Prepared for submission to JHEP

Physics4ML

Lucas Leung[a]

[a] *Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Parks Road, Oxford OX1 3PU, UK*

E-mail: lucas.leung@physics.ox.ac.uk

Abstract: In recent years machine learning and data science techniques are increasingly used to solve numerous problems in theoretical physics. Whilst the applications of the techniques in data science are interesting problems in their own right (ML4Physics), recent investigations have also shown that physical principles, such as field theory techniques, symmetries and renormalisation, can be used to understand the principles of the techniques. Here, whilst an introduction to machine learning techniques will be given, the main goal will be to discuss one broad idea in Physics for Machine Learning – neural network field theory (NNFT). Through understanding this construction we can hopefully build a better and more rigorous understanding of the basic principles of machine learning, and get closer to a detailed theory of ML.

Contents

1 Basics of ML and NNs

We are interested in Machine Learning (ML). What is ML? Mitchell gives a succinct definition.

Definition 1.1. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P; if its performance at tasks in T, as measured by P, improves with experience E.

Bit of a weird definition. Let us look at what the terms mean.

- **The task T.** This is some kind of problem that we wish to solve. Of course, typically this will be too hard to solve with fixed programs written and designed by human beings. Classes of tasks include for example: classification, regression, transcription, etc.
- **The performance measure P.** This is a quantitative measure of the performance of a computer program, specific to the task being carried out by the system. Often times we measure the accuracy of a model, using a test set that the computer program has not been trained on.

- **The experience E.** ML algorithms are broadly categorised as **supervised** and **unsupervised**. A **dataset** is the set of data drawn from the world, collected for the purposes of the task T. Whilst unsupervised learning algorithms aim to experience the data set with many features and extract useful properties of the set, supervised learning algorithms often associate each data point with an additional label or target (see example below). The line between the two is not clear, and there are many others (like for example Reinforcement Learning where the algorithm interacts with an environment).

There are many additional issues that come with ML. Studying ML involves understanding generalisation and statistical learning theory. We want to minimise both the error coming from the data being trained on (**training error**) as well as its generalisation (**generalisation error**) - and the two extremes of training are called **underfitting** and **overfitting**. A choice of model, algorithm, hyperparameters (settings), regularisation etc. all contribute to the **representational capacity** of the model. Depending on the problem, a specific choice or algorithm might work a lot better than a different one, so optimising for an algorithm for a specific task is a big challenge in applications of machine learning to solve problems. Additionally, as Bayesian inference is often used in predicting models, a good understanding of maximum likelihood estimation is also needed in designing good ML algorithms.

There are however a lot of problems with basic ML:

- Curse of dimensionality. ML problems just become exceedingly difficult due to logarithmic scaling.
- Local constancy prior is not sufficient. Often times simply requiring a smooth function is not enough (c.f. classification problems - the number of contiguous regions do not grow with training samples).
- Manifold structure. Learning seems to be hopeless when focusing on the entire region - but focusing on a subspace ('manifold') seems to be the way to go. This is not well-understood in ML.

One direction of improvement is to understand **deep learning**, learning using deep neural networks. So we should talk about neural networks.

1.1 NN basics

Artificial neural networks are loosely modelled after the structures of neurones and synapses in our brains. Let's begin with a simple definition.

Definition 1.2. A **neural network** is a function

$$\phi_\theta : \mathbb{R}^d \to \mathbb{R}^D \tag{1.1}$$

with parameters θ. We can $x \in \mathbb{R}^d$ the **input**, $\phi_\theta(x) \in \mathbb{R}^D$ the **output**, $\phi_\theta \in \mathrm{Maps}(\mathbb{R}^d, \mathbb{R}^D)$ the **network** and $\mathcal{D}$ the **data**. The **data** $\mathcal{D}$ depends on the problem and involves at least a subset of $\mathbb{R}^d$ potentially paired with labels $y \in \mathbb{R}^D$.

The easiest case of course is when $D = 1$. This is then just a scalar field - and we can already learn a lot from scalar function.

The question now we want to ask is the following.

Question: What does a NN predict?

For a set of fixed parameters θ then the answer is clear - this is just given by the function $\phi_\theta(x)$. The true answer however is complicated by two aspects in particular:

Statistics. NNs need to be initialised. When they are initialised, the parameters θ are drawn from some distribution $P(\theta)$,

$$\theta \sim P(\theta) \tag{1.2}$$

Different draws will give different θ and hence different neural networks ϕ_θ. So the prediction ϕ_θ is not fundamental - what is fundamental is the moments,

$$\mathbb{E}[\phi_\theta(x_1)...\phi_\theta(x_n)] = \int d\theta P(\theta)\phi_\theta(x_1)...\phi_\theta(x_n) \tag{1.3}$$

Of course, the first and second moments are then just the average prediction and variance respectively. Expectations are across different initialisations, so we can see that moments are fundamental quantities in the statistical distribution. We can replace $\mathbb{E}[\cdot] = \langle\cdot\rangle$ later using our language - to indicate that the moments are just correlation functions of the statistical ensemble of neural networks.

Dynamics. Of course, the ML parameters θ are updated during the training process to solve problems. So we definitely have a trajectory in the **parameter space** $\theta(t) \in \mathbb{R}^{|\theta|}$ (and therefore in the output and function space). The flow, or trajectory, will be governed by some learning dynamics determined by the optimisation algorithm and the nature of the problem. An example of an optimisation algorithm is gradient descent. For example, given a problem in supervised learning where we have the data

$$\mathcal{D} = \left\{(x_\alpha, y_\alpha) \in \mathbb{R}^d \times \mathbb{R}^D\right\}_{\alpha=1}^{|\mathcal{D}|} , \tag{1.4}$$

and a loss function

$$\mathcal{L}[\phi_\theta] = \sum_{\alpha=1}^{|\mathcal{D}|} l(\phi_\theta(x_\alpha), y_\alpha) , \tag{1.5}$$

where l is the loss function (defined as, for example, the mean square error). The parameters θ can be optimised by gradient descent,

$$\frac{d\theta_i}{dt} = -\nabla_{\theta_i}\mathcal{L}[\phi_\theta] . \tag{1.6}$$

This defines a certain flow in the parameter space as defined by the loss function and the optimisation algorithm (the gradient descent).

Now of course, we can see that the statistics of the draw can be thought of as evolving

under training as well. In particular, we can think of the parameters $\theta(t)$ as drawn from the time-dependent distribution $P(\theta(t))$,

$$\theta(t) \sim P(\theta(t)) \ , \tag{1.7}$$

where t is the training time parameter. This gives time-dependent correlators,

$$G_t^{(n)}(x) = \langle \phi_\theta(x_1)...\phi_\theta(x_n) \rangle_t \ , \tag{1.8}$$

with the subscript t indicating time-dependence and the expectation is with respect to the distribution $P(\theta(t))$. If learning helps, then the $t \to \infty$ limit should give interesting correlation functions - this is what we will focus on later and we will find that in a certain supervised setting there is an exact analytical solution for this quantity.

So far, we have only talked about initialisation and training of neural networks. There is another important aspect of neural networks that we have missed - **architecture**. In particular, the correlation functions that we have computed above, which we can now write as,

$$G^{(n)}(x_1, ..., x_n) = \frac{1}{Z_\theta} \int d\theta \phi(x_i)...\phi(x_n) P(\theta) \ , \tag{1.9}$$

where the partition function is written as $Z_\theta = \int d\theta P(\theta)$. Where does the randomness of the fixed field configuration ϕ arise from? We can think of the functions ϕ as random functions (in field theory, this will be the fields themselves). The statistics of the ensemble of the ϕ here, however, unlike in the field theory where the probability density function is specified $P(\phi) = \exp(-S[\phi])$, is instead determined by the construction of the networks and the parameter space distribution. This means that the **architecture** of the neural network defines the functional from ϕ, as well as the choice of distributions from which the parameters θ are drawn from. This begs the question:

Question: How powerful is NN?

The answer to that of course depends on the **architecture**. We can have a simple NN where we only consider linear functional forms - this is linear so will fail to express any non-linearity in our problems. So this architecture, we say, is not **expressive** enough. We therefore have a third intricate pillar that ties in with statistics and dynamics of NNs - **Expressivity**.

What does a typical neural network architecture look like? The most common (and arguably, the simplest) implementation of NNs are called **feed-forward neural networks**. This consists of a set of layers, each of which consists of a set of nodes, whose number typically varies between the layers. Feed-forward NNs are directed - the information is passed on from the **input layer** to the **output layer** via intermediate **hidden layers**, known as such as the states in such layers are hidden. The number of layers L is the **depth**, and the number of nodes per layer $N^{(i)}$ is the **width** - a NN with a $L \gg 1$ is called a deep NN, or a **multi-layer perceptron (MLP)**, or a **deep feedforward network**. The output of each layer is often set to have the following form,

$$l_\mu^{(i)} = \sigma^{(i)} \left(\sum_{\nu=1}^{n_{i-1}} w_{\mu\nu}^{(i)} l_\nu^{(i-1)} + b_\nu^{(i)} \right) \tag{1.10}$$

where $l_\nu^{(i-1)}$ are the output of the $(i-1)$th layer, the matrix entries $w_{\mu\nu}^{(i)}$ are known as **weights**, and the vector $b_\nu^{(i)}$ **bias**. The **activation function** $\sigma^{(i)}$ is non-linear function which introduces non-linearity to the NN. This is often picked depending on the purpose of the NNs (as some activation functions are shown to help with convergence in particular problems). The typical activation functions include the identity, log-sigmoid, tanh or leaky ReLU (see for example [1]). The above then defines an example of a typical NN architecture.

To summarise, we have the following three questions we want to know from neural networks.

