Statistical mechanics of Deep Learning: A Perturbative solution of a fully connected network and the role of non-linear activation functions

Chalkias, Christos

Statistical mechanics of Deep Learning

A Perturbative solution of a fully connected network and the role of non-linear activation functions

Thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

in

Physics

Author :	Christos Chalkias
Student ID :	s3482685
Supervisor :	Koenraad Schalm
Second corrector :	Subodh Patil

Leiden, The Netherlands, January 31, 2025

Statistical mechanics of Deep Learning

A Perturbative solution of a fully connected network and the role of non-linear activation functions

Christos Chalkias

Instituut-Lorentz, Leiden University

P.O. Box 9500, 2300 RA Leiden, The Netherlands

January 31, 2025

Abstract

Physics has proved to be an inspiration for understanding the inner workings of neural networks, undeniably the most indispensable tool driving the ongoing AI revolution. Ideas from statistical physics and quantum field theory provide excellent candidates for the building blocks of a theory of deep learning. This master thesis tries and shed light on the role of *non-linear activation functions* by using perturbation theory to analyze the partition function of neural networks. For brevity a shallow, 2-layer constant width architecture is studied. Easily interpretable results for restrictive cases of single arithmetic examples show an increase in temperature resistant expressivity of the network. This enables the learning algorithm to train such networks more efficiently.

Contents

1 Preliminaries **3**

1.1 Machine Learning and Physics 3

1.1.1 Supervised Learning 3

1.1.2 Deep Neural Networks 4

1.2 Partition function and statistics 7

1.2.1 The deterministic likelihood hypothesis 8

1.2.2 The uncertainty hypothesis 9

1.2.3 Partition function with sources 11

1.3 Linear Neural Networks 11

1.3.1 Effective theory for Linear neural networks 12

1.3.2 Partition function of a single entry Linear Network 12

2 Nonlinear corrections to $Z(\beta)$ **15**

2.1 Introducing sources 15

2.2 2-Layer partition function 16

2.2.1 Saddle-point approximation of the the J=0 integral 19

2.2.2 Expanding around the stationary point 21

2.3 Norm learning 25
2.3.1 Single hidden layer perturbation 27
2.3.2 Norm learning discussion 33
2.4 Multiple examples 34
2.4.1 Evaluating the source integral 34
2.4.2 First order correction in g 35
2.5 Outlook and future work 36

A Calculations & Experiments 41

Introduction

Artificial intelligence and specifically machine learning are one of those subjects that have expanded from academic circles and is now influencing the world with unprecedented rate. From the conception of the Perceptron * to state of the art architectures like ChatGPT one is presented with storyline that resembles that of how Maxwell's laws eventually gave rise to electronic computers. The only difference being that the former's rate of change has been so rapid that, even though practitioners have achieved monumental milestones, theorists have struggled to keep up.

This accelerated evolution is mostly due to a specific type of machine learning, namely Deep Neural Networks (abbreviated as DNNs). These models serve as function generators, using some 10^{11} parameters [1] to approximate functions, that are so complex that it was once believed that the tasks they correspond to only humans (and possibly not even them) would be able to achieve. One of the first instances of this was alphaGO [2] where a Deep Learning agent was able to beat world champions in the board game GO (only losing one of its 74 recorded games). Despite all this, these deep learning models are currently, for the most part, black boxes. Although its individual constituents are very well understood, *the reason* a DNN computes one function instead of another remains a mystery. It was possibly AlexNet [3] that was one of the first instances in which the world

*Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408.

witnessed this stacking of elementary operation layers (convolutional layers) that gave rise to a model with incredible predictive power and almost zero intuition of how it works.

Here is where physics comes into play. The premise is to be able to craft a theory of DNNs from first principles since physics is equipped with tools and techniques to handle extremely complex systems such as DNNs [4]. The most promising of these is trying to understand DNNs via an effective theory. This is especially helpful in the case -as it usually is with DNNs- of a nonlinear system. One can study the effect of non-linear information propagation through the network by examining a coarse grained description. This is precisely the goal of this master thesis, namely to characterize the improvement of learning of a NN with nonlinear activation functions compared to a linear one.

In chapter 1 an introduction of relevant machine learning and statistical mechanics elements is discussed, following a description of linear DNNs. In chapter 2 the method to describe non-linear activation functions is examined. Here calculations are restricted to a 2 layer network with a cubic activation function. Many calculations are omitted from the main text and make up the contents of Appendix A.

Chapter 1

Preliminaries

In this chapter a brief introduction to the contents of the thesis is presented. Some elements of statistical mechanics, Machine learning and Deep Linear Neural Networks from the perspective of physics [5] are discussed. The notation is set up for the rest of the text and the quantities of importance are presented.

1.1 Machine Learning and Physics

Machine learning is the actualization of Artificial Intelligence through electronic computers. It is a subject that has gained a lot of fame in the last decade due to its remarkable ability to perform tasks once thought impossible. In the information era, it is an indispensable tool for academics and the industry alike.

1.1.1 Supervised Learning

Supervised Learning is concerned with creating a model $\mathcal{M}$ that can learn a concept $\mathcal{C}$ from a dataset of examples and labels $\mathcal{D} = \{x^\mu, y^\mu\}$. We try to learn a concept class by admitting a specific family of functions, through the definition of a model (for example NNs). One hopes that the function

approximation task converges to the underlying function that maps the elements of the dataset as $f(x^\mu) = y^\mu$. The premise is that the concept is more general than the dataset, therefore the latter is but a finite subset of an -in principle- infinite pair of x^μs and y^μs that get mapped to one another by the function f. And so, while supervised learning aims at correctly mapping the elements of the dataset to one another (training performance) its main goal is achieving a generalization beyond the training set (generalization performance). This is the smoking gun proof that the model truly learns the concept class and not only its subset. The way this is done is by choosing a family of parameterized functions f and finding the optimal parameters that succeed in this task

$$f(x; W) \in \mathcal{M} \quad \text{s.t} \quad f(x; W^\star) \in \mathcal{C} \tag{1.1}$$

where W are the model parameters and $W^\star$ the parameters that minimize the generalization error. The choice of the model, and therefore the function family, has to be expressive enough to include the concept that one wants to be learned but at the same time simple enough so that a learning algorithm can efficiently navigate the W space to find the optimal parameters. If this convex optimization problem is solved for a given f the network is said to be trained.

1.1.2 Deep Neural Networks

Great success on this has been achieved with using DNNs as function approximators [6]. DNNs are a specific family of function approximators that consist of stacking multiple computational layers. For the purposes of this thesis one specific type of neural networks is relevant, that of *fully connected feed-forward neural networks*. Figure 1.1 shows a schematic of such a network.

We denote x^μ_a the a-th component of the μ example vector and likewise y^μ_a the a-th component of the μ example label. $W^{(l)}_{ab}$ is the l-th layer $N_l \times n_{l+1}$ weight matrix and $h^{(l)}_a$ is the l-th layer neuron preactivation such that: $h^{(0)}_{a;\mu} = x^\mu_a$, $h^{(1)}_{a;\mu} = \sum_b W^{(1)}_{ab} \phi(x^\mu_b)$ for some choice of an *activation function* ϕ, and so forth. By this definition the network's output will be:

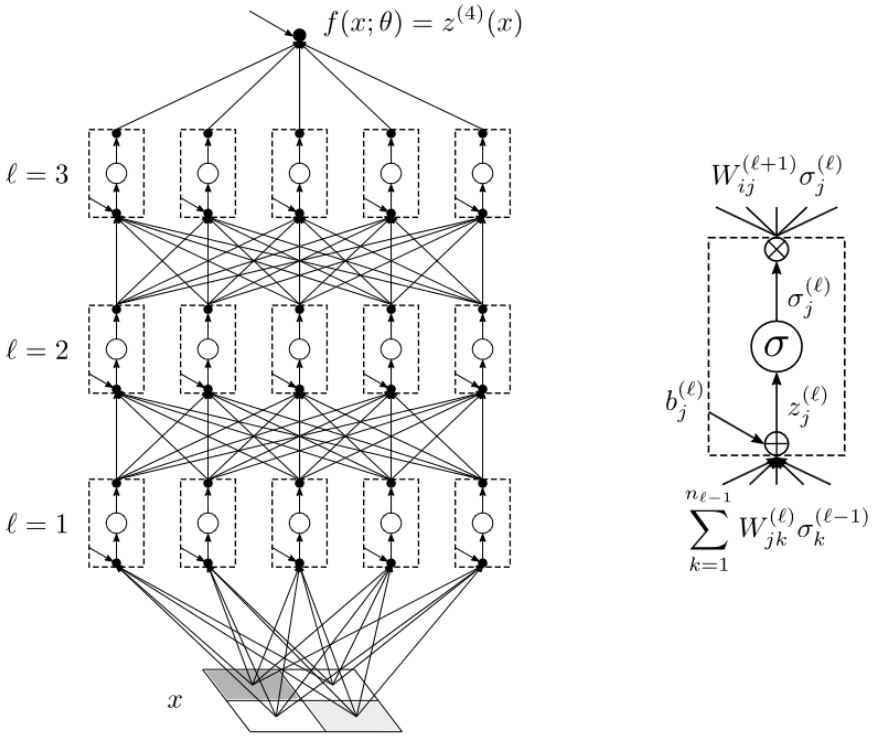


Figure 1.1: *Schematic of how a FCN operates in propagating the information of the input to form the ouput. In this specific example an input x is passed through a 4 layer NN to form the output $f(x,\theta)$. The information of the input is transformed through iterative vertical and horizontal stacking of elementary information proccesing units called neurons (left). Neurons take as an input a weighted sum of the previous neuron activations (or of the input for the first layer) and convert it through an activation function $\sigma(\cdot)$ into their own activation, which is then passed along the next layer. Source: [7]*

$$f_a\left(x^\mu; W^{(L)}, W^{(L-1)}, \cdots, W^{(1)}\right) = \sum_{b=1}^{N_l} W_{ab}^{(L)} \phi\left(h^{L-1}\left(W^{(L-1)}, \cdots, W^{1)}\right)\right) = h_a^{(L)} \tag{1.2}$$

The way the system is set up there is no bias term b_a^L *. Such networks admit a well defined limit when the width of its layers tends to infinity [8] where their behavior as well as its dynamics of training can be completely characterized [9]. However, infinite width models neither correspond to realistic, real-world models, nor do they outperform their finite width counterparts [10] as peak performance is achieved in finite widths. The usual approach is to perturb observables around the infinite width limit by supplementing $\mathcal{O}(1/N^r)$ for some power of r corrections to the observables of the network. These networks are usually initialised by drawing their parameters from zero mean Gaussian distributions

$$p\left(W_{ij}^{(l)}\right) = \sqrt{\frac{N_{l-1}}{2\pi C_W^{(l)}}} \exp\left[-\frac{N_{l-1}}{2C_W^{(l)}}\left(W_{ij}^{(l)}\right)^2\right] \tag{1.3}$$

and therefore their variances will be given by

$$\mathbb{E}\left[W_{i_1 j_1}^{(l')} W_{i_2 j_2}^{(l)}\right] = \int \mathcal{D}\mathcal{W}\, W_{i_1 j_1}^{(l')} W_{i_2 j_2}^{(l)} p\left(W_{i_1 j_1}^{(l')}\right) p\left(W_{i_2 j_2}^{(l)}\right) = \delta_{l' l}\delta_{i_1 i_2}\delta_{j_1 j_2}\frac{C_W^{(l)}}{N_{l-1}} \tag{1.4}$$

Information propagates in the network by the iterative application of the action of the weights plus the activation function of each layer. So while the I/O function of the network is described by eq. (1.2) the layerwise transformation of the input is given by:

$$h_{a,\mu}^{(l+1)} = \sum_{b=1}^{N_l} W_{ab}^{(l)} \phi_l(h_{b,\mu}^{(l)}(x^\mu)) \tag{1.5}$$

with $h_{a,\mu}^{(0)} \equiv x_a^\mu$ again being the input of the neural network. The way this is set up one might be quick to rephrase this into the language of physics as a boundary condition problem, with the

*This is partially in line with what is done in practice when training such NNs. The biases are initialized to zero without concern for neuron indistinguishability since the symmetry is broken by the fact that the weights are sampled from a (normal) random distribution

Version of January 31, 2025– Created July 12, 2025 - 19:12

boundaries being the elements of the dataset $\mathcal{D}$. Then by finding the optimal configuration of weights one finds the system that best obeys the family of boundary conditions. This what is done in physics practice where boundary conditions are usually set (square integrability, asymptotic conditions) and the possible physical systems are examined and contrasted with observation. For this and other reasons, physics machinery can be proven very effective for describing such systems. The most notable example being the statistical mechanical study of ensembles.

1.2 Partition function and statistics

We can associate a FCN with a partition function Z by introducing a functional $E[\mathcal{W}]$ similar to the energy in statistical mechanics, then:

$$Z(\beta) = \int \mathcal{D}\mathcal{W} e^{-\beta E[\mathcal{W}]} \tag{1.6}$$

an obvious choice functional E can be the square error of the system $E \sim \left(f(x^\mu) - y^\mu\right)^2$. Minimizing the energy would in turn mean minimizing the error of the neural network. It is also common practice to include a regularization term $\sim \mathcal{W}^2$ to bias towards smaller weight values. The partition function encodes the probability of the system to exhibit the correct boundary conditions, and therefore possessing a 100% accuracy on the mapping task since the posterior distribution of the networks output can be decomposed into the likelihood times the prior (see A.1.1)

$$p(h^{(L)}|x^\mu, y^\mu) = \frac{p(y^\mu|h^{(L)}, x^\mu)p(h^{(L)}|x^\mu)}{p(y^\mu|x^\mu)} \Rightarrow \tag{1.7}$$

$$p(y^\mu|x^\mu) = \int dh^{(L)} p(y^\mu|h^{(L)}, x^\mu)p(h^{(L)}|x) \tag{1.8}$$

The prior distribution takes the product form:

$$p(h^{(L)}|x) = \prod_{l=1}^{L} p\left(h^{(l)}|h^{(l-1)}\right) \tag{1.9}$$

Version of January 31, 2025– Created July 12, 2025 - 19:12

where

$$p(h^{(l)}|h^{(l-1)}) = \int \mathcal{D}W^{(l)} p\left(W^{(l)}\right) \delta\left(W^{(l)}h^{(l-1)} - h^{(l)}\right) \tag{1.10}$$

$$= \int \mathcal{D}W^{(l)} \frac{1}{\sqrt{2\pi\frac{\sigma_l^2}{N_l}}} \exp\left(-\frac{{W^{(l)}}^2}{\sqrt{2\pi\frac{\sigma_l^2}{N_l}}}\right) \int \mathcal{D}q \exp\left(-iq^{(l)}W^{(l)}h^{(l-1)} - iq^{(l)}h^{(l)}\right) \tag{1.11}$$

And therefore for the complete prior, using eq. (1.9)

$$p(h^{(L)}|x) = \int \mathcal{D}\mathcal{W} \exp\left(-\frac{{W_{ab}^{(l)}}^2}{\sqrt{2\pi\frac{\sigma_l^2}{N_l}}}\right) \int \mathcal{D}q \exp\left(-iq_a^{(l)}W_{ab}^{(l)}h_b^{(l-1)} - iq_a^{(l)}h_a^{(l)}\right) \tag{1.12}$$

Here the notation has been eased, but the following are implied

$$\mathcal{D}\mathcal{Q} = \prod_{l=1}^{L} \mathcal{D}q = \prod_{l=1}^{L} dq_a^l \tag{1.13}$$

$$\mathcal{D}\mathcal{W} = \prod_{l=1}^{L} \frac{\mathcal{D}W^{(l)}}{\sqrt{2\pi\frac{\sigma_l^2}{N_l}}} = \prod_{l=1}^{L}\prod_{a=1}^{N_l}\prod_{b=1}^{N_l} \frac{\mathcal{D}W_{ab}^{(l)}}{\sqrt{2\pi\frac{\sigma_l^2}{N_l}}} \tag{1.14}$$

$$W^{(l)}h^{(l-1)} = \prod_{b=1}^{N_l} W_{ab}^{(l)}h_b^{(l-1)} \tag{1.15}$$

with N_l being the width of layer l and L being the total length of the network.

1.2.1 The deterministic likelihood hypothesis

In the $\beta \to \infty$ limit eq. (1.12) can be supplemented with an extra $\exp\left(\frac{q_a q_a}{2\beta}\right)$ term that vanishes. One then quickly sees that if it is demanded of the system that $h^{(l)} = y^\mu$ the $\mathcal{D}q$ integral is not but the Hubbard-Stratonovich transformation on $W^{(l)}h^{(l-1)} - h^{(l)}$ and so the prior becomes:

$$p(h^{(L)}|x) = \int \mathcal{D}\mathcal{W} \exp\left(-\frac{{W_{ab}^{(l)}}^2}{\sqrt{2\pi\frac{\sigma_l^2}{N_l}}}\right) \exp\left(-\frac{\beta}{2}\left(W_{ab}^{(l)}h_b^{(l-1)} - y_a^\mu\right)^2\right) \tag{1.16}$$

This can be rewritten in the form of (1.6) with the energy being the square loss function, supplemented with an L_2 norm and since the last layer has been equated with the label, this falls under the

deterministic hypothesis regime and therefore the likelihood is a delta function $\delta(h^{(l)} - y)$ and so:

$$p(y^\mu|x^\mu) = \int dh^L p(y^\mu|y^\mu, x) p(h^L|x) = \int \mathcal{D}\mathcal{W} e^{-\beta E[\mathcal{W}]} \tag{1.17}$$

One can build an ensemble of networks with different examples μ and treat the full partition function as $Z(\beta) = \prod_\mu Z_\mu(\beta)$. However this gives rise to an uncorrelated ensemble where each network is matched with a different set of $\{x^\mu, y^\mu\}$. It is, however, known that the power of Deep Learning lies in the fact that cross correlations of the sort $x_a^\mu x_a^\nu$, $x_a^\mu x_b^\mu$, $x_a^\mu y_a^\mu$ play a major role in shaping the output distribution and finding the right concept. There is a caveat here. This information can never be encoded into the prior. Take eq. (1.16) for example. In order to arrive to this equation, one has to explicitly use the low temperature limit ($\beta \to \infty$) and the assumption that $h_a^{(L)} = y_a$. This is equivalent to admitting a *certainty hypothesis* that the network always outputs y^μ when given x^μ as its input. It should be evident that such a condition cannot hold for any μ since a single configuration of weights and network architecture cannot guarantee a perfect match between example and label.

