}_{L-1})\cdots p_{2|1}(\boldsymbol{x}_2|\boldsymbol{x}_1)p_{1|0}(\boldsymbol{x}_1|\boldsymbol{x})d\boldsymbol{x}_{L-1}\cdots d\boldsymbol{x}_1\right)d\boldsymbol{y}.\end{aligned} \tag{25}$$

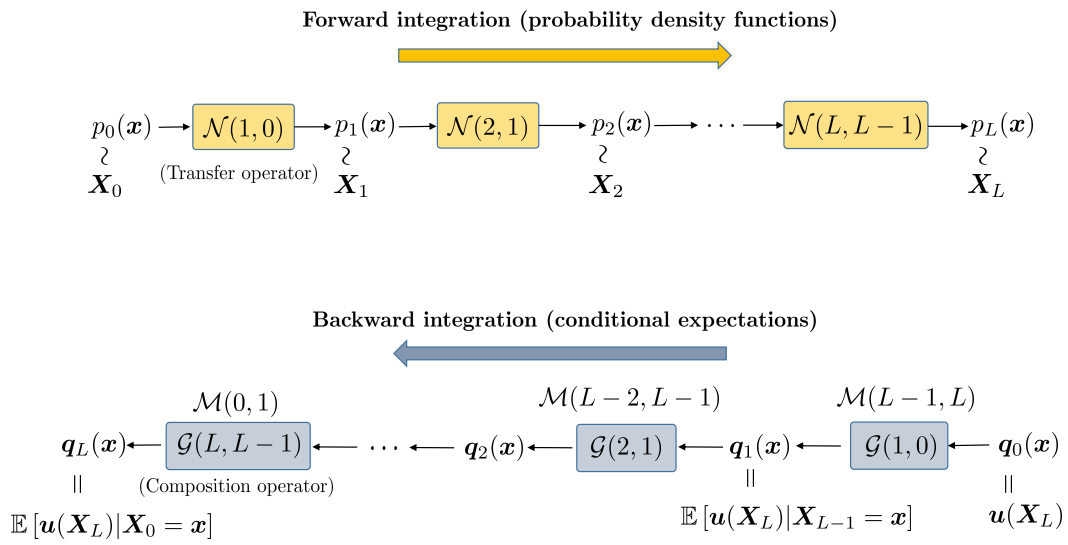

Figure 2: Sketch of the forward/backward integration process for probability density functions (PDFs) and conditional expectations. The transfer operator $\mathcal{N}(n+1,n)$ maps the PDF $p_n(\boldsymbol{x})$ of the state $\boldsymbol{X}_n$ into $p_{n+1}(\boldsymbol{x})$ forward through the neural network. On the other hand, the composition operator $\mathcal{M}$ maps the conditional expectation $\mathbb{E}[\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_{n+1}=\boldsymbol{x}]$ backwards to $\mathbb{E}[\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_n=\boldsymbol{x}]$. By defining the operator $\mathcal{G}(n,m)=\mathcal{M}(L-n,L-m)$ we can transform the backward propagation problem for $\mathbb{E}[\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_n=\boldsymbol{x}]$ into a forward propagation problem for $\boldsymbol{q}_n(\boldsymbol{x})=\mathbb{E}[\boldsymbol{u}(\boldsymbol{X}_L)|\boldsymbol{X}_{L-n}=\boldsymbol{x}]$.

Relation between composition and transfer operators. The integral operators $\mathcal{M}$ and $\mathcal{N}$ defined in (17) and (11) involve the same kernel function, i.e., the multi-layer transition probability density $p_{q|n}(\boldsymbol{x},\boldsymbol{y})$. In particular, we noticed that $\mathcal{M}(n,q)$ integrates $p_{q|n}$ "from the left", while $\mathcal{N}(q,n)$ integrates it "from the right". It is easy to show that $\mathcal{M}(n,q)$ and $\mathcal{N}(q,n)$ are adjoint to each other relative to the standard inner product in L^2 (see [3] for the continuous-time case). In fact,

$$\begin{aligned}\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_k)\} &= \int \mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_k)|\boldsymbol{X}_q=\boldsymbol{x}\}p_q(\boldsymbol{x})d\boldsymbol{x}\\ &= \int \left[\mathcal{M}(q,j)\mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_k)|\boldsymbol{X}_j=\boldsymbol{x}\}\right]p_q(\boldsymbol{x})d\boldsymbol{x}\\ &= \int \mathbb{E}\{\boldsymbol{u}(\boldsymbol{X}_k)|\boldsymbol{X}_j=\boldsymbol{x}\}\mathcal{N}(j,q)p_q(\boldsymbol{x})d\boldsymbol{x}. \end{aligned} \tag{26}$$

Therefore

$$\mathcal{M}(q,j)^*=\mathcal{N}(j,q) \qquad \forall q,j\in\{0,\ldots,L\}, \tag{27}$$

where $\mathcal{M}(q,j)^*$ denotes the operator adjoint of $\mathcal{M}(q,j)$ with respect to the L^2 inner product. By invoking the definition (23), we can also write (27) as

$$\mathcal{G}(L-q,L-j)^*=\mathcal{N}(j,q), \qquad \forall j,q\in\{0,\ldots,L\}. \tag{28}$$

In Appendix A we show that if the cumulative distribution function of each random vector $\boldsymbol{\xi}_n$ in the noise process has partial derivatives that are Lipschitz in $\mathscr{R}(\boldsymbol{\xi}_n)$ (range of $\boldsymbol{\xi}_n$), then the composition and transfer operators defined in Eqs. (17) and 11 are *bounded* in L^2 (see Proposition 0.9 and Proposition 0.10). Moreover, is possible to choose the probability density of $\boldsymbol{\xi}_n$ such that the single layer composition and transfer operators become strict *contractions*.

Conditional transition density

We have seen that the composition and the transfer operators $\mathcal{M}$ and $\mathcal{N}$ defined in Eqs. (17) and (11), allow us to push forward and backward conditional expectations and probability densities across the entire neural network. Moreover such operators are adjoint to one another (see equation (27)) [3, 22, 2], and also have the same kernel, i.e., the transition density $p_{n|q}(\boldsymbol{x}_n|\boldsymbol{x}_q)$. In this section, we determine an explicit expression for such transition density. To this end, we fist derive analytical formulas for the one-layer transition density $p_{n+1|n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n)$ for various types of neural network models. The multi-layer transition density $p_{n|q}(\boldsymbol{x}_n|\boldsymbol{x}_q)$ is then obtained by composing one-layer transition densities as follows

$$p_{n|q}(\boldsymbol{x}_n|\boldsymbol{x}_q) = \int \cdots \int p_{n|n-1}(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) \cdots p_{q+1|q}(\boldsymbol{x}_{q+1}|\boldsymbol{x}_q) d\boldsymbol{x}_{n-1} \cdots d\boldsymbol{x}_{q+1}. \tag{29}$$

Neural network with additive noise. Let us first consider the neural network model

$$\boldsymbol{X}_{n+1} = \boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n) + \boldsymbol{\xi}_n, \tag{30}$$

where $\{\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \ldots\}$ is a discrete (vector-valued) Markov process indexed by "n". By (30), $\boldsymbol{X}_{n+1}$ is the sum of two *independent* random vectors[3], i.e., $\boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n)$ and $\boldsymbol{\xi}_n$. Given any measurable function $h(\boldsymbol{x})$ we clearly have

$$\begin{aligned}
\mathbb{E}\left\{h(\boldsymbol{X}_{n+1})\right\} &= \int h(\boldsymbol{x}) p_{n+1}(\boldsymbol{x}) d\boldsymbol{x} \\
&= \int\int h(\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{w}_n) + \boldsymbol{\xi}) p_n(\boldsymbol{x}) \rho_n(\boldsymbol{\xi}) d\boldsymbol{x} d\boldsymbol{\xi} \\
&= \int\int h(\boldsymbol{y}) \underbrace{\rho_n(\boldsymbol{x} - \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{w}_n))}_{p_{n+1|n}(\boldsymbol{x}|\boldsymbol{y})} p_n(\boldsymbol{x}) d\boldsymbol{y} d\boldsymbol{x},
\end{aligned} \tag{31}$$

where $\rho_n(\boldsymbol{x})$ denotes the probability density of the random vector $\boldsymbol{\xi}_n$. Therefore, the one-layer transition density for the neural network model (30) is[4]

$$p_{n+1|n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) = \rho_n(\boldsymbol{x}_{n+1} - \boldsymbol{F}(\boldsymbol{x_n}, \boldsymbol{w}_n)). \tag{34}$$

Note that such transition density depends on the PDF of the random noise ρ_n, the neural activation function $\boldsymbol{F}$, and the neural network weights $\boldsymbol{w}_n$.