1. **Expressivity**. How powerful is NN?
2. **Statistics**. How are NN's initialised and what is the NN statistical ensemble?
3. **Dynamics**. How do NNs evolve in training?

The centre of this set of lectures is to explain how it is possible to understand these questions from a theoretical physicist's point of view, utilising the tools we have on our hand:

- Field Theory.
- Landscape Dynamics from loss functions of the parameter space.
- Symmetries of individual NNS and their ensembles.

As theoretical physicists, such a new perspective is not just helpful for us to understand how NNs work, but also to understand empirical results in machine learning where these results are barely explored. We are not thriving for a rigorous proof to all the results, but to first develop some mechanisms and toy models so we can later add in rigour. That is the main goal of the programme of **Physics4ML**.

2 The three pillars of neural networks

Let us now try and understand the three pillars of neural networks! A quick recap of the three pillars of the neural networks:

1. **Expressivity**. How powerful is NN?
2. **Statistics**. How are NN's initialised and what is the NN statistical ensemble?
3. **Dynamics**. How do NNs evolve in training?

Let us try and understand these one-by-one. This ties in closely with the recent results currently being developed in the computer science community. The expressivity of NNs is first explored by Cybenko using the Universal Approximation Theorem (UAT), where as the statistics and dynamics of NNs are recently covered by the Neural Network Gaussian Processes (NNGP) and Neural Tangent Kernel (NTK) programme. In this section, we will understand these results in these directions, with the help of physical intuition from physics.

2.1 Expressivity of NNs

We have alluded above in §1 that neural networks are big functions composed out of many simple functions given a choice of the architecture. A different architecture would give a different composition of simple functions and hence a different expression of the neural network. The question then becomes a mathematical one - how good is NN at approximating an unknown function?

2.1.1 Univeral Approximation Theorem

The Universal Approximation Theorem, UAT for short, is the first result in showing how good are NNs at approximating unknown functions. It states that a NN with a single hidden layer can approximate any continuous function on a compact domain to arbitrary accuracy. The precise form of the theorem is stated by Cybenko:

Theorem 2.1 (Cybenko)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuous function on a compact set $K \subset \mathbb{R}^d$. Then for any positive number $\epsilon > 0$ there exists a neural network with a single hidden layer of the form,*

$$\phi(x) = \sum_{i=1}^{N} \sum_{j=1}^{d} w_i^{(1)} \sigma(w_{ij}^{(0)} x_j + b_i^{(0)}) + b^{(1)} \,, \tag{2.1}$$

where the parameters are $\theta = \left\{ w_{ij}^{(0)}, w_i^{(1)}, b_i^{(0)}, b^{(1)} \right\}$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-polynomial nonlinear function such that

$$\sup_{x \in K} |f(x) - \phi(x)| < \epsilon \,. \tag{2.2}$$

Proof. See [2]. The idea of the proof is the following. In the original work, Cybenko focused on the case where σ is the sigmoid function,

$$\sigma(x) = \frac{1}{1 + e^{-x}} \,, \tag{2.3}$$

which appear in the network function in the following form,

$$\sigma(w_i^{(0)} \cdot x + b_i^{(0)}) \tag{2.4}$$

and this approximates a shifted step function as $w^{(0)} \to \infty$. So scaling this with $w^{(1)}$, this effectively gives a bump - which can be put together to approximate any function (which is very similar to the Riemann integral). □

Note that the UAT is a powerful result, but it has some big limitations. In particular, firstly, although the error ϵ improves with N the precise accuracy bound is unknown. Secondly, it tells us there is a point θ^* in parameter space that is a good approximation to any f but it doesn't tell you how to get there - we still need to answer the questions about learning dynamics.

However, this is still a first result. In particular, there are many generations of the theorem to other activations, deep NNs, and other domains.

2.1.2 Kolmogorov-Arnold Theorem

Are there other architectures that we can construct - and are there any related theorems that support such construction, if it exists? One such construction is due to Kolmogorov and Arnold - any multivariate continuous function can be represented exactly as a sum of continuous one-dimensional functions.

Theorem 2.2 (Kolmogorov-Arnold Representation Theorem)**.** *Let $f : [0,1]^n \to \mathbb{R}$ be an arbitrary multivariate continuous function. Then it has the representation*

$$f(x_1, ..., x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{2.5}$$

where Φ_q and $\phi_{q,p}$ are continuous one-dimensional functions.

A non-trivial example of the KAT is the function $f(x, y) = xy$. Since,

$$f(x, y) = xy = e^{\log x + \log y} , \tag{2.6}$$

so the theorem works for multiplications. Since this is a sum of functions, one can try and build a neural network with this theorem. However, note that $\phi_{p,q}$ lives on the connection between x_p and $x_q^{(1)} = \sum_{p=1}^{n} \phi_{q,p}(x_p)$. Before, when we had the feedforward network, we can think of the linear transformations (the weights and biases) as acting on the edges and the activation function acting on the nodes of each of the layers. Here, instead the activation functions $\phi_{q,p}$ act on the edges instead of the nodes. For distinct values of p, q the functions $\phi_{q,p}$ are independent in general as well. Recently, this architecture is explored in the proposal of **Kolmogorov-Arnold networks** (KAN) [3]. This has the advantage of interpretable architectures which can be mapped directly into a symbolic representation, improving the interpretation of the neural network.

2.2 Statistics of NNs

Now we move on to understanding what happens to the neural network at initialisation. In particular, we are interested in the statistical properties of NNs. We want to in particular answer the following question:

Question: What characterises the statistics of the NN ensemble?

One particular aspect of this is encoded in the moments or n-point functions, obtained from the partition functions as the following (using field theory notions):

$$Z[J] = \left\langle e^{\int d^d x J(x)\phi(x)} \right\rangle , \tag{2.7}$$

$$G^{(n)}(x_1, ..., x_n) = \left(\frac{\delta}{\delta J(x_1)} ... \frac{\delta}{\delta J(x_n)} Z[J] \right) \Big|_{J=0} = \langle \phi(x_1) ... \phi(x_n) \rangle , \tag{2.8}$$

where, using usual field theory notations, $J(x)$ would be the source. This expectation, as we have discussed before, is actually here averaged across different distributions $P(\theta)$ of the parameters θ, giving the partition function as

$$Z[J] = \int d\theta P(\theta) e^{\int d^d x J(x)\phi(x)} . \tag{2.9}$$

However, we are more used to functional (Feynman path) integrals of the form

$$Z[J] = \int \mathcal{D}\phi e^{-S[\phi]} e^{\int d^d x J(x)\phi(x)} \ . \tag{2.10}$$

where the action $S[\phi]$ defines a density on functions. We call the first the **parameter space partition function**. The question now is:

Question: What is the action $S[\phi]$ associated to a distribution $(\phi, P(\theta))$?

To answer this, we actually must first take a step back and understand something known as the Neural Network-Gaussian Process (NNGP) correspondence. From this, we will see that the parameter-space and function-space descriptions can be thought of as a duality effectively.

2.2.1 NNGP correspondence

In this section the aim is to discuss the correspondence between Neural Networks and Gaussian Processes. This is first pointed out by Neal [4] in his thesis, where he had the following simple observation of the a single-layer fully connected network of width N:

$$\phi(x) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \sum_{j=1}^{d} w_i^{(1)} \sigma(w_{ij}^{(0)} x_j) \tag{2.11}$$

where for simplicity I have set the biases $b_j^{(0)}$ to zero. The set of network parameters $\theta = \left\{ w_{ij}^{(0)}, w_i^{(1)} \right\}$ is independently and identically distributed. We can now use the functional form of the Central Limit Theorem (CLT) where since $\phi(x)$ is the sum of random functions that are independently and identically distributed, in the limit where the network parameters $N \to \infty$ we then have that $\phi(x)$ is a Gaussian. Let us understand these terms in more detail.

Gaussian Process. A time continuous stochastic process with variables $\{X_t\}_{t \in T}$ is a Gaussian process if and only if for every finite set of indices in the index set T, $(X_{t_1}, ..., X_{t_k})$ is a multivariate Gaussian random variable. Explicitly, recall the Bayesian approach to model physical theories, where we try and use a nonlinear function $f(x)$ parameterised by the parameters w_a to model the data (x_i, y_j). By Bayes' theorem, the inference of the function $f(x)$ is described by

$$P\left(f(x)|x_i, y_j\right) = \frac{P\left(y_j|f(x), x_i\right) P(f(x))}{P\left(y_j|x_i\right)} \ . \tag{2.12}$$

The idea of Gaussian process modelling is to place a prior $P(f(x))$ directly on the space of functions, without parameterising x. Loosely, Gaussian process is the simplest type of prior over functions where it is a generalisation of a Gaussian distribution over a finite vector space to a function space of infinite space. A Gaussian process is specified by a **mean** function $\mu(x)$ and **covariance matrix** $C(x, x')$, also known as a **kernel**. Being even a bit more mathematically imprecise, we can write a Gaussian process as the following

probability distribution on the space of functions $f(x)$:

$$P\left(y(x)|\mu(x),A\right)=\frac{1}{Z}\exp\left[-\frac{1}{2}(y(x)-\mu(x))^{T}A(y(x)-\mu(x))\right]\ , \tag{2.13}$$

where $\mu(x)$ is the mean function and A is a linear operator and the inner product of two functions is defined by

$$y(x)^{T}z(x)=\int dxy(x)z(x)\ . \tag{2.14}$$

We can in fact write

$$\log P\left(y(x)|\alpha\right)=-\frac{1}{2}y(x)^{T}Ay(x)+\textbf{const}\ , \tag{2.15}$$

where $A=[D^{p}]^{T}D^{p}$ and D is the linear derivative operator.