1.2.2 The uncertainty hypothesis

The way to fix this is by admitting to an *uncertain likelihood hypothesis*. Take eq (1.8). If one substitues the prior $p(y^\mu|h^{(L)}, x^\mu)$ with a delta function $\delta(h^{(L)} - y^\mu)$ the equation is tautologically true.

$$p(h^L|x, y) = \frac{\prod_{a=1}^{N_l} \delta\left(h_a^{(L)} - y\right) p\left(h^{(L)}|x\right)}{p(y|x)} \tag{1.18}$$

Integrating over the last layers preactivation

$$\int dh^L p(h^{(L)}|x, y) = \frac{1}{p(y|x)} \int dh^L \prod_{a=1}^{N_l} \delta(h_a^{(L)} - y) p\left(h^{(L)}\middle|x\right) = \frac{p(y|x)}{p(y|x)} = 1 \tag{1.19}$$

Relaxing the certainty hypothesis is key to incorporating a dataset with $P > 1$. One can introduce an *uncertain likelihood* $\mathcal{L}_\gamma$ that reduces to the certainty hypothesis at $\gamma = 0$. This can be

$$p\left(h^{(L)}|x, y\right) = \frac{1}{p(y|x)} \prod_{a=1}^{N_l} \frac{1}{\sqrt{2\pi\gamma^2}} e^{\frac{-1}{2\gamma^2}\left(h_a^{(L)} - y_a\right)^2} p\left(h^{(L)}\middle|x\right) \tag{1.20}$$

and integrating again over the last layer preactivations we have

$$\int dh^{(L)} p\left(h^{(L)}\middle| x,y\right) = \int dh^{(L)} \frac{1}{p(y|x)} \prod_{a=1}^{N_l} \frac{1}{\sqrt{2\pi\gamma^2}} e^{\frac{-1}{2\gamma^2}\left(h_a^{(L)}-y_a\right)^2} p\left(h^{(L)}\middle| x\right) \tag{1.21}$$

It can be argued that this should again be equal to unity since the posterior would be normalized w.r.t the last layers preactivations and so

$$\int dh^{(L)} \prod_{a=1}^{N_l} \frac{1}{\sqrt{2\pi\gamma^2}} e^{\frac{-1}{2\gamma^2}\left(h_a^{(L)}-y_a\right)^2} p\left(h^{(L)}\middle| x\right) = p(y|x) \ ^{\dagger} \tag{1.22}$$

Again γ^2 can be interpreted as the system's temperature and so substituting $\gamma^2 = 1/\beta$

$$\int dh^{(L)} \prod_{a=1}^{N_l} \sqrt{\frac{\beta}{2\pi}} e^{\frac{-\beta}{2}\left(h_a^{(L)}-y_a\right)^2} p\left(h^{(L)}\middle| x\right) = p(y|x) \tag{1.23}$$

Now using eq (1.9) and (1.10) and admitting a Gaussian prior over the weight distribution.

$$\int dh^{(L)} \prod_{a=1}^{N_l} \sqrt{\frac{\beta}{2\pi}} e^{\frac{-\beta}{2}\left(h_a^{(L)}-y_a\right)^2} \left(\prod_{l=1}^{L} \int \mathcal{D}W^{(l)} p\left(W^{(l)}\right) \delta\left(W^{(l)} h^{(l-1)} - h^{(l)}\right)\right) = p(y|x) \tag{1.24}$$

Working this out in a similar way as in 1.2 with the uncertainty hypothesis in use, the Gibbs distribution is recovered with the ordinary square loss as part of the assumption and the L_2 regularization term arising, once again as an entropic term from the prior distribution and $p(y|x)$ becomes.

$$p(y|x) = \int \mathcal{D}\mathcal{W} e^{-\beta E} \tag{1.25}$$

Which is again the partition function. Admitting to an uncertainty hypothesis the probability $p(y|x) \equiv p(y^\mu|x^\mu)$ is the partition function for any temperature regime giving rise to the Gibbs distribution. Note that for all of the above a Gaussian prior over the weights like eq.(1.3) is needed in line with how such networks are initialized in practice.

[†]This also follows from the definition, see the first line in page 3 of [11]

Version of January 31, 2025– Created July 12, 2025 - 19:12

1.2.3 Partition function with sources

Apart from characterizing the prior probability distribution, many thermodynamic properties of the system can be deducted by adding sources to the partition function. We parametrize the partition function as follows:

$$Z(J,\beta) = \int \mathcal{D}W \exp\left(-\beta E[\mathcal{W}] + J^{(l)}h^{(l)}\right) \tag{1.26}$$

The average value and variance of the predictor is then given by:

$$\langle (f(x^{\mu})\rangle = \frac{\delta}{\delta J^{(L)}} logZ(J,\beta)\bigg|_{J=0} \tag{1.27}$$

$$\langle \delta f(x^{\mu})^2\rangle = \frac{\delta^2}{\delta\left(J^{(L)}\right)^2} logZ(J,\beta)\bigg|_{J=0} \tag{1.28}$$

where again vector multiplication is implied, $J^{(l)}h^{(l)} = \sum_{l=1}^{L}\sum_{a=1}^{N_l} J_a^{(l)} h_{a;\mu}^{(l)}$. Additionally, we can examine any moment of any layer by applying the functional derivative of that layer on the partition function and also supplement the base energy functional with any arbitrary functional $F[h^{(l)}]$, specifically.

$$\langle h_{\mu}^{(l)}\rangle = \frac{\delta}{\delta J^{(l)}} logZ(J,\beta)\bigg|_{J=0} \tag{1.29}$$

$$\int \mathcal{D}W \exp\left(-\beta E[\mathcal{W}] + F(h^{(l)})\right) = \exp\left[f\left(\frac{\delta}{\delta J^{(l)}}\right)\right] Z(J,\beta)\bigg|_{J=0} \tag{1.30}$$

Such statistical characterizations make way for an elementary description of the system while also give rise to exact correspondences between deep learning and well known physics methods such as the renormalization group [12].

1.3 Linear Neural Networks

Arguably the most tractable problems are the ones where the activation function is linear. Such networks have been shown to have exact priors for their output distributions [11] and effective field

theories have been put forward to characterize the output distribution of ensembles of such networks [13, 14]

1.3.1 Effective theory for Linear neural networks

In order to build an effective theory for neural networks, many assumptions need to be made. In this section it is assumed that the system in question is a linear fully FCN with a quadratic loss function. The premise is that an effective description of the system can be obtained by marginalizing over the networks weights.

1.3.2 Partition function of a single entry Linear Network

For a linear neural network of layer width N_l and depth L the energy functional can be writen as:

$$E[W] = -\frac{1}{2}\sum_{\mu=1}^{P}\sum_{a_L=1}^{N_l}\sum_{a_{L-1}}^{N_{l-1}}\cdots\sum_{a_0=1}^{n_0}\left(W^L_{a_L a_{L-1}}W^{L-1}_{a_{L-1}a_{L-2}}\cdots W^1_{a_1 a_0}x^\mu_{a_0}-y^\mu_{a_L}\right)^2-\sum_{l=1}^{L}\sum_{a^*=1}^{N_l}\sum_{b^*=1}^{N_l}\frac{\left(W^{(l)}_{a^*b^*}\right)^2}{2\beta\sigma_l^2} \tag{1.31}$$

$$\equiv -\frac{1}{2}\left(W^{(L)}W^{(L-1)}\cdots W^{(1)}x^\mu-y^\mu\right)^2-\frac{\left(W^{(l)}\right)^2}{2\beta\sigma_l^2} \tag{1.32}$$

and therefore the partition function will be

$$Z(\beta)=\int\mathcal{D}\mathcal{W}\exp\left(-\frac{\beta}{2}\left(W^{(L)}W^{(L-1)}\cdots W^{(1)}x^\mu-y^\mu\right)^2-\frac{\left(W^{(l)}\right)^2}{2\sigma_l^2}\right) \tag{1.33}$$

where σ_l here is taken to be $\sigma_l=\rho_l/N_l$, with ρ_l a depth-independent variance so that in the infinite width limit this reduces to the ordinary square loss. Integration can be iteratively done by starting with the readout weights and moving backwards until the input. The first step is to linearize the base cost function, for that the Hubbart Strantanovich transformation is performed on every element of the readout layer preactivations $h^L_{c;\mu}$ with the dual valiable q^μ_c such that:

$$\prod_{a=1}^{N_l} \exp\left\{-\frac{\beta}{2}\left(h_a^{(L)} - y_a\right)^2\right\} = \sqrt{\frac{1}{(2\pi\beta)^{N_l}}} \int_{-\infty}^{\infty} \mathcal{D}q \exp\left[\sum_{a=1}^{N_l} -\frac{q_a^2}{2\beta} - iq_a h_a\right] \tag{1.34}$$

the partition function takes the form

$$Z(\beta) = \int_{-\infty}^{+\infty} \mathcal{D}\mathcal{W} \int_{-\infty}^{+\infty} \prod_{c=1}^{N_l} \frac{dq_c^\mu}{\sqrt{2\pi\beta}} e^{-iq_c^\mu(h_c^L - y_c^\mu) - \sum_{l^\star=1}^{L} \frac{1}{2\sigma_{l^\star}^2}(W_{ab}^{l^\star})^2} \tag{1.35}$$

$$= \int_{-\infty}^{+\infty} \prod_{c=1}^{N_l} \frac{dq_c^\mu}{\sqrt{2\pi\beta}} \int_{-\infty}^{+\infty} \mathcal{D}\mathcal{W} e^{iq_c^\mu y_c^\mu} e^{-iq_c^\mu h_c^L - \frac{(q_c^\mu)^2}{2\beta} - \sum_{l^\star=1}^{L} \frac{1}{2\sigma_{l^\star}^2}(W_{ab}^{l^\star})^2} \tag{1.36}$$

$$= \int_{-\infty}^{+\infty} \prod_{c=1}^{N_l} \frac{dq_c^\mu}{\sqrt{2\pi\beta}} e^{iq_c^\mu y_c^\mu} Z(x, q; \beta) \tag{1.37}$$

where $Z(x, q; \beta)$ is the label (y_c^μ) Fourier transformed partition function given by

$$Z(x, q; \beta) = e^{-\frac{|q_2^\mu|^2}{2\beta}} \int_{-\infty}^{+\infty} \mathcal{D}\mathcal{W} e^{-iq_c^\mu h_c^L - \sum_{l^\star=1} \frac{1}{2\sigma_{l^\star}^2}(W_{ab}^{l^\star})^2} \tag{1.38}$$

The single hidden layer network

The integral (1.38) can be solved by direct integration. Expanding, completing the square and integrating out the weights (A.1.2) yields:

$$Z(x, q; \beta) = \frac{\mathcal{N}}{det\left(\sigma_1^2\sigma_2^2 q_b^\mu x_b^\mu q_c^\nu x_c^\nu + \delta_{ca}\right)^{N_1/2}} \tag{1.39}$$

where $\mathcal{N} = e^{-\frac{(q_c^\mu)^2}{\beta}} \left(2\pi\sigma_2^2\right)^{\frac{N_1 N_2}{2}}$ a normalization factor. If this normalization factor is omitted, then equation (1.39) resembles the characteristic function for a two layer network of [11] only now supplemented to include multiple examples. Performing the inverse Hankel transform one gets back the partition function in the form of the integral

$$Z(\beta) = \int_{-\infty}^{+\infty} \frac{dq_c^\mu}{\sqrt{2\pi\beta}} e^{iq_{c;\mu} y_{c;\mu}} \frac{\mathcal{N}}{det\left(\sigma_1^2\sigma_2^2 q_b^\mu x_b^\mu q_c^\nu x_c^\nu + \delta_{ca}\right)^{N_1/2}} \tag{1.40}$$

This way the final partition function has no reference to the specific configuration of weights in the network and is only a function of the dataset $\mathcal{D}$, its cardinality P and network parameters such as the temperature

$$Z = Z\Big(x^\mu, y^\mu \in \mathcal{D}, \beta; L, \{n\}_l\Big) \tag{1.41}$$

The simplicity of linear activation functions is what allows the partition function to be expressed in closed form. It is however far from what is done in practice where linear layers play a very minor role in state of the art architectures while the majority of the layers of the model employ some sort of *non-linear* activation function.

Chapter 2

Nonlinear corrections to $Z(\beta)$

In this chapter the non-linearity in the activation function is introduced using sources in sec. 2.1 and the linear partition function is corrected in first order in the activation perturbation. The 2-layer network is considered, first for two layers in 2.2 with an easily extensible method to L layer in section 2.3. In section 2.4 this method is extended to multiple examples for the two layer network.

2.1 Introducing sources

Under certain criteria one can approximate any well behaved activation function by its Taylor series

$$\sigma(x) = \sum_{n=0}^{\infty} \frac{d^n \sigma(x)}{dx^n}\bigg|_{x=0} \frac{x^n}{n!} \tag{2.1}$$

The readout layer $h^{(1)}_{a,\mu}$ will now be considered non-linear with an activation function

$$\sigma(x) = x + gx^3 \tag{2.2}$$

which can be a first order approximation of a more practical activation function (e.g ReLU). We have

$$h^{(2)}_{b;\mu;g} = \ W_{ab}\sigma(h^{(1)}_{a;\mu}) = W_{ab}\left(h^{(1)}_{a;\mu} + g(h^{(1)}_{a;\mu})^3\right) \tag{2.3}$$

One immediately sees that the perturbed readout preactivation is $h^{(2)}_{b;\mu;g} = \left[1 + g\left(h^{(1)}_{a;\mu}\right)^2\right] h^{(2)}_{b;\mu;g=0}$ and so the perturbed energy will be

$$E_g = \left[\sum_{b=1}^{N_l}\left(1 + g|h^{(1)}_{a;\mu}|^2\right) h^{(2)}_{b;\mu} - y^{\mu}_b\right]^2 \tag{2.4}$$

and upon expanding and completing the squares this becomes

$$E_g = E_L + g|h^{(1)}_{a;\mu}|^2\left(2|h^{(2)}_{a;\mu}|^2 - 2h^{(2)}_{a;\mu}y_{a;\mu} + g|h^{(1)}_{a;\mu}|^2|h^{(2)}_{a;\mu}|^2\right) \tag{2.5}$$

and since we are doing a perturbative analysis, for small g terms containing g^2 can be ignored

$$E_g = E_L + 2g|h^{(1)}_{a;\mu}|^2\left(\left|h^{(2)}_{a;\mu}\right|^2 - h^{(2)}_{a;\mu}y_{a;\mu}\right) \tag{2.6}$$

Therefore the new partition function of interest will be

$$Z_g = \int \mathcal{D}\mathcal{W} \exp\left(-\beta E_L - 2\beta g|h^{(1)}_{a;\mu}|^2\left(|h^{(2)}_{a;\mu}|^2 - h^{(2)}_{a;\mu}y_{a;\mu}\right)\right) \tag{2.7}$$

One can generate such a function easily by introducing functional sources, specifically:

$$Z(J) = \int \mathcal{D}\mathcal{W} \exp\left(-\beta E_L + J^{(1)}_{a;\mu}h^{(1)}_{a;\mu} + J^{(2)}_{a;\mu}h^{(2)}_{a;\mu}\right) \tag{2.8}$$

And thus we can get to (2.7) from (2.8) as

$$\lim_{J_1,J_2\to 0} \exp\left[2g\left(\frac{\delta}{\delta J^{(1)}_{a;\mu}}\right)^2\left(\left(\frac{\delta}{\delta J^{(2)}_{b;\nu}}\right)^2 - \frac{\delta}{\delta J^{(2)}_{c;\lambda}}y^{\lambda}_c\right)\right] Z(J) = Z_g \tag{2.9}$$

Equation (2.7) makes it possible to encode the effect on the non-linearity of a two layer network just by acting with the functional derivatives of the sources on the linear network partition function.

2.2 2-Layer partition function

One approach for characterizing the partition function of the system would be utilizing the linear network solution to find the saddle point of the source integral and expanding around it. Effectively

expanding around the linear network solution. Here the $\mathcal{GP}$ limit corrections are taken care of by the fact that the linear solution already implements a finite width description. In the following the notation is simplified since the expressions become increasingly complex. The absence of a μ index means that one is effectively working only with one dataset vector index. The notation is further suppressed by the following norm conventions. For any two vectors u, v:

$$u^2 \equiv |u|^2 = \sum_a u_a u_a$$
$$u \equiv |u| = \sum_a u_a$$
$$uv \equiv u \cdot v = \sum_a u_a v_a$$

The absence of a vector index means that the layer index can also be moved to a subscript*.

Approximating the source integral

Now computing (2.8) first we perform the H.S transformation

$$Z(J) = \int \mathcal{D}\mathcal{W} \frac{\mathcal{D}q}{(2\pi\beta)^{N_2/2}} \exp\left(-\frac{q^2}{2\beta} - iq(W_2W_1x) - iqy - \frac{W_2^2}{2\sigma_2^2} - \frac{W_1^2}{2\sigma_1^2} + J_1W_1x + J_2W_2W_1x\right) \tag{2.10}$$

$$= \int \frac{\mathcal{D}q}{(2\pi\beta)^{N_2/2}} \exp\left(\frac{q^2}{2\beta} - iqy\right) \int \mathcal{D}\mathcal{W} \exp\left(-iq(W_2W_1x) - \frac{W_2^2}{2\sigma_2^2} - \frac{W_1^2}{2\sigma_1^2} + J_1W_1x + J_2W_2W_1x\right) \tag{2.11}$$

$$= \int \frac{\mathcal{D}q}{(2\pi\beta)^{N_2/2}} \exp\left(\frac{q^2}{2\beta} - iqy\right) \times I(q, J) \tag{2.12}$$

Now we integrate out $\mathcal{W}$ on $I(q, J)$. we will do this layer by layer starting from the readout layer

*All in all the notation transformations for this chapter for a vector u can be summarized by the mapping $u^{(l)}_{a;\mu} v^{(l)}_{a;\mu} \mapsto u_l v_l$

Version of January 31, 2025– Created July 12, 2025 - 19:12

(W_2) and then moving to the first layer.

$$I(q,J) = \int \mathcal{D}\mathcal{W} \exp\left(-iq(W_2W_1x) - \frac{W_2^2}{2\sigma_2^2} - \frac{W_1^2}{2\sigma_1^2} + J_1W_1x + J_2W_2W_1x\right) \tag{2.13}$$

$$= \int \mathcal{D}\mathcal{W} e^{-S}, \text{ where } S = -iq(W_2W_1x) - \frac{W_2^2}{2\sigma_2^2} - \frac{W_1^2}{2\sigma_1^2} + J_1W_1x + J_2W_2W_1x \tag{2.14}$$

Solving this integral is just a matter of completing the square, this can be done iterativelly starting from the readout weights and integrating back until the input weights.

Integrating out W_2

We have to complete two squares, first for W_2 (see A.1.3)

$$S = -\frac{1}{2\sigma_2^2}\left[W_2 + \left(\sigma_2^2(iq - J_2)W_1x\right)\right]^2 + \left(\frac{\sigma_2(iq-J_2)}{2}W_1x\right)^2 - \frac{W_1^2}{2\sigma_1^2} + J_1W_1x \tag{2.15}$$

The first term is now a completed square and will contribute a multiplicative term of $(2\pi\sigma_2^2)^{N_2/2}$ to the partition function.

