Xuan He

Report of the Neural Networks and Quantum Field Theory

Based and extended on the work in
https://doi.org/10.1088/2632-2153/abeca3

Report of the internship work at Shadow Creator

Shanghai, August 2021

Shadow Creator Information Technology Co.,Ltd.

Abstract

Neural networks are of great importance today in both the machine learning literature and industry. Despite the excellent performance of many kinds of neural networks such as CNN and RNN on industrial level big data, what happens inside the architecture of those models has long been a hot topic for people to discuss. Note that studying a large model in machine learning (especially when it contains uncertainty) is not only a mathematical problem, but also a physics problem because many theories in physics are aimed to interpret real world complex systems.

When thinking about how to understand neural networks, it is always common in physics literature to extend the problem to its asymptotic limit. Neural network is recognized to behave like Gaussian process when its widths become infinite. For finite neural networks, non-Gaussian process yields the distribution of network outputs. It is natural for physicist to draw a connection between neural networks and quantum field theory because of the correspondence between free field theory and GP (plus correspondence between EFT and NGP). Looking into the infinite neural networks from the perspective of FFT and finite neural networks from the perspective of EFT is the main idea of this report.

In the first chapter, we introduce the theoretic correspondence between NN and QFT. The second chapter is repeating the experiments in the paper to verify the theory. Based on the previous design of experiments, we also make extensions and more discuss about the results in the last chapter.

Contents

Abstract III

Chapter 1 Theoretic Foundation 1

1.1 Gaussian Process 1

1.1.1 Asymptotic Neural Networks and Gaussian Process 1

1.1.2 Asymptotic Neural Networks and Free Field Theory 2

1.2 Calculation of Correlation function at GP Limit 2

1.2.1 Wick Contraction 2

1.2.2 n-pt Function for Single Hidden Layer Neural Networks 2

1.2.3 GP Kernels for Typical Networks 4

1.3 NGP of the Finite Neural Networks 5

1.3.1 Perturbation Theory 5

1.3.2 n-pt Functions and Feynman Rules 5

1.3.3 Coupling Constants 6

1.3.4 Renormalization Group in Neural Networks 7

Chapter 2 Experiment Recurrence 9

2.1 Fall off to GP at Large Width 9

2.1.1 N Scaling of the Connected Part of the Correlation Function 11

2.2 Correlations in Single-Layer Networks with EFT 12

2.3 Renormalization Flow of the Coupling 13

Chapter 3 Experiment Extension **17**

3.1 Randomly Chosen Input Values 17

3.1.1 Gauss-net: larger variance brings about non-local correction terms. 17

3.2 Multiple Input Dimensions 19

3.3 Conclusion 19

Appendix A Notes for Neural networks and quantum field theory **23**

Appendix B Supplementary information for chapter 5 **39**

Chapter 1

Theoretic Foundation

1.1 Gaussian Process

Neural networks acts as a function upon the input variables.

$$f_{\theta,N} : \mathbb{R}^{d_{\text{in}}} \longrightarrow \mathbb{R}^{d_{\text{out}}} \tag{1.1}$$

where θ is the parameter and N is the width.

If we consider the network's distribution in the function space instead of the its parameters' distribution in the parameter space, then a neural network is Gaussian process if the outputs for any finite set of inputs $\mathbf{X} = (x_1, x_2, ...x_n)$ satisfy multivariate Gaussian distribution.

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mu, \mathbf{K}) \tag{1.2}$$

1.1.1 Asymptotic Neural Networks and Gaussian Process

When the neural network admits a limit $N \longrightarrow \infty$ and is properly initialized (LeCun initialization), it becomes Gaussian process with probability distribution in the function space as,

$$P[f] \sim \exp\left[-\frac{1}{2}\int d^{d_{\text{in}}} x d^{d_{\text{in}}} x' f(x) \Xi\left(x, x'\right) f\left(x'\right)\right] \tag{1.3}$$

The covariance matrix is the kernel $\Xi^{-1} = \mathbf{K}$. Suppose the distribution is always zero mean (if not, we could shift the parameter to let outputs always be zero). The correlation functions between n outputs is written as

$$G^{(n)}(x_1, ..., x_n) = \frac{\int df f_1 ... f_n e^{-S}}{Z} \tag{1.4}$$

The partition function $Z = \int df e^{-S}$ and S is the negative log-likelihood.

$$S = \frac{1}{2}\int d^{d_{in}} x d^{d_{in}} x' f(x) \Xi(x, x') f(x') \tag{1.5}$$

1.1.2 Asymptotic Neural Networks and Free Field Theory

Free field theory in physics describes GP. Thus it is direct to link the neural networks with infinite width to free field. In the table below, we draw correspondence between the infinite neural networks and Free field theory.

TABLE 1.1. Correspondence between NN and free field theory

GP/asymptotic NN	Free QFT
Input x	External space or momentum space point
Kernel K(x_1, x_2)	Feynman propagator
Asymptotic NN f(x)	Free Field $\phi(x)$
Log-likelihood	Free action S[ϕ] = $\int d^d x \phi(x)(\Box + m^2)\phi(x)$

1.2 CALCULATION OF CORRELATION FUNCTION AT GP LIMIT

1.2.1 Wick Contraction

The computation of the correlation function (n-pt function) is easy once we know the kernel function. Wick contraction gives the form of the computation function:

$$G_{GP}^{(n)}(x_1, ..., x_n) = \sum_{p \in \mathrm{Wick}(x_1,...,x_n)} K(a_1, b_1)...K(a_{n/2}, b_{n/2}) \tag{1.6}$$

which is the sum over all the product of kernel factors $K(a_i, b_i)$ for each of the pairs (a_i, b_i) that pairing up elements in $\{x_1, \ldots, x_n\}$.
For example,
$G_{\mathrm{GP}}^{(2)}(x_1, x_2) = K(x_1, x_2)$,
$G_{\mathrm{GP}}^{(4)}(x_1, x_2, x_3, x_4) =$
$K(x_1, x_2) K(x_3, x_4) + K(x_1, x_3) K(x_2, x_4) + K(x_1, x_4) K(x_2, x_3)$

1.2.2 n-pt Function for Single Hidden Layer Neural Networks

For perceptron with single hidden layer whose structure described as in Figure 1.1:

$$f_{\theta,N} : \mathbb{R}^{d_{\mathrm{in}}} \xrightarrow{W_0, b_0} \mathbb{R}^N \xrightarrow{\sigma} \mathbb{R}^N \xrightarrow{W_1, b_1} \mathbb{R}^{d_{\mathrm{out}}} \tag{1.7}$$

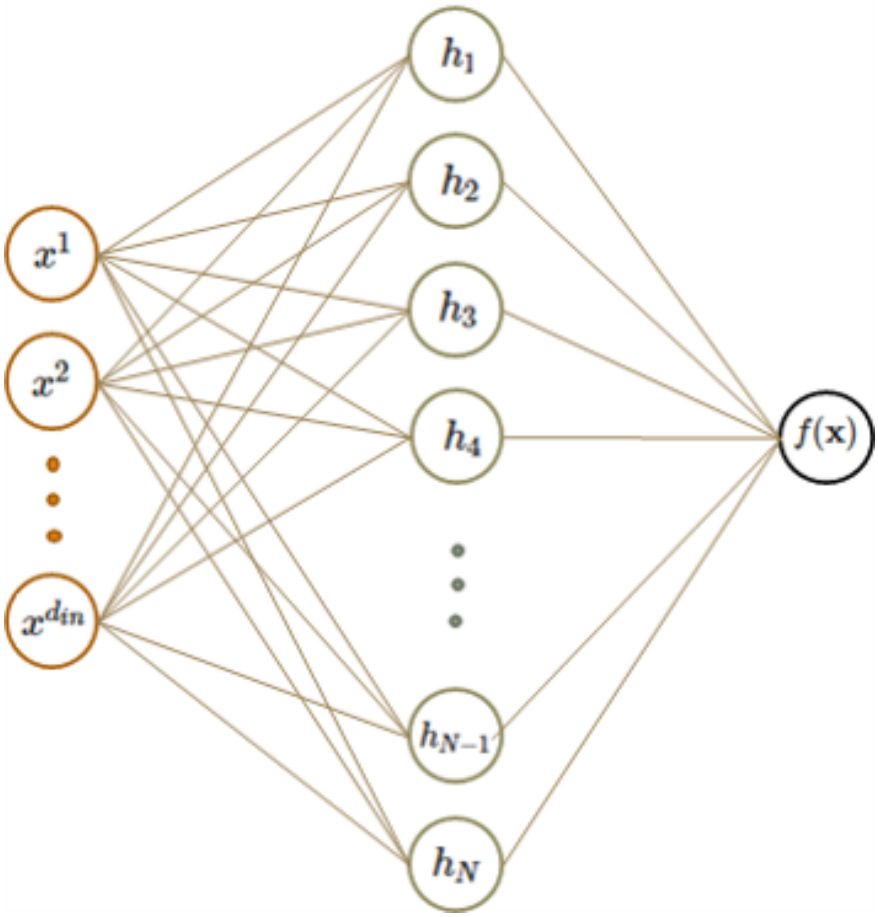


Figure 1.1. The underlying neural network structure. For simplicity, we are considering networks with single hidden layer in this report. The outputs are scalars. We can also drop this constraint without changing the main idea.

its convergence to the Gaussian process at infinite limit is intuitive by central limit theorem (CLT). Since the output layer is linear, we could write the function in two parts:

$$f = f_b + f_W \tag{1.8}$$

where the first term is ultra-local (does not depend on N) and the second term depends on the model and activation. Therefore, the n-pt function could also be written in two parts:

$$G^{(2)}(x_1, x_2) = G_b^{(2)}(x_1, x_2) + G_W^{(2)}(x_1, x_2) \tag{1.9}$$

$$G^{(4)}(x_1, x_2, x_3, x_4) = G_b^{(4)}(x_1, x_2, x_3, x_4) + G_W^{(4)}(x_1, x_2, x_3, x_4) + \sum_{i \neq j \neq k \neq l} G_b^{(2)}(x_i, x_j) G_W^{(2)}(x_k, x_l) \tag{1.10}$$

1.2.3 GP Kernels for Typical Networks

In the experiment part we work on three types of neural networks. The networks share the same model structure but differ in the weights initialization and activation function.

1. **Gauss-net** The activation function of the Gauss-net is,

$$\sigma(x) = \frac{\exp(Wx + b)}{\sqrt{\exp\left[2\left(\sigma_b^2 + \frac{\sigma_W^2}{d_{\text{in}}} x^2\right)\right]}} \tag{1.11}$$

   The GP kernel is,

$$K_{\text{Gauss}}(x, x') = \sigma_b^2 + \sigma_W^2 \exp\left[-\frac{\sigma_W^2 |x - x'|^2}{2d_{\text{in}}}\right] \tag{1.12}$$

   Gauss-net is special due to the translation invariant kernel.

2. **ReLU-net** The activation function of the ReLU-net is,

$$\sigma(z) = \begin{cases} 0 & z < 0 \\ z & z \geqslant 0 \end{cases} \tag{1.13}$$

   The GP kernel is,

$$K_{\text{ReLU}}(x, x') = \sigma_b^2 + \sigma_W^2 \frac{1}{2\pi} \sqrt{\left(\sigma_b^2 + \frac{\sigma_W^2}{d_{\text{in}}} x \cdot x\right)\left(\sigma_b^2 + \frac{\sigma_W^2}{d_{\text{in}}} x' \cdot x'\right)} \cdot (\sin\theta + (\pi - \theta)\cos\theta) \tag{1.14}$$

3. **Erf-net** The activation function of the Erf-net is,

$$\sigma(z) = \text{Erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z dt e^{-t^2} \tag{1.15}$$

The GP kernel is,

$$K_{\text{Erf}}(x, x') = \sigma_b^2 + \sigma_W^2 \frac{2}{\pi} \arcsin \left[\frac{2\left(\sigma_b^2 + \frac{\sigma_w^2}{d_{\text{in}}} x x'\right)}{\sqrt{\left(1 + 2\left(\sigma_b^2 + \frac{\sigma_w^2}{d_{\text{in}}} x^2\right)\right)\left(1 + 2\left(\sigma_b^2 + \frac{\sigma_w^2}{d_{\text{in}}} x'^2\right)\right)}} \right] \tag{1.16}$$

$$\sigma(z) = \text{Erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z dt e^{-t^2} \tag{1.17}$$

1.3 NGP OF THE FINITE NEURAL NETWORKS

1.3.1 *Perturbation Theory*

For finite width neural networks, perturbation theory gives small modification to the action,

$$S = S_{GP} + \Delta S \tag{1.18}$$

The above equation is identical to the form of what we do in EFT where an operator $O_k := g_k f(x)^k$ is introduced in the action function. The first part of S is Gaussian and is thus computable. But the second perturbation term cannot be defined accurately. Instead, we make Taylor expansion on ΔS then calculating the first few terms would be approximate the action well.

$$\Delta S = \int d^{d_{\text{in}}} x \left[g f(x)^3 + \lambda f(x)^4 + \alpha f(x)^5 + \kappa f(x)^6 + \cdots \right] \tag{1.19}$$

In fact, we write the NGP action as the sum of contributions of 4-pt and 6-pt couplings.

$$\Delta S = \int d^{d_{in}} x \left[\lambda f(x)^4 + \kappa f(x)^6 \right] \tag{1.20}$$

2-pt corrections is equal to zero because the GP kernel is the exact 2-pt function. Odd-point correction is zero because odd-point function is always zero which could be derived from the Wick rule.