NNGP Correspondence. Now that we know what is a Gaussian process is, we will now give the NNGP correspondence. This is as follows. Given a NN of single-layer, fully-connected of width N, in the limit $N\to\infty$, we have a Gaussian process:

$$\lim_{N\to\infty}\phi(x)\sim\mathcal{N}(\mu(x),K(x,y))\ , \tag{2.16}$$

with mean and covariance $\mu(x)$ and $K(x,y)$.

Explanation. The NNGP Correspondence can be thought of as the following. For many architectures, for example,

- Single layer fully-connected NN: $N=$ width
- Deep fully connected network: $N=$ width
- Convolutional NN: $N=$ channels
- Attention networks: $N=$ heads

where the limit $N\to\infty$ exists, the distribution on the functions becomes a Gaussian process. In particular, the NN outputs $\{f(x_1),...,f(x_k)\}$ evaluated on any fixed set of k inputs x_i are drawn from a multivariate Gaussian distribution $\mathcal{N}(\mu,\Xi^{-1})$,

$$\{f(x_1),...,f(x_k)\}\sim\mathcal{N}(\mu,\Xi^{-1})\ . \tag{2.17}$$

The inverse covariance matrix Ξ is determined by the kernel function $K(x,x')$. If $\mu=0$, then the GP is entirely determined by the covariance, so it is determined entirely by the kernel. To compute correlation functions, we can see that between n outputs this is expressed as,

$$G^{(n)}(x_1,...,x_n)=\frac{\int df\,f_1...f_ne^{-\frac{1}{2}f_i\Xi_{ij}f_j}}{Z} \tag{2.18}$$

with the partition function being $Z=\int df e^{-S}$ and $S=-\frac{1}{2}f_i\Xi_{ij}f_j$ being the **log-likelihood** (c.f. the action). We can of course take the continuum limit to get

$$G^{(n)}(x_1,...,x_n)=\frac{\int df\,f_1...f_ne^{-S}}{Z} \tag{2.19}$$

with the log-likelihood as (d is dimension of input space)

$$S = \frac{1}{2}\int d^d x d^d x' f(x)\Xi(x,x')f(x') \tag{2.20}$$

and $\Xi(x,x') = K^{-1}(x,x')$ being the inverse covariance function defined using the D-dimensional Dirac delta function:

$$\int d^d x' K(x,x')\Xi(x',x'') = \delta^{(d)}(x-x'') \ . \tag{2.21}$$

Let us see this explicitly worked out for the two-point correlator. We see that

$$G^{(2)}(x,y) = \frac{\Xi_2}{N}\left\langle \sigma(w_{ij}^{(0)}x_j)\sigma(w_{il}^{(0)}x_l)\right\rangle = \Xi_2\left\langle \sigma(w_{ij}^{(0)}x_j)\sigma(w_{il}^{(0)}x_l)\right\rangle \tag{2.22}$$

where we have evaluated the sum over i in the last step (so there is no summation over i in the final expression). We can of course evaluate the quantity inside the bracket to get the answers (we can always numerically obtain an answer by, for example, a Monte Carlo estimate [5]).

Relation to Field Theory. So how does all of this relate to physics? Note that the correlators are defined exactly like the ones we have in field theories in physics, apart from the fact that in physics the S is the action that is defined at the start. If we combine the parameter distribution $\theta \sim P(\theta)$ and the network architecture together this induce an implicit distribution on the function space from which the neural network is drawn from, say $\phi \sim P(\phi)$. What that means is the following. Since the NNGP Correspondence tells us that $\exp(-S[\phi])$ is Gaussian in the $N \to \infty$ limit this means that $S[\phi]$ is quadratic in the networks, and in particular we can define the associated action using the two-point correlator as

$$S[\phi] = \int d^D x d^D y \phi(x) G^{(2)}(x,y)^{-1}\phi(y) \ , \tag{2.23}$$

and the NNGP correspondence gives

$$\lim_{N\to\infty}\phi(x) \sim \mathcal{N}\left(0, G^{(2)}(x,y)\right) \ . \tag{2.24}$$

So in the $N \to \infty$ limit, certain large NNs are just functions drawn from generalised free-field theories. An example, of course, is the free scalar field theory given by

$$Z = \int D\phi e^{-S[\phi]} \tag{2.25}$$

where the action S is defined as

$$S[\phi] = \int d^d x \phi(x)(\Box + m^2)\phi(x) \tag{2.26}$$

2.2.2 Non-Gaussian processes

It is obvious that the $N \to \infty$ limit, in the most practical sense, does not hold. Violating any of the assumptions of the CLT should introduce non-Gaussian contributions, i.e. in the field theory language, interactions. In particular, the CLT is violated by

- $\frac{1}{N}$-corrections.
- Independence breaking - in the CLT proof the cumulant generating function is assumed to be the sum of all cumulant generating functions which ignores cross-correlations.

So we, in general, expect non-Gaussianities. Non-Gaussianities can be computed via the connected four-point function and using effective field theory. In particular perturbative diagrammatic methods have been used to study NNs at initialisation. Let us in the following highlight at least one of these points - the connected four-point function. The full derivation is in [6], but the result is, given an architecture,

$$\phi(x) = \sum_i w_i \varphi_i(x) \,, \tag{2.27}$$

$$G_c^{(4)}(x,y,z,w) = \frac{1}{N}\left[\Xi_4 \left\langle \varphi_i(x)\varphi_i(y)\varphi_i(z)\varphi_i(w)\right\rangle - \Xi_2^2 \left(\left\langle \varphi_i(x)\varphi_i(y)\right\rangle \left\langle \varphi_i(z)\varphi_i(w)\right\rangle + \text{ perms }\right)\right] \,. \tag{2.28}$$

Note that this is non-zero at finite-N - so there are interactions.

When non-Gaussianities are small, the theory will be close to free theory and be weakly interacting, in which case we can proceed to calculate the correlation functions using Feynman diagrams. The higher connected correlation functions which vanish in the Gaussian limit now capture non-Gaussianities - they are known as **cumulants** and can be obtained from generating functions $W[J]$ as

$$G_c^{(n)}(x_1,...,x_n) := \left(\frac{\delta}{\delta J(x_1)}...\frac{\delta}{\delta J(x_n)} W[J]\right)\Big|_{J=0} \tag{2.29}$$

For a theory with a known Lagrangian description, the correlators $G_c^{(n)} \neq 0$ for $n > 2$ still encode the presence of non-Gaussianities. For NN-FTs, since we know the parameter space description always exists, we can study non-Gaussianities using connected correlators. This allows us to:

- NN $\to$ FT: understand interactions in associated field theories.
- FT $\to$ NN: capture the statistics of finite networks and networks with correlations in the parameter descriptions which generically develop during training.

We will come back to this when we discuss the details of the NN-FT correspondence in §3.

2.2.3 Symmetries - a first encounter

Now we can ask the question.

Question: Are there any structures in the ensemble?

One type of structure we can see here at level of initialisation is the symmetry of the architecture. In particular, recall that the networks $\phi : \mathbb{R}^d \to \mathbb{R}^D$ are functions. It is perhaps possible to introduce some sort of symmetry on the NN architecture such that the

family of NNs generated will be invariant under such symmetry. Or that the symmetry can be realised by the function. To do this, we need to use the notion of equivariance in the mathematical literature.

Equivariant NNs. We say that ϕ is G**-equivariant** with respect to a group G if

$$\rho_D(g)\phi(x) = \phi(\rho_d(g)x) \ , \tag{2.30}$$

for all $g \in G$ where $\rho_d \in \text{Mat}(\mathbb{R}^d)$ and $\rho_D \in \text{Mat}(\mathbb{R}^D)$ are matrix reps of G in $\mathbb{R}^D$ and $\mathbb{R}^d$ respectively. The network is then **invariant** if $\rho_D = \mathbb{1}$, i.e. if the rep is trivial.

Global symmetries in NN ensembles. Now let us look at the symmetries that arise in ensembles of NNs. In particular, we want to leave the statistical ensemble of NNs invariant. This is equivalent to the notion of **global symmetries** in field theory. Suppose the network transform under a group action as

$$\phi \mapsto \phi_g, \quad g \in G \ . \tag{2.31}$$

The ensemble of networks then has a global symmetry group G if the partition function is invariant,

$$Z_g[J] = Z[J] \ , \forall g \in G \ , \tag{2.32}$$

or we can write this as

$$\left\langle \exp\left(\int d^d x J(x)\phi_g(x) \right) \right\rangle = \left\langle \exp\left(\int d^d x J(x)\phi(x) \right) \right\rangle , \forall g \in G \ . \tag{2.33}$$

We can see that we can redefine the network and put the symmetry if $\langle\cdot\rangle$ is invariant. In terms of FT, this means that the action $S[\phi]$ and the measure $\mathcal{D}\phi$ are both invariant. In particular, it is possible to redefine the network by transforming the parameters, as follows:

$$\int d\theta_g P(\theta_g) \left\langle \exp\left(\int d^d x J(x)\phi_{\theta_g}(x) \right) \right\rangle = \int d\theta P(\theta) \left\langle \exp\left(\int d^d x J(x)\phi_{\theta}(x) \right) \right\rangle \ , \tag{2.34}$$

so the symmetry arises with the redefinition $\theta \mapsto \theta_g$ when the parameter density and measure must be invariant.

It is perhaps important to note that the formalism above allows for transforming the inputs and outputs of the network - which corresponds to internal and spacetime symmetries of the field theory. The notion of equivariance plays a central role here - the ensemble of equivariant NNs is invariant under ρ_d on the input if the partition function is invariant under the induced ρ_D on the output.

Example 2.1. Let us perhaps look at an example with the NN architecture being of the form:

$$\phi(x) = \sum_i w_i \varphi_i(x) \tag{2.35}$$

with w_i being $P(w)$ even. The $\mathbb{Z}_2$-action $\phi \mapsto -\phi$ then may be absorbed into the parameters $w_g = -w_i$ with $dwP(w)$ invariant by the evenness, so these NNs only have even correlators.