Integrating out W_1

Now lets examine the second term

$$\left(\frac{\sigma_2(iq-J_2)}{2}W_1x\right)^2 - \frac{W_1^2}{2\sigma_1^2} + J_1W_1x = -\frac{1}{2\sigma_1^2\,(1+k)}\left[W_1 + \frac{\sigma_1^2J_1x}{1+k}\right]^2 + \frac{(1+k)J_1x}{2} \tag{2.16}$$

Putting everything back together

$$S = -\frac{1}{2\sigma_2^2}\left[W_2 + \left(\sigma_2^2(iq - J_2)W_1x\right)\right]^2 - \frac{1}{2\sigma_1^2\,(1+k)}\left[W_1 + \frac{\sigma_1^2J_1x}{1+k}\right]^2 + \frac{(1+k)J_1x}{2} \tag{2.17}$$

Version of January 31, 2025– Created July 12, 2025 - 19:12

The integral now reduces to

$$I(q,J) = \int \mathcal{D}W_2 e^{-\frac{1}{2\sigma_2^2}\left[W_2+\left(\sigma_2^2(iq-J_2)W_1x\right)\right]^2} \int \mathcal{D}W_1 e^{-\frac{1}{2\sigma_1^2(1+k)}\left[W_1+\frac{\sigma_1^2 J_1 x}{1+k}\right]^2+\frac{(1+k)J_1x}{2}} \tag{2.18}$$

$$= \frac{1}{(2\pi\sigma_2)^{N_2/2}} \frac{1}{(2\pi\sigma_1^2(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)^{N_1/2}} \exp\left(\frac{(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)J_1x}{2}\right) \tag{2.19}$$

$$= \frac{1}{(2\pi\sigma_2)^{N_2/2}} \frac{1}{(2\pi\sigma_1)^{N_1/2}} \frac{1}{(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)^{N_1/2}} \exp\left((1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)\frac{J_1x}{2}\right) \tag{2.20}$$

Relation to the partition function

One can quickly verify that in the $J_1, J_2 \to 0$ limit $I(q,J)$ converges to the linear, no-source partition function. For completeness's sake, here is the full partition function

$$Z(J) = (2\pi\beta)^{-\frac{N_2}{2}}(2\pi\sigma_2)^{-\frac{N_2}{2}}(2\pi\sigma_1)^{-\frac{N_1}{2}} \times \tag{2.21}$$

$$\int \mathcal{D}q \exp\left(-\frac{q^2}{2\beta} - iqy\right) \frac{\exp\left[\frac{1}{2}(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)J_1x\right]}{(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)^{N_1/2}} \tag{2.22}$$

Now the problem is to solve this integral if we are to revert back to the probability density. But if we restrict ourselves to the characteristic function, then

$$\phi(q,J|x) = N\exp\left(-\frac{q^2}{2\beta}\right) \frac{1}{(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)^{N_1/2}} \exp\left(\frac{1}{2}(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)J_1x\right) \tag{2.23}$$

where $N = (2\pi\beta)^{-\frac{N_2}{2}}(2\pi\sigma_2)^{-\frac{N_2}{2}}(2\pi\sigma_1)^{-\frac{N_1}{2}}$ is just a constant prefactor. Now $Z(J)$ plays the role of the normalization $p(y|x;\beta,J)$ encoding the probability of y given x for out temperature and perturbation regime. Since all dynamical variables are integrated out, the network becomes a blackbox that only produces outputs from inputs.

2.2.1 Saddle-point approximation of the the J=0 integral

The final $\mathcal{D}q$ integral can be solved by using the *saddle-point approximation*. This method is preferred for its very strong approximation power at a relatively low computational cost [15]. We start from

integral (2.22)

$$Z_J = N \int \mathcal{D}q \exp\left(-\frac{q^2}{2\beta} - iqy\right) \frac{\exp\left[\frac{1}{2}(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)J_1x\right]}{(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)^{N_1/2}} \tag{2.24}$$

we can write this as a single exponential

$$Z_J = N \int \mathcal{D}q \exp\left(-\frac{q^2}{2\beta} - iqy + \frac{1}{2}(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)J_1x - \frac{N_1}{2}ln(1+\sigma_1^2\sigma_2^2(q+iJ_2)^2x^2)\right) \tag{2.25}$$

and take the $J=0$ limit:

$$Z_{J=0} = N \int \mathcal{D}q \exp\left[-\frac{q^2}{2\beta} - iqy - \frac{N_1}{2}ln(1+\sigma_1^2\sigma_2^2q^2x^2)\right] \tag{2.26}$$

Now this integral is of the form $\int dq e^{-f_J(q)}$ and can be approximated by its saddlepoint solution , so:

$$\frac{d}{dq}f_{J=0}(q) = 0 \Rightarrow q^3 - iy\beta q^2 - \frac{(1+N_1\beta\kappa^2)}{\kappa^2}q - \frac{iy\beta}{\kappa^2} = 0 \text{ where, } \kappa^2 = \sigma_1^2\sigma_2^2x^2 \tag{2.27}$$

This is a 3-rd order polynomial of the form

$$q^3 + Pq^2 + Qq + R = 0 \tag{2.28}$$

$$\text{where, } P = -iy\beta,\ Q = \frac{1+N_1\kappa^2\beta}{\kappa^2},\ R = \frac{-iy\beta}{\kappa^2} \tag{2.29}$$

The 3 stationary points and their associated helper variables will be

$$q_{1,2,3} = -\frac{1}{3}(P + \xi^k C + \frac{\Delta_0}{\xi^k C}),\ \text{with } k \in \{0,1,2\} \tag{2.30}$$

$$\Delta_0 = P^2 - 3Q = -\beta^2y^2 - \frac{3(1+N_1\kappa^2\beta)}{\kappa^2} \tag{2.31}$$

$$\Delta_1 = 2P^3 - 9PQ + 27R = 2iy^3\beta^3 + 9iy\beta\frac{1+N_1\kappa^2\beta}{\kappa^2} - \frac{27iy\beta}{\kappa^2} \tag{2.32}$$

$$C = \left(\frac{\Delta_1 \pm \sqrt{\Delta_1^2 - 4\Delta_0^3}}{2}\right)^{1/3} \tag{2.33}$$

$$\xi = \frac{-1+\sqrt{-3}}{2} \tag{2.34}$$

One should expect 3 solutions, one real and two complex that are complex conjugates of one another. The explicit forms of these roots is given in (A.1.4). For computational simplicity, q_0 is assumed to be the value of the stationary point, the integral can now be approximated by:

$$Z_{J=0} \approx \exp\left[-f_{J=0}(q_0)\right] \int \mathcal{D}q \exp\left(\frac{1}{2}(q-q_0)^2 f''(q_0)\right) \tag{2.35}$$

The second derivative $f''(q_0) = \left.\frac{d^2 f_{J=0}(q)}{dq^2}\right|_{q=q_0}$ is again evaluated in the absence of sources. If one chooses to consider only one of the stationary points (which should be the one that yields the minimum value of the exponential) the solution is simply

$$Z_{J=0} \approx \exp\left[-f_{J=0}(q_{0;a})\right] \left(\frac{\pi}{f''(q_0^a)}\right)^{\frac{N_2}{2}} \tag{2.36}$$

The second derivative evaluates as:

$$\left.\frac{d^2 f_{J=0}(q)}{dq}\right|_{q=q_0} = \frac{-1}{\beta} - \frac{N_1\kappa^2}{1+\kappa^2 q_0^2}\left(1 - \frac{2q_0^2\kappa^2}{1+\kappa^2 q_0^2}\right) \tag{2.37}$$

and so eq. (2.36) finally becomes

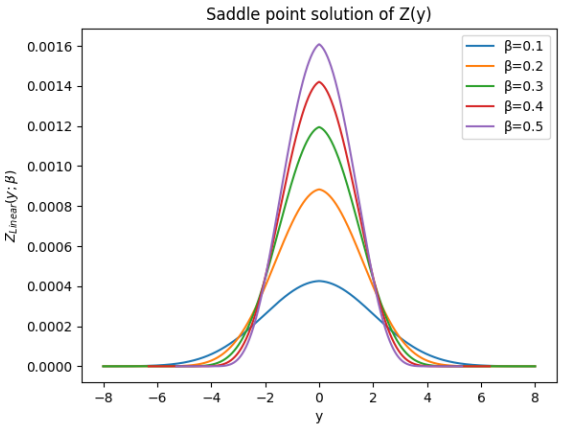
$$Z_{J=0} = \exp\left(-\frac{q_0^2}{2\beta} - iq_0 y - \frac{N_1}{2} ln(1+\kappa^2 q_0^2)\right) \frac{1}{\left(\frac{-1}{\pi\beta} - \frac{N_1\kappa^2}{\pi(1+\kappa^2 q_0^2)}\left(1 - \frac{2q_0^2\kappa^2}{1+\kappa^2 q_0^2}\right)\right)^{\frac{N_2}{2}}} \tag{2.38}$$

This is the saddle point approximation for the partition function without sources taking into account one stationary point q_0. This approximation will be the cornerstone of the non-linear activation function expansion. The β, y dependence of this function is shown in figure 2.1

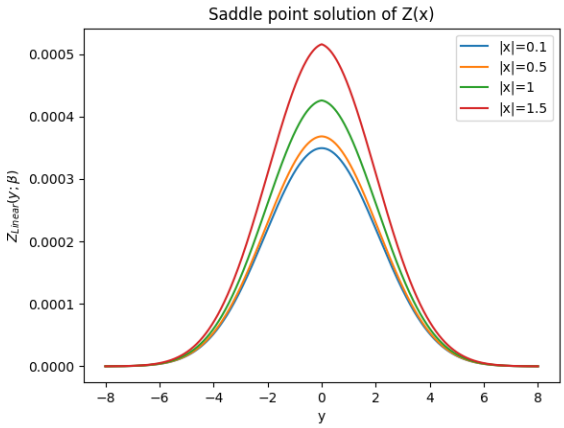
2.2.2 Expanding around the stationary point

Going back to (2.22) we perturb around the stationary point by writing

$$q = q_0 + \delta q \tag{2.39}$$



(a) *The y-dependence of the partition function encoding the un-normalized probability distribution $p(y)$. For different values of the inverse temperature β*



(b) *The x-dependence of the partition function for a fixed temperature encoding the un-normalized distribution $p(y|x)$*

Figure 2.1: *The network output in the shallow regime ($N_1 = N_2 = 10$). The (un-normalized) distribution of Z is centered around zero and as the temperature is decreased the distribution tends to a delta function. The effect of the norm of x seems to not play a crucial role here, only controlling the overall normalization of the distribution. This last part is in line with the exact solution of eq.(1.40) where the input norm only appears as a multiplicative factor and as a scaling constant on the networks' neuron activation values.*

where q_0 is one of the stationary points, and so

$$Z_J = N \int \mathcal{D}[\delta q] \exp\left(-\frac{q^2}{2\beta} - iqy\right) \frac{\exp\left[\frac{1}{2}(1+\kappa^2(q+iJ_2)^2)J_1 x\right]}{(1+\kappa^2(q+iJ_2)^2)^{N_1/2}} \tag{2.40}$$

$$= N \int \mathcal{D}[\delta q] \exp\left(-\frac{(q_0+\delta q)^2}{2\beta} - i(q_0+\delta q)y\right) \frac{\exp\left[\frac{1}{2}(1+\kappa^2((q_0+\delta q)+iJ_2)^2)J_1 x\right]}{(1+\kappa^2((q_0+\delta q)+iJ_2)^2)^{N_1/2}} \tag{2.41}$$

$$= N e^{-\frac{q_0^2}{2\beta} - iq_0 y + \frac{J_1 x}{2}} \int \mathcal{D}[\delta q] e^{-\frac{\delta q^2}{2\beta} - (\frac{q_0}{\beta} + iy)\delta q + \kappa^2 J_1 x (q_0+\delta q+iJ_2)^2 - \frac{N_1}{2} ln[1+\kappa^2(q_0+\delta q+iJ_2)^2]} \tag{2.42}$$

$$= N' \int \mathcal{D}[\delta q] e^{f(\delta q)} \tag{2.43}$$

where $N' = [(2\pi\beta)^{-\frac{N_2}{2}}(2\pi\sigma_2)^{-\frac{N_2}{2}}(2\pi\sigma_1)^{-\frac{N_1}{2}}]e^{-\frac{q_0^2}{2\beta} - iq_0 y + \frac{J_1 x}{2}}$ and

$$f(\delta q) = -\frac{\delta q^2}{2\beta} - (\frac{q_0}{\beta} + iy)\delta q + \kappa^2 J_1 x (q_0+\delta q+iJ_2)^2 - \frac{N_1}{2} ln[1+\kappa^2(q_0+\delta q+iJ_2)^2] \tag{2.44}$$

We aim to write f as: $f(\delta q) = A\delta q^2 + B\delta q + C$, so we can solve it as an ordinary Gaussian integral. One immediately sees the problem that lies with the logarithmic term. The Mercator series only converges for $-1 \leq x \leq 1$ so we can only write

$$-\frac{N_1}{2} ln[1 + \kappa^2(q_0 + \delta q + iJ_2)^2] = \frac{N_1}{2} \sum_{n=1}^{\infty} \frac{(-1)^n}{n} \kappa^{2n} (q_0 + \delta q + iJ_2)^{2n} \tag{2.45}$$

we use the following approximation for the logarithm

$$ln[1 + \kappa^2(q_0 + iJ_2 + \delta q)^2] \approx ln[1 + \kappa^2(q_0 + iJ_2)^2)] \tag{2.46}$$

$$+ \delta q \frac{d}{d(\delta q)} ln[1 + \kappa^2(q_0 + iJ_2 + \delta q)^2]\Big|_{\delta q=0} \tag{2.47}$$

$$+ \frac{\delta q^2}{2} \frac{d^2}{d(\delta q)^2} ln[1 + \kappa^2(q_0 + iJ_2 + \delta q)^2]\Big|_{\delta q=0} \tag{2.48}$$

and so, writing $Q_J = q_0 + iJ_2$ to try to ease the notation further:

$$-\frac{N_1}{2} ln[1 + \kappa^2(Q_J + \delta q)^2] \approx -\frac{N_1}{2} \left(ln[1 + \kappa^2 Q_J^2] + \delta q \frac{2\kappa^2 Q_J}{1 + \kappa^2 Q_J^2} \right) \tag{2.49}$$

$$+ \frac{\delta q^2}{2} \left(\frac{2\kappa^2}{1 + \kappa^2 Q_J^2} - \frac{4\kappa^2 Q_J}{(1 + \kappa^2 Q_J^2)^2} \right) \tag{2.50}$$

$$= -\frac{N_1}{2} \left(ln[1 + \kappa^2 Q_J^2] + \delta q \frac{2\kappa^2 Q_J}{1 + \kappa^2 Q_J^2} + \delta q^2 \frac{\kappa^2 \left(1 + \kappa^2 Q_J^2\right) - 2\kappa^2 Q_J^2}{(1 + \kappa^2 Q_J^2)^2} \right) \tag{2.51}$$

And now eq. (2.44) can be written as a second degree polynomial in δq as:

$$f(\delta q) = \delta q^2 \left(\frac{N_1}{2} \frac{\kappa^2 \left(1 + \kappa^2 Q_J^2\right) - 2\kappa^2 Q_J^2}{(1 + \kappa^2 Q_J^2)^2} + \kappa^2 J_1 x - \frac{1}{2\beta} \right) \tag{2.52}$$

$$+ \delta q \left(\frac{N_1}{2} \frac{2\kappa^2 Q_J}{1 + \kappa^2 Q_J^2} - \frac{q_0}{\beta} - iy + 2i\kappa^2 J_2 J_1 x + 2q_0 \kappa^2 J_1 x \right) \tag{2.53}$$

$$+ \kappa^2 J_1 x Q_J^2 - \frac{N_1}{2} ln[1 + \kappa^2 Q_J^2] \tag{2.54}$$

This is of the form $A\delta q^2 + B\delta q + \Gamma$ which can be written as $A\left(\delta q + \frac{B}{2A}\right)^2 + \Gamma - \frac{B^2}{2A}$ And now we can complete the square and finally calculate the integral (2.40):

$$Z(J) = \exp\left(\kappa^2 J_1 x Q_J^2 - \frac{N_1}{2} ln[1+\kappa^2 Q_J^2] - \frac{\left(-\frac{N_1\kappa^2 Q_J}{1+\kappa^2 Q_J^2} - \frac{q_0}{\beta} - iy + 2i\kappa^2 J_2 J_1 x + 2q_0\kappa^2 J_1 x\right)^2}{2\left(\frac{\kappa^2\left(1+\kappa^2 Q_J^2\right)-2\kappa^2 Q_J^2}{\left(1+\kappa^2 Q_J^2\right)^2} + \kappa^2 J_1 x - \frac{1}{2\beta}\right)}\right) \times \frac{1}{\left(4\pi^2\beta\sigma_2\right)^{\frac{N_2}{2}}\left(2\pi\sigma_2\right)^{\frac{N_1}{2}}} \frac{\pi^{\frac{N_2}{2}}}{\left(\frac{\kappa^2\left(1+\kappa^2 Q_J^2\right)-2\kappa^2 Q_J^2}{\left(1+\kappa^2 Q_J^2\right)^2} + \kappa^2 J_1 x - \frac{1}{2\beta}\right)^{\frac{N_2}{2}}} \tag{2.55}$$

We can simplify the expression further by defining

$$c = \frac{\pi^{\frac{N_2}{2}}}{\left(4\pi^2\beta\sigma_2\right)^{\frac{N_2}{2}}\left(2\pi\sigma_2\right)^{\frac{N_1}{2}}} \tag{2.56}$$

$$D(J) = \left(\frac{\kappa^2\left(1+\kappa^2 Q_J^2\right) - 2\kappa^2 Q_J^2}{\left(1+\kappa^2 Q_J^2\right)^2} + \kappa^2 J_1 x - \frac{1}{2\beta}\right) \tag{2.57}$$

$$S(J) = \left(-\frac{N_1\kappa^2 Q_J}{1+\kappa^2 Q_J^2} - \frac{q_0}{\beta} - iy + 2i\kappa^2 J_2 J_1 x + 2q_0\kappa^2 J_1 x\right) \tag{2.58}$$

Then the partition function becomes:

$$Z(\beta; J) = \frac{c}{D(J)^{\frac{N_2}{2}}} \exp\left(\kappa^2 J_1 x Q_J^2 - \frac{N_1}{2} ln[1+\kappa^2 Q_J^2] - \frac{S(J)^2}{2D(J)}\right) \tag{2.59}$$

This is the partition function for a two layer network for an arbitrary choice of activation function, controlled by the potential action of $\delta/\delta J_{1,2}$ derivatives. Notice that in the $J \to 0$ limit equation (2.59) reduces to eq.(2.40) as $S(J)$ becomes the stationary point equation. This is simply quoted here as a generalization of eq. (2.36). One can get the desired non-linear partition function, with lower computational cost, by employing a procedure we name *Norm Learning*.