Neural network with random weights and random biases. Next, consider the recurrent neural network model

$$\boldsymbol{X}_{n+1} = \boldsymbol{\varphi}(\boldsymbol{W}_n(\omega)\boldsymbol{X}_n + \boldsymbol{b}_n(\omega)), \tag{35}$$

[3] Recall that $\boldsymbol{X}_n$ and $\boldsymbol{\xi}_n$ are statistically independent random vectors. Hence, $\boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n)$ and $\boldsymbol{\xi}_n$ are statistically independent random vectors.

[4] Equation (34) can be derived in a more general setting by recalling the conditional probability identity

$$p_{n+1|n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) = \int p_{\boldsymbol{X}_{n+1}|\boldsymbol{X}_n,\boldsymbol{\xi}_n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n, \boldsymbol{z}) \rho_n(\boldsymbol{z}) d\boldsymbol{z}. \tag{32}$$

The conditional density of $\boldsymbol{X}_{n+1}$ given $\boldsymbol{X}_n = \boldsymbol{x}_n$ and $\boldsymbol{\xi}_n = \boldsymbol{z}$, i.e., $p_{\boldsymbol{X}_{n+1}|\boldsymbol{X}_n,\boldsymbol{\xi}_n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n, \boldsymbol{z})$, can be immediately computed by using (30) as

$$p_{\boldsymbol{X}_{n+1}|\boldsymbol{X}_n,\boldsymbol{\xi}_n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n, \boldsymbol{z}) = \delta(\boldsymbol{x}_{n+1} - \boldsymbol{F}(\boldsymbol{x}_n, \boldsymbol{w}_n) - \boldsymbol{z}), \tag{33}$$

where $\delta(\boldsymbol{x})$ is the multivariate Dirac delta function. Substituting (33) into (32), and integrating over $\boldsymbol{z}$ yields (34).

where $\boldsymbol{W}_n(\omega)$ are random weight matrices and $\boldsymbol{b}_n(\omega)$ are random bias vectors. The PDF of $\boldsymbol{X}_{n+1}$ given $\boldsymbol{X}_n = \boldsymbol{x}_n$, i.e., the conditional density we are interested in, can be obtained first by computing the PDF of the random vector

$$\boldsymbol{Z}_n(\omega) = \boldsymbol{W}_n(\omega)\boldsymbol{X}_n + \boldsymbol{b}_n(\omega), \tag{36}$$

i.e., a linear mapping between independent random variables, and then computing the PDF of $\boldsymbol{X}_{n+1} = \boldsymbol{\varphi}(\boldsymbol{Z}_n)$, where $\boldsymbol{\varphi}$ is the (invertible) activation function. By using the methods we have seen in chapter 1, it is rather straightforward to obtain and expression for the conditional density of $\boldsymbol{X}_{n+1}$ given $\boldsymbol{X}_n = \boldsymbol{x}_n$ for specific probability distributions of $\boldsymbol{W}_n(\omega)$ (random matrix ensembles) and $\boldsymbol{b}_n$.

General neural networks. The transition density of a general neural network of the form

$$\boldsymbol{X}_{n+1} = \boldsymbol{H}(\boldsymbol{X}_n, \boldsymbol{w}_n, \boldsymbol{\xi}_n), \tag{37}$$

where $\{\boldsymbol{\xi}_0, \dots, \boldsymbol{\xi}_{L-1}\}$ are statistically independent and do not depend on $\{\boldsymbol{X}_j\}$ can be written as (see, e.g., [7])

$$p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) = \int \underbrace{\delta\left(\boldsymbol{x}_{n+1} - \boldsymbol{H}(\boldsymbol{x}_n, \boldsymbol{w}_n, \boldsymbol{\xi}_n)\right)}_{p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n, \boldsymbol{\xi}_n)} p(\boldsymbol{\xi}_n) d\boldsymbol{\xi}_n. \tag{38}$$

Remark: The transition density (34) associated with the neural network model (30) can be computed explicitly once we choose a probability model for $\boldsymbol{\xi}_n \in \mathbb{R}^N$. For instance, if we assume that $\{\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \dots, \}$ are i.i.d. Gaussian random vectors with PDF

$$\rho_n(\boldsymbol{\xi}) = \frac{1}{(2\pi)^{N/2}} e^{-\boldsymbol{\xi}^T\boldsymbol{\xi}/2} \qquad \text{for all } n = 0, \dots, L \tag{39}$$

then we can explicitly write the one-layer transition density (34) as

$$p_{n+1|n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) = \frac{1}{(2\pi)^{N/2}} \exp\left[-\frac{\left[\boldsymbol{x}_{n+1} - \boldsymbol{F}(\boldsymbol{x}_n, \boldsymbol{w}_n)\right]^T \left[\boldsymbol{x}_{n+1} - \boldsymbol{F}(\boldsymbol{x}_n, \boldsymbol{w}_n)\right]}{2}\right]. \tag{40}$$

In Appendix A we provide an analytical example of transition density for a neural network with two layers (one neuron per layer), $\tanh(\cdot)$ activation function, and uniformly distributed random noise.

The zero noise limit An important question is what happens to the neural network as we send the amplitude of the noise to zero. To answer this question consider the system (30) and the introduce the parameter $\epsilon \geq 0$, i.e.,

$$\boldsymbol{X}_{n+1} = \boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n) + \epsilon\boldsymbol{\xi}_n, \tag{41}$$

We are interested in studying the orbits of this system as $\epsilon \to 0$. To this end, we assume the $\boldsymbol{\xi}_n$ to be independent random vectors each having the same density $\rho(\boldsymbol{x})$. This implies that for all $n = 0, \dots, L-1$, the PDF of $\epsilon\boldsymbol{\xi}_n$ is

$$\epsilon\boldsymbol{\xi}_n \sim \frac{1}{\epsilon}\rho_n\left(\frac{\boldsymbol{x}}{\epsilon}\right). \tag{42}$$

It is shown in [11, Proposition 10.6.1] that the operator $\mathcal{N}(n+1, n)$ defined in (11)

$$\begin{aligned} p_{n+1}(\boldsymbol{x}) =& \mathcal{N}(n+1, n) p_n(\boldsymbol{x}) \\ =& \int \frac{1}{\epsilon}\rho_n\left(\frac{\boldsymbol{x} - \boldsymbol{F}(\boldsymbol{z}, \boldsymbol{w}_n)}{\epsilon}\right) p_n(\boldsymbol{z}) d\boldsymbol{z} \end{aligned} \tag{43}$$

converges in norm to the Frobenious-Perron operator corresponding to $\boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n)$ as $\epsilon \to 0$. Indeed, in the limit $\epsilon \to 0$ we have, formally

$$\lim_{\epsilon\to 0} p_{n|n+1}\left(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n\right) = \lim_{\epsilon\to 0} \int \frac{1}{\epsilon}\rho_n\left(\frac{\boldsymbol{x}_{n+1}-\boldsymbol{F}(\boldsymbol{x}_n,\boldsymbol{w}_n)}{\epsilon}\right) = \delta\left(\boldsymbol{x}_{n+1}-\boldsymbol{F}(\boldsymbol{x}_n,\boldsymbol{w}_n)\right). \tag{44}$$

Substituting this expression into (11), one gets,

$$p_{n+1}(\boldsymbol{x}) = \mathcal{N}(n+1,n)p_n(\boldsymbol{x}) = \int \delta\big(\boldsymbol{x}-\boldsymbol{F}(\boldsymbol{z},\boldsymbol{w}_n)\big)p_n(\boldsymbol{z})d\boldsymbol{z}. \tag{45}$$

Similarly, a substitution into equation (24) yields

$$\boldsymbol{q}_{n+1}(\boldsymbol{x}) = \mathcal{G}(n+1,n)\boldsymbol{q}_n(\boldsymbol{x}) = q_n\left(\boldsymbol{F}(\boldsymbol{x},\boldsymbol{w}_{L-n-1})\right), \tag{46}$$

i.e, the familiar function composition representation of neural network mappings

$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_0\Big(\boldsymbol{F}\big(\boldsymbol{F}(\cdots\boldsymbol{F}(\boldsymbol{x},\boldsymbol{w}_{L-n})\cdots,\boldsymbol{w}_{L-1}),\boldsymbol{w}_L\big)\Big). \tag{47}$$

Training over weights versus training over noise

By adding random noise to the output of each layer in a neural network we are essentially adding an infinite number of degrees of freedom to our system. This allows us to rethink the process of training the neural network from a probabilistic perspective. In particular, instead of optimizing a performance metric[5] relative to the neural network weights $\boldsymbol{w} = \{\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{L-1}\}$ for fixed noise, we can now optimize the transition density[6] $p_{n+1|n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n)$. Clearly, such transition density is connected to the neural network weights, e.g., by equation (34). Hence, if we prescribe the PDF of the random noise, i.e., $\rho_n(\cdot)$ in (34), then the transition density $p_{n+1|n}$ is uniquely determined by the functional form of the activation function $\boldsymbol{F}$, and by the weights $\boldsymbol{w}_n$. On the other hand, we can *optimize* ρ_n (probability density of the random noise $\boldsymbol{\xi}_n$) while keeping the weights $\boldsymbol{w}_n$ *fixed*. As we shall see hereafter, this process opens the possibility approximately encode and decode secret messages in a fully trained neural network using random noise.