1.3.2 *n-pt Functions and Feynman Rules*

Using Taylor expansions of the NGP part, the n-pt NGP function is,

$$G^{(n)}(x_1, ..., x_n) = \frac{\int df f(x_1) ... f(x_n) \left[1 - \int d^{d_{in}} x g_k f(x)^k + O(g_k^2)\right] e^{-S_{GP}} / Z_{GP,0}}{\int df \left[1 - \int d^{d_{in}} x g_k f(x)^k + O(g_k^2)\right] e^{-S_{GP}} / Z_{GP,0}} \tag{1.21}$$

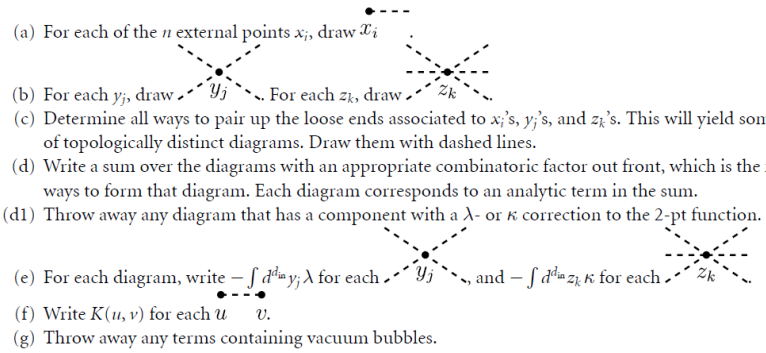


FIGURE 1.2. Feynman rules

Considering the correction terms $\int df f(x_1)...f(x_n) \int d^{d_{in}} x g_k f(x)^k$ that displays similar form as the GP integral, we could carry out calculation by combining all possible wick pairs. Note now we are having internal points that lie inside the integral. The routine to compute NGP n-pt functions is called Feynman rules (Figure 1.2).

As in the first section, we could also draw a perfect analogy between the finite neural networks and the EFT.

TABLE 1.2. Correspondence between finite NN and effective field theory

NGP NN	EFT
Input x	External space or momentum space point
Kernel $K(x_1, x_2)$	Free or exact propagator
NN output $f(x)$	Interacting Field
Non-Gaussianities	Interactions
Non-Gaussian coefficients	Coupling strengths
Log-likelihood	EFT action S

1.3.3 *Coupling Constants*

The λ and κ in S are called coupling constants in QFT theory. However, they are not always constants - more generally depending on the input value. By technical naturalness:

- Conjecture: couplings in NGPs associated to neural network architectures are constants (or nearly constants) if the kernel $K(x, y)$ associated with their GP limit is translationally invariant.

Since Gauss-net has translational invariant kernel, the coupling constants would be actual constants according to technical naturalness.

Given Feynman rules in the previous section,

- if λ is constant

$$\lambda = \frac{K(x_1,x_2)K(x_3,x_4)+K(x_1,x_3)K(x_2,x_4)+K(x_1,x_4)K(x_2,x_3)-G^{(4)}(x_1,x_2,x_3,x_4)}{24\int d^{d_{in}}yK_W(x_1,y)K_W(x_2,y)K_W(x_3,y)K_W(x_4,y)} \tag{1.22}$$

- if λ is not constant

$$\lambda(y) = \bar{\lambda} + \delta\lambda(y) \tag{1.23}$$

$$\begin{aligned}\bar{\lambda} = &\frac{K(x_1,x_2)K(x_3,x_4)+K(x_1,x_3)K(x_2,x_4)+K(x_1,x_4)K(x_2,x_3)}{24\int d^{d_{in}}y\ \Delta_{1234y}} \\ &-\frac{G^{(4)}\ (x_1,x_2,x_3,x_4)}{\int d^{d_{in}}y\ \Delta_{1234y}} - \frac{\int d^{d_{in}}y\ \delta\lambda(y)\ \Delta_{1234y}}{\int d^{d_{in}}y\ \Delta_{1234y}}\end{aligned} \tag{1.24}$$

Where $\Delta_{1234y} = K_W(x_1,y)K_W(x_2,y)K_W(x_3,y)K_W(x_4,y)$

1.3.4 Renormalization Group in Neural Networks

The integral in the NGP action $\Delta S = \int d^{d_{in}}x\left[\lambda f(x)^4 + \kappa f(x)^6\right]$ might diverge when integrate from $-\infty$ to ∞ . There is a trick in QFT that deal with the divergence by simply cutting off the integral limits.

$$\Delta S_{W,\Lambda} = \int_{-\Lambda}^{\Lambda} d^{d_{in}}x \sum_{l\leqslant k} g_{O_l}(\Lambda)O_l \tag{1.25}$$

Note that the bias term is always Gaussian thus ignorable in the following analysis. Substituting the kernels in the n-pt function, the correlation for our network is,

$$G^{(4)}\ (x_1,...,x_4) = \gamma_{4,0} + \varrho_{4,0} - \int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{4,\lambda} - \kappa\int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{4,\kappa} \tag{1.26}$$

$$G^{(6)}\ (x_1,...,x_6) = \gamma_{6,0} + \varrho_{6,0} - \int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{6,\lambda} - \kappa\int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{6,\kappa} \tag{1.27}$$

The n-pt function should be independent of the choice of limits Λ, therefore RG equation requires,

$$\frac{\partial\lambda}{\partial\log\Lambda}\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left[\frac{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\varrho_{6,\lambda}}{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\varrho_{6,\kappa}}-\frac{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\varrho_{4,\lambda}}{\int_{-\Lambda}^{\Lambda} dd_{\text{in}}x\varrho_{4,\kappa}}\right]+\lambda\left[\frac{1}{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\varrho_{6,\kappa}}\frac{\partial\left(\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\varrho_{6,\lambda}\right)}{\partial\log\Lambda}\right.$$
$$\left.-\frac{1}{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\varrho_{4,\kappa}}\frac{\partial\left(\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\varrho_{4,\lambda}\right)}{\partial\log\Lambda}\right]=0 \tag{1.28}$$

RG equation gives the relation between coupling constants and integral limits. Scaling analysis yields similar results.

Chapter 2

Experiment Recurrence

In this chapter, we will do experiments based on the same design described in the paper to verify the results.

2.1 Fall off to GP at Large Width

To test the convergence to Gaussian process, we make measurements on the normalized deviation of the n-pt function.

$$m_n(x_1, ..., x_n) = \Delta G^{(n)}(x_1, ..., x_n)/G^{(n)}_{GP}(x_1, ..., x_n) \tag{2.1}$$

where,

$$\Delta G^{(n)}(x_1, ..., x_n) = G^{(n)}(x_1, ..., x_n) - G^{(n)}_{GP}(x_1, ..., x_n) \tag{2.2}$$

The experimental measurements of the n-pt function is the expectation of the product of n outputs.

$G^{(n)}(x_1, ..., x_n) = \mathcal{E}[f_\alpha(x_1)...f_\alpha(x_n)]$ is measured 100(number of experiments)$*$ 10^5(number of randomly intialized networks per experiment) times in our experiments.

Weights and biases are drawn from Gaussian distribution with mean equals zero and std equals $1/N_{l-1}$ (0 for ReLU-net). N_{l-1} is the width of the previous layer.

Note that the set of inputs for each activation is specified before computing the n-pt function. The inputs used for experiments were chosen where the finite width NGP is well approximated by local-operator correction terms to the associated log-likelihood.

For simplicity, the inputs are 1-dimension (we will increase the input dimension in the third chapter). Inputs for each kind of network are:

- Gauss-net: $\{-0.01, -0.006, -0.002, +0.002, +0.006, +0.01\}$
- ReLU-net: $\{+0.2, +0.4, +0.6, +0.8, +1.0, +1.2\}$
- Erf-net: $\{+0.002, +0.004, +0.006, +0.008, +0.010, +0.012\}$

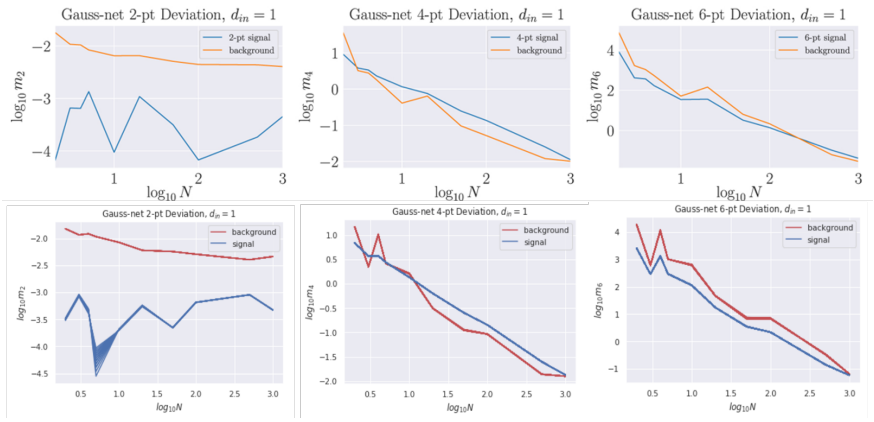


FIGURE 2.1. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of Gauss-net. Signals represent the elements of the tensor m_n , and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements. The first row is the results in the paper, the second row is the results in our experiment.

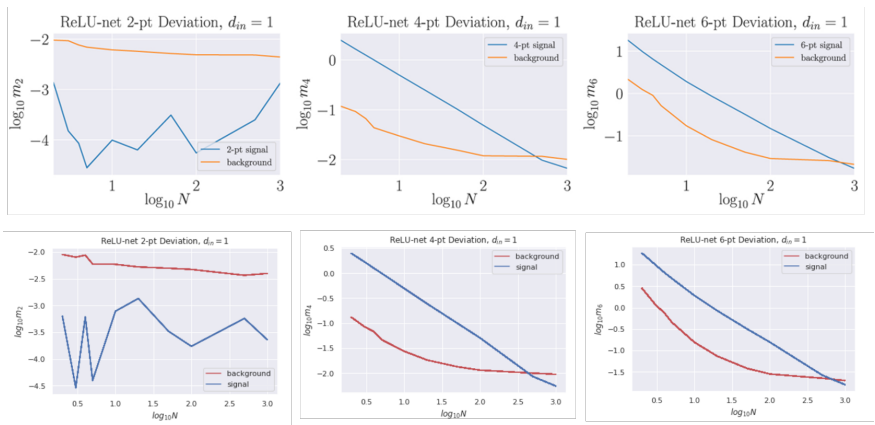


FIGURE 2.2. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of ReLU-net. Signals represent the elements of the tensor m_n , and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements. The first row is the results in the paper, the second row is the results in our experiment.