There are many more examples in [6].

2.3 Dynamics of NNs

Now we move on to the next aspect of NNs. What happens to NNs during training? As explained §1, it is clear that NNs evolve in some way under gradient descent. This is often simplified in a scheme known as Neural Tangent Kernel (NTK) where we will then use to solve a model exactly in the case of MSE loss. In particular, we will focus on the problem of dynamics of supervised learning with gradient descent, with data

$$\mathcal{D} = \{(x_\alpha, y_\alpha)\}_{\alpha=1}^{|\mathcal{D}|} \ , \tag{2.36}$$

and the loss function

$$\mathcal{L}[\phi] = \frac{1}{|\mathcal{D}|} \sum_{\alpha=1}^{|\mathcal{D}|} l\left(\phi(x_\alpha), y_\alpha\right) \ . \tag{2.37}$$

We optimise the network parameters θ by a gradient descent procedure,

$$\frac{d\theta_i}{dt} = -\eta \nabla_{\theta_i} \mathcal{L}[\phi] \ , \tag{2.38}$$

where let us define the natural object of gradient descent in function space as

$$\Delta(x) = -\frac{\delta l(\phi(x), y)}{\delta \phi(x)} \tag{2.39}$$

which kind of gives the gradient of the loss in space of learned outputs. This yields,

$$\frac{d\theta_i}{dt} = \frac{\eta}{|\mathcal{D}|} \sum_{\alpha=1}^{|\mathcal{D}|} \Delta(x_\alpha) \frac{\partial \phi(x_\alpha)}{\partial \theta_i} \ . \tag{2.40}$$

Could we possibly use this equation to derive a quantity to simplify our analysis?

2.3.1 Neural tangent kernel

Let us define a quantity in ML called the **Neural Tangent Kernel (NTK)**. By training the network by gradient descent and using Eq.(2.40), we have,

$$\frac{d\phi(x)}{dt} = \frac{\partial \phi(x)}{\partial \theta_i} \frac{d\theta_i}{dt} = \frac{\eta}{|\mathcal{D}|} \sum_{\alpha=1}^{|\mathcal{D}|} \Delta(x_\alpha) \Theta(x, x_\alpha) \ , \tag{2.41}$$

where the NTK is

$$\Theta(x, x_\alpha) = \frac{\partial \phi(x)}{\partial \theta_i} \frac{\partial \phi(x_\alpha)}{\partial \theta_i} \ . \tag{2.42}$$

Recall what a kernel is. We can define a kernel as a map $V \times V \to \mathrm{Sym}(V \otimes V)$ where some of the dimensions can be suppressed. This maps the pair (x, x') to an $n_L \times n_L$ symmetric matrix $K(x, x')$. The dual space of V is spanned by the set of linear forms $\mu : V \to \mathbb{R}$ where $\mu = \langle v, \cdot \rangle$. So we can define a map $\Phi_K : V^* \to V$ as

$$f_\mu(\cdot) = \Phi_K(\mu(x)) = \mu K(x, \cdot) = \langle d, K(x, \cdot) \rangle \ . \tag{2.43}$$

where $\cdot$ is now the variable. Now defining the **cost function** C as a functional $C : V \to R$ and therefore the **kernel gradient** which is defined as

$$\nabla_K C\big|_{f_0} = \Phi_K \left(\partial_f C\big|_{f_0}\right) \ , \tag{2.44}$$

which we can write,

$$\nabla_K C\big|_{f_0}(x) = \frac{1}{N}\sum_{j=1}^{N} K(x, x_j) d\big|_{f_0}(x_j) \tag{2.45}$$

and we say a time-dependent function $f(t)$ follows the **kernel gradient descent w.r.t** K if it satisfies,

$$\partial_t f(t) = -\nabla_K C\big|_{f(t)} \ . \tag{2.46}$$

The NTK is a fundamental object. But it is physically terrible to work with. Why?

- Parameter-dependence. We need to sum over bi-jillions of parameters.
- Time-dependent. The NTK time evolves.
- Stochastic. The NTK inherits the statistical uncertainty at initialisation.
- Non-local. It communicates information between x and x_α (in particular, the correlation between the gradients of the function $\phi(x)$ and $\phi(x_\alpha)$).

All this makes NTK unwieldy. But there is a fix - the NTK simplifies in the $N \to \infty$ limit, where we call the lazy regime $|\theta(t) - \theta(0)| \ll 1$ as the number of parameters is large but the evolution keeps them in a local neighbourhood. The NN in this case is approximately a linear-in-parameters model where

$$\lim_{N\to\infty} \phi(x) \cong \phi_{\theta_0}(x) + (\theta - \theta_0)_i \left.\frac{\partial \phi(x)}{\partial \theta_i}\right|_{\theta_0} \ , \tag{2.47}$$

so then

$$\lim_{N\to\infty} \Theta(x, x') \cong \Theta(x, x')\big|_{\theta_0} \cong \beta_\theta(x, x') \ , \tag{2.48}$$

where we have approximated the last term by an expectation value and hence the network dynamics are now governed by the frozen NTK $\overline{\Theta}$,

$$\frac{d\phi(x)}{dt} = -\frac{\eta}{|\mathcal{D}|}\sum_{\alpha=1}^{|\mathcal{D}|} \frac{\delta l(\phi(x_\alpha), y_\alpha)}{\delta \phi(x_\alpha)} \overline{\Theta}(x, x_\alpha) \ . \tag{2.49}$$

where I have defined the frozen NTK as

$$\overline{\Theta}(x, x') = \beta_\theta(x, x') \ . \tag{2.50}$$

This is not good though. What Eq. (2.49) means for evolution is this - since $\overline{\Theta}$ is a fixed function that can be calculated exactly (it is deterministic), the NN effectively has zero parameters. So the NN actually does not learn any features in the hidden dimensions and therefore there is no non-trivial evolution that would cause the NTK to evolve. So we

have, effectively, learnt nothing. An exact solvable model for frozen-NTK dynamics, in particular, can be found in [6] with MSE loss, where the converged network is

$$\phi_\infty(x) = \phi_0(x) + \overline{\Theta}(x, x_\alpha)\overline{\Theta}(x_\alpha, x_\beta)^{-1}(y_\beta - \phi_0(x_\beta)) \,, \tag{2.51}$$

which is known as kernel regression.

However, all is not lost. In particular, there are two key results that we can infer from here.

1. The NTK, in the limit $N \to \infty$, converges to an explicit deterministic limit. The existence of such is merely shown but not proven by the example we have above (see [7]), but this explicitly connects NTK with NNGP.

2. The NTK stays asymptotically constant during training in the same limit. This means that the frozen NTK behaviour is in fact universal in the limit, and as the intermediate layer pre-activations $z_\alpha^{(l)}$ are also Gaussian distributed, the statistics of a randomly initialised NN is described by a sequence of generalised free field theories where correlations are propagated down the network according to a recursion relation.

We see that the dynamics of this family of NNs (in this $N \to \infty$ limit) is consistent with the behaviour of free field theories. The initialisation is akin to the initial set-up, and the evolution of free theories are deterministic and described by a kernel. The question now becomes, what happens when we move away from the limit?

2.3.2 Feature learning

To extend to a general study of learning dynamics of NNs, we need to specifically engineer tractable properties so we can do wise N-scaling/expansions to understand the behaviour. In particular let us start with three properties:

1. Finite initialisation pre-activations. This means that the pre-activations $z^{(l)}(x_\alpha)$ in each layer, the inputs to the activation functions, will be set to $O_N(1)$.

2. Learning in finite time. We want to train the NN in a reasonable time-frame, $\frac{d\phi(x)}{dt} \sim O_N(1)$.

3. Feature learning in finite time. This is the same as above but we want to explicitly learn the features of the system, i.e. on the pre-activations $\frac{dz^{(l)}}{dt} \sim O_N(1)$.

From [6] and its cited works, the key point is that the constraints listed above have a one-parameter family of solutions which is completely fixed when the learning rate is additionally assumed. In particular, it is possible to undergo a detailed N-scaling analysis to see how we can lead to richer learning regimes known as dynamical mean field theory or the maximal update parametrisation.

3 NN-FT correspondence

Neural networks, as discussed above, generally have $\mathbb{R}^n$ as their domain and as a result live in a space with Euclidean signature. This means they define statistical field theories (in the first instance) that may or may not have analytic continuations to quantum field theories in the Lorentzian signature. But it is clear that there is a direct correspondence between NNs and field theories [8].

The question that us theoretical physicist are most concerned about, however, is when neural architecture defines a quantum field theory (QFT), i.e. does there exist an analytic continuation to Lorentzian signature that defines a QFT. The key is the Osterwalder-Schrader (OS) theorem of axiomatic field theory which gives the necessary and sufficient conditions, expressed in terms of the correlators, for the existence of a QFT after continuation. The axioms include the following:

- **Euclidean Invariance.** Correlation functions must be invariant in Euclidean signature. This ensures it is Lorentz-invariant after analytic continuation.

- **Permutation Symmetry.** Correlation functions must be invariant under permutation of their arguments (collection of points in Euclidean space). Of course, this is automatic with scalar outputs.

- **Reflection Positivity.** Correlation functions must satisfy a positivity condition known as reflection positivity. This is necessary for unitarity and the absence of negative-norm states in the analytically-continued theory. For the Gaussian case, this simply requires the two-point function to be reflection positive, which means requiring
$$\int d^d x d^d y f^*(x) f(y) G^{(2)}(x^\theta, y) \geq 0 \, . \tag{3.1}$$
for any complex function $f(x)$ with support only on $\tau > 0$ and $x^\theta = (-\tau_x, \mathbf{x})$ being the reflection of x in imaginary time.