Encoding secret messages in neural networks using random noise. An interesting question is whether random noise added to the output of each layer in the neural network can enhance features of the output, or allow us to encode/decode secret signals in the network. The interaction between random noise and the nonlinear dynamics modeled by the network can yield indeed many surprising results. For example, in stochastic resonance [16, 20] it is well known that random noise added to a properly tuned bi-stable system can induce a peak in the Fourier power spectrum of the output, hence effectively amplifying the signal. Similarly, random noise added to a neural network can have remarkable effects. In particular, it allows us to re-purpose (to some extent) a previously trained network by hiding a secret signal in it, which can be approximately encoded and decoded by using random noise. To this end, it is sufficient to optimize

[5] In a supervised learning setting the neural network weights are usually determined by minimizing a dissimilarity measure between the output of the network and a target function. Such measure may be an entropy measure, the Wasserstein distance, the Kullback–Leibler divergence, or other measures defined by classical L^p norms.

[6] In a deterministic setting, the transition density for a neural network model of the form $\boldsymbol{X}_{n+1} = \boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n)$ is simply

$$p_{n+1|n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) = \delta\left(\boldsymbol{x}_{n+1}-\boldsymbol{F}(\boldsymbol{x}_n,\boldsymbol{w}_n)\right), \tag{48}$$

where $\delta(\cdot)$ is the Dirac delta function. Such density does not have any degree of freedom other than $\boldsymbol{w}_n$. On the other hand, in a stochastic setting we are free to *choose* the PDF of $\boldsymbol{\xi}_n$. For a neural network model of the form $\boldsymbol{X}_{n+1} = \boldsymbol{F}(\boldsymbol{X}_n, \boldsymbol{w}_n) + \boldsymbol{\xi}_n$ the transition density has the form

$$p_{n+1|n}(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) = \rho_n\left(\boldsymbol{x}_{n+1}-\boldsymbol{F}(\boldsymbol{x}_n,\boldsymbol{w}_n)\right), \tag{49}$$

where $\rho_n(\boldsymbol{\xi})$ is the PDF of $\boldsymbol{\xi}_n$. We are clearly free to choose the functional form of ρ_n.

the PDF of the noise appropriately, and then train the conditional expectation of the output over such PDF.

To describe the method, suppose that we are given a fully trained deterministic neural network with only two layers, and weights chosen to represent an input-output map defined on some domain $\Omega \subseteq \mathbb{R}^d$. In the absence of noise we can write the output of the neural network as

$$q_2(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{F}(\boldsymbol{F}_0(\boldsymbol{x}, \boldsymbol{w}_0), \boldsymbol{w}_1) \tag{50}$$

where $\{\boldsymbol{\alpha}, \boldsymbol{w}_0, \boldsymbol{w}_1\}$ can be optimized to minimize the distance between $q_2(\boldsymbol{x})$ and a given target function $f(\boldsymbol{x})$ ($\boldsymbol{x} \in \Omega$) . Injecting noise $\boldsymbol{\xi}_0$ in the output of the first layer yields the input-output map

$$h_2(\boldsymbol{x}) = \boldsymbol{\alpha}^T \int \boldsymbol{F}(\boldsymbol{y} + \boldsymbol{F}_0(\boldsymbol{x}, \boldsymbol{w}_0), \boldsymbol{w}_1) \rho_0(\boldsymbol{y})\, d\boldsymbol{y}, \tag{51}$$

where $\{\boldsymbol{\alpha}, \boldsymbol{w}_1, \boldsymbol{w}_0\}$ here are fixed, and ρ_0 is the PDF of $\boldsymbol{\xi}_0$. Equation (51) resembles a Fredholm integral equation of the first kind. In fact, it can be written as

$$h_2(\boldsymbol{x}) = \int \kappa_2(\boldsymbol{x}, \boldsymbol{y}) \rho_0(\boldsymbol{y})\, d\boldsymbol{y}, \tag{52}$$

where

$$\kappa_2(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\alpha}^T \boldsymbol{F}(\boldsymbol{y} + \boldsymbol{F}_0(\boldsymbol{x}, \boldsymbol{w}_0), \boldsymbol{w}_1). \tag{53}$$

However, differently from standard Fredholm equations of the first kind, here we have $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d$ while $\boldsymbol{y} \in \mathbb{R}^N$, i.e., the integral operator with kernel κ_2 maps functions in N variables into functions in d variables. We are interested in finding a PDF $\rho_0(\boldsymbol{y})$ that solves (51) for a given function $h_2(\boldsymbol{x})$. In other words, we are re-purposing the neural network (50) with output $q_2(\boldsymbol{x}) \simeq f(\boldsymbol{x})$ to approximate now a different function $h_2(\boldsymbol{x}) \simeq v(\boldsymbol{x})$, without modifying the weights $\{\boldsymbol{\alpha}, \boldsymbol{w}_1, \boldsymbol{w}_0\}$ but rather simply adding noise $\boldsymbol{\xi}_0$ and averaging the output over the PDF ρ_0 of the noise (Eq. (52)). Equation (52) is unfortunately ill-posed in the space of probability distributions. In other words, for a given kernel κ_2 and a given target q_2 there is (in general) no PDF ρ_0 that satisfies (52) exactly. However, one can proceed by optimization. For instance, ρ_0 can be determined by solving the constrained least squares problem[7]

$$\rho_0 = \underset{\rho}{\operatorname{argmin}} \left\| h_2(\boldsymbol{x}) - \int \kappa_2(\boldsymbol{x}, \boldsymbol{y}) \rho(\boldsymbol{y}) d\boldsymbol{y} \right\|_{L^2(\Omega)} \qquad \text{subject to} \qquad \|\rho\|_{L^1(\mathbb{R}^N)} = 1 \qquad \rho \geq 0. \tag{54}$$

[7]The optimization problem (54) is a quadratic program with linear constraints if we represent ρ_0 in the span of a basis made of positive functions, e.g., Gaussian kernels [1].

Appendix A: Functional setting

Let $(\mathcal{S},\mathcal{F},\mathscr{P})$ be a probability space. Consider the neural network model (see Figure 1)

$$\boldsymbol{X}_1=\boldsymbol{F}_0(\boldsymbol{X}_0,\boldsymbol{w}_0)+\boldsymbol{\xi}_0 \qquad \boldsymbol{X}_{n+1}=\boldsymbol{F}(\boldsymbol{X}_n,\boldsymbol{w}_n)+\boldsymbol{\xi}_n \qquad n=1,\ldots,L-1, \tag{55}$$

where $\{\boldsymbol{\xi}_0,\ldots,\boldsymbol{\xi}_{L-1}\}$ is a discrete (vector-valued) Markov process. Suppose we are interested in using the model (55) to approximate a multivariate function $f(\boldsymbol{x})$. This is usually done by taking a linear combination of the network output, e.g., Eq. (4). In this setting, the neural network can be thought of as a process of constructing an adaptive basis by function composition. Here we consider the case where the function we are approximating is defined on a compact subset Ω_0 of $\mathbb{R}^d$. This means that the input vector of the neural network, i.e. $\boldsymbol{X}_0$, is and element of Ω_0. We assume that the following conditions are satisfied

1. $\boldsymbol{X}_0\in\Omega_0\subseteq\mathbb{R}^d$ (Ω compact), $\boldsymbol{X}_n\in\mathbb{R}^N$ for $n=1,\ldots,L-1$;
2. The image of $\boldsymbol{F}_0$ and $\boldsymbol{F}$ is the hyper-cube $[-1,1]^N$.

For example, if $\boldsymbol{F}$ in (55) is of the form

$$\boldsymbol{F}(\boldsymbol{x},\boldsymbol{w})=\tanh(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}) \qquad \boldsymbol{w}=\{\boldsymbol{W},\boldsymbol{b}\}, \tag{56}$$

then conditions 1. and 2. imply that $\boldsymbol{W}_0\in M_{N\times d}(\mathbb{R})$ and $\boldsymbol{W}_n\in M_{N\times N}(\mathbb{R})$ for $n=1,\ldots L-1$, while the biases are $\boldsymbol{b}_n\in M_{N\times 1}(\mathbb{R})$ for $n=0,\ldots L-1$. The random vectors $\{\boldsymbol{\xi}_0,\ldots,\boldsymbol{\xi}_{L-1}\}$ added to the output of each layer make $\{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_K\}$ a discrete Markov process (each $\boldsymbol{X}_i$ is a random vector). The range of $\boldsymbol{X}_{n+1}$ depends essentially on the range of $\boldsymbol{\xi}_n$, as the image of $\boldsymbol{F}$ is the hyper-cube $[-1,1]^N$ (see condition 2. above). Let us define[8]

$$\begin{aligned}\Omega_{n+1}=&[-1,1]^N+\mathscr{R}(\boldsymbol{\xi}_n)\\ =&\{\boldsymbol{c}\in\mathbb{R}^N\,:\,\boldsymbol{c}=\boldsymbol{a}+\boldsymbol{b}\quad \boldsymbol{a}\in[-1,1]^N,\,\boldsymbol{b}\in\mathscr{R}(\boldsymbol{\xi}_n)\},\end{aligned} \tag{58}$$

where $\mathscr{R}(\boldsymbol{\xi}_n)$ denotes the range of the random vector $\boldsymbol{\xi}_n$, i.e.,

$$\mathscr{R}(\boldsymbol{\xi}_n)=\{\boldsymbol{\xi}_n(\omega)\in\mathbb{R}^N:\omega\in\mathcal{S}\}. \tag{59}$$

Clearly, the range of the random vector $\boldsymbol{X}_{n+1}$ is a subset[9] of Ω_{n+1}, i.e., $\mathscr{R}(\boldsymbol{X}_{n+1})\subseteq\Omega_{n+1}$. This implies the following lemma.