Version of January 31, 2025– Created July 12, 2025 - 19:12

2.3 Norm learning

A limitation of the above analysis is that one restricts themselves to a 2-layer network by construction, at the same time numerical constants appear in highly non-linear ways obscuring the interpretability of the model. An iterative procedure of Fourier transforming each layers activations without integrating them out solves both of those problems. The calculations are eased further by again decoupling the "frequencies" of the layers from the input, effectively treating the input as a single neuron. One can think of this as a phantom 0-th layer that transforms the N_0 dimensional input x_{N_0} to a single dimensional input $|x|^2$. The general form of the partition function for a linear network after HST on the readout layer is:

$$Z(\beta) = \prod_{\mu=1}^{P} (2\pi\beta)^{\frac{-N_l}{2}} \int \mathcal{D}\mathcal{W}\mathcal{D}q^{(L)} \exp\left(-\frac{{q^{(L)}}^2}{2\beta} - iq^{(L)}y - iq^{(L)}h^{(L)} - \sum_l \frac{{W^{(L)}}^2}{2P\sigma_l^2} + \sum_l J_a^{(l)} h_a^{(l)}\right) \tag{2.60}$$

After integrating out the weights of the last two layers the expression reads:

$$\begin{aligned} Z(\beta) = \prod_{\mu=1}^{P} N \int \mathcal{D}q^{(L)}\mathcal{D}q^{(L-1)}\mathcal{D}\mathcal{W}^{(l<L-1)} e^{-\frac{{q^{(L)}}^2}{2\beta} - \frac{{q^{(L-1)}}^2}{2P\sigma_L^2} - iq^{(L)}y - \sum_{l<L-1} \frac{{W^{(L-1)}}^2}{2P\sigma_l^2} + J_a^{(l)}h_a^{(l)}} \\ \times \ (2\pi P\sigma_{L-1}^2)^{\frac{N_1^2}{2}} \exp\left(P\sigma_{L-1}^2 \left[-iq^{(L-1)}\left(q^{(L)} - iJ^{(L)}\right) + J^{(L-1)}\right]^2 {h^{(L-2)}}^2\right) \end{aligned} \tag{2.61}$$

The intermediate steps of the calculation are done in the Appendix (A.1.5). Everytime a dual variable $q^{(l)}$ is introduced a prefactor of $(2pP\sigma_{l+1}^2)^{N_l/2}$ appears. Next, every time a Gaussian integration on $\mathcal{W}^{(l)}$ is performed a prefactor of $(2\pi P\sigma_l^2)^{N_l^2/2}$ appears. In total this prefactor is

$$\mathcal{N} = \prod_{l=1}^{L} \frac{(2\pi P\sigma_l^2)^{\frac{N_l^2}{2}}}{(2\pi P\sigma_{l+1}^2)^{\frac{N_l}{2}}} \tag{2.62}$$

Where a "convention" of taking $\sigma_{L+1} = \beta/P$ is used. Now as for the rest of the expression this appears to have a nested structure. Breaking it down, on readout a term appears as a result of integration of the readout layer weights

$$\exp\left(P\sigma_L^2 \left[\left(J^{(L)} - iq^{(L)}\right) h^{(L-1)}\right]^2\right)$$

moving backwards a layer to the pre-readout layer, this term becomes

$$\exp\left(P\sigma_{L-1}^2\left[-iq^{(L-1)}\left(q^{(L)}-iJ^{(L)}\right)+J^{(L-1)}\right]^2 {h^{(L-2)}}^2\right)$$

If one treats these canonically conjugate variables as orthogonal to each other $\left(\frac{dq^{(k)}}{dq^{(j)}}=\delta_{jk}\right)$ and performs the integrations while defining the following helper quantities

$$\begin{aligned}
Q^{(L)} &= q^{(L)} - iJ^{(L)} \\
Q^{(L-1)} &= \left(-iq^{(L-1)}\left(q^{(L)}-iJ^{(L)}\right)+J^{(L-1)}\right)^2 = \left(-iq^{(L-1)}Q^{(L)}+J^{(L-1)}\right)^2 \\
&\dots \\
Q^{(1)} &= \left(-iq^{(1)}Q^{(2)}+J^{(1)}\right)^2
\end{aligned}$$

Then the partition function with the weights fully integrated out and all indices, products and sums printed out explicitly will be:

$$Z=\prod_{\mu=1}^{P}\frac{1}{(2\pi\beta)^{\frac{N_l}{2}}}\left(\prod_{l=1}^{L-1}\frac{(2\pi P\sigma_l^2)^{\frac{N_l^2}{2}}}{(2\pi P\sigma_{l+1}^2)^{\frac{N_l}{2}}}\right)\int\left(\prod_{l=1}^{L}\prod_{a=1}^{N_l}dq_a^{(l)}\right)e^{\sum_{l=1}^{L}\sum_{a=1}^{N_l}\left(\frac{{q_{a;\mu}^{(l)}}^2}{2\sigma_{(l+1)}^2}-iq_{a;\mu}^{(L)}y^\mu\right)+P\sigma_1^2Q_\mu^{(1)}|x^\mu|^2} \tag{2.63}$$

the μ product here applies to the whole expression, Q_1 is a scalar which results from the integration of the total weights of the system.

$$Q_\mu^{(1)}=\left(\sum_{a=1}^{N_1}-iq_\mu^{(1)}Q_\mu^{(2)}+J_a^{(1)}\right)^2 \tag{2.64}$$

Its upper index (1 in this case) can be interpreted as denoting the total amount of effective layers left after the integration. If one were to stop at say $L-5$ integration then the information about the first 5 layers of the networks' weights would still be explicitly present in the partition function. This is where the interactions between q's are generated and will be the first stepping stone for

Version of January 31, 2025– Created July 12, 2025 - 19:12

the perturbative expansion. One can see the nested structure, for an $L = 5$ network (swapping momentarily to lower layer-indices for notational simplicity):

$$Q_1 = \left(J_1 - iq_1\left(J_2 - iq_2\left(J_3 - iq_3\left(J_4 - iq_4\left(J_5 - iq_5\right)\right)^2\right)^2\right)^2\right)^2 \xrightarrow{\text{J=0}} Q_1\Big|_{J=0} = q_1^2 q_2^4 q_3^6 q_4^8 q_5^{10}$$

And in general for the linear case where the sources are all set to zero

$$Q^{(1)} = \prod_{l=1}^{L} {q^{(l)}}^{2l} \tag{2.65}$$

notice that the first layer dual variable is always a square and that the readout layer dual scales with L. This is a testiment to the Gaussianity of the first layer, the further accumilation of non-Gaussianities here is expressed with higher and higher orders of the dual variables for each layer.

The dual representation

In this last section, the formalism of the system has changed in that the neural network is now completely described by the canonical conjugate of the neuron activations. Using this dual representation gives way to a very straightforward way of integrating out the weights. The advantage of this is twofold in that: a) any type of fully connected architecture can be described this way, as N_l, L can take any value, b) a perturbative analysis of non-linear activations can easily be performed and c) this representation gives rise to equations that bare a lot of resemblance with scalar field theory.

2.3.1 Single hidden layer perturbation

While in principle one can examine any number of layers, what is more relevant for the scope of this thesis is the single hidden layer case. For $L = 2$ we can stop at equation (2.61) and consider only $L = 2$. This is equivalent to treating both $\int \mathcal{D}W^{(l<L-1)}\ e^{-\sum_{l<L}\frac{{W^{(L-1)}}^2}{2P\sigma_l^2} + J_a^{(l)}h_a^{(l)}}$ and $h^{(l<L-1)}$ as the identity operator and letting $J^{(l<L-1)} = 0$. Then the partition function for the two layer network

Version of January 31, 2025– Created July 12, 2025 - 19:12

reads as $Z = \prod_{\mu=1}^{P} Z_\mu$ where:

$$Z_\mu = N \int \mathcal{D}q^{(2)}\mathcal{D}q^{(1)} \exp\left(-\frac{{q^{(2)}}^2}{2\beta} - \frac{{q^{(1)}}^2}{2P\sigma_l^2} - iq^{(2)}y + \left(P\sigma_1^2\left[J^{(1)} - iq^{(1)}\left(q^{(L)} - iJ^{(L)}\right)\right]^2 x^2\right)\right)$$
$$= N \int \mathcal{D}q^{(2)}\mathcal{D}q^{(1)} \exp\left(S[q^{(1)}, q^{(2)}]\right) \tag{2.66}$$

where $N = \frac{(2\pi P\sigma_L^2)^{\frac{N_2^2}{2}}(2\pi P\sigma_1^2)^{\frac{N_1^2}{2}}}{(2\pi\beta)^{\frac{N_2}{2}}(2\pi P\sigma_2^2)^{\frac{N_1}{2}}}$ is a numerical factor arising from the two integrations and two H.S transformations and S is the two field action:

$$S[q^{(1)}, q^{(2)}] = -\frac{{q^{(2)}}^2}{2\beta} - \frac{{q^{(1)}}^2}{2P\sigma_2^2} - iq^{(2)}y + \left(P\sigma_1^2\left[J^{(1)} - iq^{(1)}\left(q^{(2)} - iJ^{(2)}\right)\right]^2 x^2\right) \tag{2.67}$$

The first two terms correspond to mass terms of the free dual fields of the neurons activations while all the rest are interaction terms. The most trivial of which is the interaction of the last layer with the output labels. The last term, in the big parentheses is the one of interest. One can see that even in the absence of external sources the fields are still coupled, i.e:

$$S[q^{(1)}, q^{(2)}]_{J_1,J_2=0} = -\frac{{q^{(2)}}^2}{2\beta} - \frac{{q^{(1)}}^2}{2P\sigma_2^2} - iq^{(2)}y - P\sigma_1^2 x^2\left(q^{(1)}q^{(2)}\right)^2 \tag{2.68}$$

Note: One can go back to the real field (neuron activation) description anytime by performing the inverse HST, namely substituting

$$\exp\left(-\frac{{q^{(l)}}^2}{2\beta}\right) = \left(\frac{\beta}{2\pi}\right)^{\frac{N_l}{2}} \int \mathcal{D}h^{(l)} \exp\left(-\beta {h^{(l)}}^2 - ih^{(l)}q^{(l)}\right) \tag{2.69}$$

and so the partition function including the neuron activations would read:

$$Z_\mu = N \int \mathcal{D}q\mathcal{D}h \exp\left(-2\beta {h^{(2)}}^2 - 2P\sigma_2^2 {h^{(1)}}^2 - iq^{(1)}h^{(1)} - iq^{(2)}h^{(2)} - iq^{(2)}y\right) \tag{2.70}$$
$$\times \exp\left(P\sigma_1^2\left[J^{(1)} - iq^{(1)}\left(q^{(L)} - iJ^{(L)}\right)\right]^2 x^2\right)$$

where $\mathcal{D}q\mathcal{D}h = \mathcal{D}q^{(1)}\mathcal{D}h^{(1)}\mathcal{D}q^{(2)}\mathcal{D}h^{(2)}$ However, one can now easily integrate out the dual variable and be left with only the activation integral. These two approaches are equivalent, as they result in probability distributions that are the Fourier transform of one another. However, due to its simplicity the dual variable representation is preferred here.

The Gaussian Process limit

This interaction comes from the finite width of the network. In the infinite width (N_1) limit the last term vanishes as $\sigma_1 \sim \frac{1}{N_1}$. As for the rest of the terms, the hidden layer dual is a Gaussian and becomes a delta function in the large N_2 limit and so what is left is the output layer dual, which becomes a shifted Gaussian. In other words, the infinite width limit of a Gaussian process is strictly recovered for the dual variable $q^{(2)}$ and since the Fourier transform of a Gaussian is still a Gaussian with an inverse variance, the system admits a Gaussian process limit in the output $h^{(2)}$. To be more specific:

$$\lim_{N_1\to\infty} Z_\mu = \lim_{N_2\to\infty} N \int \mathcal{D}q^{(2)}\mathcal{D}q^{(1)} \exp\left(S[q^{(1)}, q^{(2)}]\right) \tag{2.71}$$

$$= N \int \mathcal{D}q^{(2)}\mathcal{D}q^{(1)} \exp\left(-\frac{{q^{(2)}}^2}{2\beta} - \frac{{q^{(1)}}^2}{2P\sigma_2^2} - iq^{(2)}y\right)$$

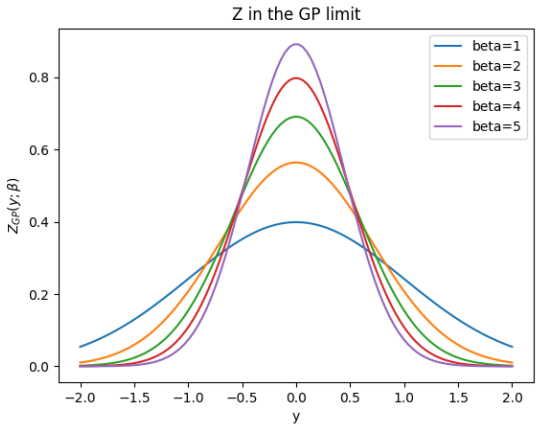
$$= Ne^{\frac{-\beta y^2}{2}} \int \mathcal{D}q^{(2)} \exp\frac{\left(q^{(2)} - i\beta y\right)^2}{2\beta} \int \mathcal{D}q^{(1)} \exp\left(-\frac{{q^{(1)}}^2}{2P\sigma_2^2}\right)$$

$$= N\frac{1}{(\pi\beta)^{N_2/2}} \exp\left(\frac{-\beta y^2}{2}\right) \tag{2.72}$$

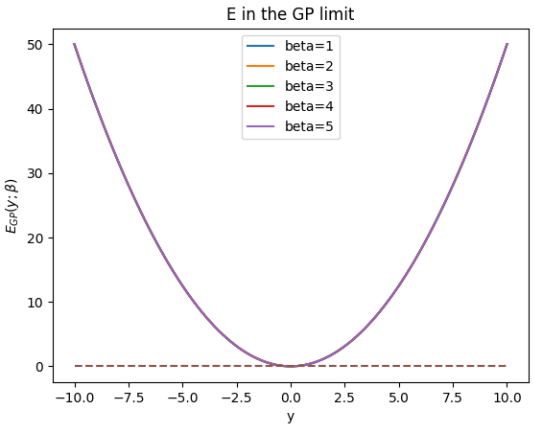
A graph of the partition function as well as the average value of the error are shown in figure (2.2)

There are two facts of importance that arise from this, one conceptual and one computational. The conceptual being that the $\mathcal{D}q^{(1)}$ results in a delta function of the weights of the second to last layer when we admit an infinite width limit for the last layer. One can interpret this in the following way: if the last layer is an infinite collection numbers drawn from the normal distribution, the simplest configuration that can still retrieve the GP limit is by having all the neurons in the previous layer be uniform. Also, by considering the $N_{l-1} \to \infty$ limit one gets rid of the interaction term so the "free theory" can still be obtained by a finite sized last layer. For any limit of N_l we recover the same result, it is the N_{l-1} scaling that is of importance here.

The computational fact is that the output layer dual distribution is a shifted Gaussian. This means that when one goes about to compute correlation functions of the sort $\langle q^{(2)}q^{(2)}\rangle$ the result ought not



(a) *The partition function of an infinite width 2-layer neural network for different temperatures. Notice that the partition function is a Gaussian distribution of the outputs.*



(b) *The energy associated with an infinite width NN. Notice that the function retains its parabolic shape with a minimum at zero for all values of the temperature*

Figure 2.2: *The partition function (a) and average value of the error (b) for the Gaussian process limit.*

be a simple factor of $\sqrt{\pi}/2$ as usual. Therefore, in order to keep the calculation simple any $n-$point function should be on the shifted variable: i.e be of the form $\left\langle \left(q^{(2)} - i\beta y\right)^n \left(q^{(2)} - i\beta y\right)^n \right\rangle$

Verification of the result (direct integration)

Of course, if one were to evaluate the integral we should expect the same result as in the case where only one H.S transformation is performed on the ouput layer, and indeed this holds true (see A.1.6). In addition, one sees the ratio P/N appearing as the perturbative parameter, in line with what is done in [13] where this ratio is kept fixed as N tends to infinity.

Corrections to the GP limit

A perturbative expansion of the coupling term in $\frac{1}{N_1}$ can give the finite width corrections to the GP a double scaling of $\frac{1}{N_2}$, can also be examined, giving rise to a forced delta constrain on the previous layer. This is not mandated though since, as stated, generally the last layer width is kept

at $N_l \sim \mathcal{O}(1)$. Examining the perturbative analysis on $\frac{1}{N_1}$ only, take again eq (2.67), this can be split into a free and an interacting part

$$S[q^{(1)}, q^{(2)}] = S_{\text{free}} + S_{\text{int}}, \tag{2.73}$$

$$S_{\text{free}} = -\frac{{q^{(2)}}^2}{2\beta} - \frac{{q^{(1)}}^2}{2P\sigma_l^2} - iq^{(2)}y, \tag{2.74}$$

$$S_{\text{int}} = -P\sigma_1^2 x^2 \left(J^{(1)} - iq^{(1)}\left(J^{(2)} - iq^{(2)}\right)\right)^2 \tag{2.75}$$

Notice that both the input x and the sources J only appear in the interaction term. This a testament on the fact that in the infinite width limit it does not matter what the input x is, or the activation for all that matter since the output is always a Gaussian with width $1/\beta$ .The interacting part, in the absence of sources, can be perturbed for small $\lambda = P\sigma_1^2|x^2| \sim \frac{1}{N_2}$ and the partition function now reads

$$Z_{NGP} = Ne^{\frac{-\beta y^2}{2}} \int \mathcal{D}q^{(2)} \int \mathcal{D}q^{(1)} \exp \frac{\left(q^{(2)} - i\beta y\right)^2}{2\beta} \exp\left(-\frac{{q^{(1)}}^2}{2P\sigma_2^2}\right) \sum_{n=1}^{\infty} \frac{\lambda^2}{n!} \left(q^{(1)}q^{(2)}\right)^{2n} \tag{2.76}$$

$$= N'e^{-\frac{\beta^2 y^2}{2}} \left[1 + \lambda \left\langle q^{(1)}q^{(1)}q^{(2)}q^{(2)} \right\rangle - \lambda\beta^2 y^2 \left\langle q^{(1)}q^{(1)} \right\rangle + \lambda \left\langle \left(q^{(1)}q^{(1)}q^{(2)}\right) \right\rangle + \mathcal{O}(\lambda^2)\right] \tag{2.77}$$

where the average values are assumed to be taken over centered Gaussians of the free action

$$\langle q^n \rangle = \int \mathcal{D}q^{(2)}\mathcal{D}q^{(1)} \left(q^n\right) e^{S_{\text{free}}[q^{(1)}, q^{(2)}]} \tag{2.78}$$

so the last term of eq (2.77) vanishes and the rest terms can be calculated from Wick's theorem to finally obtain

$$Z_{NGP} = N'exp\left(-\frac{\beta y^2}{2}\right)\left(1 + \lambda(\frac{1}{2} - \beta^2 y^2)\right) \tag{2.79}$$

Note that this is a function of x and y in contrast to the Gaussian process limit where the output is a Gaussian in y for any input. Here the introduction of the non-zero coupling λ introduces the input into the partition function. This is still a uniform distribution in the infinite temperature limit and a delta function in the zero temperature limit. For intermediate values of β this mostly looks like a Gaussian in y except in the strong coupling (small width) regime where two minima appear as can

be seen in figure (2.3). These two minima are the targets of the learning algorithm and their specific values are determined by the dataset.