- **Cluster Decomposition.** Correlation functions must satisfy cluster decomposition, which states that interactions must shut off at infinite distance. For connected correlators, this imposes,
$$\lim_{b\to\infty} G_c^{(n)}(x_1, ..., x_p, x_{p+1} + b, ..., x_n + b) \to 0 \, , \tag{3.2}$$
for any value of $1 < p < n$.

In particular, our field theory only consists of fields and correlation functions at this stage, and is, really, an ensemble of functions with a way to compute their correlators. In the Euclidean picture, this is, additionally, a statistical ensemble of functions. We can already construct the partition function,
$$Z[J] = \left\langle \exp\left(\int d^d x J(x) \phi(x) \right) \right\rangle \, , \tag{3.3}$$

where we can use to compute correlators. The NN-GP correspondence now gives us the following correspondence of partition functions:

$$\int \mathcal{D}\phi e^{-S[\phi]+\int d^d x J(x)\phi(x)} \longleftrightarrow \int d\theta P(\theta) e^{\int d^d x J(x)\phi(x)} , \tag{3.4}$$

where we can now try and work out the associated action given $(\phi_\theta, P(\theta))$. Now I hear you complain - well, surely, we need more things! The list of things X may include,

- Quantumness
- Lagrangian Description
- Symmetries
- Locality

But these are all add-ons remember. Different physicists might argue that different things are important, and will hence (basically) require a different NN architecture and statistical ensemble to make FT + X work. But FT is enough here. For now.

So now we know that there is an NN-FT correspondence, and there are additional things that different physicists might want in NNs to add extra spice to their field theories. Fine. We have constructed free field theories above, so the question is how could we possibly (perhaps the easiest way) get non-Gaussianities? Turns out it is possible to insert an operator to any local potential $V(\phi)$ which deforms the action in the following way:

$$Z[J] = \int d\theta P(\theta) e^{\int d^d x V(\phi_\theta(x))} e^{\int d^d x J(x)\phi_\theta(x)} \tag{3.5}$$

$$= \int d\theta \tilde{P}(\theta) e^{\int d^d x J(x)\phi_\theta(x)} \tag{3.6}$$

where we have used the architecture equation to define a new parameter density in the second line. The interactions in $V(\phi)$ will now break the Gaussianity of the NNGP as the statistical independence is now violated - giving the $Z[J]$ an interacting NN-QFT structure!

In this section, we will approach the NN-FT correspondence in the following different steps.

1. **Parametrically breaking Gaussianities.** We first analyse the interactions in NN-FT by parametrically breaking the necessary conditions for the central limit theorem to hold.

2. **EFTs and Feynman diagrams.** We then develop a general field theory technique for computing the action diagrammatically, through perturbation theory. This involves using a new type of Feynman diagrams. We also take the perspective of EFTs when analysing these non-Gaussianities.

3. **Designing NN architectures.** We then go backwards, instead develop techniques to design architectures that realise a particular type of action. This is done by deforming the parameter distribution of NN.

We will then provide a few examples and analyse some field theory features of the NN-FT correspondence. The discussion in this section is mainly based on [8], with inputs also taken from [9, 10].

3.1 Parametrically breaking Gaussianities

Understanding non-Gaussianities requires understanding the essential aspects of the Central Limit Theorem. The Central Limit Theorem (CLT) is one of the most fundamental theorems in statistics. The essentials of the theorem is stated as follows. Consider N random variables X_i, being identical, independent, mean-free and having finite-variance. Then the standardised sum,

$$\phi = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} X_i \tag{3.7}$$

is drawn from a Gaussian distribution in the limit $N \to \infty$. The moment generating function of ϕ is defined as

$$Z_\phi[J] = \mathbb{E}\left[e^{J\phi}\right] = \mathbb{E}\left[e^{J\sum_i X_i/\sqrt{N}}\right] , \tag{3.8}$$

where we extract the moments by taking derivatives

$$\mu_r^\phi := \mathbb{E}\left[\phi^r\right] = \left(\frac{\partial}{\partial J}\right)^r Z_\phi[J] . \tag{3.9}$$

Of course, in physics language, J is the source, $Z_\phi[J]$ is the partition function, and μ_r^ϕ is the r-th correlator of ϕ. The cumulant generating functional (CGF) of ϕ is the logarithm of the moment generating functional,

$$W_\phi[J] = \log Z_\phi[J] \tag{3.10}$$

and the cumulants are computed by taking derivatives,

$$\kappa_r^\phi = \left(\frac{\partial}{\partial J}\right)^r W_\phi[J] . \tag{3.11}$$

A random variable is Gaussian only if $\kappa_{r>2}^\phi$ vanish. Note the following properties of CGFs,

$$W_{X+c}[J] = cJ + W_X[J] , \tag{3.12}$$

$$W_{cX}[J] = \log \mathbb{E}[e^{Jcx}] = W_X[cJ] , \tag{3.13}$$

where $c \in \mathbb{R}$ is a constant, implying,

$$\kappa_a^{X+c} = \kappa_1^X + c , \tag{3.14}$$

$$\kappa_{r>1}^{X+c} = \kappa_{r>1}^X , \tag{3.15}$$

$$\kappa_r^{cX} = c^r \kappa_r^X . \tag{3.16}$$

For independent variables the moment generating factorises, so the CGF and cumulants can be written as a sum:

$$Z_{\sum_i X_i}[J] = \prod_i^N Z_{X_i}[J] \Rightarrow W_{\sum_i}[J] = \sum_i W_{X_i}[J] \,, \tag{3.17}$$

$$\kappa_r^{\sum_i X_i} = \sum_i \kappa_r^{X_i} \,. \tag{3.18}$$

Therefore we can write

$$\kappa_r^{\phi} = \frac{\sum_i \kappa_r^{X_i}}{\sqrt{N^r}} \,, \tag{3.19}$$

therefore where the X_i are identical, we will have

$$\kappa_r^{\phi} = \frac{\kappa_r^{X_i}}{N^{r/2-1}} \tag{3.20}$$

hence the cumulants $\kappa_{r>2}^{\phi}$ vanish in the $N \to \infty$ limit.

How do we analyse the emergence of non-Gaussianities in the system? We will need to parametrise the breaking of statistical independence. Let $p(X;\alpha)$ be a family of joint distributions on X_i, parametrised by a hyperparameter α. Choose the family of joint distribution to be of the form,

$$p(X;\alpha=0) = \prod_i p(X_i) \,, \tag{3.21}$$

such that $p(X)$ is independent in the $\alpha \to 0$ limit. The hyperparameter α therefore controls the breaking of independence. We now write,

$$W_\phi[J] = \log \int \prod_j dX_j p(X;\alpha) e^{J\sum_i X_i/\sqrt{N}} \,, \tag{3.22}$$

so expanding around $\alpha = 0$ gives,

$$W_\phi[J] = \log \left[\prod_j \mathbb{E}_{p(X,\alpha=0)}[e^{JX_j/\sqrt{N}}] + \sum_{k=1}^{\infty} \frac{\alpha^k}{k!} \int \prod_j dX_j e^{J\sum_i X_i/\sqrt{N}} \partial_\alpha^k p(X;\alpha)|_{\alpha=0} \right] \,, \tag{3.23}$$

where the first term have used the independence of $p(X;\alpha=0)$. We now utilise the trick $p\partial_\alpha \log p = \partial_\alpha p$, such that $\partial_\alpha \mathbb{E}[\mathcal{O}] = \mathbb{E}[\mathcal{O}\partial_\alpha \log p]$. So defining $\mathcal{P}_k = \frac{1}{p}\partial_\alpha^k p$, we can now write Eq. (3.23) as,

$$W_\phi[J] = \log \left[\prod_j \mathbb{E}_{p(X,\alpha=0)}[e^{JX_j/\sqrt{N}}] + \sum_{k=1}^{\infty} \frac{\alpha^k}{k!} \mathbb{E}_{p(X,\alpha=0)} \left[e^{J\sum_i X_i/\sqrt{N}} \mathcal{P}_k|_{\alpha=0} \right] \right] \,. \tag{3.24}$$

Of course, in the limit $\alpha \to 0$, the X_j become independent and we recover the same result as before. So we have seen that non-Gaussianities can be generated by both $\frac{1}{N}$-corrections and independence breaking, but the latter requires more caution as that also depends on

N and the hyperparameter α. For example, if the leading corrections to κ_r^ϕ scale as αN^{a_r} where $a_r < 0$ for all $r > 2$, then ϕ will be Gaussian regardless of independence breaking. An example where $a_r = 0$ for all r and the non-Gaussianities are generated by independence breaking alone can be found in [8].