Lemma 0.1. *Let $\lambda(\Omega_{n+1})$ the Lebesgue measure of the set* (58)*. The Lebesgue measure of the range of $\boldsymbol{X}_{n+1}$ satisfies*

$$\lambda(\mathscr{R}(\boldsymbol{X}_{n+1}))\leq\lambda(\Omega_{n+1}). \tag{60}$$

Proof. The proof follows from the inclusion $\mathscr{R}(\boldsymbol{X}_{n+1})\subseteq\Omega_{n+1}$. □

[8]The notation $[-1,1]^N$ denotes a Cartesian product of N one-dimensional domains $[-1,1]$, i.e.,

$$[-1,1]^N=\bigtimes_{k=1}^{N}[-1,1]=\underbrace{[-1,1]\times[-1,1]\times\cdots\times[-1,1]}_{N\text{ times}}. \tag{57}$$

[9]We emphasize that if we are given a specific form of the activation function $\boldsymbol{F}$ together with suitable bounds on the neural network weights and biases $\{\boldsymbol{W},\boldsymbol{b}\}$ then we can easily identify a domain that is smaller than Ω_n, and that still contains $\mathscr{R}(\boldsymbol{X}_n)$. This allows us to construct a tighter bound for $\lambda(\mathscr{R}(\boldsymbol{X}_{n+1})$ in Lemma 0.1, which depends on the activation function and on the bounds we set on neural network weights and biases.

The L^∞ norm of the random vector $\boldsymbol{\xi}$ is defined as the largest value of $r \geq 0$ that yields a nonzero probability on the event $\{\omega \in \mathcal{S} : \|\boldsymbol{\xi}(\omega)\|_\infty > r\} \in \mathcal{F}$, i.e.,

$$\|\boldsymbol{\xi}\|_\infty = \sup_{r\in\mathbb{R}}\{\, \mathscr{P}(\{\omega \in \mathcal{S} : \|\boldsymbol{\xi}(\omega)\|_\infty > r\}) > 0\}. \tag{61}$$

This definition allows us to bound the Lebesgue measure of Ω_{n+1} as follows.

Proposition 0.2. *The Lebesgue measure of the set Ω_{n+1} defined in* (58) *can be bounded as*

$$\lambda(\Omega_{n+1}) \leq \left(\sqrt{N} + \|\boldsymbol{\xi}_n\|_\infty\right)^N \frac{\pi^{N/2}}{\Gamma(1+N/2)}, \tag{62}$$

where N is the number of neurons and $\Gamma(\cdot)$ is the Gamma function.

Proof. As is well known, the length of the diagonal of the hypercube $[-1,1]^N$ is $\sqrt{N}$. Hence, $\sqrt{N} + \|\boldsymbol{\xi}_n\|_\infty$ is the radius of a ball that encloses all elements of Ω_{n+1}. The Lebesgue measure of such ball is obtained by multiplying the Lebesgue measure of the unit ball in $\mathbb{R}^N$, i.e., $\pi^{N/2}/\Gamma(1+N/2)$ by the scaling factor $\left(\sqrt{N} + \|\boldsymbol{\xi}_n\|_\infty\right)^N$.

□

Lemma 0.3. *If $\mathscr{R}(\boldsymbol{\xi}_n)$ is bounded then $\mathscr{R}(\boldsymbol{X}_{n+1})$ is bounded.*

Proof. The image of the activation function $\boldsymbol{F}$ is a bounded set. If $\mathscr{R}(\boldsymbol{\xi}_n)$ is bounded then Ω_{n+1} in (58) is bounded. $\mathscr{R}(\boldsymbol{X}_{n+1}) \subseteq \Omega_{n+1}$ and therefore $\mathscr{R}(\boldsymbol{X}_{n+1})$ is bounded.

□

Clearly, if $\{\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{L-1}\}$ are i.i.d. random variables then there exists a domain $V = \Omega_1 = \cdots = \Omega_L$ such that

$$\mathscr{R}(\boldsymbol{\xi}_n) \subseteq \mathscr{R}(\boldsymbol{X}_{n+1}) \subseteq V \qquad \forall n = 0, \ldots, L-1. \tag{63}$$

In fact, if $\{\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{L-1}\}$ are i.i.d. random variables then we have

$$\mathscr{R}(\boldsymbol{\xi}_0) = \mathscr{R}(\boldsymbol{\xi}_1) = \cdots = \mathscr{R}(\boldsymbol{\xi}_{L-1}), \tag{64}$$

which implies that all of Ω_i defined in (58) are the same. If the range of each random vector $\boldsymbol{\xi}_n$ is a tensor product of one-dimensional domain, e.g., if the components of $\boldsymbol{\xi}_n$ are statistically independent, then $V = \Omega_1 = \cdots = \Omega_L$ becomes particularly simple, i.e., a hypercube.

Lemma 0.4. *Let $\{\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{L-1}\}$ be i.i.d. random variables with bounded range and suppose that each $\boldsymbol{\xi}_k$ has statistically independent components with range $[a,b]$. Then all domains $\{\Omega_1, \ldots, \Omega_L\}$ defined in equation* (58) *are the same, and they are equivalent to*

$$V = \bigtimes_{k=1}^{N} [-1+a, 1+b]. \tag{65}$$

V includes the range of all random vectors $\boldsymbol{X}_n$ ($n = 1, \ldots, L$) and has Lebesgue measure

$$\lambda(V) = (2+b-a)^N. \tag{66}$$

Proof. The proof is trivial and therefore omitted.

□

Remark: It is worth noticing that if each $\boldsymbol{\xi}_k$ is a uniformly distributed random vector with statistically independent components in $[-1,1]$, then for $N = 10$ (number of neurons) the upper bound in (62) is 3.98×10^6 while the exact result (66) gives 1.05×10^6. Hence the estimate (62) is quite sharp in the case of uniform random vectors.

Boundedness of composition and transfer operators

Lemma 0.3 states that if we perturb the output of the n-th layer of a neural network by a random vector $\boldsymbol{\xi}_n$ with finite range then we obtain a random vector $\boldsymbol{X}_{n+1}$ with finite range. In this hypothesis it is straightforward to show that that the composition and transfer operators defined in (17) and (11) are bounded. We have seen that these operators can be written as

$$\mathcal{M}(n,n+1)v = \int_{\mathscr{R}(\boldsymbol{X}_{n+1})} v(\boldsymbol{y}) p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x}) d\boldsymbol{y}, \qquad \mathcal{N}(n+1,n)v = \int_{\mathscr{R}(\boldsymbol{X}_n)} p_{n+1|n}(\boldsymbol{x}|\boldsymbol{y}) v(\boldsymbol{y}) d\boldsymbol{y}, \tag{67}$$

where $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x}) = \rho_n(\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{x},\boldsymbol{w}_n))$ is the conditional transition density of $\boldsymbol{X}_{n+1}$ given $\boldsymbol{X}_n$, and ρ_n is the joint PDF of the random vector $\boldsymbol{\xi}_n$. The conditional transition density $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})$ is always non-negative, i.e.,

$$p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x}) \geq 0 \qquad \forall \boldsymbol{y} \in \mathscr{R}(\boldsymbol{X}_{n+1}), \quad \forall \boldsymbol{x} \in \mathscr{R}(\boldsymbol{X}_n). \tag{68}$$

Moreover, the conditional density $p_{n+1|n}$ is defined on the set

$$\mathscr{B}_n = \{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{R}(\boldsymbol{X}_n) \times \mathscr{R}(\boldsymbol{X}_{n+1}) : (\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{x},\boldsymbol{w}_n)) \in \mathscr{R}(\boldsymbol{\xi}_n)\}. \tag{69}$$

It is also important to emphasize that $\boldsymbol{y} \in \mathscr{R}(\boldsymbol{X}_{n+1})$ and $\boldsymbol{x} \in \mathscr{R}(\boldsymbol{X}_n)$. Both $\mathscr{R}(\boldsymbol{X}_{n+1})$ and $\mathscr{R}(\boldsymbol{X}_n)$ depend on Ω_0 (domain of the input), the neural network weights, and the noise amplitude. Thanks to Lemma 0.1, we have that

$$\mathscr{B}_n \subseteq \Omega_n \times \Omega_{n+1}. \tag{70}$$

The Lebesgue measure of $\mathscr{B}_n$ can be calculated as follows.