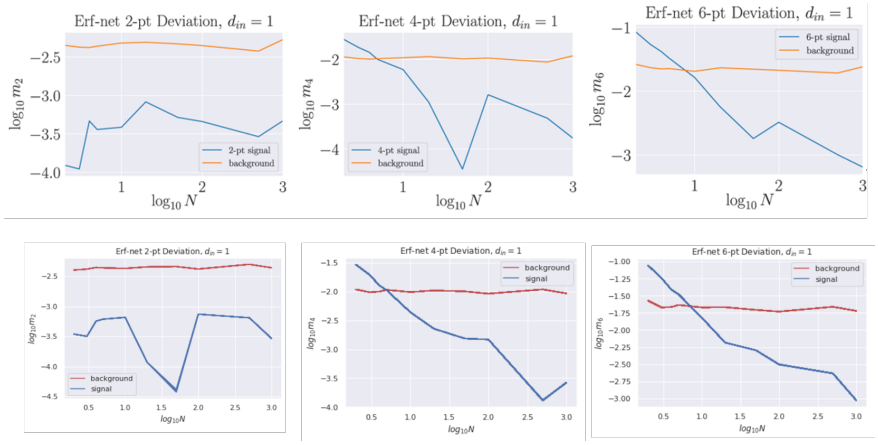


FIGURE 2.3. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of Erf-net. Signals represent the elements of the tensor m_n , and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements. The first row is the results in the paper, the second row is the results in our experiment.

Figure 2.1, Figure 2.2, Figure 2.3 showed that the kernels in three networks are always equal to the 2-pt function. While the deviation in higher point function exhibit a $1/N$ fall off when N becomes large.

2.1.1 *N Scaling of the Connected Part of the Correlation Function*

To understand the scale of deviation from the Gaussian Process predictions, we could compute connected piece of the n-pt functions. In general, we draw following conclusions:

- 2-pt function is exactly the kernel even away from GP (can be proved)
- $\Delta G^{(n)} \propto N^{-1}$ for $n = 4, 6$ (can be proved)
- Connected contribution of the 4-pt function is same as $\Delta G^{(4)} \propto N^{-1}$ (can be proved)
- Connected contribution of the 6-pt function is $\propto N^{-2}$ (can be proved)

The connected part of the 4-pt and 6-pt function is:

$$G^{(4)}(x_1, ..., x_4)|_{\text{connected}} = \Delta G^{(4)} \propto N^{-1} \tag{2.3}$$

$$G^{(6)}(x_1, ..., x_6)|_{\text{connected}} = \Delta G^{(6)} - \sum G^{(4)}(x_1, ..., x_4)|_{\text{connected}}\, G^{(2)}(x_5, x_6) \propto N^{-2} \tag{2.4}$$

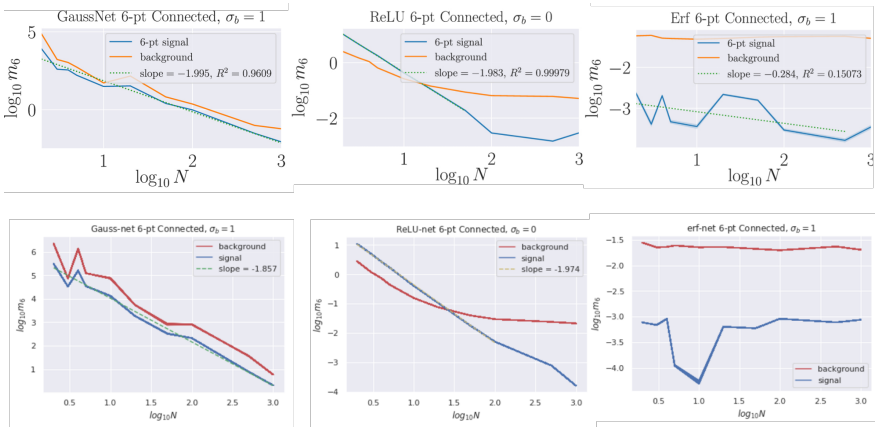


FIGURE 2.4. Normalized connected part of 6-pt function of Erf-net, ReLU-net and Gauss-net. Signals represent the elements of the normalized connected-$G^{(6)}$ tensor, and backgrounds are obtained using the equation below. Slopes are about 2, in agreement with theory, whenever signal is not well below background; for Erf-net, it is for all N. The first row is the results in the paper, the second row is the results in our experiment.

respectively.

In theory,

$$G^{(2k)}(x_1, ..., x_{2k})|_{\text{connected}} = \left[G^{(2k)}(x_1, ..., x_{2k}) - S(x_1, ..., x_{2k})\right]|_{\text{internal indices same}} \tag{2.5}$$

$$G^{(2k)}(x_1, ..., x_{2k})|_{\text{connected}} \propto \frac{1}{N^{k-1}} \tag{2.6}$$

The error in the connected 6-pt functions is the combination of 6-pt error and 4-pt error.

$$\delta G^{(6)}\Big|_{\text{connected}} = \sqrt{\left(\delta G^{(6)}\right)^2 + \left(G^{(2)}\delta G^{(4)}\right)^2} \tag{2.7}$$

2.2 CORRELATIONS IN SINGLE-LAYER NETWORKS WITH EFT

To obtain the coupling constants, we need to compute the coupling terms of the n-pt function.

Note that we are assuming the coulping constants independent of the input.

$$\lambda = \frac{K(x_1,x_2)K(x_3,x_4) + K(x_1,x_3)K(x_2,x_4) + K(x_1,x_4)K(x_2,x_3) - G^{(4)}(x_1,x_2,x_3,x_4)}{24\int d^{d_{in}}y K_W(x_1,y)K_W(x_2,y)K_W(x_3,y)K_W(x_4,y)} \tag{2.8}$$

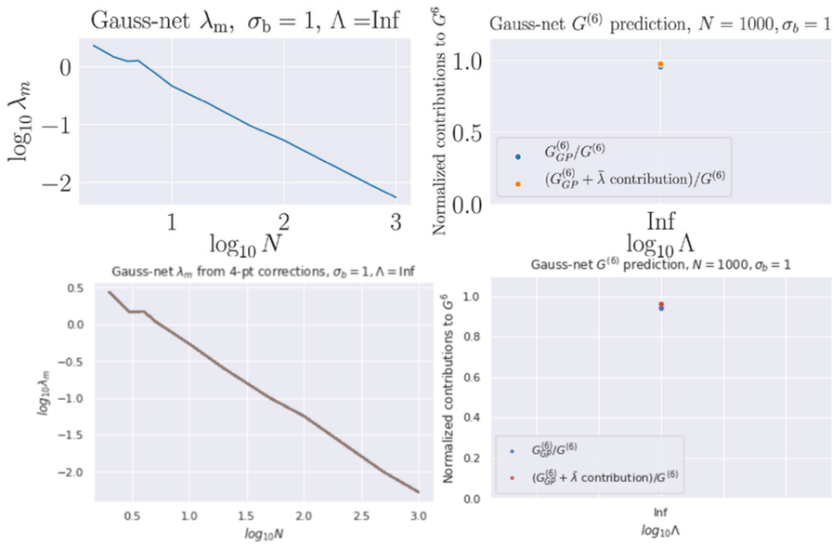


FIGURE 2.5. (left) Measured λ_m tensor elements at different widths for Gauss-net with infinite cutoff. (right) GP prediction alone, and GP prediction $+\bar{\lambda}$ correction of 6-pt function $G^{(6)}$ for Gauss-net at width N =1000, normalized with respect to $G^{(6)}$. The first row is the results in the paper, the second row is the results in our experiment.

$$= \frac{K(x_1,x_2)K(x_3,x_4) + K(x_1,x_3)K(x_2,x_4) + K(x_1,x_4)K(x_2,x_3) - G^{(4)}\ (x_1,x_2,x_3,x_4)}{24 \int d^{d_{in}} y\ \Delta_{1234y}} \tag{2.9}$$

In Figure 2.5, Figure 2.6, and Figure 2.7, an integral limit Λ is specified for each network when computing the denominator. The width is chosen to be $N \in \{2, 3, 4, 5, 10, 20, 50, 100, 500, 1000\}$ We could also fix the width and vary the limit, as shown in the right graph.

2.3 RENORMALIZATION FLOW OF THE COUPLING

In this section, we discover the relationship between the coupling strength λ_m and the integral bound Λ. Scaling dimensions of λ_m could be obtained from the dimensionlessness of the term $\Delta S = \sum \int d^{d_{in}} x O_k$, where operator $O_k := g_k f(x)^k$.

$[\Delta S] = 0$ requires $d_{\text{in}} + k[f] + [g_k] = 0$.

For $[f]$, we know that,

$$P[f] = e^{-S_{\text{GP}}\ [f]} = \exp\left[-\frac{1}{2}\int d^{d_{\text{in}}}\ x d^{d_{\text{in}}}\ y f(x)\Xi(x,y)f(y)\right] \tag{2.10}$$

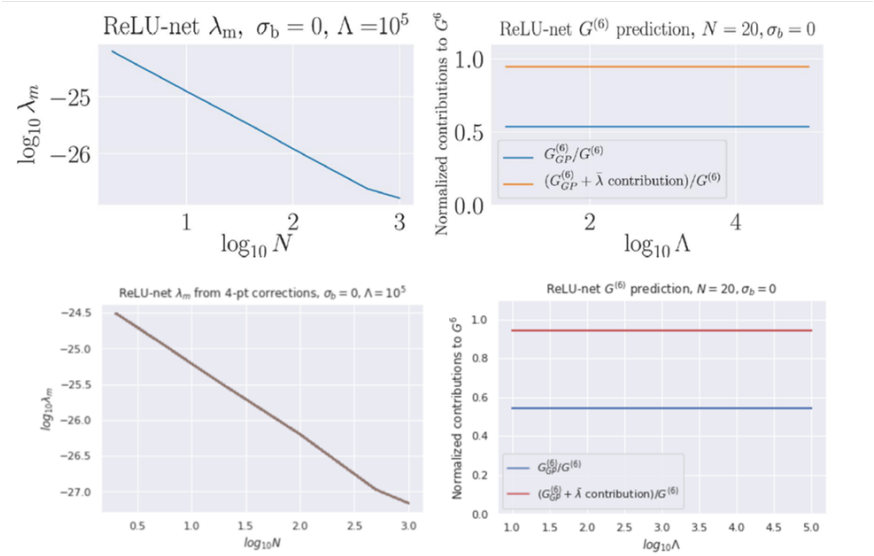


FIGURE 2.6. (left) Measured λ_m tensor elements at different widths for ReLU-net with fixed cutoff. (right) GP prediction alone, and GP prediction $+\bar{\lambda}$ correction of 6-pt function $G^{(6)}$ for ReLU-net at width N = 20, normalized with respect to $G^{(6)}$. The first row is the results in the paper, the second row is the results in our experiment.

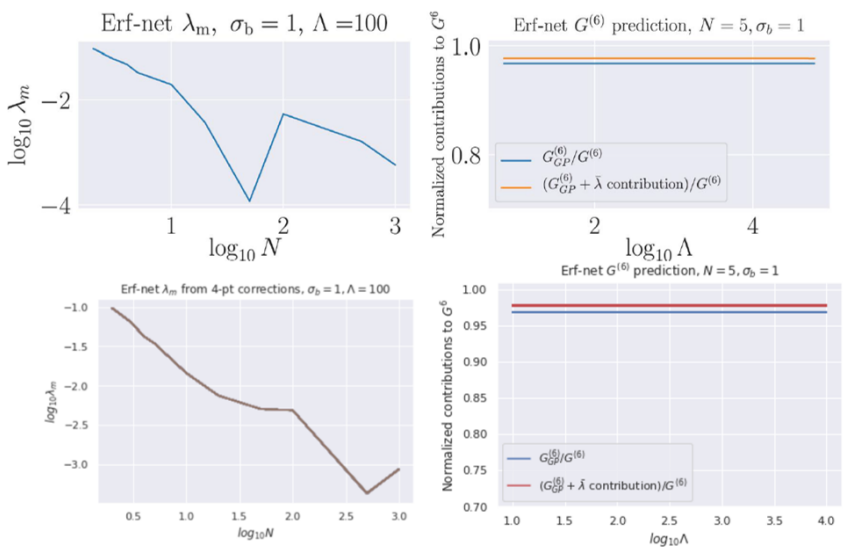


FIGURE 2.7. (left) Measured λ_m tensor elements at different widths for Erf-net with fixed cutoff. (right) GP prediction alone, and GP prediction $+\bar{\lambda}$ correction of 6-pt function $G^{(6)}$ for Erf-net at width N =5, normalized with respect to $G^{(6)}$. The first row is the results in the paper, the second row is the results in our experiment.