The next step is to generalise this to continuum NN-FT, i.e. we want to conceptually pass this from a single random variable ϕ (0d FT) to first a discrete number of random variable ϕ_i (LFT) and then finally to a continuous number of random variables $\phi(x)$ (FT). Consider the case where $\phi_i(x)$ is built out of neurones $h_i(x)$ as

$$\phi(x) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} h_i(x) \ . \tag{3.25}$$

If $h_i(x)$ are independent, then of course we have $\phi(x)$ being a Gaussian process in the limit $N \to \infty$. It is now useful to study the non-Gaussianities arising from finite-N corrections and breaking of the independence condition.The cumulant generating functional of $\phi(x)$ is

$$W_\phi[J] = \log Z_\phi[J] = \sum_{r=1}^{\infty} \int \prod_{i=1}^{r} d^d x_i \frac{J(x_1)...J(x_r)}{r!} G_c^{(r)}(x_1, ..., x_r) \ . \tag{3.26}$$

The connected four-point function $G_c^{(4)}(x_1, ..., x_4)$, when the odd terms vanish, is sufficient in capturing a lot of the leading-order Gaussianities in most examples. Firstly, non-Gaussianities can arise when the neurones $h_i(x)$ are i.d.d. but N is finite. This arise, for example, in single hidden layer networks. Then we can write the CGF in terms of the connected correlation functions of the neurones,

$$\begin{aligned} W_{\phi(x)}[J] &= \log \mathbb{E}\left[\exp\left(\frac{1}{\sqrt{N}} \sum_{i=1}^{N} \int d^d x J(x) h_i(x) \right) \right] \\ &= \sum_{r=1}^{\infty} \int \prod_{i=1}^{r} d^d x_i \frac{J(x_1)...J(x_r)}{r!} \frac{G_{c,h_i}^{(r)}(x_1, ..., x_r)}{N^{r/2-1}} \ , \end{aligned} \tag{3.27}$$

note that we have used (since we have i.d.d. h_i),

$$G_c^{(r)}(x_1, ..., x_r) = \frac{G_{c,h_i}^{(r)}(x_1, ..., x_r)}{N^{r/2-1}} \ . \tag{3.28}$$

Note that the scaling of N here implies that,

$$\lim_{N \to \infty} G_c^{(r>2)}(x_1, ..., x_r) = 0 \ . \tag{3.29}$$

Examples of evaluating this on the ReLU-net, $\phi(x) = W_i^1 R(W_{ij}^0 x_j)$ where $R(z) = zH(z)$ with H being the Heaviside step function and the Cos-net, $\phi(x) = W_i^1 \cos\left(W_{ij}^0 x_j + b_i^0\right)$ can be found in [8]. Of course, non-Gaussianities will also arise, as discussed above, from independence breaking. Since we want to perturb around the Gaussian fixed point we require,

$$P(h; \alpha = 0) = \prod_i P(h_i) \ , \tag{3.30}$$

where the hyperparameter vector α is chosen as part of the architecture definition. The limit $\alpha \to 0$ means the neurones become independent. Now the CGF is,

$$W_\phi[J] = \log\left[\int\left(\prod_{i=1}^{N} Dh_i\right) P(h;\alpha) e^{\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\int dx h_i(x)J(x)}\right] . \tag{3.31}$$

For small α we can expand $P(h;\alpha)$ which gives

$$W_\phi[J] = \log\left[e^{W_{\phi,\alpha=0}} + \sum_{r=1}^{\infty}\sum_{s_1,..,s_r=1}^{q}\frac{\alpha_{s_1}...\alpha_{s_r}}{r!}\prod_{i=1}^{N}\mathbb{E}_{P_i(h_i)}\left[e^{\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\int dx h_i(x)J(x)}\cdot\mathcal{P}_{r,\{s_1,...,s_r\}}|_{\alpha=0}\right]\right] , \tag{3.32}$$

where I have defined,

$$\mathcal{P}_{r,\{s_1,...,s_r\}} = \frac{1}{P(h;\alpha)}\partial_{\alpha_{s_1}}...\partial_{\alpha_{s_r}}P(h;\alpha) . \tag{3.33}$$

This is the form that combines both of the α and N-dependencies for non-Gaussianities. We can further approximate using $\log(1+x) \approx x$ where,

$$W_\phi[J] = W_{\phi,\alpha=0}[J] + \sum_{s=1}^{q}\frac{\alpha_s}{e^{W_{\phi,\alpha=0}[J]}}\prod_{i=1}^{N}\mathbb{E}_{P_i(h_i)}\left[e^{\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\int dx h_i(x)J(x)}\cdot\mathcal{P}_{1,s}|_{\alpha=0}\right] \tag{3.34}$$

and the cummulants look like,

$$\begin{aligned} G_c^{(r)}(x_1,...,x_r) &= \left.\frac{\partial^r W_\phi[J]}{\partial J(x_1)...\partial J(x_r)}\right|_{J=0} \\ &= G_c^{(c),\text{i.i.d. }c+\alpha\cdot\Delta G^{(r)}} + O(\alpha^2) . \end{aligned} \tag{3.35}$$

So we see that the leading order correction is proportional to α. You can again find an example (single layer Cos-net) in [8].

3.2 Feynman diagrams for NN-FT

Now we ask the following question.

Question: Given a NN-FT architecture, can we now calculate the action from the connected correlators?

The answer to that is unsurprisingly yes, but there are several things that complicate our analysis. In particular, we would like to follow the following scheme.

1. **Computing NN correlation functions.** At initialisation it is possible to compute the correlators of the neural network by using EFT-techniques and perturbative diagrammatic methods.

2. **Edgeworth Expansion.** We then use the knowledge of these correlators, or cumulants, to approximate the probability density and compute the action. A different kind of Feynman diagrams can be developed to compute these partition functions.

3. **Specialising to NNFTs.** We finally specialise the analysis above to NNFTs, especially limiting to the case of non-Gaussianities.

We will discuss these concepts one by one.

Computing NN correlation functions. From the previous sections we see that the non-Gaussianities are encoded in the high-order correlators of the theory. It is therefore useful to develop a general method to compute such correlators. As shown in for example [9], it is possible to compute NN-correlators by using an effective theory approach. We consider a family of NN architectures with learnable parameters θ and discrete hyperparameter N,

$$f_{\theta,N} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}} \,, \tag{3.36}$$

and we focus on the state of the system at initialisation where the parameters are drawn $\theta \sim P(\theta)$. At the limit $N \to \infty$, the NN becomes a Gaussian process in which on a fixed set of k inputs $\{x_1, ..., x_k\}$ are drawn from a multivariate Gaussian distribution $\mathcal{N}(\mu, \Xi^{-1})$,

$$\{f(x_1), ..., f(x_k)\} \sim \mathcal{N}(\mu, \Xi^{-1}) \,. \tag{3.37}$$

Here the inverse covariance matrix Ξ is determined by the kernel function $K(x, x')$ as $(\Xi^{-1})_{ij} = K_{ij} = K(x_i, x_j)$, which if $\mu = 0$, entirely determines the distribution. Borrowing from the QFT notations, and anticipating the correspondence in NNFT, we can now write the n-point correlator as,

$$G^{(n)}(x_1, ..., x_n) = \frac{\int df \phi(x_1)...\phi(x_n)e^{-S}}{Z} \,, \tag{3.38}$$

where the log-likelihood determines the form of the process we are looking at. Here I have replaced $f(x)$ with $\phi(x)$ to indicate that this looks like a 'field' in our correspondence. In the Gaussian limit, we will have,

$$S_{\text{GP}} = \frac{1}{2} \int d^d x d^d x' \phi(x) \Xi(x, x') \phi(x') \,, \tag{3.39}$$

where $\Xi(x, x') = K^{-1}(x, x')$ is the inverse covariance function and I further drop the subscript of d_{in} for simplicity. To relate this to field theories, suppose $\Xi(x, x') = \Xi(x) L_\sigma(x, x')$ where $L_\sigma(x, x')$ is any family of functions that limits to $L_0(x, x') = \delta^{(d_{\text{in}})}(x - x')$ (say this is a family of normalised Gaussians, for example). In this limit Eq. (3.39) will reduce to

$$S_{\text{local-GP}}[\phi] = \frac{1}{2} \int d^{d_{\text{in}}} x \phi(x) \Xi(x) \phi(x) \,, \tag{3.40}$$

which when comparing this with the free-field theory action,

$$S[\phi] = \int d^d x \phi(x)(\Box + m^2)\phi(x) \,, \tag{3.41}$$

so we see that how the Gaussian process is closely related to a free field theory. Eq. 3.40 refers in particular to a **local** Gaussian process, which is normally what is imposed on a QFT.

Using this correspondence it is now possible to analyse the correlation functions using the perturbative diagrammatic methods from QFT. In particular, the partition function of the Gaussian process is,

$$Z_{\text{GP}}[J] = \frac{\int df \exp\left(-S_{\text{GP}} - \frac{1}{2}\int d^{d_{\text{in}}} J(x)\phi(x) - \frac{1}{2}\int d^{d_{\text{in}}} J(y)\phi(y)\right)}{Z_{\text{GP},0}} \,, \tag{3.42}$$

where the source is $J(x)$, $Z_{\text{GP},0}$ is the partition function of the Gaussian process integral given by Eq. (3.39). The n-point correlator is hence defined by,

$$G^{(n)}_{\text{GP}}(x_1, ..., x_n) = \frac{\int \mathcal{D}\phi\phi(x_1)...\phi(x_n)e^{-S_{\text{GP}}}}{Z_{\text{GP},0}} \,, \tag{3.43}$$

This of course can be evaluated exactly in the GP case, which yields

$$Z_{\text{GP}}[J] = \exp\left(\frac{1}{2}\int d^dx d^dy J(x)K(x,y)J(y)\right) \,, \tag{3.44}$$

and

$$G^{(n)}_{\text{GP}}(x_1, ..., x_n) = \left[\left(-\frac{\delta}{\delta J(x_1)}\right)...\left(-\frac{\delta}{\delta J(x_n)}\right)Z_{\text{GP}}\right] \,. \tag{3.45}$$

To calculate the correlators, we perform Wick contractions where,

$$\text{Wick}(x_1, ..., x_n) = \{P \in \text{Partitions}(x_1, ..., x_n) \,|\, |p| = 2, \ \forall p \in P\} \,, \tag{3.46}$$

so

$$G^{(n)}_{\text{GP}}(x_1, ..., x_n) = \sum_{p\in\text{Wick}(x_1,...,x_n)} K(a_1, b_1)...K(a_{n/2}, b_{n/2}) \,. \tag{3.47}$$

This exactly mirrors what happens in free-field theories. For small deformations away from the Gaussian process by introducing non-Gaussianities, we can use a similar method to compute correlation functions. The log-likelihood (action) now has two parts,

$$S = S_{\text{GP}} + \Delta S \,, \tag{3.48}$$

and to construct the effective action of an NGP associated to a NN architecture that admits a known GP limit requires the following Wilsonian EFT rules [9]:

- Determine the symmetries respected by the system.
- Fix an upper bound k on the dimension of any operators appearing in ΔS.
- Define ΔS to contain all operators of dimension $\leq k$ that respect the symmetries determined.