Lemma 0.5. *The Lebesgue measure of the set $\mathscr{B}_n$ defined in* (69) *is equal to the product of the measure of $\lambda(\mathscr{R}(\boldsymbol{X}_n))$ and the measure of $\mathscr{R}(\boldsymbol{\xi}_n)$, i.e.,*

$$\lambda(\mathscr{B}_n) = \lambda(\mathscr{R}(\boldsymbol{X}_n))\lambda(\mathscr{R}(\boldsymbol{\xi}_n)). \tag{71}$$

Moreover, $\lambda(\mathscr{B}_n)$ is bounded by $\lambda(\mathscr{R}(\Omega_n))\lambda(\mathscr{R}(\boldsymbol{\xi}_n))$, which is independent of the neural network weights.

Proof. Let χ_n be the indicator function of the set $\mathscr{R}(\boldsymbol{\xi}_n)$, $\boldsymbol{y} \in \mathscr{R}(\boldsymbol{X}_{n+1})$ and $\boldsymbol{x} \in \mathscr{R}(\boldsymbol{X}_n)$. Then

$$\begin{aligned}
\lambda(\mathscr{B}_n) =& \int_{\mathscr{R}(\boldsymbol{X}_{n+1})} \int_{\mathscr{R}(\boldsymbol{X}_n)} \chi_n(\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{x},\boldsymbol{w}_n)) d\boldsymbol{x} d\boldsymbol{y} \\
=& \lambda(\mathscr{R}(\boldsymbol{\xi}_n)) \int_{\mathscr{R}(\boldsymbol{X}_n)} d\boldsymbol{x} \\
=& \lambda(\mathscr{R}(\boldsymbol{X}_n))\lambda(\mathscr{R}(\boldsymbol{\xi}_n)).
\end{aligned} \tag{72}$$

By using Lemma 0.1 we conclude that $\lambda(\mathscr{B}_n)$ is bounded from above by $\lambda(\mathscr{R}(\Omega_{L-m}))\lambda(\mathscr{R}(\boldsymbol{\xi}_n))$, which is independent of the neural network weights.

□

Remark: The result (71) has a straightforward geometrical interpretation in two dimensions. Pick a ruler of length $r = \lambda(\mathscr{R}(\xi_n))$ with endpoints that can leave markings if we slide it on a rectangular table with side lengths $s_b = \lambda(\mathscr{R}(X_{n+1}))$ (horizontal side) $s_h = \lambda(\mathscr{R}(X_n))$ (vertical side). Then slide the ruler from the top to the bottom of the table, while keeping it horizontal, i.e., parallel to the horizontal sides of the table (see Figure 3). The area of the domain defined by the two curves drawn by the endpoints of the ruler is always $r \times s_h$ independently of the way we slide the ruler laterally – provided the ruler never gets out of the table.

Lemma 0.6. *If the range of $\boldsymbol{\xi}_{n-1}$ is a bounded subset of $\mathbb{R}^N$ then the transition density $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})$ is an element of $L^1\left(\mathscr{R}(\boldsymbol{X}_{n+1}) \times \mathscr{R}(\boldsymbol{X}_n)\right)$.*

Proof. Note that

$$\int_{\mathscr{R}(\boldsymbol{X}_{n+1})} \int_{\mathscr{R}(\boldsymbol{X}_n)} p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x}) d\boldsymbol{y} d\boldsymbol{x} = \lambda\left(\mathscr{R}(\boldsymbol{X}_n)\right) \leq \lambda(\Omega_n). \tag{73}$$

The Lebesgue measure $\lambda(\Omega_n)$ can be bounded as (see Proposition 0.2)

$$\lambda(\Omega_n) \leq \left(\sqrt{N} + \|\boldsymbol{\xi}_{n-1}\|_\infty\right)^N \frac{\pi^{N/2}}{\Gamma(1+N/2)}. \tag{74}$$

Since the range of $\boldsymbol{\xi}_{n-1}$ is bounded by hypothesis we have that there exists a finite real number $M > 0$ such that $\|\boldsymbol{\xi}_{n-1}\|_\infty \leq M$. This implies that the integral in (73) is finite, i.e., that the transition kernel $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})$ is in $L^1\left(\mathscr{R}(\boldsymbol{X}_{n+1}) \times \mathscr{R}(\boldsymbol{X}_n)\right)$.

□

Theorem 0.7. *Let $C_{\boldsymbol{\xi}_n}(\boldsymbol{x})$ be the cumulative distribution function $\boldsymbol{\xi}_n$. If $C_{\boldsymbol{\xi}_n}(\boldsymbol{x})$ is Lipschitz continuous on $\mathscr{R}(\boldsymbol{\xi}_n)$ and the partial derivatives $\partial C_{\boldsymbol{\xi}_n}/\partial x_k$ ($k = 1, \ldots, N$) are Lipschitz continuous in x_1, x_2, ..., x_N, respectively, then the joint probability density function of $\boldsymbol{\xi}_n$ is bounded on $\mathscr{R}(\boldsymbol{\xi}_n)$.*

Proof. By using Rademacher's theorem we have that if $C_{\boldsymbol{\xi}_n}(\boldsymbol{x})$ is Lipschitz on $\mathscr{R}(\boldsymbol{\xi}_n)$ then it is differentiable almost everywhere on $\mathscr{R}(\boldsymbol{\xi}_n)$ (except on a set with zero Lebesgue measure). Therefore the partial derivatives $\partial C_{\boldsymbol{\xi}_n}/\partial x_k$ exist almost everywhere on $\mathscr{R}(\boldsymbol{\xi}_n)$. If, in addition, we assume that $\partial C_{\boldsymbol{\xi}_n}/\partial x_k$ are Lipschitz continuous with respect to x_k (for all $k = 1, \ldots, N$) then by applying [15, Theorem 9] recursively we conclude that the joint probability density function of $\boldsymbol{\xi}_n$ is bounded.

□

Lemma 0.8. *Under the same assumptions of Theorem 0.7 we have that the conditional PDF $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x}) = \rho_n(\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{w}))$ is bounded on $\mathscr{R}(\boldsymbol{X}_{n+1}) \times \mathscr{R}(\boldsymbol{X}_n)$.*

Proof. Theorem 0.7 states that ρ_n is a bounded function. This implies that the conditional density $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x}) = \rho_n(\boldsymbol{y} - \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{w}))$ is bounded on $\mathscr{R}(\boldsymbol{X}_{n+1}) \times \mathscr{R}(\boldsymbol{X}_n)$.

□

Proposition 0.9. *Let $\mathscr{R}(\boldsymbol{\xi}_n)$ and $\mathscr{R}(\boldsymbol{\xi}_{n-1})$ be bounded subsets of $\mathbb{R}^N$. Then, under the same assumptions of Theorem 0.7, we have that the composition and the transfer operators defined in* (67) *are bounded in L^2.*

Proof. Let us first prove that $\mathcal{M}(n,n+1)$ is a bounded linear operator from $L^2(\mathscr{R}(\boldsymbol{X}_{n+1}))$ into $L^2(\mathscr{R}(\boldsymbol{X}_n))$. To this end, note that

$$\begin{aligned}\|\mathcal{M}(n,n+1)v\|^2_{L^2(\mathscr{R}(\boldsymbol{X}_n))} &= \int_{\mathscr{R}(\boldsymbol{X}_n)}\left|\int_{\mathscr{R}(\boldsymbol{X}_{n+1})} v(\boldsymbol{y})p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}\right|^2 d\boldsymbol{x}\\ &\leq \|v\|^2_{L^2(\mathscr{R}(\boldsymbol{X}_{n+1}))}\underbrace{\int_{\mathscr{R}(\boldsymbol{X}_n)}\int_{\mathscr{R}(\boldsymbol{X}_{n+1})} p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})^2 d\boldsymbol{y}d\boldsymbol{x}}_{K_n}\\ &= K_n\|v\|^2_{L^2(\mathscr{R}(\boldsymbol{X}_{n+1}))}. \end{aligned} \tag{75}$$