Note that this is the finite width correction to Z_μ for a single example. To implement multiple examples the partition function reads

$$Z_{NGP} = N'' \prod_{\mu=1}^{P} e^{-\frac{\beta y_\mu^2}{2}} \left[1 + P\sigma_1^2 x_\mu^2 \left\langle q_\mu^{(1)} q_\mu^{(1)} q_\mu^{(2)} q_\mu^{(2)} \right\rangle - \lambda\beta^2 y_\mu^2 \left\langle q_\mu^{(1)} q_\mu^{(1)} \right\rangle + P\sigma_1^2 x_\mu^2 \left\langle \left(q_\mu^{(1)} q_\mu^{(1)} q_\mu^{(2)} \right) \right\rangle \right] \tag{2.80}$$

$$= N'' \prod_{\mu=1}^{P} \exp\left(-\frac{\beta y_\mu^2}{2}\right) \left(1 + P\sigma_2^2 x_\mu^2 (\frac{1}{2} - \beta^2 y^2)\right) + \mathcal{O}\left(\frac{1}{N_1^2}\right) \tag{2.81}$$

Here N_1 is kept finite, in the double scaling limit the $\beta^2 y^2$ term vanishes and one is left with just an ordinary Gaussian for any pair of $\{\beta, \lambda\}$. One can follow with higher order corrections to the infinite width limit but since the end goal is to perturb a non-linear solution around the linear partition function, the correction of up to $\mathcal{O}(\frac{1}{N^2})$ is kept, which is good enough for relatively wide networks. Now, this result can be pushed forward by taking into account non-linearities arising from the source derivatives of Z_{NGP}.

Cubic activation

A perturbative expansion of the coupling term in $\frac{1}{N_1}$ can give the finite width corrections to the $\mathcal{GP}$ limit. Some intuition has been obtained by the fact that, in studying the finite width corrections one moves from a free theory to an interacting $\phi^2\psi^2$ theory. It is natural to ask what the interaction looks like when we consider the non-linear corrections. We observe that the sources only appear in the interacting component of the action

$$\frac{\delta S}{\delta J_1} = 2\lambda \left(J^{(1)} - \mathrm{i}q^{(1)} \left(J^{(2)} - \mathrm{i}q^{(2)}\right)\right), \quad \frac{\delta S}{\delta J_2} = -2\mathrm{i}\lambda q^{(1)} \left(J^{(2)} - \mathrm{i}q^{(2)}\right) \tag{2.82}$$

Version of January 31, 2025– Created July 12, 2025 - 19:12

And so now the first order correction in g to the partition function, with a $\sigma(x) = 1 + gx^3$ activation function will be

$$Z_{NL} = \left[1 + 2g\left(\frac{\delta}{\delta J_1}\right)^2 \left(\left(\frac{\delta}{\delta J_2}\right)^2 - \frac{\delta}{\delta J_2} y + \mathcal{O}(g^2)\right)\right] Z\Big|_{J_1,J_2=0} \tag{2.83}$$

$$= Z + 48g\lambda^3\beta y \left\langle q^{(1)}q^{(1)}q^{(1)}q^{(1)}\ q^{(2)}q^{(2)}q^{(2)}q^{(2)}\right\rangle + 16g\lambda^3 \left\langle \beta^3 y^3\right\rangle \tag{2.84}$$

One can see than now moments of $\left\langle {q^{(1)}}^4 {q^{(2)}}^4\right\rangle$ are encoded. It is curious to see that this correction is of third order in lambda and so the resulting partition function is $\sim \mathcal{O}(g)\cdot\mathcal{O}\left(\frac{1}{N_1^3}\right)\cdot\mathcal{O}\left(\frac{1}{N_2}\right)$. At the end the partition function be

$$Z_{NL} \sim N e^{-\frac{\beta y^2}{2}} \left(1 + \frac{\lambda}{2}(1 - 2\beta^2 y^2)\right)\left(1 + g(24\lambda^3\beta y + 16\lambda^3\beta^3 y^3)\right) \tag{2.85}$$

Figure (2.4) summarizes the results of the introduction of the non-linearity. The minima that appear in the $\mathcal{NGP}$ correction get accentuated and resist more effectively in the "aligning" tendency of the system as it cools down. Since training can be seen as cooling the system down to the ground-state, this means that the introduction of the non-linearity will help the learning algorithm land on the target minima.

2.3.2 Norm learning discussion

In the infinite width limit the output can only be a simple Gaussian. In the first order expansion of the finite width correction we see the inclusion of $\beta^2 y^2$ which creates two more extrema in the function (therefore some values of y start to be preferred more than in the Gaussian). The position of these extremas is controlled by the width perturbation parameter λ and therefore by $|x|$ so we see a first improvement in *learning* as the expressivity of the network increases. A further improvement in learning is obtained by the introduction of the first order in g where now the minima in the energy become more steep and less resistant to temperature.

Version of January 31, 2025– Created July 12, 2025 - 19:12

2.4 Multiple examples

One problem of the above analysis is that all the resulting expressions posses a rotational invariance in the input (only the norm of the input appears). We know that this is not the case in the real world and networks can distinguish between combinatorial equivalent input examples. In cases like image recognition for example, simply rearranging the pixels can produce a vastly different image and even simple architectures can detect that. This means that the arrangement of neurons matter. In the above treatment though any norm preserving operation on the input would give rise to the same resulting partition function. In the linear case this is evident since the integral of $\left(W^{(l)}h^{(l-1)}(x) - h^{(l)}(x)\right)^2$ is an even function of x. This also holds true to the non-linear case and is an artifact of the choice to decouple the input from the dual variables by considering only a single input example. While this gives rise to simple expressions and interpretation, it is far from machine learning practice since we want these networks to learn concepts i.e functions $f(x) \mapsto y^\mu \ \forall \mu$ and not a simple mapping $f(x) \mapsto y$. In section 2.4.1 we again compute the source integral for multiple examples and in section 2.4.2 we quote the small g approximation to the partition function.

2.4.1 Evaluating the source integral

From now on since all the indices need be recovered, Einstein summation is heavily employed. When same indexes are not summed this will be stated explicitly. The full partition function for the two-layer network reads (A.1.9):

$$Z = \mathcal{N} \int \mathcal{D}q \ \exp\left(-\frac{q_a^\mu q_a^\mu}{2\beta} - i q_a^\mu y_a^\mu\right) \phi\left(J^{(1)}, J^{(2)}\right) \tag{2.86}$$

where $\mathcal{N} = (\sigma_2/\beta)^{N_2 N_1/2}(2\pi\beta)^{-N_2 N_1/2}$ is a constant prefactor and ϕ encodes the source dependence:

$$\phi\left(J^{(1)}, J^{(2)}\right) = |M|^{-\frac{N_1}{2}} \exp\left(\frac{1}{2} J^{(1)}_{b;\mu} x_k^\mu M^{-1}_{kf} J^{(1)}_{b;\nu} x_f^\nu\right) \tag{2.87}$$

Where $M_{kf} = x_k^\mu x_f^\nu Q_a^\mu Q_a^\nu$. Notice again that in the $J \to 0$ limit equation (2.86) reduces to (1.40)

Version of January 31, 2025– Created July 12, 2025 - 19:12

and one recovers the linear network limit.

2.4.2 First order correction in g

The first order correction in the partition function for a cubic activation function for multiple examples is given by

$$Z_g(\beta) \approx \lim_{J_1,J_2\to 0}\left[1+2g\left(\frac{\delta}{\delta J^{(1)}_{a;\mu}}\right)^2\left(\left(\frac{\delta}{\delta J^{(2)}_{a;\mu}}\right)^2-\frac{\delta}{\delta J^{(2)}_{a;\mu}}y^\mu_a\right)+\mathcal{O}(g^2)\right]Z(J) \tag{2.88}$$

Plugging Z from eq. (2.86) into (2.88) we get, in simplified notation:

$$Z_g(\beta) = Z(\beta)\Big|_{j,J=0} + 2g\left[\sum_{a,b,c,\mu,\nu,\lambda}\left(\frac{\delta}{\delta J^\nu_b}\frac{\delta}{\delta J^\nu_b}-y^\lambda_c\frac{\delta}{\delta J^\lambda_c}\right)\frac{\delta}{\delta j^\mu_a}\frac{\delta}{\delta j^\mu_a}Z_L\right]_{j,J=0} \tag{2.89}$$

$$\tag{2.90}$$

Evaluating the derivatives (see A.1.10) the final partition function becomes

$$\begin{aligned}\tilde{Z}_g(\beta) = \tilde{Z}(\beta)\Big|_{j,J=0} - \frac{2g}{|M|^{\frac{N_1}{2}}}\Bigg[&\frac{N_1}{2}\frac{S^{\lambda\lambda}}{|M|^2}\Bigg(-\left(\frac{N_1}{2}+1\right)\frac{1}{|M|}\left(M^{-1}\partial M^\lambda\right)^2 \\ &+\left(M^{-1}\partial\partial M^\lambda - M^{-1}\partial M M^{-1}\partial M\right)\Bigg)+\frac{N_1}{2|M|}\left(M^{-1}\partial M^\lambda X^\lambda M^{-1}\partial M^\lambda M^{-1}X^\lambda\right) \\ &+X^\lambda\left(M^{-1}\partial M^\lambda M^{-1}-M\partial\partial M^\lambda M^{-1}+M^{-1}\partial M^\lambda M^{-1}\partial M^\lambda M^{-1}\right)X^\lambda \\ &-Y^\lambda M^{-1}\partial M^\lambda + X^\lambda M^{-1}Y^\lambda\partial M^\lambda M^{-1}X^\lambda\Bigg]\end{aligned} \tag{2.91}$$

where $\tilde{Z}$ denotes the Fourier transform of the partition function, $\left(\partial M^\lambda\right)_{ij} = \frac{\partial M_{ij}}{\partial J^\lambda_d}$, $\left(\partial\partial M^\lambda\right)_{ij} = \frac{\partial^2 M_{ij}}{\partial J^\lambda_d \partial J^\lambda_d}$, $\left(X^\lambda\right)_i = x^\lambda_i$, $\left(Y^\lambda\right)_i = y^\lambda_i$. The explicit calculation can be found in the appendix A.1.10. The partition function is of order $\mathcal{O}\left(N_1^{-N_1/2}\right)$, it is however beyond the scope of this thesis to further

evaluate this function, as the large complexity of the expression would mean numerical errors would dominate over the small increase in performance the non-linearity offers (cf. A.2.3).

What we have achieved is a marginalization of the partition function, with the dynamical variables completely integrated out. For any choice of activation function that includes cubic terms in its Taylor expansion, equation (2.91) would constitute the first order correction to the linear model in terms of the added non-linearity.

2.5 Outlook and future work

In this thesis, two are the main points of interest. How and in which regime does the partition function of a neural network correspond to a probability distribution of its output and what is the effects of a small polynomial non-linearity on that probability distribution. We see that the deterministic likelihood is not enough to account for multiple examples but also that, even for a shallow network, the resulting marginalized probability distributions are very complex. Nonetheless, for single inputs the effect of the non-linearity is very apparent in shaping the output distribution. The question remains, how is this generalized to the multiple example case and what happens when one uses more realistic activation functions?

Version of January 31, 2025– Created July 12, 2025 - 19:12

References

[1] OpenAI, *GPT-4 Technical Report*, 2024.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, *Mastering the Game of Go with Deep Neural Networks and Tree Search*, 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012.

[4] D. A. Roberts, *Why is AI hard and Physics simple?*, 2021.

[5] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, *A high-bias, low-variance introduction to Machine Learning for physicists*, Physics Reports **810**, 1 (2019).

[6] J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61**, 85 (2015).

[7] D. A. Roberts, S. Yaida, and B. Hanin, *The Principles of Deep Learning Theory*, Cambridge University Press, 2022, https://deeplearningtheory.com.

[8] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, *Deep Neural Networks as Gaussian Processes*, 2018.

[9] A. Jacot, F. Gabriel, and C. Hongler, *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*, 2020.

[10] S. Yaida, *Non-Gaussian processes and neural networks at finite widths*, 2020.

[11] J. Zavatone-Veth and C. Pehlevan, *Exact marginal prior distributions of finite Bayesian neural networks*, in *Advances in Neural Information Processing Systems*, edited by M. Ranzato,

A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, volume 34, pages 3364–3375, Curran Associates, Inc., 2021.

[12] P. Mehta and D. J. Schwab, *An exact mapping between the Variational Renormalization Group and Deep Learning*, 2014.

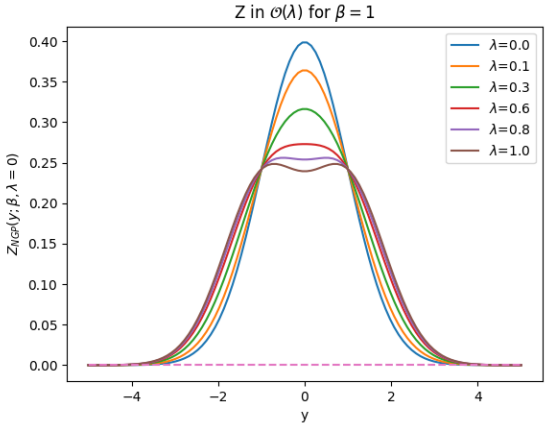
[13] Q. Li and H. Sompolinsky, *Statistical Mechanics of Deep Linear Neural Networks: The Back-propagating Kernel Renormalization*, Physical Review X **11** (2021).

[14] J. A. Zavatone-Veth, A. Canatar, and C. Pehlevan, *Asymptotics of representation learning in finite Bayesian neural networks*, CoRR **abs/2106.00651** (2021).

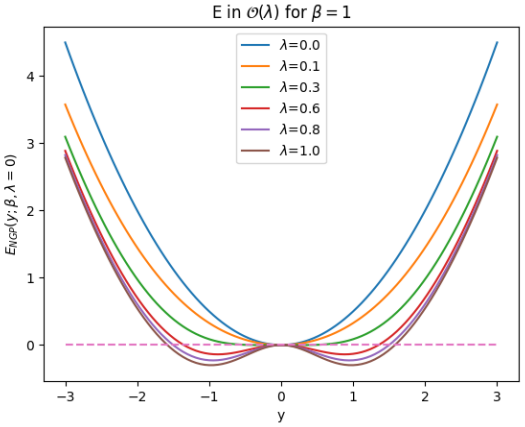
[15] A. Alhejaili and A. AlGhamedi, *A Review of Saddle-Point Approximation: Theory and Applications*, Advances and Applications in Statistical Sciences **18**, 67 (2025).

[16] A. Byerly, T. Kalganova, and I. Dear, *A Branching and Merging Convolutional Network with Homogeneous Filter Capsules*, CoRR **abs/2001.09136** (2020).

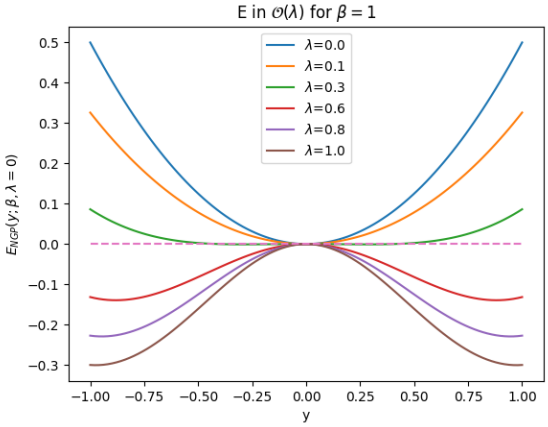
[17] *Meta AI, Papers with code*, https://paperswithcode.com/sota/image-classification-on-mnist, Licenced under CC-BY-SA, Accessed: 2025-01-27.



(a) *The partition function of a finite width 2-layer neural network for different values of the perturbing parameter λ. Notice that the partition function is still centered around zero but now dips to and below the $Z = 0$ line (dashed line).*

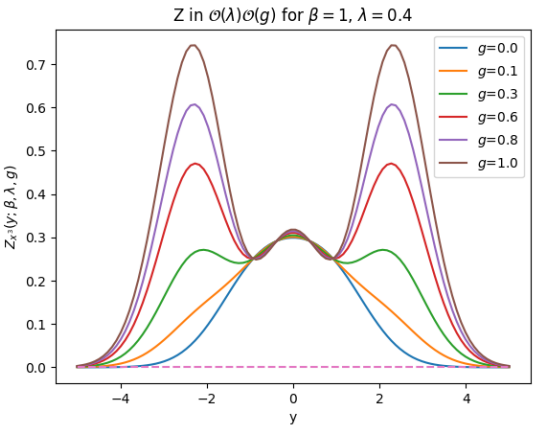


(b) *The energy associated with an infinite width NN. Notice that the function retains its parabolic shape with a minimum at zero for all values of the temperature but fluctuates wildly for some values of y*

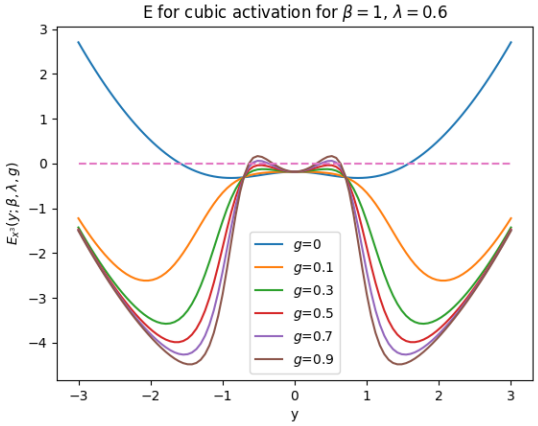


(c) *Zoomed in version of graph (2.3b) Notice that around 0 the function behaves as a parabola with a minima at 0*

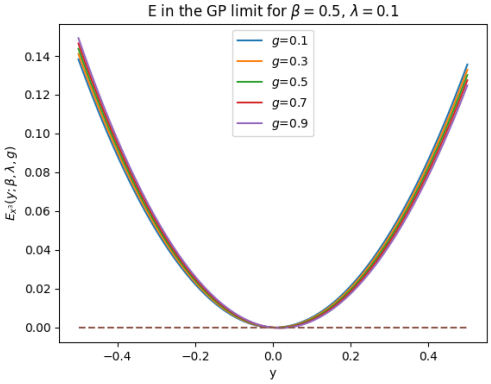
Figure 2.3: *The partition function (2.3a) and average value of the error (2.3b) for the non-Gaussian process limit. It is apparent that the preference of the system is still at $y = 0$ where the minimum of the energy lies. However, we can relax the preference for an output of 0, at a fixed temperature, by changing the perturbing parameter λ.*



(a) *The partition function of a finite width, non-linear 2-layer neural network for different values of the perturbing parameter λ. Notice that the partition function is not centered around zero anymore and exhibits multiple peaks and troughs.*



(b) *The energy associated with an infinite width NN. Notice that the function has now shifted its minimum away from 0. So while the first order correction in λ could not shift the maximum likelihood output away from 0, the first order correction in g can.*



(c) *The average error of the system for a smaller values of the perturbing parameter λ. One can see the diminishing effect this has on shifting the minimum of the error and in the infinite width limit ($\lambda = 0$) the effect of the non-linearity is completely suppressed.*

Figure 2.4: *The partition function and energy in $\mathcal{O}(g)$. One can see the error minimum shifting away from zero*

Appendix A

Calculations & Experiments

A.1 Calculations

In this section some intermediate steps between calculations are included

A.1.1 Expanding the posterior

In order to arrive to equation (1.7) start with Bayes rule and substitute $\mathcal{S} = h \cup x := h, x$

$$p(\mathcal{S}|y) = \frac{p(y|\mathcal{S})p(\mathcal{S})}{p(y)} \tag{A.1}$$

$$p(h, x|y) = \frac{p(y|h, x)p(h, x)}{p(y)} \tag{A.2}$$

Since one wants the posterior to be conditioned on the boundary conditions like $p(h|x, y)$ observe that

$$p(h|x, y) = \frac{p(h, x, y)}{p(x, y)} = \frac{p(h, x, y)}{p(y)} \frac{p(y)}{p(x, y)} = \frac{p(h, x|y)}{p(x|y)} \tag{A.3}$$

rearanging

$$p(h, x|y) = p(h|x, y)p(x|y) \tag{A.4}$$

Now the LHS of equations (A.2) and (A.4) are the same so equating the RHS of both:

$$\frac{p(y|h,x)p(h,x)}{p(y)} = p(h|x,y)p(x|y) \tag{A.5}$$

solving for the posterior

$$p(h|x,y) = \frac{p(y|h,x)p(h,x)}{p(y)p(x|y)} = \frac{p(y|h,x)p(h|x)p(x)}{p(y)p(x|y)} \tag{A.6}$$

The only thing left to prove to arrive at (1.7) is that $p(x)/p(y)p(x|y) = p(y|x)$ but this is just Bayes theorem so we finally get

$$p(h|x,y) = \frac{p(y|h,x)p(h|x)}{p(y|x)} \tag{A.7}$$

Which is exactly equation (1.7) with the μ indices suppressed.