Of course, the NGP n-point correlation functions are similar defined as in Eq. 3.45,

$$G^{(n)}(x_1, ..., x_n) = \frac{\int \mathcal{D}\phi\phi(x_1)...\phi(x_n)e^{-S}}{Z_0} \,, \tag{3.49}$$

where $Z_0 = \int \mathcal{D}\phi e^{-S}$. It is possible to linearise the expression for a small $\Delta S = \int d^dx g_k\phi(x)^k$ where $k > 2$ to approximate the n-point correlator up to some order in g_k the coupling constant. The integrals can now be evaluated with the help of Wick's theorem, which is sketched out in Appendix A of [9]. Additionally, cutoffs need to be chosen to remove naïvely yielded divergences. This requires the coefficients in the log-likelihood to obey a set of Wilsonian renormalisation group equations. In practice, however, one would compute the correlation functions in perturbation theory after determining the allowed operators from the symmetries of the architecture, and then utilise Feynman rules (which one can derive from the perturbative methods as sketched out in [9]) to compute the correlators. There are many more technical issues such as the technical naturalness of the operators in the theory, but we leave the reader to [9] for details.

Edgeworth Expansion, finding the corresponding action. What we have calculated before is more of an approximation as to what the corresponding action on the FT side of the correspondence. Instead, we can try and directly sample the cumulants (correlation functions) of the NNs, and then look at what the corresponding action is. To do this, we need to apply the Edgeworth expansion to field theory. Writing the generating function in terms of,

$$e^{W[J]} = \int d\phi e^{-S[\phi]+J\phi} \,, \tag{3.50}$$

We can add a source term in the exponent, mapping $J \mapsto iJ$ and deforming the integral back to real J to give,

$$P[\phi] = e^{-S[\phi]} = \int dJ e^{W[J]-J\phi} \,, \tag{3.51}$$

which gives the probability density and action in terms of $W[J]$. We now apply the Edgeworth expansion as follows. Writing,

$$W[J] = \sum_{r=1}^{\infty} \frac{\kappa_r}{r!} J^r \,, \tag{3.52}$$

which gives,

$$P[\phi] = \exp\left[\sum_{r=3}^{\infty} \frac{\kappa_r}{r!}(-\partial_\phi)^r\right] e^{-\frac{(\phi-\kappa_1)^2}{2\kappa_2}} \,, \tag{3.53}$$

and extending this to the field theory case yields,

$$e^{-S[\phi]} = \frac{1}{Z} \exp\left[\sum_{r=3}^{\infty} \frac{(-1)^r}{r!} \int \prod_{i=1}^{r} d^d x_i G_c^{(r)}(x_1, ..., x_r) \frac{\delta}{\delta\phi(x_1)} ... \frac{\delta}{\delta\phi(x_r)}\right] e^{-S_{\rm GP}[\phi]} \tag{3.54}$$

with the Gaussian process action defined as above in Eq. (3.39). The expansion in Eq. (3.54) is known as the Edgeworth expansion, and it is related directly to the partition function defined as,

$$e^{W[J]} = Z[J] = c' \exp\left[\sum_{r=3}^{\infty} \int \prod_{i=1}^{r} d^d x_i g_c^{(r)}(x_1, ..., x_r) \frac{\delta}{\delta\phi(J_1)} ... \frac{\delta}{\delta\phi(J_r)}\right] e^{-S_0[J]} \tag{3.55}$$

with, using the notations as above in Eq.(3.39) but replacing $f(x) \mapsto \phi(x)$ as the fields (NNs),

$$S_{\rm GP}[\phi] = \frac{1}{2} \int d^{d_{\rm in}} x d^{d_{\rm in}} x' \phi(x) \Xi(x, x') \phi(x') \,, \tag{3.39}$$

$$S_0[J] = \frac{1}{2} \int d^{d_{\rm in}} x d^{d_{\rm in}} x' J(x) K(x, x') J(x') \,, \tag{3.39}$$

Here I have defined $K(x, x')$ as the inverse of the inverse covariance operator $\Xi(x, x')$, which is just the free propagator. We have done nothing but done the simplest Legendre transformation on the variables $\phi(x) \leftrightarrow J(x)$, i.e. going between the field and source pictures. The corresponding CGFs and cumulant can be then summarised in the following table. Earlier in the section we have developed a method to calculate the cumulants

	Field Picture	Source Picture
Field	$\phi(x)$	$J(x)$
CGF	$W[J] = \log(Z[J])$	$S[\phi] = \log(P[\phi])$
Cumulant	$G_c^{(r)}(x_1, ..., x_r)$	$g_r(x_1, ..., x_r)$

Table 3.1: A table summarising the Edgeworth expansion for $P[\phi]$ and the interaction expansion for $Z[J]$ - this is just the Legendre transformation in action. Taken from [8].

$G_c^{(r)}(x_1, .., x_r)$ which are the correlators of the theory. What we really want is to compute the interaction term coupling constants,

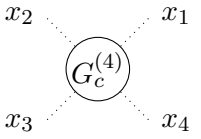
$$S_{\text{int}} = \sum_{r=3}^{\infty} \int \prod_{i=1}^{r} d^d x_i g_r(x_1, ..., x_r)\phi(x_1)...\phi(x_r) \ . \tag{3.56}$$

Using the Feynman rules we have established for the effective theory approach, we can now use the fact that the structure is similar for both pictures under the Legendre transformation to establish Feynman rules for computing the cumulants $g_r(x_1, .., x_r)$, the interaction couplings, in terms of the correlators $G_c^{(r)}(x_1, .., x_r)$. The details of the Feynman rules can be found in [8], and an explicit example using the ϕ^4 theory can be also found there.

Generating Interacting Actions for NNFT. It remains to actualise specialise the scheme to NNFTs. As we have discussed in §3.1 in detail, the non-Gaussianities are generated by two mechanisms - finite N corrections and independence breaking. In the first case, the non-Gaussianities due to $1/N$-corrections can be evaluated to scale as,

$$G_c^{(r)}(x_1, ..., x_r) \propto \frac{1}{N^{r/2-1}} \ . \tag{3.57}$$

For example, the quartic coupling $g_4(x_1, ..., x_4)$ at leading order in $G_c^{(4)} \propto \frac{1}{N}$, is,

$g_4(x_1, x_2, x_3, x_4) =$ 

$+ O\left(\frac{1}{N^2}\right)$ (3.58)

For the details see [8]. We have therefore deduced that the leading-order $1/N$-action is,

$$S = S_{\text{GP}} + \int d^d x_1 ... d^d x_4 g_4(x_1, x_2, x_3, x_4)\phi(x_1)...\phi(x_4) + O\left(\frac{1}{N^2}\right) \ . \tag{3.59}$$

Of course, non-Gaussianities can also be generated by independence-breaking. From Eq. (3.35), we know that the connected correlation functions scale as,

$$G_c^{(r)}(x_1, ..., x_r) \propto \alpha \ , \tag{3.60}$$

for all $r > 2$ where α is the hyperparameter controlling the degree of independence-breaking. It is actually possible to show that each coupling $g_r(x_1, ..., x_r)$ receives contributions from

tree-level diagrams of all connected correlators at leading order in α. This makes it impossible to approximate $g_r(x_1, .., x_r)$ using a finite number of terms via a perturbative expansion in α unless the theory exhibits some additional structure. It is perhaps possible to ask whether a resummation of the series can be obtained to allow for a consistent perturbative expansion (see [8] for details).

3.3 Engineering actions in NN-FT

We now want to the reverse and answer the following question.

Question: Is it possible to engineer an architecture to realise an action? The first part of the previous section, §3.2, actually gives a good description into how one would start. Given an effective field theory and an NN architecture which gives the free propagator, what we have done is to come out with a diagrammatic method to calculate the perturbative expansion up to some order in an expansion hyperparameter. We now want to completely invert this, i.e. to actually realise the operator as some parameter density deformation of a NNFT. This will allow us to engineer local interactions. But to do this we will need to develop the exact mechanism for it.

We begin by looking at the partition function of a Gaussian theory

$$Z_G[J] = \mathbb{E}_G\left[e^{\int d^dx J(x)\phi(x)}\right] , \tag{3.61}$$

in which we now deform it by an operator insertion of the following form,

$$Z[J] = \mathbb{E}_G\left[e^{-\lambda\int d^dx_1...d^dx_r\mathcal{O}_\phi(x_1,...,x_r)}e^{\int d^dx J(x)\phi(x)}\right] , \tag{3.62}$$

with $\mathcal{O}_\phi$ being a non-local operator in general depending on the function $\phi(x)$ and its derivatives. Then in function space we have the partition function deformed to,

$$Z[J] = \int D\phi e^{-S[\phi]+\int d^dx J(x)\phi(x)} , \tag{3.63}$$

where the action has been deformed to

$$S_G[\phi] \mapsto S_G[\phi] + \lambda\int d^dx_1...d^dx_r\mathcal{O}_\phi(x_1, ..., x_r) = S[\phi] . \tag{3.64}$$

What we really want to know is how the deformation is translated in parameter space. The Gaussian partition function in parameter space is,

$$Z_G[J] = \int d\theta P_G(\theta)e^{\int d^dx J(x)\phi(x)} , \tag{3.65}$$

so the deformation gives,

$$Z[J] = \int d\theta P_G(\theta)e^{-\lambda\int d^dx_1...d^dx_r\mathcal{O}_\phi(x_1,...,x_r)}e^{\int d^dx J(x)\phi(x)} . \tag{3.66}$$

How do we describe the deformation in parameter space? We can define a deformed parameter distribution as,

$$P(\theta) = P_G(\theta)e^{-\lambda\int d^dx_1...d^dx_r\mathcal{O}_\phi(x_1,...,x_r)} , \tag{3.67}$$

where the architecture remains the same but the parameter distribution $P(\theta)$ has been deformed. So the non-Gaussian theory indeed must receive corrections only from independence breaking - the non-Gaussianities solely come from the deformation of the distribution. It is unclear whether it is more natural for non-Gaussianities to arise this way, but we know that in the proof of UAT the error in the approximation goes down with increasing N. Hence it is expected that there is at least one finite-action configuration $\phi(x)$ in ϕ^4 theory that cannot be realised by a finite-N neural network of fixed architecture, giving us the conclusion that the true NN-FT realisation of a particular theory, say ϕ^4-theory, must occur at infinite-N.