Clearly, $K_n<\infty$. In fact, if $\mathscr{R}(\boldsymbol{\xi}_n)$ and $\mathscr{R}(\boldsymbol{\xi}_{n-1})$ are bounded then $\mathscr{R}(\boldsymbol{X}_{n+1})$ and $\mathscr{R}(\boldsymbol{X}_n)$ are bounded. Moreover, thanks to Lemma 0.8 we have that $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})$ is bounded on $\mathscr{R}(\boldsymbol{X}_{n+1})\times\mathscr{R}(\boldsymbol{X}_n)$. Hence, K_n is the integral of the square of a bounded function defined on a bounded domain, and therefore it is finite. By following the same steps it is straightforward to show that the transfer operator $\mathcal{N}$ is a bounded linear operator. Alternatively, simply recall that $\mathcal{N}$ is the adjoint of $\mathcal{M}$, and the adjoint of a bounded linear operator is bounded. Specifically we have,

$$\|\mathcal{N}(n+1,n)p\|^2_{L^2(\mathscr{R}(\boldsymbol{X}_{n+1}))}\leq K_n\|p\|^2_{L^2(\mathscr{R}(\boldsymbol{X}_n))}. \tag{76}$$

□

Remark: The integrals

$$K_n=\int_{\mathscr{R}(\boldsymbol{X}_n)}\int_{\mathscr{R}(\boldsymbol{X}_{n+1})} p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})^2 d\boldsymbol{y}d\boldsymbol{x} \tag{77}$$

can be computed by noting that

$$p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})=\rho_n(\boldsymbol{y}-\boldsymbol{F}(\boldsymbol{x},\boldsymbol{w})) \tag{78}$$

is essentially a *shift* of the PDF ρ_n by a quantity $\boldsymbol{F}(\boldsymbol{x},\boldsymbol{w})$ that depends on $\boldsymbol{x}$ and $\boldsymbol{w}$ (see, e.g., Figure 3). Such a shift does not influence the integral with respect to $\boldsymbol{y}$, meaning that the integral of $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})$ or $p_{n+1|n}(\boldsymbol{y}|\boldsymbol{x})^2$ with respect to $\boldsymbol{y}$ is the same for all $\boldsymbol{x}$. Hence, by changing variables we have that the integral (77) is equivalent to

$$K_n=\lambda(\mathscr{R}(\boldsymbol{X}_n))\int_{\mathscr{R}(\boldsymbol{\xi}_n)}\rho_n(\boldsymbol{x})^2 d\boldsymbol{x}, \tag{79}$$

where $\lambda(\mathscr{R}(\boldsymbol{X}_n))$ is the Lebesgue measure of $\mathscr{R}(\boldsymbol{X}_n)$, and $\mathscr{R}(\boldsymbol{\xi}_n)$ is the range of $\boldsymbol{\xi}_n$. Note that K_n depends on the neural net weights only through the Lebesgue measure of $\mathscr{R}(\boldsymbol{X}_n)$. Clearly, since the set Ω_n includes $\mathscr{R}(\boldsymbol{X}_n)$ we have by Lemma 0.1 that $\lambda(\mathscr{R}(\boldsymbol{X}_n))\leq\lambda(\Omega_n)$. This implies that

$$K_n\leq\lambda(\Omega_n)\int_{\mathscr{R}(\boldsymbol{\xi}_n)}\rho_n(\boldsymbol{x})^2 d\boldsymbol{x}. \tag{80}$$

The upper bound here does not depend on the neural network weights. The following lemma summarizes all these remarks.

Proposition 0.10. *Under the same assumptions of Theorem 0.7, we have that the composition and the transfer operators defined in* (67) *can be bounded as*

$$\|\mathcal{M}(n,n+1)\|^2\leq K_n,\qquad \|\mathcal{N}(n+1,n)\|^2\leq K_n, \tag{81}$$

where

$$K_n=\lambda(\mathscr{R}(\boldsymbol{X}_n))\int_{\mathscr{R}(\boldsymbol{\xi}_n)}\rho_n(\boldsymbol{x})^2 d\boldsymbol{x}. \tag{82}$$

Moreover, K_n can be bounded as

$$K_n \leq \lambda(\Omega_n) \int_{\mathscr{R}(\boldsymbol{\xi}_n)} \rho_n(\boldsymbol{x})^2 d\boldsymbol{x}, \tag{83}$$

where Ω_n is defined in (58) *and ρ_n is the PDF of $\boldsymbol{\xi}_n$. The upper bound in* (83) *does not depend on the neural network weights and biases.*

Under additional assumptions on the PDF $\rho_n(\boldsymbol{x})$ it is also possible to bound the integrals at the right hand side of (82) and (83). Specifically we have the following sharp bound.

Lemma 0.11. *Let*

$$s_n = \inf_{\boldsymbol{x}\in\mathscr{R}(\boldsymbol{\xi}_n)} \rho_n(\boldsymbol{x}), \qquad S_n = \sup_{\boldsymbol{x}\in\mathscr{R}(\boldsymbol{\xi}_n)} \rho_n(\boldsymbol{x}). \tag{84}$$

If $s_n > 0$ then under the same assumptions of Theorem 0.7 we have that

$$\int_{\mathscr{R}(\boldsymbol{\xi}_n)} \rho_n(\boldsymbol{x})^2 d\boldsymbol{x} \leq \frac{1}{\lambda(\mathscr{R}(\boldsymbol{\xi}_n))} \frac{(S_n + s_n)^2}{4 S_n s_n}. \tag{85}$$

Proof. First we notice that if the random vector $\boldsymbol{\xi}_n$ satisfies the assumptions of Theorem 0.7 then the upper bound S_n is finite. By using the definition (84) we have

$$(\rho_n(\boldsymbol{x}) - s_n)(S_n - \rho_n(\boldsymbol{x})) \geq 0 \quad \text{for all} \quad \boldsymbol{x} \in \mathscr{R}(\boldsymbol{\xi}_n). \tag{86}$$

This implies

$$\int_{\mathscr{R}(\boldsymbol{\xi}_n)} \rho_n(\boldsymbol{x})^2 d\boldsymbol{x} \leq (S_n + s_n) - S_n s_n \lambda(\mathscr{R}(\boldsymbol{\xi}_n)), \tag{87}$$

where we used the fact that the PDF ρ_n integrates to one over $\mathscr{R}(\boldsymbol{\xi}_n)$. Next, define

$$R_n = \frac{1}{\lambda(\mathscr{R}(\boldsymbol{\xi}_n))} \frac{(S_n + s_n)^2}{4 S_n s_n}. \tag{88}$$

Clearly,

$$R_n \left(1 - \frac{2 S_n s_n}{s_n + S_n} \lambda(\mathscr{R}(\boldsymbol{\xi}_n))\right)^2 = R_n - (S_n + s_n) + S_n s_n \lambda(\mathscr{R}(\boldsymbol{\xi}_n)) \geq 0 \tag{89}$$

which implies that

$$(S_n + s_n) - S_n s_n \lambda(\mathscr{R}(\boldsymbol{\xi}_n)) \leq R_n. \tag{90}$$

A substitution of (90) into (87) yields (85).

□

An example. Let $X_0 \in \Omega_0 = [-1, 1]$ and consider

$$X_1 = \tanh(X_0 + 3) + \xi_0, \qquad X_2 = \tanh(2X_1 - 1) + \xi_1, \tag{91}$$

where ξ_0 and ξ_1 are uniform random variables with range $\mathscr{R}(\xi_0) = \mathscr{R}(\xi_1) = [-2, 2]$. In this setting,

$$\begin{aligned}
\mathscr{R}(X_1) =& [\tanh(2) - 2, \tanh(4) + 2], \\
\mathscr{R}(X_2) =& [\tanh(2\tanh(2) - 5) - 2, \tanh(2\tanh(4) + 3) + 2].
\end{aligned}$$

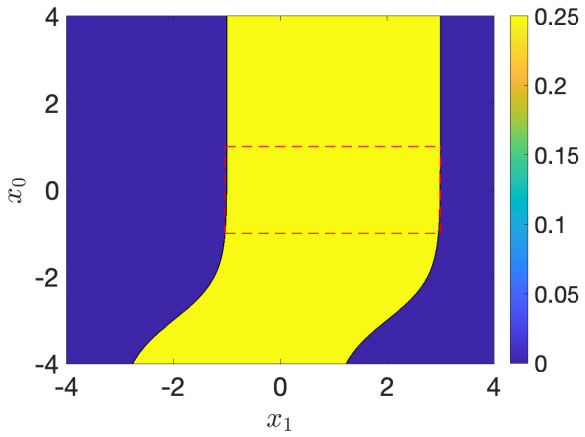

Figure 3: Conditional probability density function $p_{1|0}(x_1|x_0)$ defined in equation (92). The domain $\mathscr{R}(X_1) \times \mathscr{R}(X_0)$ is the interior of the rectangle delimited by dashed red lines.