A.1.2 Computing the multi entry linear partition function

The partition function can be computed by direct integration upon completing the square

$$Z(x,q;\beta) = e^{-\frac{|q_2^\mu|^2}{2\beta}} \int_{-\infty}^{+\infty} \mathcal{D}\mathcal{W} e^{-iq_c^\mu(\sum_{a_1=1}^{n_l}\sum_{a_2=1}^{n_l} W_{ca_1}^{(2)} W_{a_1a_2}^{(1)} x_{a_2}^\mu) - \sum_{l^\star=1} \frac{1}{2\sigma_{l^\star}^2}\left(W_{ab}^{l^\star}\right)^2} \tag{A.8}$$

$$= e^{-\frac{|q_2^\mu|^2}{2\beta}} \prod_{a_1=1}^{n_l}\prod_{a_2=1}^{n_l} \int_{-\infty}^{+\infty} \mathcal{D}\mathcal{W} e^{-iq_c^\mu(W_{ca_1}^{(2)} W_{a_1a_2}^{(1)} x_{a_2}^\mu) - \frac{1}{2\sigma_1^2}\left(W_{a_2a_1}^{(1)}\right)^2 - \frac{1}{2\sigma_2^2}\left(W_{a_2a_1}^{(2)}\right)^2} \tag{A.9}$$

$$= e^{-\frac{|q_2^\mu|^2}{2\beta}} \prod_{a_1,a_2}^{n_l} \int_{-\infty}^{+\infty} \mathcal{D}\mathcal{W} e^{\frac{1}{2\sigma_2^2}\left(W_{ca_1}^{(2)} - iq_c^\mu W_{a_1a_2}^{(1)} x_{a_2}^\mu\right)^2 + \frac{\sigma_2^2}{2}\left(q_c^\mu W_{a_1a_2}^{(1)} x_{a_2}^\mu\right)^2 + \frac{1}{\sigma_1^2}\left(W_{a_2a_1}^{(1)}\right)^2} \tag{A.10}$$

$$= e^{-\frac{|q_2^\mu|^2}{2\beta}} \prod_{a_1,a_2}^{n_l} \int_{-\infty}^{+\infty} \mathcal{D}\mathcal{W} e^{\frac{1}{2\sigma_2^2}\left(W_{ca_1}^{(2)} - iq_c^\mu W_{a_1a_2}^{(1)} x_{a_2}^\mu\right)^2} e^{\frac{1}{\sigma_1^2} W_{a_2c}^{(1)}\left(\sigma_2^2\sigma_1^2 q_c^\mu x_c^\mu q_c^\nu x_{a_2}^\nu + \delta_{ca_2}\right) W_{a_2a_1}^{(1)T}} \tag{A.11}$$

A.1.3 Integrating out W

Simplifying the expressions amounts to completing the square first in terms of W_2 and then in terms of W_1.

Version of January 31, 2025– Created July 12, 2025 - 19:12

Integrating out W_2

$$S = -iq(W_2 W_1 x) - \frac{W_2^2}{2\sigma_2^2} - \frac{W_1^2}{2\sigma_1^2} + J_1 W_1 x + J_2 W_2 W_1 x \tag{A.12}$$

$$S = -\frac{1}{2\sigma_2^2}\left[W_2^2 + 2\sigma_2^2(iqW_2 W_1 x - J_2 W_2 W_1 x)\right] - \frac{W_1^2}{2\sigma_1^2} + J_1 W_1 x \tag{A.13}$$

$$= -\frac{1}{2\sigma_2^2}\left[W_2^2 + 2W_2\sigma_2^2(iq - J_2)W_1 x\right] - \frac{W_1^2}{2\sigma_1^2} + J_1 W_1 x \tag{A.14}$$

$$= -\frac{1}{2\sigma_2^2}\left[W_2^2 + 2W_2\sigma_2^2(iq - J_2)W_1 x + \left(\sigma_2^2(iq - J_2)W_1 x\right)^2\right] + \left(\frac{\sigma_2(iq - J_2)}{2}W_1 x\right)^2 - \frac{W_1^2}{2\sigma_1^2} + J_1 W_1 x \tag{A.15}$$

$$= -\frac{1}{2\sigma_2^2}\left[W_2 + \left(\sigma_2^2(iq - J_2)W_1 x\right)\right]^2 + \left(\frac{\sigma_2(iq - J_2)}{2}W_1 x\right)^2 - \frac{W_1^2}{2\sigma_1^2} + J_1 W_1 x \tag{A.16}$$

Integrating out $W1$

$$\left(\frac{\sigma_2(iq - J_2)}{2}W_1 x\right)^2 - \frac{W_1^2}{2\sigma_1^2} + J_1 W_1 x = -\left(\frac{\sigma_2(q + iJ_2)}{2}W_1 x\right)^2 - \frac{W_1^2}{2\sigma_1^2} + J_1 W_1 x \tag{A.17}$$

$$= -\frac{1}{2\sigma_1^2}\left[W_1^2 + W_1^2\sigma_1^2\sigma_2^2(q + iJ_2)^2 x^2 + 2\sigma_1^2 J_1 W_1 x\right] \tag{A.18}$$

$$= -\frac{1}{2\sigma_1^2}\left[W_1^2\left(1 + \sigma_1^2\sigma_2^2(q + iJ_2)^2 x^2\right) + 2\sigma_1^2 J_1 W_1 x\right] \tag{A.19}$$

$$= -\frac{1}{2\sigma_1^2\left(1 + \sigma_1^2\sigma_2^2(q + iJ_2)^2 x^2\right)}\left[W_1^2 + 2W_1(1 + \sigma_1^2\sigma_2^2(iq - J_2)^2 x^2)\sigma_1^2 J_1 x\right] \tag{A.20}$$

$$= -\frac{1}{2\sigma_1^2(1 + k)}\left[W_1^2 + 2W_1(1 + k)\sigma_1^2 J_1 x\right], \text{ where } k = \sigma_1^2\sigma_2^2(q + iJ_2)^2 x^2 \tag{A.21}$$

$$= -\frac{1}{2\sigma_1^2(1 + k)}\left[W_1^2 + 2W_1(1 + k)\sigma_1^2 J_1 x + \left((1 + k)\sigma_1^2 J_1 x\right)^2\right] + \frac{(1 + k)J_1 x}{2} \tag{A.22}$$

$$= -\frac{1}{2\sigma_1^2(1 + k)}\left[W_1 + \frac{\sigma_1^2 J_1 x}{1 + k}\right]^2 + \frac{(1 + k)J_1 x}{2} \tag{A.23}$$

A.1.4 The stationary points

The values of the stationary points are obtained by solving the equation (2.28). This results in 3 roots, their exact value given by:

$$q_0 = \sqrt[3]{\frac{\sqrt{\left(27n_1^2y^2\beta^4+108n_1^3\beta^3\right)\kappa^6+\left(108y^4\beta^4+540n_1y^2\beta^3+324n_1^2\beta^2\right)\kappa^4+\left(324n_1\beta-216y^2\beta^2\right)\kappa^2+108}}{54\kappa^3}+\frac{\frac{(\kappa^2\beta n_1+1)\mathrm{i}y\beta}{\kappa^2}-\frac{3\beta y\mathrm{i}}{\kappa^2}}{6}+\frac{\mathrm{i}y^3\beta^3}{27}} - \frac{\frac{\kappa^2\beta n_1+1}{3\kappa^2}+\frac{y^2\beta^2}{9}}{\sqrt[3]{\frac{\sqrt{\left(27n_1^2y^2\beta^4+108n_1^3\beta^3\right)\kappa^6+\left(108y^4\beta^4+540n_1y^2\beta^3+324n_1^2\beta^2\right)\kappa^4+\left(324n_1\beta-216y^2\beta^2\right)\kappa^2+108}}{54\kappa^3}+\frac{\frac{(\kappa^2\beta n_1+1)\mathrm{i}y\beta}{\kappa^2}-\frac{3\beta y\mathrm{i}}{\kappa^2}}{6}+\frac{\mathrm{i}y^3\beta^3}{27}}} - \frac{1\mathrm{i}y\beta}{3} \tag{A.24}$$

$$q_1 = \left(\frac{\sqrt{3}\,\mathrm{i}}{2}-\frac{1}{2}\right)q_0 \tag{A.25}$$

$$q_2 = \left(\frac{-\sqrt{3}\,\mathrm{i}}{2}-\frac{1}{2}\right)q_0 \tag{A.26}$$

Version of January 31, 2025– Created July 12, 2025 - 19:12

A.1.5 Norm learning, iterations

The procedure for marginalizing over W in order to retrieve the dual variable description for a two layer network is given below

$$Z(\beta) = \prod_{\mu=1}^{P} (2\pi\beta)^{\frac{-n_L}{2}} \int \mathcal{D}W \mathcal{D}q^{(L)} \exp\left(-\frac{{q^{(L)}}^2}{2\beta} - iq^{(L)}y - iq^{(L)}h^{(L)} - \sum_l \frac{{W^{(L)}}^2}{2P\sigma_l^2} + \sum_l J_a^{(l)} h_a^{(l)} \right) \tag{A.27}$$

$$= \prod_{\mu=1}^{P} (2\pi\beta)^{\frac{-n_L}{2}} \int \mathcal{D}q^{(L)} \mathcal{D}\mathcal{W} e^{-\frac{{q^{(L)}}^2}{2\beta} - iq^{(L)}y - \sum_{l<L} \frac{{W^{(l)}}^2}{2P\sigma_l^2} + J_a^{(l)} h_a^{(l)}} \exp\left(\left(J^{(L)} - iq^{(L)} \right) W^{(L)} h^{(L-1)} - \frac{{W^{(L)}}^2}{2P\sigma_L^2} \right) \tag{A.28}$$

$$= \prod_{\mu=1}^{P} (2\pi\beta)^{\frac{-n_L}{2}} \int \mathcal{D}q^{(L)} \mathcal{D}\mathcal{W}^{(l<L)} e^{-\frac{{q^{(L)}}^2}{2\beta} - iq^{(L)}y - \sum_{l<L} \frac{{W^{(l)}}^2}{2P\sigma_l^2} + J_a^{(l)} h_a^{(l)}} \int \mathcal{D}W^{(L)} \exp\left(\left(J^{(L)} - iq^{(L)} \right) W^{(L)} h^{(L-1)} - \frac{{W^{(L)}}^2}{2P\sigma_L^2} \right) \tag{A.29}$$

$$= \prod_{\mu=1}^{P} (2\pi\beta)^{\frac{-n_L}{2}} \int \mathcal{D}q^{(L)} \mathcal{D}\mathcal{W}^{(l<L)} e^{-\frac{{q^{(L)}}^2}{2\beta} - iq^{(L)}y - \sum_{l<L} \frac{{W^{(l)}}^2}{2P\sigma_l^2} + J_a^{(l)} h_a^{(l)}} (2\pi P\sigma_L^2)^{\frac{N_L^2}{2}} \exp\left(P\sigma_L^2 \left[\left(J^{(L)} - iq^{(L)} \right) h^{(L-1)} \right]^2 \right) \tag{A.30}$$

$$= \prod_{\mu=1}^{P} (2\pi\beta)^{\frac{-n_L}{2}} \int \mathcal{D}q^{(L)} \mathcal{D}\mathcal{W}^{(l<L)} e^{-\frac{{q^{(L)}}^2}{2\beta} - iq^{(L)}y - \sum_{l<L} \frac{{W^{(l)}}^2}{2P\sigma_l^2} + J_a^{(l)} h_a^{(l)}} (2\pi P\sigma_L^2)^{\frac{N_L^2}{2}} \exp\left(-P\sigma_L^2 \left[\left(q^{(L)} - iJ^{(L)} \right) h^{(L-1)} \right]^2 \right) \tag{A.31}$$

$$= \prod_{\mu=1}^{P} \frac{(2\pi P\sigma_L^2)^{\frac{N_L^2}{2}}}{(2\pi\beta)^{\frac{N_L}{2}} (2\pi P\sigma_L^2)^{\frac{N_{L-1}}{2}}} \int \mathcal{D}q^{(L)} \mathcal{D}q^{(L-1)} \mathcal{D}\mathcal{W}^{(l<L)} e^{-\frac{{q^{(L)}}^2}{2\beta} - iq^{(L)}y - \sum_{l<L} \frac{{W^{(l)}}^2}{2P\sigma_l^2} + J_a^{(l)} h_a^{(l)}} \tag{A.32}$$

$$\times \exp\left(-\frac{{q^{(L-1)}}^2}{2P\sigma_L^2} - iq^{(L-1)} \left(q^{(L)} - iJ^{(L)} \right) h^{(L-1)} \right) \tag{A.33}$$

$$= \prod_{\mu=1}^{P} \frac{(2\pi P\sigma_L^2)^{\frac{N_L^2}{2}}}{(2\pi\beta)^{\frac{N_L}{2}} (2\pi P\sigma_L^2)^{\frac{N_{L-1}}{2}}} \int \mathcal{D}q^{(L)} \mathcal{D}q^{(L-1)} \mathcal{D}\mathcal{W}^{(l<L-1)} e^{-\frac{{q^{(L)}}^2}{2\beta} - \frac{{q^{(L-1)}}^2}{2P\sigma_L^2} - iq^{(L)}y - \sum_{l<L-1} \frac{{W^{(l)}}^2}{2P\sigma_l^2} + J_a^{(l)} h_a^{(l)}} \tag{A.34}$$

$$\times \int \mathcal{D}W^{(L-1)} \exp\left(\left[-iq^{(L-1)} \left(q^{(L)} - iJ^{(L)} \right) + J^{(L-1)} \right] W^{(L-1)} h^{(L-2)} - \frac{{W^{(L-1)}}^2}{2P\sigma_{L-1}^2} \right) \tag{A.35}$$

$$= \prod_{\mu=1}^{P} \frac{(2\pi P\sigma_L^2)^{\frac{N_L^2}{2}}}{(2\pi\beta)^{\frac{N_L}{2}} (2\pi P\sigma_L^2)^{\frac{N_{L-1}}{2}}} \int \mathcal{D}q^{(L)} \mathcal{D}q^{(L-1)} \mathcal{D}\mathcal{W}^{(l<L-1)} e^{-\frac{{q^{(L)}}^2}{2\beta} - \frac{{q^{(L-1)}}^2}{2P\sigma_L^2} - iq^{(L)}y - \sum_{l<L-1} \frac{{W^{(L-1)}}^2}{2P\sigma_l^2} + J_a^{(l)} h_a^{(l)}} \tag{A.36}$$

$$\times (2\pi P\sigma_{L-1}^2)^{\frac{N_1^2}{2}} \exp\left(P\sigma_{L-1}^2 \left[-iq^{(L-1)} \left(q^{(L)} - iJ^{(L)} \right) + J^{(L-1)} \right]^2 {h^{(L-2)}}^2 \right) \tag{A.37}$$

One can see this iterative approach of: performing the H.S transformation → performing the Gaussian integral → move to next layer and repeat.

Version of January 31, 2025– Created July 12, 2025 - 19:12

A.1.6 Recovering the ZVP result

Integrating out one of the dual variables yields:

$$N\int \mathcal{D}q^{(2)}\mathcal{D}q^{(1)}\exp\left(S[q^{(1)},q^{(2)}]_{J_1,J_2=0}\right) \tag{A.38}$$

$$= N\int \mathcal{D}q^{(2)}\mathcal{D}q^{(1)}\exp\left(-\frac{{q^{(2)}}^2}{2\beta}-\frac{{q^{(1)}}^2}{2P\sigma_2^2}-iq^{(2)}y-P\sigma_1^2x^2\left(q^{(1)}q^{(2)}\right)^2\right) \tag{A.39}$$

$$= N\int \mathcal{D}q^{(2)}\exp\left(-\frac{{q^{(2)}}^2}{2\beta}-iq^{(2)}y\right)\int \mathcal{D}q^{(1)}\exp\left(-{q^{(1)}}^2\left(\frac{1}{2P\sigma_2^2}+P\sigma_1^2x^2{q^{(2)}}^2\right)\right) \tag{A.40}$$

$$= N\int \mathcal{D}q^{(2)}\exp\left(-\frac{{q^{(2)}}^2}{2\beta}-iq^{(2)}y\right)\left(\frac{1}{2P\sigma_2^2}\left(1+2P^2\sigma_1^2\sigma_2^2x^2{q^{(2)}}^2\right)\right)^{-\frac{N_1}{2}} \tag{A.41}$$

$$= \frac{N}{(2P\sigma_2^2)^{\frac{N_1}{2}}}\int \mathcal{D}q^{(2)}\exp\left(-\frac{{q^{(2)}}^2}{2\beta}-iq^{(2)}y-\frac{N_1}{2}ln\left(1+2P\sigma_1^2\sigma_2^2|x|^2{q^{(2)}}^2\right)\right) \tag{A.42}$$

with $\kappa^2 = 2P^2\sigma_1^2\sigma_2^2x^2$ this result is identical to the one obtained by only one H.S transformation, see eq. (1.40) for reference.