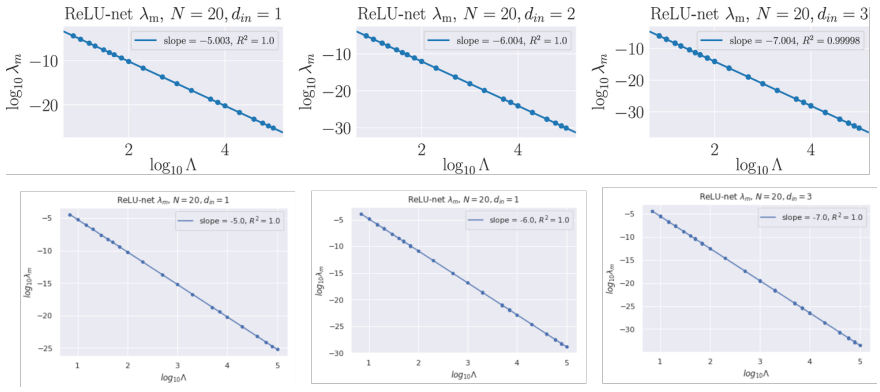


FIGURE 2.8. RG equation for ReLU-net: $\log(\lambda_m)$ as a function of $\log(\Lambda)$ for $d_{\text{in}} = 1, 2, 3$. The first row is the results in the paper, the second row is the results in our experiment.

should have the exponential term dimensionless too. Then, $[S] = 2d_{\text{in}} + 2[f] + [\Xi] = 0$.

$\int d^{d_{\text{in}}} x' K(x, x') \Xi(x', x'') = \delta^{(d_{\text{in}})}(x - x'')$, Ξ should satisfy $d_{\text{in}} + [\Xi] + [K] = -d_{\text{in}}$.

Thus the scaling dimension of the correction terms,

$$[g_k] = -d_{\text{in}} - \frac{k[K]}{2} \tag{2.11}$$

Consider the effective function $S = S_{GP} + \int d^{d_{in}} x \left[\lambda f(x)^4 + \kappa f(x)^6\right]$. The couplings of 4-pt and 6-pt correction have input-space dimensions:

$$[\lambda] = -d_{in} - 2[K], \quad [\kappa] = -d_{in} - 3[K]$$

$[\lambda]$ depends on the input dimension. This is analogous to a well-known effect from QFT, where the dimension of spacetime affects the RG flow.

In previous experiment, we compute the relationship between couplings and boundary terms on one dimensional input. The relationship on 2d and 3d input are to be explored in this section.

The RG flow for ReLU-net is,

$$\log \lambda = -(d_{\text{in}} + 4) \log \Lambda + p_1 \tag{2.12}$$

which is consistent with the experimental result Figure 2.8.

Chapter 3

Experiment Extension

Apart from the last section, all the experiments in the paper uses specially chosen inputs values with only one dimension and few numbers. It would be better if we made more extensions beyond these constraints to test the generalizability of the experimental results.

3.1 Randomly Chosen Input Values

First, what happens if we randomly choose the inputs? Will the n-pt function behave similarly as previously? Input values are randomly generated from Gaussian distribution with zero mean each time. Change the variance of X and verify the NGP $\longrightarrow$ GP behavior of the n-pt function.

3.1.1 Gauss-net: larger variance brings about non-local correction terms.

From Figure 3.3 and Figure 3.4 we can see that when X variance becomes larger, the normalized deviation disperse.

This might be caused by the non-locality of the coupling terms when the inputs become larger. Let's verify this by computing the coupling constant λ_m .

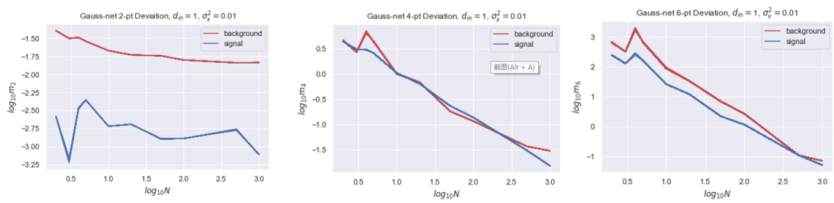


Figure 3.1. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of Gauss-net. Signals represent the elements of the tensor m_n , and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements. The input values is generated Gaussianly with std = 0.01.

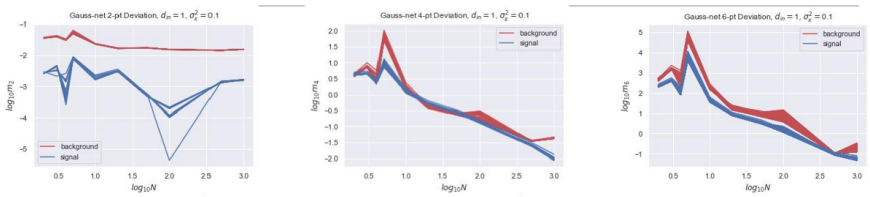


FIGURE 3.2. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of Gauss-net. Signals represent the elements of the tensor m_n , and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements. The input values is generated Gaussianly with std = 0.1.

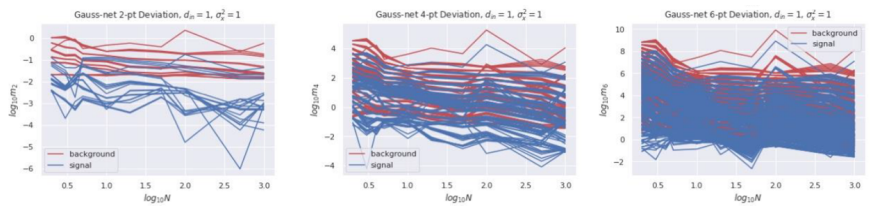


FIGURE 3.3. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of Gauss-net. Signals represent the elements of the tensor m_n, and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements. The input values is generated Gaussianly with std = 1.

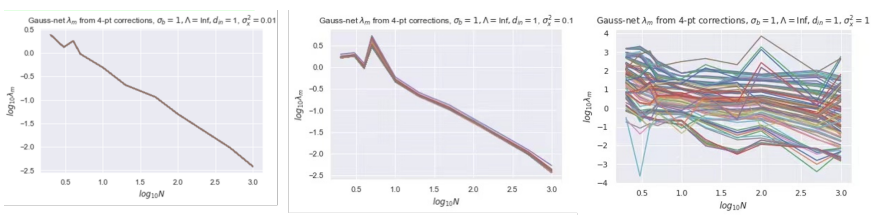


FIGURE 3.4. Measured λ_m tensor elements at different widths for Gauss-net with fixed cut-off. When the variance increases, the coupling constants are not independent of inputs anymore.

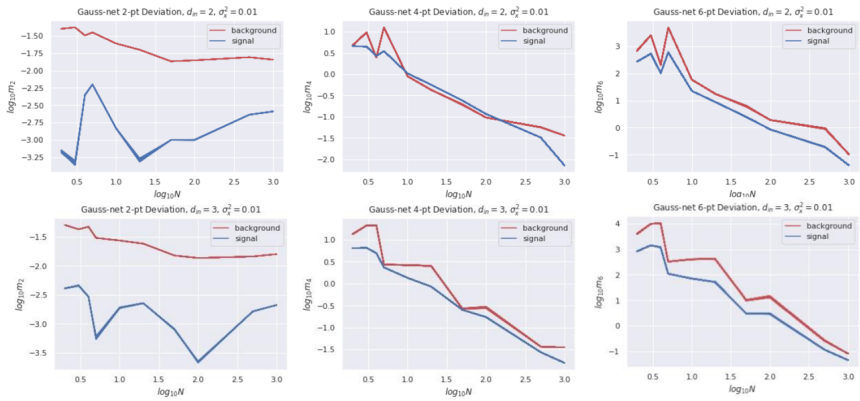


FIGURE 3.5. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of Gauss-net. Signals represent the elements of the tensor m_n , and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements. The input values is generated Gaussianly with std = 0.01.

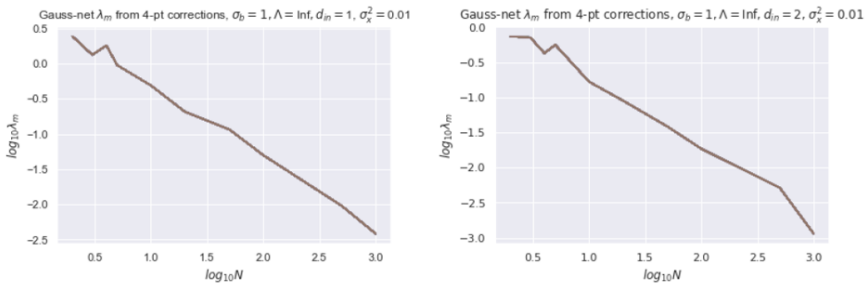


FIGURE 3.6. Measured λ_m tensor elements at different widths for Gauss-net with fixed cutoff for 1d and 2d inputs. When the variance increases, the coupling constants are not independent of inputs anymore.

3.2 MULTIPLE INPUT DIMENSIONS

Next we increase input dimension and look into its effect on the GP convergence and coupling constants. Figure 3.5 shows that n-pt function still converge to the GP with $1/N$ rate when increasing input dimension.

Figure 3.6 compares λ_m for 1d input and 2d input, we find that while they are both linear to the inverse of the width, coupling constant becomes smaller for higher dimension.

3.3 CONCLUSION

In Chapter 1, we introduced the theoretic ideas of relating neural networks to quantum field theory. Then experimental results in the paper is repeated

and verified in the second chapter. The last chapter is to give more insights to the theory by loosing the constraints and developing the experiments to more dimensions. We could still dig into the experiments along this direction. Moreover, we are not computing multiple hidden layer perceptron, neither other structures such as CNN or RNN. Their kernel function might be complicated to solve. However, it is still important to discover their behaviors under infinite width assumption experimentally. And one more limitation of the whole framework is that we are only considering randomly initialized neural network. The training process of asymptotic neural network is also studied, nevertheless through a different kernel (neural tangent kernel). Including trained network into QFT frame would be another job to do.

Appendix A

Notes for Neural networks and quantum field theory

This is the notes taking from https://doi.org/10.1088/2632-2153/abeca3

Gaussian Process

Concept of Gaussian Process

Prior over functions: $p(f)$

For training: minimize negative log marginal likelihood $\mathcal{L}(\theta)$ (θ is hyperparameters and noise level)

Can we use Beyesian regression? $p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{P(\mathcal{D})}$

Gaussian process is a set of random variables f, indexed by a continuous variable $f = f(x)$. Any subset of finite function variables $\{f_n\}_{n=1}^N$ has joint (zero mean) Gaussian distribution.

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(0, \mathbf{K})$$

infinite width	$\Longleftrightarrow$	GP	$\Longleftrightarrow$	free field theory
finite width	$\Longleftrightarrow$	NGP	$\Longleftrightarrow$	effective field theory

Function over x : $f(x)$

$f(x)$ has a distribution: $p(f)$

x is like time/space coordinates. Infinite choices of x, but for any finite subset of x, the finite function variables $\{f_i\}_{i=1}^m$ has joint (zero mean) Gaussian distribution.

$$p(\mathbf{f}|\mathbf{X}) \sim \mathcal{N}$$

For neural networks, x is our input data, f is the network.

$f_{\theta,N} : R^{d_{in}} \to R^{d_{out}}$

parameter space distribution $\to$ function space distribution

Neural Network Corresponds to Gaussian Process

parameter space distribution $\to$ function space distribution

$f_{\theta,N} : R^{d_{in}} \to R^{d_{out}}$

Consider one hidden later of N units, and output dimension is 1:

$$f(\mathbf{x}) = b_1 + \sum_{j=1}^{N} W_1^j \sigma(\mathbf{x}; \mathbf{W}_0^j, b_0^j)$$

$\mathbf{W}_0^j$ is i.i.d., b_0^j, b_1 and W_1^j is zero mean and independent.

In this paper, $b_0^j, b_1 \sim \mathcal{N}(0, \sigma_b^2)$, $\mathbf{W}_0^j \sim \mathcal{N}(0, \sigma_W^2/d_{in})$, $\mathbf{W}_1^j \sim \mathcal{N}(0, \sigma_W^2/N)$

(normalize w.r.t. input dimension)

kernel function dependents on the activation function.

- For neural networks with $N \to \infty$: Gaussian distribution on function space (Gaussian Process) (distribution depends on the GP kernel)
- For neural networks with $N < \infty$: distribution receives $1/N$ corrections $\to$ non-Gaussian Process (NGP)

Learning $\to$ data-induced flow of function space distribution (remain NGP during training)

NN-QFT correspondence

EFT approach can be used for any architecture admitting a GP limit.