The conditional density of X_1 given X_0 is given by

$$p_{1|0}(x_1|x_0) = \begin{cases} \frac{1}{4} & \text{if } |x_1 - \tanh(x_0+3)| \leq 2 \\ 0 & \text{otherwise} \end{cases} \tag{92}$$

This function is plotted in Figure 3 together with the domain $\mathscr{R}(X_1) \times \mathscr{R}(X_0)$ (interior of the rectangle delimited by dashed red lines). Clearly, the integral of the conditional PDF (92) is

$$\int_{\mathscr{R}(X_0)} \int_{\mathscr{R}(X_1)} p_{1|0}(x_1|x_0) dx_1 dx_0 = \lambda(\mathscr{R}(X_0)) = 2, \tag{93}$$

where $\lambda(\mathscr{R}(X_0))$ is the Lebesgue measure of $\mathscr{R}(X_0) = [-1, 1]$. The L^2 norm of the operators $\mathcal{N}$ and $\mathcal{M}$ is bounded by[10]

$$K_0 = \int_{\mathscr{R}(X_0)} \int_{\mathscr{R}(X_1)} p_{1|0}(x_1|x_0)^2 dx_1 dx_0 = \frac{\lambda(\mathscr{R}(X_0))}{\lambda(\mathscr{R}(\xi_0))} = \frac{1}{2}. \tag{96}$$

Hence, both operators $\mathcal{N}(1,0)$ and $\mathcal{M}(0,1)$ are contractions (Proposition 0.10). On the other hand,

$$K_1 = \frac{\lambda(\mathscr{R}(X_1))}{\lambda(\mathscr{R}(\xi_1))} = 1 + \frac{\tan(4) - \tan(2)}{4} > 1. \tag{97}$$

Next, define V as in Lemma 0.4, i.e., $V = [-3, 3]$. Clearly, both $\mathscr{R}(X_0)$ and $\mathscr{R}(X_1)$ are subsets of V. If we integrate the conditional PDF shown in Figure 3 in $V \times V$ we obtain

$$\int_V \int_V p_{1|0}(x_1|x_0)^2 dx_1 dx_0 = \frac{\lambda(V)}{\lambda(\mathscr{R}(\xi_0))} = \frac{3}{2}. \tag{98}$$

[10] For uniformly distributed random variables we have that

$$\int_{\mathscr{R}(\boldsymbol{\xi}_n)} \rho_n(\boldsymbol{x})^2 d\boldsymbol{x} = \frac{1}{\lambda(\mathscr{R}(\boldsymbol{\xi}_n))}. \tag{94}$$

Therefore equation (82) yields

$$K_n = \frac{\lambda(\mathscr{R}(\boldsymbol{X}_n))}{\lambda(\mathscr{R}(\boldsymbol{\xi}_n))} \leq \frac{\lambda(\Omega_n)}{\lambda(\mathscr{R}(\boldsymbol{\xi}_n))}. \tag{95}$$

Depending on the ratio between the Lebesgue measure of $\mathscr{R}(\boldsymbol{X}_n)$ and $\mathscr{R}(\boldsymbol{\xi}_n)$ one can have K_n smaller or larger than 1.

Random noise can induce contractions

In this section we prove a result on neural networks perturbed by random noise of increasing amplitude which states that it is possible to make both operators $\mathcal{N}$ and $\mathcal{M}$ in (67) contractions[11] if the noise is properly chosen. To this end, we begin with the following lemma.

Lemma 0.12. *Let*

$$\|\rho_n\|^2_{L^2(\mathscr{R}(\boldsymbol{\xi}_n))} = \int_{\mathscr{R}(\boldsymbol{\xi}_n)} \rho_n(\boldsymbol{x})^2 d\boldsymbol{x}. \tag{99}$$

If

$$\|\rho_n\|^2_{L^2(\mathscr{R}(\boldsymbol{\xi}_n))} \leq \frac{\kappa}{\lambda(\Omega_n)} \qquad 0 \leq \kappa < 1 \tag{100}$$

then $\mathcal{M}(n, n+1)$ and $\mathcal{N}(n+1, n)$ are contractions. The condition (100) is independent of the neural network weights.

Proof. The proof follows from equation (83).

□

Hereafter we specialize Lemma 0.12 to neural network perturbed by uniformly distributed random noise.

Proposition 0.13. *Let $\{\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_{L-1}\}$ be independent random vectors. Suppose that the components of each $\boldsymbol{\xi}_n$ are zero-mean i.i.d. uniform random variables with range $[-b_n, b_n]$ ($b_n > 0$). If*

$$b_0 \geq \frac{1}{2}\left(\frac{\lambda(\Omega_0)}{\kappa}\right)^{1/N} \qquad \text{and} \qquad b_n \geq \frac{b_{n-1}+1}{\kappa^{1/N}} \qquad n = 1, \ldots, L-1, \tag{101}$$

where Ω_0 is the domain of the neural network input, $0 \leq \kappa < 1$, and N is the number of neurons in each layer, then both operators $\mathcal{M}(n, n+1)$ and $\mathcal{N}(n+1, n)$ defined in (67) are contractions for all $n = 0, \ldots, L-1$, i.e., their norm can be bounded by a constant $K_n \leq \kappa$, independently of the weights of the neural network.

Proof. If $\boldsymbol{\xi}_n$ is uniformly distributed then from (82) we have that

$$K_n = \frac{\lambda\left(\mathscr{R}(\boldsymbol{X}_n)\right)}{\lambda\left(\mathscr{R}(\boldsymbol{\xi}_n)\right)}. \tag{102}$$

By using Lemma 0.4 we can bound K_n as

$$K_n \leq \left(\frac{1+b_{n-1}}{b_n}\right)^N, \tag{103}$$

where N is the number of neurons in each layer of the neural network. Therefore, if $b_n \geq (b_{n-1}+1)/\kappa^{1/N}$ $(n = 1, \ldots, L-1)$ we have that K_n is bounded by a quantity κ smaller than one. Regarding b_0, we notice that

$$K_0 = \frac{\lambda\left(\mathscr{R}(\boldsymbol{X}_0)\right)}{\lambda\left(\mathscr{R}(\boldsymbol{\xi}_0)\right)} = \frac{\lambda(\Omega_0)}{(2b_0)^N}, \tag{104}$$

where Ω_0 is the domain of the neural network input. Hence, if b_0 satisfies (102) then $K_0 \leq \kappa$.

□

[11] An linear operator is called a contraction if its operator norm is smaller than one.

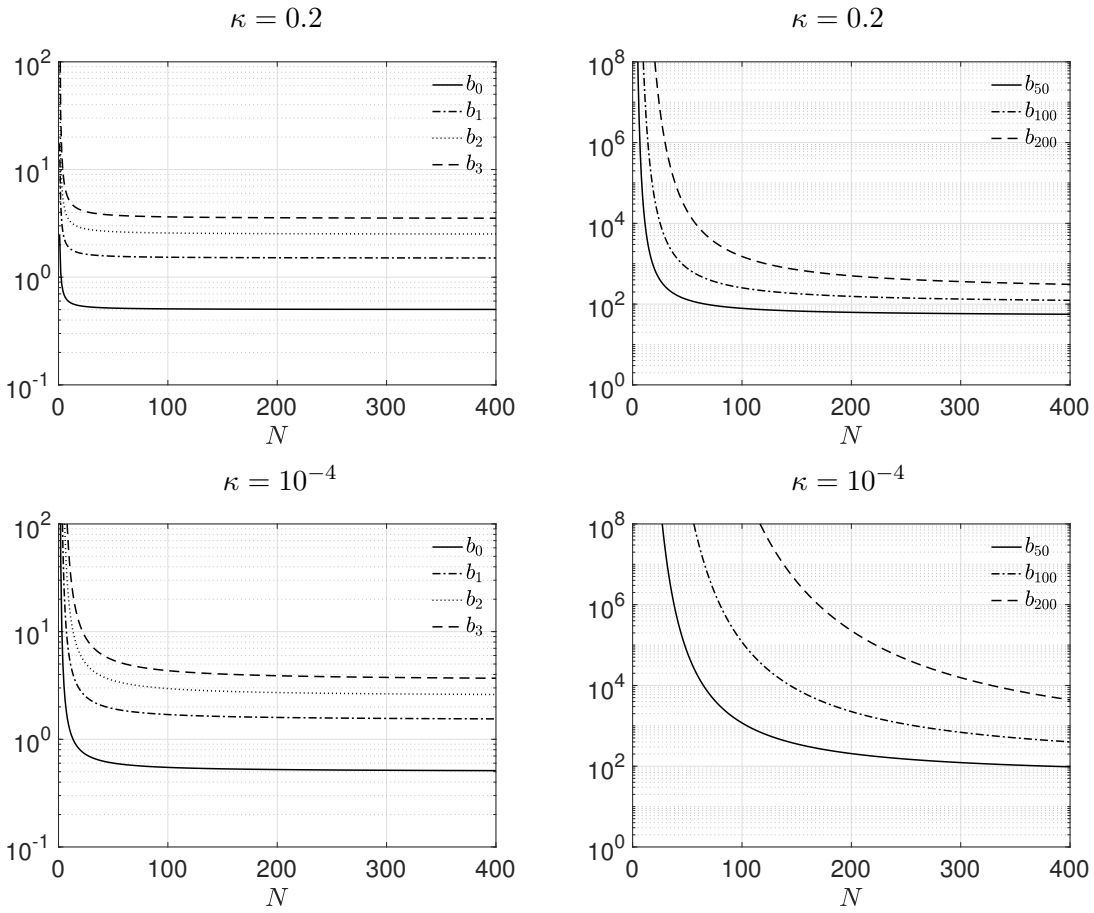

Figure 4: Lower bound on the coefficients b_n defined in (106) for $\lambda(\Omega_0) = 1$ as a function of the number of neurons N and number of layers of the neural network. With such values of b_n the operator $\mathcal{G}(L-n+1, L-n)$ is a contraction satisfying $\|\mathcal{G}(L-n+1, L-n)\|^2 \leq \kappa$. Shown are results for $\kappa = 0.2$ and $\kappa = 10^{-4}$ (contraction index).