A.1.7 GP corrections

$$Z_{NGP} = Ne^{\frac{-\beta^2y^2}{2}}\int \mathcal{D}q^{(2)}\int \mathcal{D}q^{(1)}\exp\frac{\left(q^{(2)}-i\beta y\right)^2}{2\beta}\exp\left(-\frac{{q^{(1)}}^2}{2P\sigma_2^2}\right)\sum_{n=1}^{\infty}\frac{\lambda^2}{n!}\left(q^{(1)}q^{(2)}\right)^{2n} \tag{A.43}$$

$$\approx Ne^{\frac{-\beta^2y^2}{2}}\int \mathcal{D}q^{(2)}\int \mathcal{D}q^{(1)}\exp\frac{\left(q^{(2)}-i\beta y\right)^2}{2\beta}\exp\left(-\frac{{q^{(1)}}^2}{2P\sigma_2^2}\right)\left(1+\lambda\left(q^{(1)}q^{(2)}\right)^2\right) \tag{A.44}$$

$$= Ne^{\frac{-\beta^2y^2}{2}}\int \mathcal{D}q^{(2)}\int \mathcal{D}q^{(1)}\exp\frac{\left(q^{(2)}-i\beta y\right)^2}{2\beta}\exp\left(-\frac{{q^{(1)}}^2}{2P\sigma_2^2}\right) \tag{A.45}$$

$$\times\left(1+\lambda\left(q^{(1)}(q^{(2)}-i\beta y)\right)^2+\lambda\left(q^{(1)}\right)^2\left(\beta^2y^2+2i\beta yq^{(2)}\right)\right) \tag{A.46}$$

$$= N'e^{-\frac{\beta^2y^2}{2}}\left[1+\lambda\left\langle q^{(1)}q^{(1)}q^{(2)}q^{(2)}\right\rangle-\lambda\beta^2y^2\left\langle q^{(1)}q^{(1)}\right\rangle+\lambda\left\langle\left(q^{(1)}q^{(1)}q^{(2)}\right)\right\rangle\right] \tag{A.47}$$

Version of January 31, 2025– Created July 12, 2025 - 19:12

A.1.8 Non-linear corrections

$$Z_{NL} = \left[1 + 2g\left(\frac{\delta}{\delta J_1}\right)^2\left(\left(\frac{\delta}{\delta J_2}\right)^2 - \frac{\delta}{\delta J_2}y + \mathcal{O}(g^2)\right)\right] Z\Big|_{J_1,J_2=0} \tag{A.48}$$

$$= \left\langle 1 + 2g\left[4\lambda^2 {q^{(1)}}^2 {q^{(2)}}^2\left({q^{(1)}}^3 {q^{(2)}}^2 - 2i\lambda y {q^{(1)}}^2 q^{(2)}\right)\right]\right\rangle \tag{A.49}$$

$$= \langle 1\rangle + 8g\lambda^2\left\langle\left({q^{(1)}}^5 {q^{(2)}}^4 - 2i\lambda {q^{(1)}}^4 {q^{(2)}}^3\right)\right\rangle \tag{A.50}$$

$$= Z - 16ig\lambda^3\left\langle {q^{(1)}}^4 {q^{(2)}}^3\right\rangle \tag{A.51}$$

$$= Z - 16ig\lambda^3\left\langle {q^{(1)}}^4\left(q^{(2)} + i\beta y\right)^3\right\rangle_{\text{both centered Gaussians}} \tag{A.52}$$

$$= Z - 16ig\lambda^3\left\langle {q^{(1)}}^4\left(3i\beta y {q^{(2)}}^2 - i\beta^3 y^3\right)\right\rangle \tag{A.53}$$

$$= Z + 48g\lambda^3\beta y\left\langle q^{(1)}q^{(1)}q^{(1)}q^{(1)}\ q^{(2)}q^{(2)}q^{(2)}q^{(2)}\right\rangle + 16g\lambda^3\left\langle\beta^3 y^3\right\rangle \tag{A.54}$$

A.1.9 Computing the partition function

$$Z(\beta, J_a^\mu | x_a^\mu, y_a^\mu) = \int \mathcal{D}W^{(1)}\mathcal{D}W^{(2)} \exp\left(-\beta\left|W_{ab}^{(2)}W_{bc}^{(1)}x_c^\mu - y_a^\mu\right|^2 - \frac{{W_{ab}^{(2)}}^2}{2\sigma_2^2} - \frac{{W_{ab}^{(1)}}^2}{2\sigma_1^2}\right) \tag{A.55}$$

$$\times \exp\left(+J_{a;\mu}^{(2)}W_{ab}^{(2)}W_{bc}^{(1)}x_c^\mu + J_{a;\mu}^{(1)}W_{ab}^{(1)}x_b^\mu\right) \tag{A.56}$$

$$= \mathcal{N}_1 \int \mathcal{D}W^{(1)}\mathcal{D}W^{(2)}\mathcal{D}q \exp\left(-\frac{q_a^\mu q_a^\mu}{2\beta} + iq_a^\mu\left(W_{ab}^{(2)}W_{bc}^{(1)}x_c^\mu - y_a^\mu\right) - \frac{{W_{ab}^{(1)}}^2}{2\sigma_1^2} - \frac{{W_{ab}^{(2)}}^2}{2\sigma_2^2}\right) \tag{A.57}$$

$$\times \exp\left(+J_{a;\mu}^{(2)}W_{ab}^{(2)}W_{bc}^{(1)}x_c^\mu + J_{a;\mu}^{(1)}W_{ab}^{(1)}x_b^\mu\right) \tag{A.58}$$

$$= \mathcal{N}_1 \int \mathcal{D}W^{(1)}\mathcal{D}W^{(2)}\mathcal{D}q \exp\left(-\frac{q_a^\mu q_a^\mu}{2\beta} - iq_a^\mu + iW_{ab}^{(2)}\left(q_{a;\mu}W_{bc}^{(1)}x_c^\mu - iJ_{a;\mu}^{(2)}W_{bc}^{(1)}x_c^\mu\right)\right) \tag{A.59}$$

$$\times \exp\left(J_{a;\mu}^{(1)}W_{ab}^{(1)}x_b^\mu - \frac{{W_{ab}^{(1)}}^2}{2\sigma_1^2} - \frac{{W_{ab}^{(2)}}^2}{2\sigma_2^2}\right) \tag{A.60}$$

where $\mathcal{N}_1 = (2\pi\beta)^{-N_2 N_1/2}$. Integrating out $W^{(2)}_{ab}$ and redefining $Q^\mu_a = q_{a;\mu} - iJ^{(l)}_{a;\mu}$

$$Z = \mathcal{N}_2 \int \mathcal{D}W^{(1)}\mathcal{D}q \exp\left(-\frac{q^\mu_a q^\mu_a}{2\beta} + iq^\mu_a\right) \exp\left(-\frac{\sigma_2^2}{2}\left(Q^\mu_a W^{(1)}_{bc} x^\mu_c\right)^2 - \frac{\left(W^{(1)}_{ab}\right)^2}{2\sigma_1^2} + J^{(1)}_{a;\mu} W^{(1)}_{ab} x^\mu_b\right) \tag{A.61}$$

with $\mathcal{N}_2 = (\sigma_2/\beta)^{N_2 N_1/2}$. Notice that the two square terms can be written in a compact form as

$$\exp\left(-\frac{{W^{(1)}_{ab}}^2}{2\sigma_1^2} - \frac{\sigma_2^2}{2}\left(Q^\mu_a W^{(1)}_{bc} x^\mu_c\right)^2\right) = -\frac{\left(W^{(1)^T}\right)_{db} W^{(1)}_{ab}}{2\sigma_1^2} - \left(W^{(1)^T}\right)_{bd} \frac{\sigma_2^2}{2} x^\nu_d Q^\nu_a Q^\mu_a x^\mu_c \left(W^{(1)}_{bc}\right) \tag{A.62}$$

$$= -\frac{\left(W^{(1)^T}\right)_{eb} W^{(1)}_{bc}}{2\sigma_1^2}\left(\delta_{ce} + \sigma_1^2\sigma_2^2 x^\nu_e Q^\nu_a Q^\mu_a x^\mu_c\right) = \frac{\left(W^{(1)^T}\right)_{eb} W^{(1)}_{bc}}{2} M_{ce} \tag{A.63}$$

where M_{ce} is an $N_0 \times N_0$ matrix given by

$$M_{ce} = \frac{\delta_{ce}}{\sigma_1^2} + \sigma_2^2 x^\nu_e Q^\nu_a Q^\mu_a x^\mu_c \tag{A.64}$$

And so

$$Z = \mathcal{N}_2 \int \mathcal{D}W^{(1)}\mathcal{D}q \exp\left(-\frac{q^\mu_a q^\mu_a}{2\beta} + iq^\mu_a\right) \exp\left(J^{(1)}_{a;\mu} W^{(1)}_{ab} x^\mu_b - \frac{1}{2} W^{(1)}_{bc} M_{ce} {W^{(1)}_{eb}}^T\right) \tag{A.65}$$

The last step is to complete the square again and perform the $W^{(1)}$ integral. We can write $W^{(1)}_{ab} = \frac{1}{2}\left(W^{(1)}_{ab} + W^{(1)T}_{ba}\right)$ and so

$$\exp\left(J^{(1)}_{a;\mu} W^{(1)}_{ab} x^\mu_b - \frac{1}{2} W^{(1)}_{bc} M_{ce} {W^{(1)}_{eb}}^T\right) = \frac{1}{2} W^{(1)}_{ab} J^{(1)}_{a;\mu} x^\mu_b + \frac{1}{2} J^{(1)}_{a;\mu} x^\mu_b {W^{(1)}_{ba}}^T - \frac{1}{2} W^{(1)}_{bc} M_{ce} {W^{(1)}_{eb}}^T \tag{A.66}$$

$$= -\frac{1}{2}\left(W^{(1)}_{bc} - J^{(1)}_{b;\mu} x^\mu_k M^{-1}_{kc}\right) M_{ce} \left({W^{(1)}_{eb}}^T - M^{-1}_{ef} J^{(1)}_{b;\nu} x^\mu_f\right) + \frac{1}{2} J^{(1)}_{b;\mu} x^\mu_k M^{-1}_{kf} J^{(1)}_{b;\nu} x^\nu_f \tag{A.67}$$

The first term of eq. (A.66) is just a shifted Gaussian and therefore, the partition function with the weights integrated gives equation (2.86) in the main text.

Version of January 31, 2025– Created July 12, 2025 - 19:12

A.1.10 Calculation of the source derivatives

The source-including part o f the partition function is (2.87) reprinted here

$$\phi\left(J^{(1)},J^{(2)}\right)=|M|^{-\frac{N_1}{2}}\exp\left(\frac{1}{2}\sum_{b,\nu,\mu,k,f}j_b^\mu x_k^\mu M_{kf}^{-1}j_b^\nu x_f^\nu\right) \tag{A.68}$$

Where the matrix M encodes the J dependence

$$M_{ce}=\frac{\delta_{ce}}{\sigma_1^2}+\sigma_2^2 x_e^\nu Q_a^\nu Q_a^\mu x_f^\mu\equiv\frac{1}{\sigma_1^2}\left(\delta_{ce}+\kappa^2\sum_{\mu,\nu}x_e^\nu\sum_a\left(Q_a^\nu Q_a^\mu\right)x_f^\mu\right),\quad\kappa^2=\sigma_1^2\sigma_2^2 \tag{A.69}$$

Notice that the the dataset "Greek indices" are cross contracted so we can simplify the notation further:

$$S^{\mu\nu}=\sum_{k,f}x_k^\mu M_{kf}^{-1}x_f^\nu \tag{A.70}$$

$$\sum_{b,\mu,\nu,k,f}j_b^\mu x_k^\mu(M_{kf})^{-1}j_b^\nu x_f^\nu=\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\Rightarrow \tag{A.71}$$

$$\phi\left(j,J\right)=|M|^{-\frac{N_1}{2}}\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right) \tag{A.72}$$

Calculating the hidden layer source derivatives

The hidden layer source dependence is only encoded in the exponential term. Therefore the following apply

$$\frac{\delta j_a^\mu}{\delta j_b^\nu}=\delta_{ab}\delta^{\mu\nu},\quad\frac{\delta M_{ab}}{\delta j_c^\lambda}=0 \tag{A.73}$$

Calculating the first derivative with respect to j:

$$\frac{\delta\phi}{\delta j_c^\lambda} = |M|^{-\frac{N_1}{2}} \frac{\delta}{\delta j_c^\lambda}\left(\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu} S^{\mu\nu} j_b^\mu j_b^\nu\right)\right) \tag{A.74}$$

$$= |M|^{-\frac{N_1}{2}}\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}\left(\delta_{bc}\delta^{\mu\lambda}S^{\mu\nu}j_b^\nu\right) + \frac{1}{2}\sum_b\sum_{\mu,\nu}\left(\delta_{bc}\delta^{\nu\lambda}S^{\mu\nu}j_b^\mu\right)\right) e^{\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu} \tag{A.75}$$

$$= |M|^{-\frac{N_1}{2}}\left(\frac{1}{2}\sum_\nu S^{\lambda\nu}j_c^\nu + \frac{1}{2}\sum_\mu S^{\mu\lambda}j_c^\mu\right) e^{\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu} \tag{A.76}$$

$$= |M|^{-\frac{N_1}{2}}\left(\sum_\nu S^{\lambda\nu}j_c^\nu\right) e^{\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu} \xrightarrow{j\to 0} 0 \tag{A.77}$$

And the second derivative:

$$\frac{\delta^2\phi}{\delta j_c^\lambda \delta j_d^\xi} = |M|^{-\frac{N_1}{2}}\left(\delta_{cd}S^{\lambda\xi}e^{\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu} + \left(\sum_\nu S^{\xi\nu}j_d^\nu\right)e^{\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu}\right) \xrightarrow{j\to 0} |M|^{-\frac{N_1}{2}}\delta_{cd}S^{\lambda\xi} \tag{A.78}$$

One can see the pattern here, the only part of the functional derivative that survives is the one where j does not appear as a multiplicative term. Repeated differentiation will result is a derivative of the form:

$$\frac{\delta^n\phi}{\delta j_{a_1}^{\nu_1}\delta j_{a_2}^{\nu_2}\cdots\delta j_{a_n}^{\nu_n}} = \delta_{a_1a_2}\delta^{\nu_1\nu_2}S^{\nu_1\nu_2}\cdots\delta_{a_{n-1}a_n}\delta^{\nu_{n-1}\nu_n}S^{\nu_{n-1}\nu_n} \tag{A.79}$$

Calculating the output source derivative

$$\left(\frac{\delta}{\delta J}\right)^n\phi = \left(\frac{\delta}{\delta J}\right)^{n-1}\left(\frac{\delta}{\delta J}\right)\left(|M|^{-\frac{N_1}{2}}\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right)\right) \tag{A.80}$$

$$= \left(\frac{\delta}{\delta J}\right)^{n-1}\left[|M|^{-\frac{N_1}{2}}\frac{\delta}{\delta J}\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right) + \exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right)\frac{\delta|M|^{-\frac{N_1}{2}}}{\delta J}\right] \tag{A.81}$$

But in the $j \to 0$ limit the first term goes to zero as

$$\frac{\delta}{\delta J}\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right) = \frac{\delta}{\delta S^{\xi\chi}}\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right)\frac{\delta S^{\xi\chi}}{\delta J} \sim j\cdot\frac{\delta S^{\xi\chi}}{\delta J}\to 0 \tag{A.82}$$

$$= \sum_b\sum_{\mu\nu} j_b^\mu j_b^\nu \frac{\delta S^{\mu\nu}}{\delta S^{\xi\chi}}\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right)\frac{\delta S^{\xi\chi}}{\delta J} \tag{A.83}$$

$$= \sum_b j_b^\xi j_b^\chi \exp\left(\frac{1}{2}\sum_b S^{\xi\chi}j_b^\xi j_b^\chi\right)\frac{\delta S^{\xi\chi}}{\delta J} \sim j\cdot j\to 0 \tag{A.84}$$

In essence the only terms that survive are of the sort

$$\exp\left(\frac{1}{2}\sum_b\sum_{\mu,\nu}S^{\mu\nu}j_b^\mu j_b^\nu\right)\frac{\delta^n}{\delta j^n}|M(j)|^{-\frac{N_1}{2}} \to \frac{\delta^n}{\delta j^n}|M(0)|^{-\frac{N_1}{2}} \tag{A.85}$$

In the case that different combinations need be considered, the calculation is more involved. Specifically for the cubic activation case one has:

$$Z_{NL} = \left[1+2g\left(\frac{\partial}{\partial J_d^\xi}\frac{\partial}{\partial J_d^\xi}\frac{\partial}{\partial j_c^\lambda}\frac{\partial}{\partial j_c^\lambda} + y_d^\xi\frac{\partial}{\partial J_d^\xi}\frac{\partial}{\partial j_c^\lambda}\frac{\partial}{\partial j_c^\lambda}\right)\right]Z_L \tag{A.86}$$

Examining the terms one by one, the linear partition function is corrected by

$$\frac{\partial}{\partial J_d^\xi}\frac{\partial}{\partial J_d^\xi}\frac{\partial}{\partial j_c^\lambda}\frac{\partial}{\partial j_c^\lambda}\phi = \frac{\partial}{\partial J_d^\xi}\frac{\partial}{\partial J_d^\xi}\left(|M|^{-\frac{N_1}{2}}\delta_{cc}S^{\lambda\lambda}\exp\left(\frac{1}{2}S^{\mu\nu}j_b^\mu j_b^\nu\right)+\mathcal{O}(j)\right) \tag{A.87}$$

$$= \frac{\partial}{\partial J_d^\xi}\left(\left(\frac{\partial|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi}S^{\lambda\lambda}+|M|^{-\frac{N_1}{2}}\frac{\partial S^{\lambda\lambda}}{\partial J_d^\xi}+\mathcal{O}(j)\right)\exp\left(\frac{1}{2}S^{\mu\nu}j_b^\mu j_b^\nu\right)\right) \tag{A.88}$$

$$= \left(\frac{\partial^2|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi\partial J_d^\xi}S^{\lambda\lambda}+2\frac{\partial|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi}\frac{\partial S^{\lambda\lambda}}{\partial J_d^\xi}+|M|^{-\frac{N_1}{2}}\frac{\partial^2 S^{\lambda\lambda}}{\partial J_d^\xi\partial J_d^\xi}\right)e^{\frac{1}{2}S^{\mu\nu}j_b^\mu j_b^\nu}+\mathcal{O}(j) \tag{A.89}$$