Admit GP limits:

- multipayer perceptrons, recurrent NN, skip connections,, convolutions, graph convolutions, pooling, batch/layer normalization, attention.
- both randomly initialized NN or appropriately trained networks.

Meaning of NN-QFT Correspondence

$P[f] \sim e^{-S}$

overparameterization (increasingly large numbers of parameters) $\Leftrightarrow$ neural network likelihood simplicity (simple distributions)

because of:

- asymptotic NN $\to$ GP
- non-Gaussian correction is $1/N$ (actually need only a single number to correct correlation functions)

Asymptotic NN and Free Field Theory

GP/asymptotic NN and Free QFT correspondence

GP/asymptotic NN	Free QFT
Input x	External space or momentum space point
Kernel $K(x_1, x_2)$	Feynman propagator
Asymptotic NN $f(x)$	Free Field

GP/asymptotic NN

$\theta \sim P(\theta)$ + network architecture $\rightarrow P(f)$

$\{f(x_1), \ldots, f(x_k)\} \sim \mathcal{N}(\mu, \Xi^{-1})$

assumption: $\mu = 0$

$(\Xi^{-1})_{ij} = K_{ij}$

- Correlation function (n-pt functions)

$G^{(n)}(x_1, \ldots, x_n) = \frac{\int df f_1 \cdots f_n e^{-S}}{Z}$

- partition function $Z = \int df e^{-S}$
- discrete action $S = \frac{1}{2} f_i \Xi_{ij} f_j$ (Einstein summation)
- continuous action $S = \frac{1}{2} \int d^{d_{in}} x d^{d_{in}} x' f(x) \Xi(x, x') f(x')$
- inverse covariance function $\int d^{d_{in}} x' K(x, x') \Xi(x', x'') = \delta^{(d_{in})}(x - x'')$
- local GP $S = \frac{1}{2} \int d^{d_{in}} x f(x) \Xi(x) f(x)$ where $\Xi(x, x') = \Xi(x) L_\sigma(x, x')$ and $L_0(x, x') = \delta^{(d_{in})}(x - x')$
- ultra-local GP: f is constant $S = \frac{1}{2} f \Xi f$ where $K\Xi = 1$

question: what are propagator (probability or amplitude of propagation of a particle from one point to another) and action in QFT

Free Field Theory

- quantum field $\phi(x)$
- path integral $Z = \int D\phi e^{-S[\phi]}$
- action $S[\phi] = \int d^d x \phi(x)(\Box + m^2)\phi(x)$

question A.21

Computing correlation function

$G^{(n)}_{GP}(x_1, \ldots, x_n) = \sum_{p \in \mathrm{Wick}(x_1, \ldots, x_n)} K(a_1, b_1) \ldots K(a_{n/2}, b_{n/2})$ (can be derived)

Diagrammatic representaions: pairs of points, due to the Gaussian nature of Z_{GP}

Question: how to understand

GP in neural networks

- Why GP: by CLT

$f_{\theta,N}(x) = z_1^k = \sum_{j=1}^N W_1^{jk} x_1^j + b_1^k$

- f has two parts

$f = f_b(\text{ultra-local}) + f_W(\text{depends on model and activation}) \rightarrow$
$G^{(2)}(x_1, x_2) = G_b^{(2)}(x_1, x_2) + G_W^{(2)}(x_1, x_2)$
$G^{(4)}(x_1, x_2, x_3, x_4) = G_b^{(4)}(x_1, x_2, x_3, x_4) + G_W^{(4)}(x_1, x_2, x_3, x_4) + \sum_{i \neq j \neq k \neq l} G_b^{(2)}(x_i, x$

Three networks

- Erf-net
- ReLu-net
- Gauss-net (translation invariant kernel)

Experiments of GP limit

How to measure

- normalized deviation m_n

$m_n(x_1, \ldots, x_n) = \Delta G^{(n)}(x_1, \ldots, x_n) / G_{GP}^{(n)}(x_1, \ldots, x_n)$
$\Delta G^{(n)}(x_1, \ldots, x_n) = G^{(n)}(x_1, \ldots, x_n) - G_{GP}^{(n)}(x_1, \ldots, x_n)$

$G^{(n)}(x_1, \ldots, x_n) = \mathcal{E}[f_\alpha(x_1) \ldots f_\alpha(x_n)]$ is measured in the experiment
$100(\text{experiments}) * 10^5(\text{nets})$ times

weights and biases is drawn from Gaussian dist. with mean equals zero and std equals 1

$G_{GP}^{(n)}(x_1, \ldots, x_n)$ is computed using Wick contraction

$d_{in} = d_{out} = 1$

- Measure m_2, m_4, m_6 multiple times.
- Compute mean and std.
- Compare mean and std.

Inputs

Table 2. Parameters that define the networks used in GP and non-GP experiments in sections 2.4 and 3.5.

	Inputs $\{x_i\}$	(σ_W^2, σ_b^2)
Gauss-net	$\{-0.01, -0.006, -0.002, +0.002, +0.006, +0.01\}$	$(1, 1)$
Erf-net	$\{+0.002, +0.004, +0.006, +0.008, +0.010, +0.012\}$	$(1, 1)$
ReLU-net	$\{+0.2, +0.4, +0.6, +0.8, +1.0, +1.2\}$	$(1, 0)$

- inputs of ReLu-net and Erf-net are chosen to be positive so that the kernel is always positive
- inputs are chosen where the finite width NGP is well approximated by local-operator correction terms to the associated log-likelihood,

question local-operator correction terms

Results

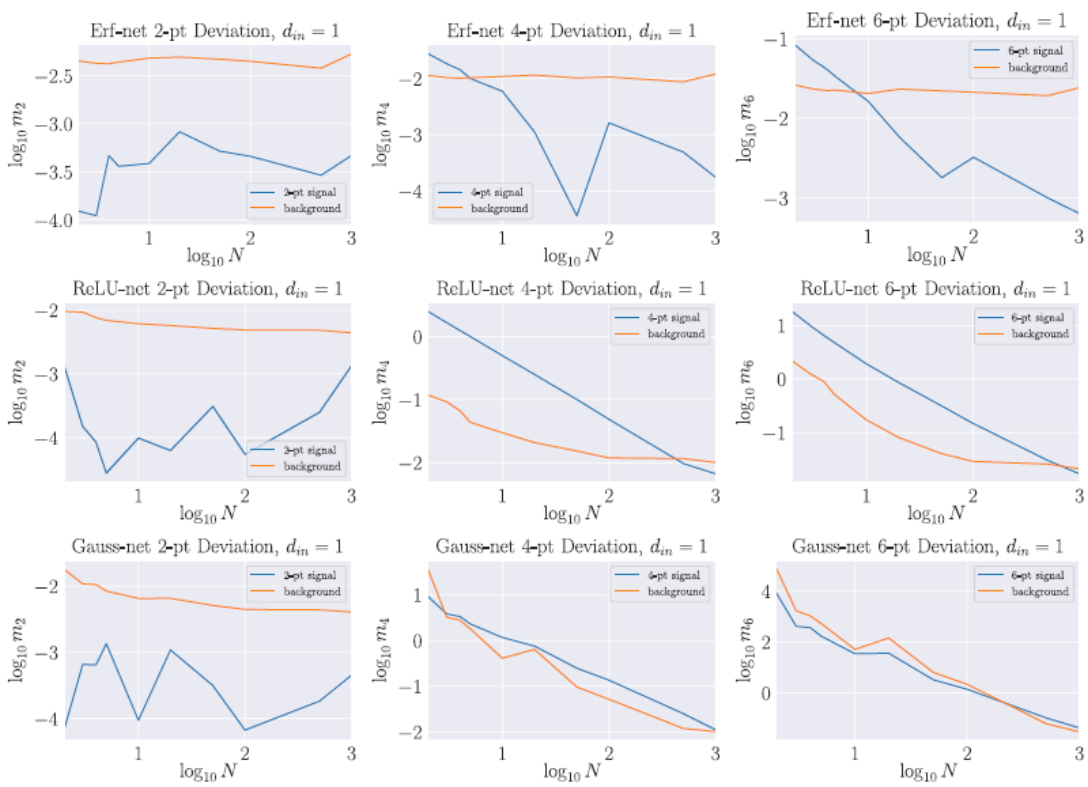


Figure 2. Normalized 2-pt deviation, 4-pt deviation and 6-pt deviation of Erf-net, ReLU-net and Gauss-net. Signals represent the elements of the tensor m_n, and backgrounds are the average elementwise standard deviation across 100 experiments of the tensor elements.

- 2-pt function is exactly the kernel even away from GP (can be proved)
- background is average std across 100 experiments of m_n
- $\Delta G^{(n)} \propto N^{-1}$ for $n = 4, 6$ (can be proved)
- connected contribution of the 4-pt function is same as $\Delta G^{(4)} \propto N^{-1}$ (can be proved)
- connected contribution of the 6-pt function is $\propto N^{-2}$ (can be proved)
- $G^{(2k)}(x_1, \ldots, x_{2k})|_{\text{connected}} = \left[G^{(2k)}(x_1, \ldots, x_{2k}) - S(x_1, \ldots, x_{2k})\right]|_{\text{internal indices s}}$
- $G^{(2k)}(x_1, \ldots, x_{2k})|_{\text{connected}} \propto \frac{1}{N^{k-1}}$
- $G^{(2k)}(x_1, \ldots, x_{2k})|_{\text{connected}}$ is computed from expectation of layer postactivation value

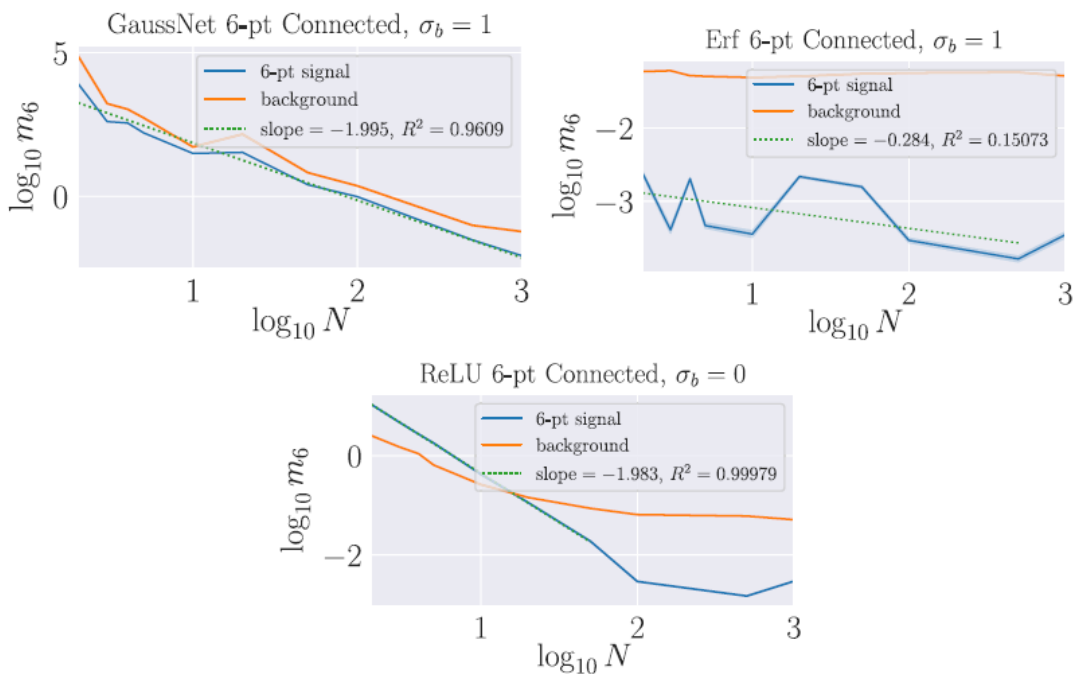


Figure 3. Normalized connected part of 6-pt function of Erf-net, ReLU-net and Gauss-net. Signals represent the elements of the normalized connected-$G^{(6)}$ tensor, and backgrounds are obtained using (57). Slopes are about -2, in agreement with theory, whenever signal is not well below background; for Erf-net, it is for all N.