In practice, we deform the Gaussian theory by deforming the independent Gaussian parameter densities $P_G(\theta_1)...P_G(\theta_N)$ to a non-trivial joint density given by,

$$P(\{\theta_i\}) = \prod_i (P_G(\theta_i)) \times e^{\lambda \int d^d x_1 ... d^d x_r \mathcal{O}_\phi(x_1,...,x_r)} \ . \tag{3.68}$$

We then follow this by writing the partition function for the deformed theory in parameter space.

Example 3.1 (ϕ^4 theory as deformed parameter distribution)**.** Let us illustrate this with an example. We want to realise a local ϕ^4 theory, with the action,

$$S[\phi] = \int d^d x \left[\phi(x) \left(\nabla^2 + m^2 \right) \phi(x) + \frac{\lambda}{4!} \phi^4(x) \right] \ . \tag{3.69}$$

We first engineer the NNGP. Take,

$$\phi_{a,b,c}(x) = \sum_i \frac{a_i \cos\left(b_{ij} x_j + c_i\right)}{\sqrt{\mathbf{b}_i^2 + m^2}} \tag{3.70}$$

where the sum runs from 1 to infinity $N \to \infty$ and b_i is the i-th row vector of the matrix b_{ij}. The parameter densities of the Gaussian theory are,

$$\begin{aligned} P_G(a) &= \prod_i e^{-\frac{N}{2\sigma_a^2} a_i a_i}, \\ P_G(b) &= \prod_i P_G(\mathbf{b}_i), & P_G(\mathbf{b}_i) &= \mathcal{U}(B_\Lambda^d) \qquad (3.71) \\ P_G(c) &= \prod_i P_G(c_i), & P_G(c_i) &= \mathcal{U}([-\pi, \pi]) \qquad (3.72) \end{aligned}$$

where $\mathcal{U}$ indicates the uniform distribution and B_Λ^d is a d-sphere with radius Λ. The two-point function now has the form,

$$G^{(2)}(p) = \frac{\sigma_a^2 (2\pi)^d}{2\mathrm{vol}(B_\Lambda^d)} \frac{1}{p^2 + m^2} \tag{3.73}$$

which becomes the standard free scalar result if we rescale the fields to give,

$$\phi_{a,b,c}(x) = \sqrt{2\mathrm{vol}(B_\Lambda^d)} \sigma_a^2 (2\pi)^d \sum_{i,j} \frac{a_i \cos\left(b_{ij} x_j + c_i\right)}{\sqrt{\mathbf{b}_i^2 + m^2}} \ . \tag{3.74}$$

Here Λ acts as the hard UV cutoff on the momentum. Now we insert the operator,

$$\exp\left(-\frac{\lambda}{4!}\int d^d x \phi_{a,b,c}(x)^4\right) , \tag{3.75}$$

so we deform the parameter density to,

$$P(a,b,c) = P_G(a)P_G(b)P_G(c)\exp\left(-\frac{\lambda}{4!}\int d^d x \phi_{a,b,c}(x)^4\right) \tag{3.76}$$

on the RHS of which we integrate all dependencies of x leaving a function of a, b, c only. We can then write the partition function for the deformed theory as,

$$Z[J] = \int dadbdcP(a,b,c)e^{\int d^d x J(x)\phi_{a,b,c}(x)} . \tag{3.77}$$

So the architecture specified by $\phi_{a,b,c}(x)$ and the parameter density $P(a,b,c)$ together realise the local ϕ^4 theory.

There is more discussion on cluster decomposition, a weaker form of locality, which requires studying space-dependent field theory which has different statistics at every point in space at initialisation in [8]. We will not discuss these issues here.

3.4 Classical field configurations and Landscapes

We have seen how the correspondence between NN and FTs work. This is highly non-trivial, and a priori the action is unknown. Is there a different way of analysing classical field configurations? We now use the last physical principle we mentioned at the start of the seminar - namely 'Landscape Dynamics', to analyse the issues concerning field configurations.

Classical field configurations. We can think of classical field configurations as local maxima in probability (parameter space probability). For neural field $\phi_\theta(x)$ with parameters $\theta \sim P(\theta)$ drawn from some distribution, the set of classical configurations are of then of the form,

$$S_c = \{\phi_{\theta^*} \,|\, \theta^* \in \max(P(\theta))\} \tag{3.78}$$

where $\max(P(\theta))$ is the set of local maxima of $P(\theta)$. Of course, the dominant configurations in S_c will then be $\phi_{\theta^*_d}$ with $\theta^*_d = \arg_\theta \max(P(\theta))$, so the global maxima.

In particular, the neurones h have their own classical configurations which influences the field configuration. If we write,

$$\phi(x) = \sum_{i=1}^{N} a_i h_i(x) , \tag{3.79}$$

with $a \sim P(a)$. Then,

$$S^h_c = \{h_{\theta^*} \,|\, \theta^*_h \in \max(P(\theta_h))\} . \tag{3.80}$$

We can write the classfical field configurations as,

$$S_c = \left\{ \phi_{a^*,h_i} \,\middle|\, a^* \in \max(P(a)),\, h_i \in S_c^h \right\} \,. \tag{3.81}$$

The number of classical configurations is then bounded by,

$$|S_c| \leq |\max(P(a))||S_c^h|^N \,, \tag{3.82}$$

where a priori we have chosen the N classical neurones and maxima independently. The number however may be different:

- $|S_c|$ is much smaller due to redundancies in the system.
- If $\max(P_a)$ is only solved by one solution given by $a_i = 0$, $\forall i$ then there is no non-trivial classical field configuration. This arises for example in a_i being i.d.d. draws with maximum only at $a_i = 0$.
- Consider the case where $|S_c^h| > N$ and ϕ is constructed out of a set of classical neurones h_i without repeats. Then,

$$|S_c| \geq |\max(P(a))|^N \frac{|S_c^h|!}{N!(N - |S_c^h|)!} \,, \tag{3.83}$$

  so if $P(a)$ is multimodal then the number of classical configurations is exponentially large in N. This is how large landscapes of classical configurations can be generated.

Spontaneous Symmetry Breaking (SSB). This arises when an architecture exhibits a symmetry in its correlators but not in its classical configurations. For example, we can have a D-dimensional real scalar field (NN),

$$\phi_i(x) = \sum_{j=1}^{N} a_{ij} h_j(x) \,, \tag{3.84}$$

where $a \sim \mathcal{N}(0, \sigma_a^2/N)$ and this exhibits an $SO(D)$ invariance of the Gaussian. Taking the rows of a_{ij} as $\mathbf{a}_j$ as,

$$\mathbf{a}_j \sim P(\mathbf{a}_j) \propto \exp\left(-\frac{1}{2\sigma_a^2} \mathbf{a}_j \cdot \mathbf{a}_j \right) \,, \tag{3.85}$$

gives a factorisation into products of Gaussians, but taking

$$\mathbf{a}_0 \sim P(\mathbf{a}_0) \propto \exp\left[-\frac{1}{2\sigma_a^2} \left(\mathbf{a}_0 \cdot \mathbf{a}_0 - v^2 \right)^2 \right] \,, \tag{3.86}$$

and the other distributions the same leaves the correlators $SO(D)$-invariant but the classical distribution of $\mathbf{a}_0$ to be on S_v^{D-1}. The correlation functions of the fluctuations around this associated classical configuration are then no-longer $SO(D)$-invariant. This is the simplest example of realising SSO in NNs.

There is perhaps a lot more that we can talk about regarding NNFTs. This field is indeed very young, and it has some very exciting prospects in the future. I hope this is sufficient in giving a good first introduction to what this construction is, and I really hope that you will perhaps be inspired to contribute to this framework in the future.

References

[1] F. Ruehle, "Data science applications to string theory," *Phys. Rept.*, vol. 839, pp. 1–117, 2020.

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, pp. 303–314, Dec. 1989.

[3] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks," 4 2024.

[4] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer New York, 1996 ed., 1996.

[5] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[6] J. Halverson, "TASI Lectures on Physics for Machine Learning," 7 2024.

[7] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[8] M. Demirtas, J. Halverson, A. Maiti, M. D. Schwartz, and K. Stoner, "Neural network field theories: non-Gaussianity, actions, and locality," *Mach. Learn. Sci. Tech.*, vol. 5, no. 1, p. 015002, 2024.

[9] J. Halverson, A. Maiti, and K. Stoner, "Neural Networks and Quantum Field Theory," *Mach. Learn. Sci. Tech.*, vol. 2, no. 3, p. 035002, 2021.

[10] J. Halverson, "Building Quantum Field Theories Out of Neurons," 12 2021.