Version of January 31, 2025– Created July 12, 2025 - 19:12

Examining, once again, the terms one by one, for the first term:

$$\frac{\partial^2|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi \partial J_d^\xi} S^{\lambda\lambda} = S^{\lambda\lambda}\frac{\partial}{\partial J_d^\xi}\left(-\frac{N_1}{2}\frac{|M|^{-\frac{N_1}{2}}}{|M|} tr\left(M_{ij}^{-1}\frac{\partial M_{jk}}{\partial J_d^\xi}\right)\right) \tag{A.90}$$

$$= -\frac{N_1}{2}S^{\lambda\lambda}\left(-\left(\frac{N_1}{2}+1\right)\frac{|M|^{-\frac{N_1}{2}}}{|M|^2} tr^2\left(M_{ij}^{-1}\frac{\partial M_{jk}}{\partial J_d^\xi}\right) + \frac{|M|^{-\frac{N_1}{2}}}{|M|} tr\left(M_{ij}^{-1}\frac{\partial^2 M_{jk}}{\partial J_d^\xi \partial J_d^\xi} + \frac{\partial M_{ij}^{-1}}{\partial J_d^\xi}\frac{\partial M_{jk}}{\partial J_d^\xi}\right)\right) \tag{A.91}$$

$$= -\frac{N_1}{2}S^{\lambda\lambda}\frac{|M|^{-\frac{N_1}{2}}}{|M|^2}\left(-\left(\frac{N_1}{2}+1\right)\frac{1}{|M|}\left(M_{ij}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi}\right)^2 + \left(M_{ij}^{-1}\frac{\partial^2 M_{ji}}{\partial J_d^\xi \partial J_d^\xi} - M_{im}^{-1}\frac{\partial M_{mn}}{\partial J_d^\xi}M_{mj}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi}\right)\right) \tag{A.92}$$

where in the first step Jacobi's formula is used to calculate the derivative of the determinant and in the second step the derivative of the inverse is calculated and the trace is replaced by the repeated use of the cross-contracted index i. Now the expression involves only M, M^{-1}, the determinant and the derivative of M. Now for the second term

$$2\frac{\partial|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi}\frac{\partial S^{\lambda\lambda}}{\partial J_d^\xi} = -2\frac{N_1}{2}\left(\frac{|M|^{-\frac{N_1}{2}}}{|M|}M_{ij}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi}x_k^\lambda x_f^\lambda\frac{\partial M_{kf}^{-1}}{\partial J_d^\xi}\right) \tag{A.93}$$

$$= -N_1\frac{|M|^{-\frac{N_1}{2}}}{|M|}\left(M_{ij}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi}x_k^\lambda x_f^\lambda M_{kg}^{-1}\frac{\partial M_{ge}}{\partial J_d^\xi}M_{ef}^{-1}\right) \tag{A.94}$$

here note that λ is a free index, again the expression involves only M, M^{-1}, the determinant and the derivative of M. And now for the third term:

$$|M|^{-\frac{N_1}{2}}\frac{\partial^2 S^{\lambda\lambda}}{\partial J_d^\xi \partial J_d^\xi} = |M|^{-\frac{N_1}{2}}\frac{\partial}{\partial J_d^\xi}\left(x_k^\lambda x_f^\lambda M_{kg}^{-1}\frac{\partial M_{ge}}{\partial J_d^\xi}M_{ef}^{-1}\right) \tag{A.95}$$

$$= -|M|^{-\frac{N_1}{2}}x_k^\lambda x_f^\lambda\left(M_{km}^{-1}\frac{\partial M_{mn}}{\partial J_d^\xi}M_{ng}^{-1} - M_{kg}\frac{\partial^2 M_{ge}}{\partial J_d^\xi \partial J_d^\xi}M_{ef}^{-1} + M_{kg}^{-1}\frac{\partial M_{ge}}{\partial J_d^\xi}M_{eu}^{-1}\frac{\partial M_{uv}}{\partial J_d^\xi}M_{vf}^{-1}\right) \tag{A.96}$$

Joining the 3 terms together:

$$\left(\frac{\partial^2|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi \partial J_d^\xi}S^{\lambda\lambda} + 2\frac{\partial|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi}\frac{\partial S^{\lambda\lambda}}{\partial J_d^\xi} + |M|^{-\frac{N_1}{2}}\frac{\partial^2 S^{\lambda\lambda}}{\partial J_d^\xi \partial J_d^\xi}\right)e^{\frac{1}{2}S^{\mu\nu}j_b^\mu j_b^\nu} + \mathcal{O}(j) \tag{A.97}$$

$$\overset{j,J\to 0}{=} -\frac{N_1}{2}S^{\lambda\lambda}\frac{|M|^{-\frac{N_1}{2}}}{|M|^2}\left(-\left(\frac{N_1}{2}+1\right)\frac{1}{|M|}\left(M_{ij}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi}\right)^2 + \left(M_{ij}^{-1}\frac{\partial^2 M_{ji}}{\partial J_d^\xi \partial J_d^\xi} - M_{im}^{-1}\frac{\partial M_{mn}}{\partial J_d^\xi}M_{mj}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi}\right)\right)$$

$$-\frac{N_1|M|^{-\frac{N_1}{2}}}{|M|}\left(M_{ij}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi}x_k^\lambda x_f^\lambda M_{kg}^{-1}\frac{\partial M_{ge}}{\partial J_d^\xi}M_{ef}^{-1}\right)$$

$$-x_k^\lambda x_f^\lambda\left(M_{km}^{-1}\frac{\partial M_{mn}}{\partial J_d^\xi}M_{ng}^{-1} - M_{kg}^{-1}\frac{\partial^2 M_{ge}}{\partial J_d^\xi \partial J_d^\xi}M_{ef}^{-1} + M_{kg}^{-1}\frac{\partial M_{ge}}{\partial J_d^\xi}M_{eu}^{-1}\frac{\partial M_{uv}}{\partial J_d^\xi}M_{vf}^{-1}\right)\Bigg]|M|^{-\frac{N_1}{2}}$$

And lastly for the second term of A.86:

$$y_d^\xi\frac{\partial}{\partial J_d^\xi}\frac{\partial}{\partial j_c^\lambda}\frac{\partial}{\partial j_c^\lambda}\phi = y_d^\xi\left(\frac{\partial|M|^{-\frac{N_1}{2}}}{\partial J_d^\xi}S^{\lambda\lambda} + |M|^{-\frac{N_1}{2}}\frac{\partial S^{\lambda\lambda}}{\partial J_d^\xi} + \mathcal{O}(j)\right)\exp\left(\frac{1}{2}S^{\mu\nu}j_b^\mu j_b^\nu\right) \tag{A.98}$$

$$= |M|^{-\frac{N_1}{2}}y_d^\xi M_{ij}^{-1}\frac{\partial M_{ji}}{\partial J_d^\xi} - x_k^\mu x_f^\nu M_{ki}^{-1}y_d^\xi\frac{\partial M_{ij}}{\partial J_d^x i}M_{jf}^{-1} \tag{A.99}$$

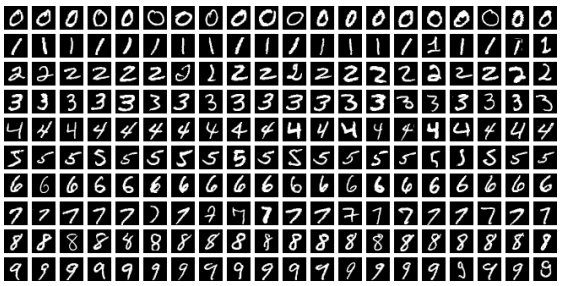
Now adding eq. (A.97) to eq. (A.98) we get the non-linear partition function in first order (A.86) concluding the derivation.

A.2 Numerical Experiments

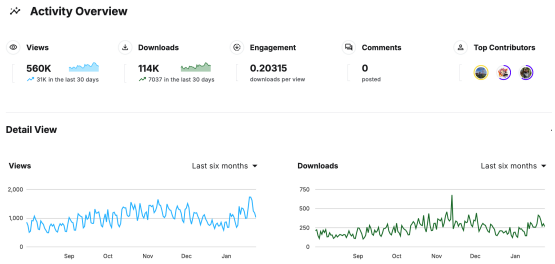
A number of numerical experiments have been performed to test known results, supplemental to the main text, using the MNIST dataset described in A.2.1. We use -a truncated version of- it to probe the distribution of weights in A.2.2 and examine some aspects of the training dynamics in A.2.3.

A.2.1 The MNIST dataset

The MNIST dataset contains 60.000 training images and 10.000 testing 28×28 pixel images of handwritten digits (see fig. A.1a) and will be used as the base dataset for our numerical experiments.



(a) *An instance of the MNIST dataset containing 200 (20 images of each digit) . By Suvanjanprasai - Own work, CC BY-SA 4.0, (Source).*



(b) *Activity overview of the MNIST dataset in Kaggle averaging just below 10.000 downloads a month with a steady popularity as a standard evaluation dataset (Source).*

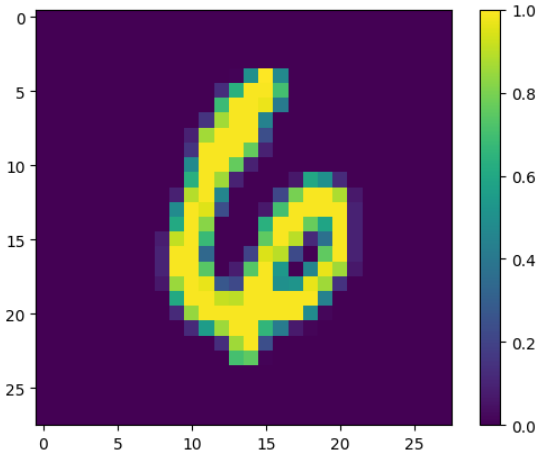
Figure A.1: *Some examples from the MNIST dataset (left) in gray-scale and an indicative activity overview in one of the most popular dataset websites.*

It is a widely used dataset for image classification tasks as well general machine learning tasks with a steady popularity (see fig. A.1b). State of the art models have achieved a 99.87% accuracy on the dataset [16] while many of the top ranking models outperform humans in this task [17]. Such models make extensive use of *convolutional layers* in the model architecture, but in our case fully connected layers are used and therefore the first step in every computation is to translate the 28×28 image into a 1×784 array.

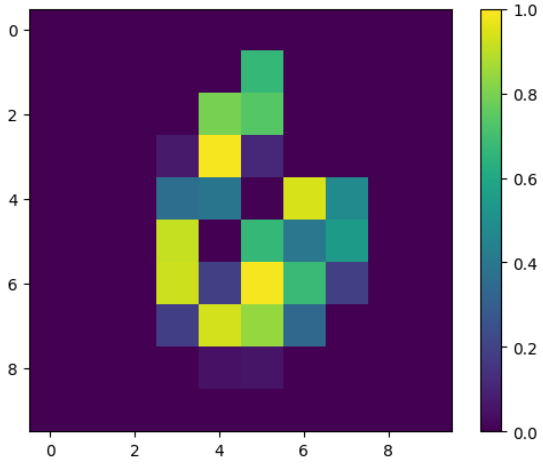
A.2.2 Weight Gaussianity

Since our approach focuses on a marginalized description where the weights have been integrated out, it has value to see the behavior of the values of these weights during actual training. To keep the computations simple in the beginning, we downgrade the MNIST dataset to a 10×10 scale (see A.2) and only keep the digits `'3'` and `'7'` for computational ease. We consider $L = 3$ layers with $N_L = 2$ output neurons to match the number of classes and train for $e = 10$ and $e = 100$ epochs.

The initial weights are drawn from Gaussian distributions (1.3), therefore a natural metric to keep track of would be how much do the weights deviate from Gaussianity during training. Thus we



(a) *An instance of the original* 28×28 *image from MNIST. This original dataset is used all of the numerics except A.2.2.*



(b) *The smaller* 10×10 *instance of the truncated dataset used in A.2.2.*

Figure A.2: *An instance of a digit from the dataset (left) and a downsized version of it (right). We used bilinear interpolation to downsample the image to test initialization and weight configuration properties of ensembles of networks.*

examine the relative entropy S between the weight distribution and the normal distribution, defined as:

$$S\left(P(\mathcal{W}^{(l)})\middle\|\mathcal{G}\right) = \sum_{ij} P\left(W_{ij}^{(l)}\right) ln\left(P\left(W_{ij}^{(l)}\right)e^{x^2}\right) \tag{A.100}$$

The results are summarized in table A.1. Notice that for a given choice of activation function the weight distribution becomes increasingly less Gaussian when one moves from input to output layer. This is true for both choices of training epochs. Interestingly, for $e = 300$ the increase in g seems to not correlate with an increased non-Gaussianty, as is the case with $e = 10$. This hints to the fact that the choice of activation function influences the convergence of the algorithm to the minimum. But given enough epochs any choice of activation function has an equal impact on the weight distribution when compared to the Gaussian.

Table A.1: *KL divergence of the network layers weights from the Gaussian distribution. The output dimension N_L being very low is the reason for the relative entropy of the last layer being large.*

	10 Epochs				*300 Epochs*			
	Linear	$0.1 \cdot x^3$	$0.5 \cdot x^3$	ReLU	Linear	$0.1 \cdot x^3$	$0.5 \cdot x^3$	ReLU
Input layer	6.29	8.45	10.50	21.54	20.51	12.50	17.05	17.39
Hidden layer	39.25	22.65	30.30	49.27	43.20	43.36	43.55	45.42
Output layer	146.66	130.03	142.60	138.16	135.89	141.71	147.67	134.57

The deviation of the weights can only be informative about the degree that the learning algorithm has impacted the initialized distribution, however, it is not indicative of the model performance.

A.2.3 Training

To assess model performance, we need to reformulate the classification problem of MNIST into a regression one, since, the output layer is of dimension different than the number of classes. This means that we need to transform the labels into arrays of the same size as the output layer. Since

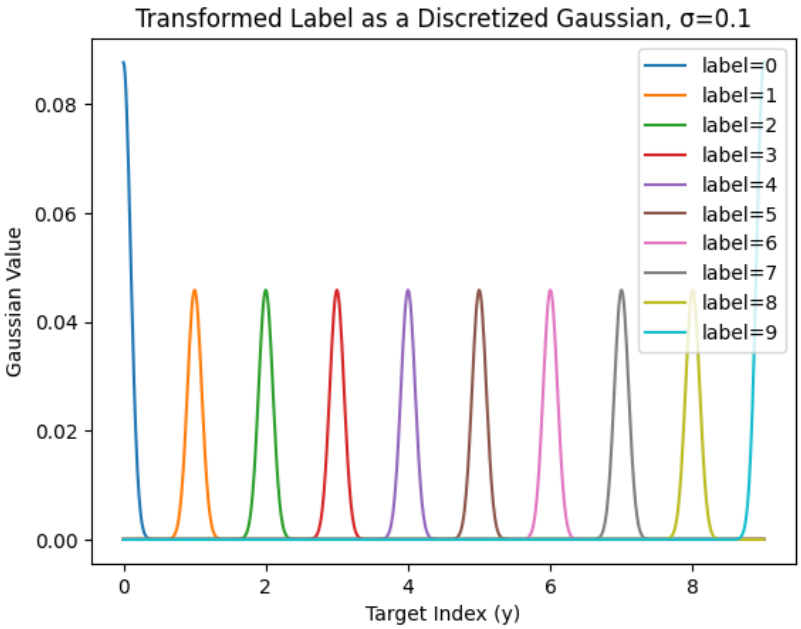
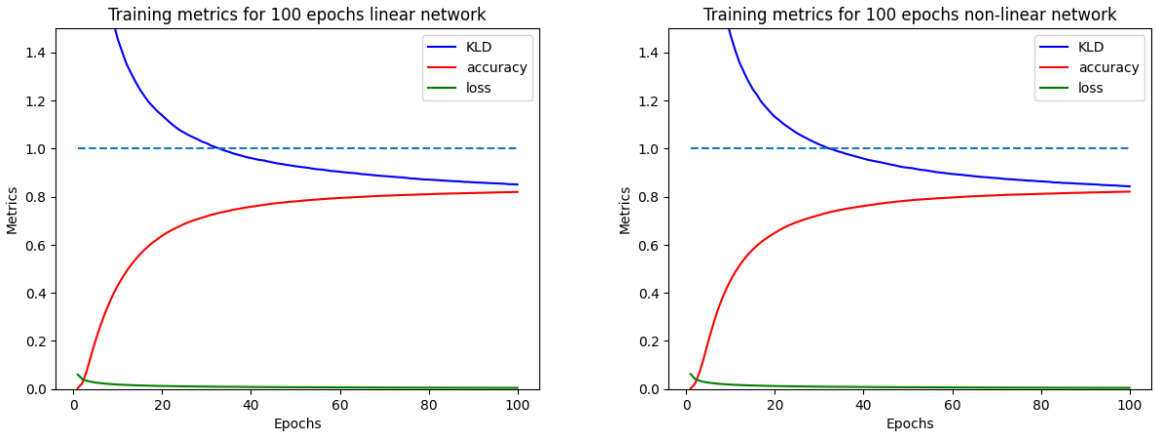


Figure A.3: *The distributions of the target labels for the regression task. Here a standard deviation of* 0.001 *has been chosen to eliminate overlap between the labels. Note that equal areas are assigned to each label forcing and so the classes on the edge practcally occupy less neurons of* $h^{(L)}$.

the output layer `h` is of dimension 784, we define equal dimension arrays and transform the labels y into Gaussian distributions around `int(len(h)/10)` (cf. fig. A.3).

We train the network for $e = 100$ epochs and plot the accuracy, loss and mutual entropy in figure A.4. See that for both the linear (A.4a) and non-linear cases (A.4b) the curve starts to flatten out around $e = 60$ epochs. We choose this number of epochs to extract the training metrics. The network is thus trained for $e = 60$ epochs for 3 choices of activation functions, $\phi(x) = x + gx^3$ for $g = 0, 0.1, 0.5$. The resulting training metrics are shown in table A.2. One can see an increase in train and validation accuracy of the model by $\sim 1\%$ accompanied by a small decrease in mutual entropy. This is a barely noticeable increase in performance however itis persistent with multiple runs of the network.



(a) *Training metric curves for* ***linear*** *network.* ***(b)*** *Training metric curves for* ***non-linear*** *network.*

Figure A.4: *The (training) accuracy, loss, and relative entropy (KLD) for a linear and non-linear ($g = 0.1$) FCN.*

Table A.2: *Train/Test metrics for $e = 60$ epochs of a linear and non-linear FCN using the MNIST data (total runtime 98 minutes). The non linear network uses $\phi(x) = x + g \cdot x^3$ as its activation function. a graphic representation of the metric curves is shown in figure A.4. Notice the similarity in the training metrics. However the non-linear network exhibits slightly performance.*

	Accuracy	S	Loss	Val Accuracy	Val S	Val Loss
$g = 0$	0.808	0.975	0.097	0.820	0.922	0.096
$g = 0.1$	0.811	0.947	0.097	0.822	0.898	0.096
$g = 0.5$	0.815	0,880	0.097	0.828	0.842	0.096