$$\Delta G^{(2)} \propto 0$$

$$\Delta G^{(4)} \propto N^{-1}$$

$$\Delta G^{(6)} \propto N^{-1} + N^{-2} \propto N^{-1}$$

$$G^{(4)}(x_1, \ldots, x_4)|_{\text{connected}} = \Delta G^{(4)} \propto N^{-1}$$

$$G^{(6)}(x_1, \ldots, x_6)|_{\text{connected}} = \Delta G^{(6)} - \sum G^{(4)}(x_1, \ldots, x_4)|_{\text{connected}}\, G^{(2)}(x_5, x_6) \propto N^{-2}$$

Normalize:

$$\frac{G^{(4)}(x_1,\ldots,x_4)|_{\text{connected}}}{G^{(4)}(x_1,\ldots,x_4)} \propto N^{-1}$$

$$\frac{G^{(6)}(x_1,\ldots,x_6)|_{\text{connected}}}{G^{(6)}(x_1,\ldots,x_6)} \propto N^{-2}$$

Non-Gaussian processes with effective field theory

NGP NN and EFT correspondence

NGP NN	EFT
Input x	External space or momentum space point
Kernel $K(x_1, x_2)$	Free or exact propagator
NN output $f(x)$	Interacting Field
Non-Gaussianities	Interactions
Non-Gaussian coefficients	Coupling strengths
Log-likelihood	EFT action S

EFT: scales

$[S_{GP}] = 0 \to [f] = -\frac{2d_{in}+[\Xi]}{2}$

for $\mathcal{O}_k := g_k f(x)^k$ in ΔS as $\int d^{d_{in}}x\mathcal{O}_k$

$[g_k] = -d_{in} + \frac{k(2d_{in}+[\Xi])}{2} = -d_{in} - \frac{k([K])}{2}$

$[\Xi] + [K] = -2d_{in}$

$\mathcal{O}_k$ can be ignored for sufficiently large k

How to construct the effective action of an NGP

- Determine the symmetries respected by the system of interest.
- Fix and upper bound k on the dimentison of any operator appearing in ΔS.
- Define ΔS to contrain all operators of dimension $\leqslant k$ that respect the symmetries.

Perturbation Theory

$G^{(n)}(x_1,\ldots,x_n) = \frac{\int dff_1\ldots f_n e^{-S}}{Z_0}$

$Z_0 = \int dfe^{-S}$

$S = S_{GP} + \Delta S$

S_{GP} is Gaussian so the first part is computable, but ΔS is not computable.

But ΔS is small $\rightarrow$ perturbation theory.

Cutoffs

$S \rightarrow S_\Lambda$ to deal with divergences.

should be insensitive to the choice of $\Lambda \rightarrow$ obey RGEs (ReLu-net satisfies)

$\Delta S = \int d^{d_{in}}x \left[\lambda f(x)^4 + \kappa f(x)^6\right]$

question: why $k \geqslant 4$

- odd-order coeefficients are zero. Odd-point function must be zero because means of weights and biases are zero or S must have $f \rightarrow -f$ symmetry.

Represent correlation function diagrammatically - Feynman rules

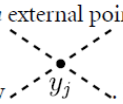
$O(\lambda^l \kappa^m)$ correction to $G^{(n)}$, l 4-pt interaction vertices and m 6-pt interaction vertices.

$G^{(2)}(x_1, x_2) = K(x_1, x_2) \ \rightarrow \ S = S_G + \Delta S$

$$G^{(n)}(x_1, \ldots, x_n) = \frac{\int df f(x_1) \ldots f(x_n) \left[1 - \int d^{d_{in}}x g_k f(x)^k + O(g_k^2)\right] e^{-S_{GP}/Z_{GP,0}}}{\int df \left[1 - \int d^{d_{in}}x g_k f(x)^k + O(g_k^2)\right] e^{-S_{GP}/Z_{GP,0}}}$$

- Feynman rules

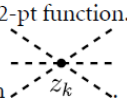
(a) For each of the n external points x_i, draw

(b) For each y_j, draw  . For each z_k, draw .

(c) Determine all ways to pair up the loose ends associated to x_i's, y_j's, and z_k's. This will yield some number of topologically distinct diagrams. Draw them with dashed lines.

(d) Write a sum over the diagrams with an appropriate combinatoric factor out front, which is the number of ways to form that diagram. Each diagram corresponds to an analytic term in the sum.

(d1) Throw away any diagram that has a component with a λ- or κ correction to the 2-pt function.

(e) For each diagram, write $-\int d^{d_{in}} y_j \lambda$ for each , and $-\int d^{d_{in}} z_k \kappa$ for each  .

(f) Write $K(u, v)$ for each .

(g) Throw away any terms containing vacuum bubbles.

- Feynman lines from --- to - - $\Leftrightarrow$ propagator of $S_G \rightarrow$ 2-pt function
- Expand to leading order in λ and κ

question eq.70 why linear? Taylor expansion?

Symmetries and Interaction strength coefficient (also called coupling constants in QFT)

- Technical naturalness: a coupling g appearing in ΔS may be small relative to Λ if a symmetry is restored when g is set to zero.

$\Delta g = 0$ if S has a symmetry.

$g \to 0 \Rightarrow \Delta g \to 0$

- Technical naturalness for NGP neural networks:

$\lambda(x) = \bar{\lambda} + \delta\lambda(x)$

if there is symmetry T in $\delta\lambda \to 0$ limit

then $\frac{\delta\lambda}{\lambda} \leqslant 1$

which means the couplings in NGPs are (near) constants. (not proven but can be tested: coupling are effectively constants in Gauss-net)

$T : K(x, y)$ is translation invariant

$\lambda,\ \kappa$ is constants by symmetry at GP and Technical Naturalness.

Independence of single-layer networks

$S_{GP} = S^b_{GP} + S^W_{GP} = \frac{1}{2}\int d^{d_{in}}x d^{d_{in}}y\,[f_b(x)\Xi_b(x,y)f_b(y) + f_W(x)\Xi_W(x,y)f_W(y)]$

$K(x, y) = K_b(x, y) + K_W(x, y)$

$\Delta S = \Delta S_b + \Delta S_W = \Delta S_W$

similarly $G^{(n)}(x_1, \ldots, x_n) = \frac{\int df f(x_1)\ldots f(x_n)\left[1-\int d^{d_{in}}x g_k f_W(x)^k + O(g_k^2)\right]e^{-S_{GP}/Z_{GP,0}}}{\int df\left[1-\int d^{d_{in}}x g_k f_W(x)^k + O(g_k^2)\right]e^{-S_{GP}/Z_{GP,0}}} \Rightarrow$

Feynman diagrams

dashed lines in Fetnman diagram correspond to $K_W(u, v)$

Experiments of NGP

$\lambda = \frac{K(x_1,x_2)K(x_3,x_4)+K(x_1,x_3)K(x_2,x_4)+K(x_1,x_4)K(x_2,x_3)-G^{(4)}(x_1,x_2,x_3,x_4)}{24\int d^{d_{in}}y K_W(x_1,y)K_W(x_2,y)K_W(x_3,y)K_W(x_4,y)} = \frac{K(x_1,x_2)K(x_3,x_4)+K(x_1,}{}$

compute coupling constants

- if λ is constant

$$\lambda = \frac{K(x_1,x_2)K(x_3,x_4)+K(x_1,x_3)K(x_2,x_4)+K(x_1,x_4)K(x_2,x_3)-G^{(4)}(x_1,x_2,x_3,x_4)}{24\int d^{d_{in}}y K_W(x_1,y)K_W(x_2,y)K_W(x_3,y)K_W(x_4,y)}$$

- if λ is not constant

$$\lambda(y) = \bar{\lambda} + \delta\lambda(y)$$

$$\bar{\lambda} = \frac{K(x_1,x_2)K(x_3,x_4)+K(x_1,x_3)K(x_2,x_4)+K(x_1,x_4)K(x_2,x_3)-G^{(4)}(x_1,x_2,x_3,x_4)}{24\int d^{d_{in}}y\ \Delta_{1234y}} - \frac{\int d^{d_{in}}y\ \delta\lambda(y)\ \Delta_{1234y}}{\int d^{d_{in}}y\ \Delta_{1234y}}$$

Where $\Delta_{1234y} = K_W(x_1,y)K_W(x_2,y)K_W(x_3,y)K_W(x_4,y)$

- if $\delta\lambda(y)$ is small

$$\lambda \backsimeq \bar{\lambda} \backsimeq \mathrm{mean}(\lambda_m(x_1,x_2,x_3,x_4))$$

- for κ

$$\delta'(x_1,\ldots,x_6) := G^{(6)}\ (x_1,\ldots,x_6) - \sum_{15\text{ combinations}} \left[K(x_i,x_j)K(x_k,x_l)K(x_i,x_j) - 24\int d^{d_{in}}y\lambda K_W(x_i,y)K_W(x_j,y)K_W(x\right.$$

$$\delta(x_1,\ldots,x_6) := \frac{\delta'(x_1,\ldots,x_6)}{G^{(6)}\ (x_1,\ldots,x_6)}$$

$\delta(x_1,\ldots,x_6) \to 0$ in large N and large Λ limits

Results

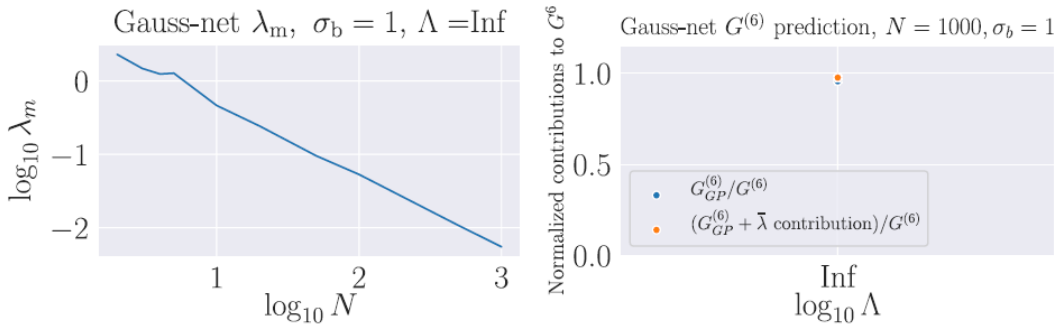


Figure 4. (left) Measured λ_m tensor elements at different widths for Gauss-net with infinite cutoff. (right) GP prediction alone, and GP prediction + $\bar{\lambda}$ correction of 6-pt function $G^{(6)}$ for Gauss-net at width $N = 1000$, normalized with respect to $G^{(6)}$.

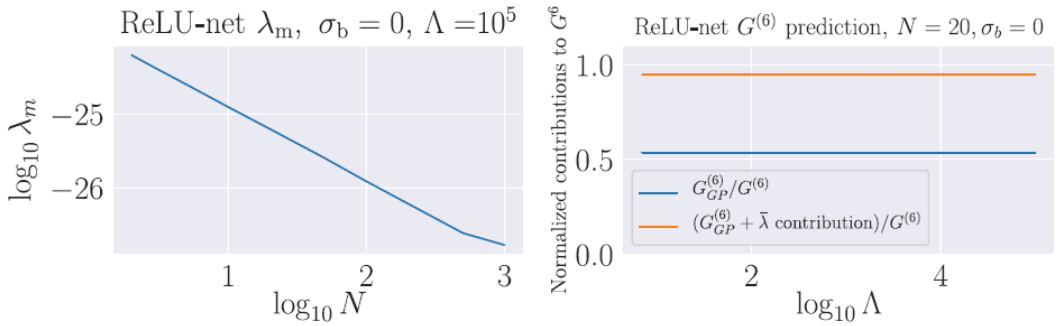


Figure 5. (left) Measured λ_m tensor elements at different widths for ReLU-net with fixed cutoff. (right) GP prediction alone, and GP prediction + $\bar{\lambda}$ correction of 6pt function $G^{(6)}$ for ReLU-net at width $N = 20$, normalized with respect to $G^{(6)}$.