One consequence of Proposition 0.13 is that the L^2 norm of the neural network output decays with both the number of layers and the number of neurons if the noise amplitude from one layer to the next increases as in (101). For example, if we represent the input-output map as a sequence of conditional expectations (see (20)), and set $u(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{x}$ (linear output) then we have

$$q_0(\boldsymbol{x}) = \mathcal{M}(0,1)\mathcal{M}(1,2)\cdots\mathcal{M}(L-1,L)(\boldsymbol{\alpha}^T\boldsymbol{x}). \tag{105}$$

By iterating the inequalities (101) in Proposition 0.13 we find that

$$b_n \geq \frac{1}{2\kappa^{n/N}} \left(\frac{\lambda(\Omega_0)}{\kappa}\right)^{1/N} + \sum_{k=1}^{n} \frac{1}{\kappa^{k/N}} \qquad n = 0, \ldots, L-1, \tag{106}$$

In Figure 4 we plot the lower bound at the right hand side of (106) for $\kappa = 0.2$ and $\kappa = 10^{-4}$ as a function of the number of neurons (N). With b_n given in (106) we have that the operator norms of $\mathcal{M}(n, n+1)$ and $\mathcal{N}(n+1, n)$ ($n = 0, \ldots, L-1$) are bounded exactly by κ (see Lemma 0.12). Hence, by taking the L^2 norm of (105), and recalling that $\|\mathcal{M}(n, n+1)\|^2 \leq \kappa$ we obtain

$$\|q_0\|^2_{L^2(\Omega_0)} \leq Z^2 \, \|\boldsymbol{\alpha}\|_2^2 \, \kappa^L, \tag{107}$$

where[12]

$$Z^2 = \sum_{k=1}^{N} \int_{\mathscr{R}(\boldsymbol{X}_L)} x_k^2 d\boldsymbol{x} \qquad \text{and} \qquad \|\boldsymbol{\alpha}\|_2^2 = \sum_{k=1}^{N} \alpha_k^2. \tag{109}$$

The inequality (107) shows that the 2-norm of the vector of weights $\boldsymbol{\alpha}$ must increase exponentially fast with the number of layers L if we chose the noise amplitude as in (106). As shown in the following Lemma, the growth rate of b_n that guarantees that both $\mathcal{M}$ and $\mathcal{N}$ are contractions is linear (asymptotically with the number of neurons).

Lemma 0.14. *Consider a neural network satisfying the hypotheses of Proposition 0.13. Then, in the limit of an infinite number of neurons ($N \to \infty$), the noise amplitude* (106) *satisfies*

$$\lim_{N\to\infty} b_n = \frac{1}{2} + n, \tag{110}$$

independently of the contraction factor κ and the domain Ω_0. This means that for a finite number of neurons the noise amplitude b_n that guarantees that $\|\mathcal{M}(n, n+1)\| \leq \kappa$ is bounded from below ($\kappa < 1$) or from above ($\kappa > 1$) by a function that increases linearly with the number of layers.

Proof. The proof follows by taking the limit of (106) for $N \to \infty$. □

An example: Set $k = 10^{-4}$, $L = 4$ (four layers) and $N = 10$ neurons per layer. The factor κ^L in (107) is 10^{-16}. If we are interested in representing a d-dimensional function in the unit cube $\Omega_0 = [0,1]^d$ then we have[13]

$$\lambda(\Omega_0) = 1 \qquad Z^2 \leq N \frac{2^N (1+b_3)^{N+2}}{3}, \tag{112}$$

where $b_3 = 3.684$ (see Figure 4 for $N = 10$). Hence, the norm of the output (107) is bounded by

$$\|q_L\|_{L^2(\Omega_0)} \leq C \|\boldsymbol{\alpha}\|_2 \qquad C = 10^{-16} \sqrt{10 \frac{2^{10}(1+b_3)^{12}}{3}} \simeq 6.17 \times 10^{-11}. \tag{113}$$

This means that the 2-norm of the output coefficients $\boldsymbol{\alpha}$ has to be of the order of 10^{11} to represent, e.g., a two-dimensional function of norm about one on the square $\Omega_0 = [0,1]^2$.

[12] In equation (107) we used the Cauchy-Schwarz inequality

$$\left\| \boldsymbol{\alpha}^T \boldsymbol{x} \right\|_{L^2(\mathscr{R}(\boldsymbol{X}_L))}^2 \leq Z^2 \|\boldsymbol{\alpha}\|_2^2 . \tag{108}$$

[13] In fact,

$$\sum_{i=1}^{N} \int_{\mathscr{R}(\boldsymbol{X}_4)} x_i^2 d\boldsymbol{x} \leq \sum_{i=1}^{N} \int_{\Omega_4} x_i^2 d\boldsymbol{x} = \sum_{i=1}^{N} \underbrace{\int_{-1-b_3}^{1+b_3} \cdots \int_{-1-b_3}^{1+b_3}}_{N \text{ times}} x_i^2 d\boldsymbol{x} = \frac{2^N N (1+b_3)^{N+2}}{3}. \tag{111}$$

References

[1] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *Ann. Stat.*, 38(5):2916–2957, 2010.

[2] C. Brennan and D. Venturi. Data-driven closures for stochastic dynamical systems. *J. Comp. Phys.*, 372:281–298, 2018.

[3] J. M. Dominy and D. Venturi. Duality and conditional expectations in the Nakajima-Mori-Zwanzig formulation. *J. Math. Phys.*, 58(8):082701, 2017.

[4] W. E. A proposal on machine learning via dynamical systems. *Commun. Math. Stat.*, 5:1–10, 2017.

[5] W. E, J. Han, and Q. Li. A mean-field optimal control formulation of deep learning. *Res. Math. Sci.*, 6:1–41, 2019.

[6] L. Gonon, L. Grigoryeva, and J.-P. Ortega. Risk bounds for reservoir computing. *JMLR*, 21(240):1–61, 2020.

[7] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F. Radar and Signal Processing*, 140(2):107–113, 1993.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 770–778, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[10] I. Kobyzev, S. J. D. Prince, and M. A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[11] A. Lasota and M. C. Mackey. *Chaos, fractals and noise: stochastic aspects of dynamics*. Springer–Verlag, second edition, 1994.

[12] Q. Li, L. Chen, C. Tai, and W. E. Maximum principle based algorithms for deep learning. *JMLR*, 18:1–29, 2018.

[13] Q. Li, T. Lin, and Z. Shen. Deep learning via dynamical systems: An approximation perspective. *J. Eur. Math. Soc.*, (published online first), 2022.

[14] Y. Lu, Zhong A, Q. Li, and B. Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv:1710.10121*, 2017.

[15] E. Minguzzi. The equality of mixed partial derivatives under weak differentiability conditions. *eal Anal. Exch.*, 40(1):81–98, 2014/2015.

[16] D. Nozaki, D. J. Mar, P. Grigg, and J. J. Collins. Effects of colored noise on stochastic resonance in sensory neurons. *Phys. Rev. Lett.*, 82(11):2402–2405, 1999.

[17] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[18] E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Comm. Math. Sci.*, 8(1):217–233, 2010.

[19] N. Črnjarić-Žic, S. Maćešić, and I. Mezić. Koopman operator spectrum for random dynamical systems. *J. Nonlinear Sci.*, 30:2007–2056, 2020.

[20] D. Venturi and G. E. Karniadakis. Convolutionless Nakajima-Zwanzig equations for stochastic analysis in nonlinear dynamical systems. *Proc. R. Soc. A*, 470(2166):1–20, 2014.

[21] T. Yu, Y. Yang, D. Li, T. Hospedales, and T. Xiang. Simple and effective stochastic neural networks. In *Proc. Innov. Appl. Artif. Intell. Conf.*, volume 35, pages 3252–3260, 2021.

[22] Y. Zhu, J. M. Dominy, and D. Venturi. On the estimation of the Mori-Zwanzig memory integral. *J. Math. Phys*, 59(10):103501, 2018.

[23] Y. Zhu and D. Venturi. Hypoellipticity and the Mori-Zwanzig formulation of stochastic differential equations. *J. Math. Phys.*, 62:1035051, 2021.