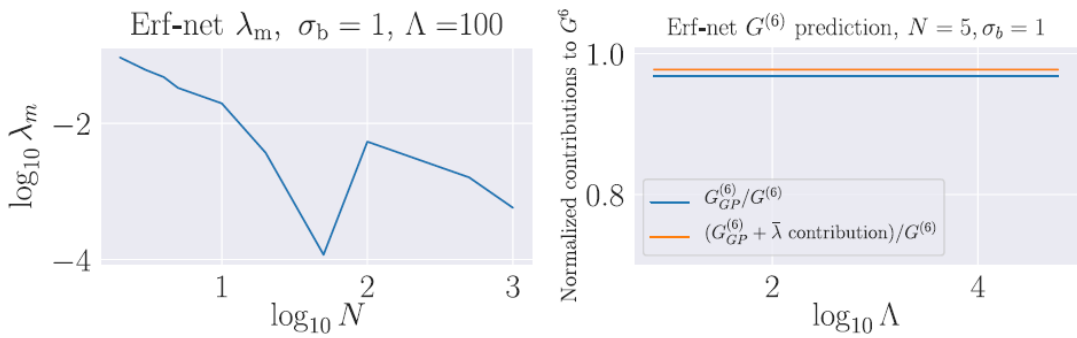


Figure 6. (left) Measured λ_m tensor elements at different widths for Erf-net with fixed cutoff. (right) GP prediction alone, and GP prediction + $\bar{\lambda}$ correction of 6pt function $G^{(6)}$ for Erf-net at width $N = 5$, normalized with respect to $G^{(6)}$.

Experiments of fitting EFT parameters

Compute deviation from GP in 4-pt function

$\Delta S_W = \int d^{d_{in}} x \lambda(x) f_W(x)^4$

design three functional forms of λ

$\Delta G^{(4)}_{EFT} = \lambda_0 T_0(x1,..,x_4) + \lambda_2 T_2(x1,..,x_4) + \lambda_{NL} T_{NL}(x1,..,x_4)$

- $T_0(x1,..,x_4) = 24 \int d^{d_{in}} x K_W(x_1,x) K_W(x_2,x) K_W(x_3,x) K_W(x_4,x)$
- $T_2(x1,..,x_4) = 24 \int d^{d_{in}} x x^2 K_W(x_1,x) K_W(x_2,x) K_W(x_3,x) K_W(x_4,x)$
- nonlocal term:
  $T_{NL}(x1,..,x_4) = 8$
  $\int d^{d_{in}} x d^{d_{in}} y \, [K_W(x_1,x) K_W(x_2,x) K_W(x_3,y) K_W(x_4,y) + K_W(x_1,x) K_W(x_2,y)$
  $+ K_W(x_1,x) K_W(x_2,y) K_W(x_3,y) K_W(x_4,x)]$

Predict $\Delta G^{(4)}$

Experimental measurements of deviation

$\Delta G^{(4)}_{exp}$ is averaged over 10^7 experiments

thus we can find values of λ to minimize the mean squared error between $\Delta G^{(4)}_{exp}$ and $\Delta G^{(4)}_{EFT}$

test effectiveness of measurement value by making predictions for the test set

Wilsonian Renormalization

How couplings change with Λ

- goal of RG: solve divergences arising from integrals over the space of inputs

using cutoff: $\Delta S_\Lambda = \int_{-\Lambda}^{\Lambda} d^{d_{in}} x \sum_{l \leqslant k} g_{\mathcal{O}_l}(\Lambda) \mathcal{O}_l$

- How to understand cutoff: MNIST maximum brightness and darkness scale.
- We can extract values of $g_{\mathcal{O}_l}(\Lambda)$ from experimental result of $G^{(n)}(x_1, \ldots, x_n)$
- ΔS_Λ should satisfy $\frac{dG^{(n)}(x_1,\ldots,x_n)}{d \log \Lambda} = 0$
- β -function: $\beta(g_{\mathcal{O}_l}) := \frac{d(g_{\mathcal{O}_l}(\Lambda))}{d \log \Lambda}$
- RG flow: flow in the couplings induced by varying Λ
- Along a direction of flow, couplings can be irrelevant (decrease), relevant (increase), marginal (same).
- Since $[\lambda] = -d_{in} - 2[K], \quad [\kappa] = -d_{in} - 3[K]$. When $[K] > 0$, κ decrease more quickly than λ

β - functions

- Kernels have model independent term and model dependent term

$$K(x,x') = \alpha + \varsigma(x,x')$$

- Thus correlation functions can be written in combination of model independent and dependent terms

$$G^{(4)}\left(x_1,\ldots,x_4\right) = \gamma_{4,0} + \varrho_{4,0} - \int_{-\Lambda}^{\Lambda} d^{d_{in}}x(\gamma_{4,\lambda} + \varrho_{4,\lambda}) - \kappa \int_{-\Lambda}^{\Lambda} d^{d_{in}}x(\gamma_{4,\kappa} + \varrho_{4,\kappa})$$

$$G^{(6)}\left(x_1,\ldots,x_6\right) = \gamma_{6,0} + \varrho_{6,0} - \int_{-\Lambda}^{\Lambda} d^{d_{in}}x(\gamma_{6,\lambda} + \varrho_{6,\lambda}) - \kappa \int_{-\Lambda}^{\Lambda} d^{d_{in}}x(\gamma_{6,\kappa} + \varrho_{6,\kappa})$$

- take derivative and cancel out the last term (κ is negligible)

$$\frac{\partial\lambda}{\partial\log\Lambda}\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left[\frac{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{6,\lambda}+\varrho_{6,\lambda}\right)}{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{6,\kappa}+\varrho_{6,\kappa}\right)} - \frac{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{4,\lambda}+\varrho_{4,\lambda}\right)}{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{4,\kappa}+\varrho_{4,\kappa}\right)}\right]$$
$$+\lambda\left[\frac{1}{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{6,\kappa}+\varrho_{6,\kappa}\right)}\frac{\partial(\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{6,\lambda}+\varrho_{6,\lambda}\right))}{\partial\log\Lambda}\right.$$
$$\left.-\frac{1}{\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{4,\kappa}+\varrho_{4,\kappa}\right)}\frac{\partial(\int_{-\Lambda}^{\Lambda} d^{d_{\text{in}}}x\left(\gamma_{4,\lambda}+\varrho_{4,\lambda}\right))}{\partial\log\Lambda}\right] = 0.$$

RG in neural networks

- Because bias term is Gaussian

$$\Delta S_{W,\Lambda} = \int_{-\Lambda}^{\Lambda} d^{d_{in}}x \sum_{l\leqslant k} g_{\mathcal{O}_l}(\Lambda)\mathcal{O}_l$$

$$[\lambda] = -d_{in} - 2[K_W], \quad [\kappa] = -d_{in} - 3[K_W].$$

- For network architecture

$$K_W(x,x') = \varsigma(x,x')$$

$$G^{(4)}\left(x_1,\ldots,x_4\right) = \gamma_{4,0} + \varrho_{4,0} - \int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{4,\lambda} - \kappa \int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{4,\kappa}$$

$$G^{(6)}\left(x_1,\ldots,x_6\right) = \gamma_{6,0} + \varrho_{6,0} - \int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{6,\lambda} - \kappa \int_{-\Lambda}^{\Lambda} d^{d_{in}}x\varrho_{6,\kappa}$$

- RG equations

$$\frac{\partial\lambda}{\partial\log\Lambda}\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\left[\frac{\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{6,\lambda}}{\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{6,\kappa}}-\frac{\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{4,\lambda}}{\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{4,\kappa}}\right]+\lambda\left[\frac{1}{\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{6,\kappa}}\frac{\partial(\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{6,\lambda})}{\partial\log\Lambda}\right.$$
$$\left.-\frac{1}{\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{4,\kappa}}\frac{\partial(\int_{-\Lambda}^{\Lambda}d^{d_{\rm in}}x\,\varrho_{4,\lambda})}{\partial\log\Lambda}\right]=0\,.$$

RG flows in Gauss-net

$$K_{\rm Gauss}(x,x')=\sigma_b^2+\sigma_W^2\exp\left[-\frac{\sigma_W^2\,|x-x'|^2}{2d_{in}}\right]$$

- $K_{\rm Gauss}$ converges for large integration value, thus $\Lambda\to\infty$
- for small x and x', $[K_W]<0$

RuLU-net

$$K_W(x,x')=\varsigma(x,x')=h_1(x,x')h_2(\theta)$$

$$h_1(x,x')=\sqrt{(\sigma_b^2+\frac{\sigma_W^2}{d_{in}}x\cdot x)(\sigma_b^2+\frac{\sigma_W^2}{d_{in}}x'\cdot x')}$$

$$h_2(\theta)=(sin\theta+(\pi-\theta)cos\theta)$$

- $[K_W(x,x')]=2$
- $[\lambda]=-d_{in}-4,\ [\kappa]=-d_{in}-6$

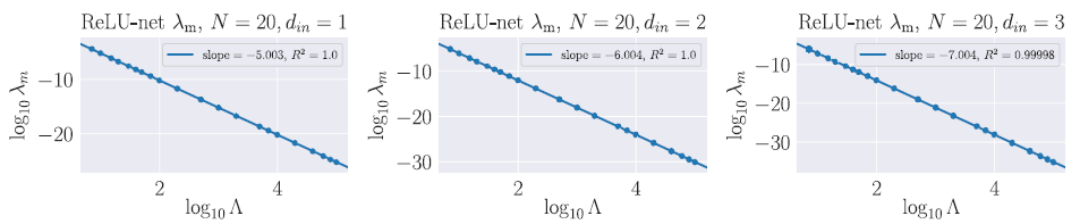


Figure 7. RG equation for ReLU-net: $\log(\lambda_m)$ as a function of $\log(\Lambda)$ for $d_{\rm in}=1,2,3$.

$d_{in}=3$

$$\text{for ReLU-net, }\sigma_W^2,\sigma_b^2=1,0$$
$$K(x_1,x_2)=\frac{1}{2d_{in}}|x_1||x_2|$$

$$\lambda_m = \frac{\frac{3}{16}|x_1||x_2||x_3||x_4| - G^{(4)}}{24\int_{-\Lambda}^{\Lambda} d^2y|x_1||x_2||x_3||x_4||y|^4} = \frac{\frac{3}{16}|x_1||x_2||x_3||x_4| - G^{(4)}}{24|x_1||x_2||x_3||x_4|\int_{-\Lambda}^{\Lambda}\int_{-\Lambda}^{\Lambda} dy_1dy_2(y_1^2+y_2^2)^2}$$

$$\lambda_m = \frac{\frac{3}{36}|x_1||x_2||x_3||x_4| - G^{(4)}}{24\int_{-\Lambda}^{\Lambda} d^3y|x_1||x_2||x_3||x_4||y|^4} = \frac{\frac{3}{36}|x_1||x_2||x_3||x_4| - G^{(4)}}{24|x_1||x_2||x_3||x_4|\int_{-\Lambda}^{\Lambda}\int_{-\Lambda}^{\Lambda}\int_{-\Lambda}^{\Lambda} dy_1dy_2dy_3(y_1^2 -}$$

Appendix B

Supporting information for chapter 5

This is the notes taking from arXiv:1608.08225v4

Introduction

$$\mathrm{x} \sim \mathcal{N}(\mu, \sigma^2)$$

Deep learning algorithm has seen significant success since the past decades. However, what did the neural network does during the training process is obscure to human. The complexity of its structure makes deep learning hard to be intepreted. One thing interesting to us is that, despite of large number of possible functions, why can a neural network approximate the funciton so well with relatively much fewer parameters than the dimensions and size of its input data?

This paper manages to develop some physical insights on deep learning and its success. Here, we illustrate several inspiring ideas from it.

Neural Network is trying to approximate the Hamiltonian of the data.

$$H_y(\mathrm{x}) \equiv -\ln p(\mathrm{x} \mid y),$$
$$\mu_y \equiv -\ln p(y).$$

The Hamiltonian gives Maxwell-Boltzman distribution of x given y. Combining with Bayes rule, the distribution of y given x is

$$p(y \mid \mathbf{x}) = \frac{1}{N(\mathbf{x})} e^{-[H_y(\mathbf{x})+\mu_x]}$$

where,

$$N(\mathrm{x}) \equiv \sum_y e^{-[H_y(\mathbf{x})+\mu_y]}$$

If the last layer is softmax layer,

$$\tilde{\sigma}(\mathrm{x}) \equiv \frac{e^{\mathrm{x}}}{\sum_i e^{y_i}}$$

then the layer before it computes the Hamiltonian vector $\mathbf{H}(\mathbf{x})$. The μ -vector simply becomes the bias term in the final layer.

The Hamiltonian that a neural network approximates is simple, specifically, polynomial. Let's expand a Hamiltonian as a power series,

$$H_y(\mathrm{x}) = h + \sum_i h_i x_i + \sum_{i \le j} h_{ij} x_i x_j + \sum_{i \le j \le k} h_{ijk} x_i x_j x_k + \cdots$$

Multiplication Gate

Continuous input

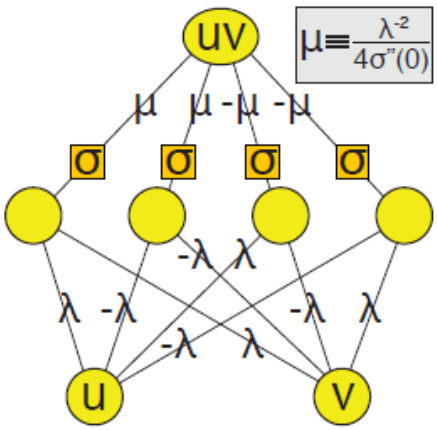
Taylor expand the activation around the origin (if the inputs are large, we could make them small by scaling):

$$\sigma(u) = \sigma_0 + \sigma_1 u + \sigma_2 \frac{u^2}{2} + \mathcal{O}\left(u^3\right)$$

Then we can approximate the multiplication,

$$m(u,v) \equiv \frac{\sigma(u+v) + \sigma(-u-v) - \sigma(u-v) - \sigma(-u+v)}{a_\sigma}$$

Continuous multiplication gate:



Binary input

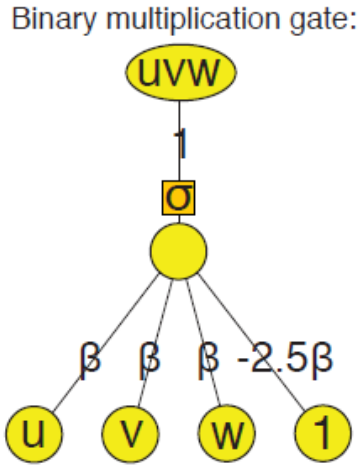
The non-trival terms are that all variables have at most the first order.

$$H_y(\mathrm{x}) = h + \sum_i h_i x_i + \sum_{i \le j} h_{ij} x_i x_j + \sum_{i \le j \le k} h_{ijk} x_i x_j x_k + \cdots$$

The Hamiltonian is a finite series with 2^n terms.

The multiplication approximator could be constructed as:

$$\prod_{i \in K} x_i = \lim_{\beta \to \infty} \sigma\left[-\beta\left(k - \frac{1}{2} - \sum_{x \in K} x_i\right)\right]$$



Neural network is able to approximate polynomials

Corollary: For any given multivariate polynomial and any tolerance $\epsilon > 0$, there exists a neural network of fixed finite size N (independent of ϵ) that approximates the polynomial to accuracy better than ϵ. Furthermore, N is bounded by the complexity of the polynomial, scaling as the number of multiplications required times a factor that is typically slightly larger than 4.

Hamiltonian

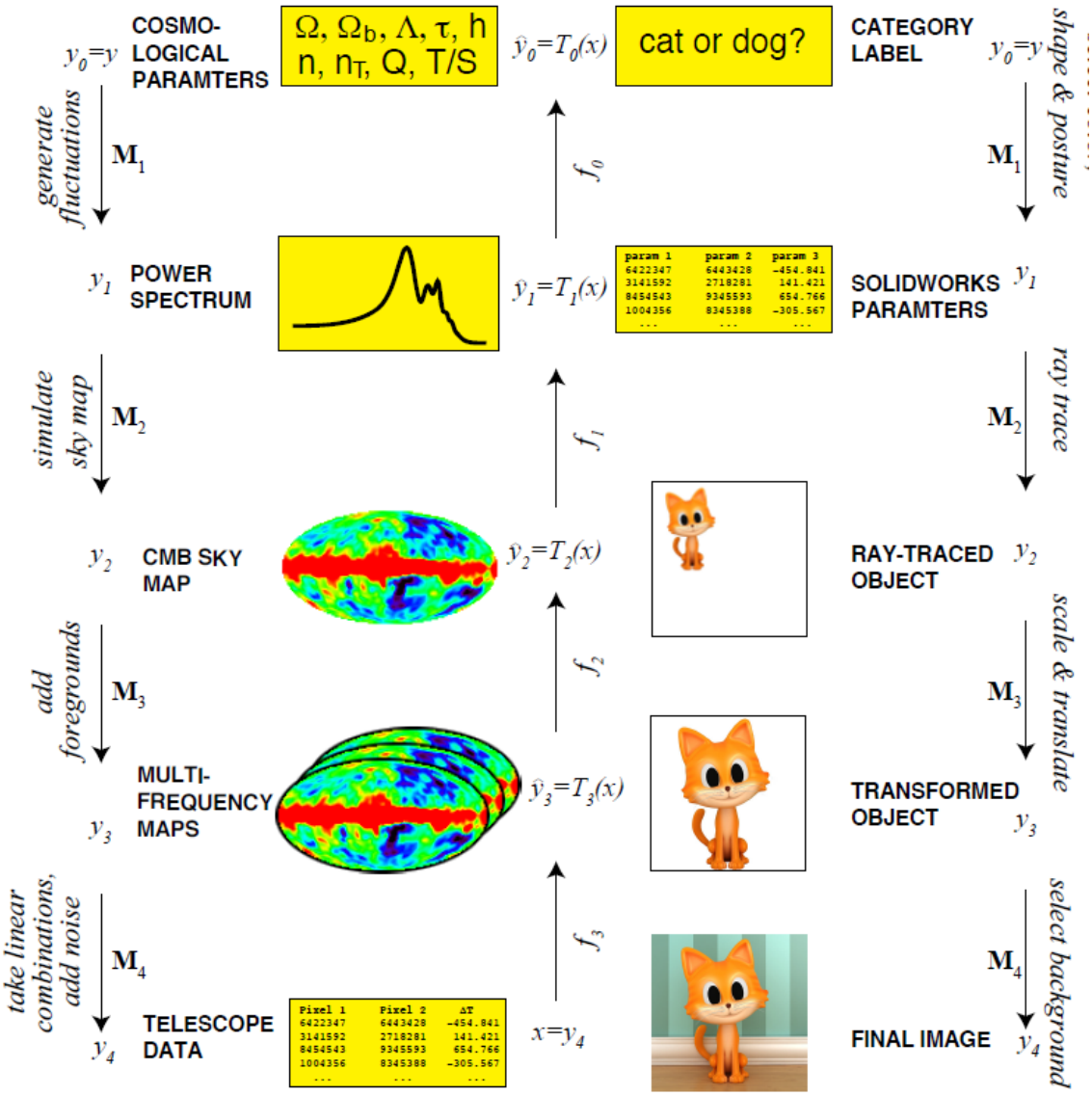
What will the Hamiltonian be in most situation?

- Low polynomial order
- Local
- Symmetry

Deep neural network is more efficient

Data generated by hierarchical process

$$\mathbf{p}_n = \mathbf{M}_n \mathbf{M}_{n-1} \cdots \mathbf{M}_1 \mathbf{p}_0$$



Deep learning aims to reverse the hierarchical process

Deep learning inferences the input y from the output x.

- **Sufficient statistics**: $T(x)$ is all the information about y contained in x.

The data processing inequality $[30]$ states that for any function f and any random variables x, y ,

$$I(x,y) \geq I(x, f(y))$$

where I is the mutual information:

$$I(x,y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

A sufficient statistic $T(x)$ is a function $f(x)$ for which " $\geq$ " gets replaced by "=", i.e., a function retaining all the information about y.

- **Minimal sufficient statistic**: sufficient statistic of all other sufficient statistic. $T_*(y) = f(T(y))$
- **Theorem**: Given a Markov chain described by our notation above, let T_i be a minimal sufficient statistic of $P(y_i \mid y_n)$. Then there exists some functions f_i such that $T_i = f_i \circ T_{i+1}$. More casually speaking, the generative hierarchy of Figure 3 can be optimally reversed one step at a time: there are functions f_i that optimally undo each of the steps, distilling out all information about the level above that was not destroyed by the Markov process.
- **Corollary**: Define the function $f_0(T_0) = P(y_0 \mid T_0)$ and $f_n = T_{n-1}$. Then
$$P(y_0 \mid y_n) = (f_0 \circ f_1 \circ \cdots \circ f_n)(y_n)$$

Distillation could be near sufficient

If not being strict sufficient could reduce complexity of the Hamiltonian to a great extent, this procedure called renormalization is a useful method to approximate distillation.

The working definition of RG consists of a vector y of random variables, a course-graining operation R and a requirement that this operation leaves Hamiltonian invariant except for parameter changes.

$$H_y(x) \to H'(R(x)) = H_{r(y)}(R^n(x))$$

- Renormalization flow: $R^n = R \circ R \circ \cdots R$
- Renormalization is a supervised learning

Example of renormalization:

The Hamiltonian of a **simple image model** is:

$$H_y(\phi) = \int \left[y_0 \phi^2 + y_1 (\nabla \phi)^2 + y_2 \left(\nabla^2 \phi\right)^2 + \cdots \right] d^2 r$$

$\phi(\mathbf{r})$ describes the image by intensity field. In neural network, the derivatives are approximated by differences. The translational symmetry of the image implies a convnet. Thus the course graining operation is replacing each block of $b \times b$ pixels by its average and dividing the result by b^2, the Hamiltonian retains the form of previous equation but with the parameters y_i replaced by

$$y_i' = b^{2-2i} y_i$$

This means that all parameters y_i with $i \geq 2$ decay exponentially with b as we repeatedly renormalize and b keeps increasing, so that for modest b, one can neglect all but the first few y_i

's. The neural network now becomes finite if we only consider relevant operators and ignore irrelavant ones.

Cost of Flattening

- Flattening cost

$\mathbf{f}_{\mathrm{c}}^{l}$ is a **neuron-efficient flattening** if:

$$C_n(\mathbf{f}, \ell, \epsilon) \equiv \min_{\mathbf{f}_{\mathrm{c}}^{l}} \frac{N_n\left(\mathbf{f}_{\epsilon}^{\ell}\right)}{N_n(\mathbf{f})} < 1$$

where $\mathbf{f}$ is the original deep neural network.

$\mathbf{f}_{\mathrm{c}}^{l}$ is a **synapse-efficient flattening** if:

$$C_s(\mathbf{f}, \ell, \epsilon) \equiv \min_{\mathbf{f}_{c}^{\ell}} \frac{N_s\left(\mathbf{f}_{\epsilon}^{\ell}\right)}{N_s(\mathbf{f})} < 1$$

No-flattening theorem for linear models

Neuron efficiency satisfies.

- factorization of matrix

fully flattening a network that performs a FFT of n variables increases the synapse count N_s from $\mathcal{O}(nlogn)$ to n.

- matrix multiplication

from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^{2.3728639})$

- low rank matrix multiplication

$\mathbf{A}$ is a rank k $n \times n$ matrix. $\mathbf{A} = \mathbf{BC}$ where $\mathbf{B}$ is $n \times k$ and $\mathbf{C}$ is $k \times n$. Thus $C_s = n/2k$

- sparse matrix

> Finally, let us consider flattening a network $\mathrm{f} = \mathrm{AB}$, where A and B are random sparse $n \times n$ matrices such that each element is 1 with probability p and 0 with probability $1-p$. Flattening the network results in a matrix $F_{ij} = \sum_k A_{ik} B_{kj}$, so the probability that $F_{ij} = 0$ is $\left(1-p^2\right)^n$. Hence the number of non-zero components will on average be $\left(1-\left(1-p^2\right)^n\right) n^2$, so
> $$C_s = \frac{\left[1-\left(1-p^2\right)^n\right] n^2}{2n^2 p} = \frac{1-\left(1-p^2\right)^n}{2p}$$
> Note that $C_s \leq 1/2p$ and that this bound is asymptotically saturated for $n \gg 1/p^2$. Hence in the limit where n is very large, flattening multiplication by sparse matrices $p \ll 1$ is horribly inefficient.

No-flattening theorem for polynomial model

Multiplication of n variables can be implemented by a flat network with 2^n neurons in the hidden layer.

$$\prod_{i=1}^{n} x_i = \frac{1}{2^n} \sum_{\{s\}} s_1 \ldots s_n \sigma\left(s_1 x_1 + \ldots + s_n x_n\right)$$

which is necessary and sufficient.

Therefore, polynomials are exponentially expensive to flatten. $\mathcal{O}(4n) \rightarrow \mathcal{O}(2^n)$