EFT on Transformers

Gijs van Seeventer

Oktober 2024

Abstract

This project investigated the multi-head self-attention (MHSA) component of transformers, focusing on its theoretical and numerical characterization using effective field theory (EFT). The forward equation for MHSA at leading order revealed deviations from Gaussian assumptions, further supported by challenges in deriving the equation for the first hidden layer. Numerical analysis using a simplified linear MLP confirmed non-Gaussian behavior even in this "easy" case. These findings challenge claims that MHSA can be modeled as Gaussian and suggest complexities in its theoretical description. Additionally, by accident it was found that using the standard deviation instead of variance in the theory, the MLP alignment with theory improved, warranting further exploration. Note, this is a report written while working to retain the information, it is not a properly finalised report.

Contents

1 Introduction

This paper focuses on better understanding machine learning (ML). To achieve a better understanding in this context means to have a theoretical framework that describes neural networks (NNs). In this paper we therefore apply Richard Kenway's developed effective-field theory (EFT) [1] on understanding the claim in [2] that transformers can be properly described by (an) EFT at leading order with respect tot the width n. Given the complexity of MLPs and the need for multiple layers combined with the known complexity of depth when using EFTs to describe NNs, we focused on one component of the transformer: the multi-head self-attention (MHSA). The MHSA is the most interesting component with respect to EFTs that are known to describe multi-linear perceptrons (MLP) quite well. This paper provides context on the paper [2], an introduction to the architecture of the transformer and an introduction to the formalism of Kenway. Besides this context the paper shows that a key intermediate result of [2] doesn't hold. Furthermore the paper applies Kenway's formalism to the MHSA and finds that gaussians are probably not a good description of the MHSA. This understanding is both reached from a theoretical and numerical experimental perspective, although the latter is limited to a linear activation function. Finally, the paper by accident encountered the interesting result that EFTs describe MLPs with a linear activation function better if the standard deviation instead of the covariance is used.

Note, this is a report written while working to retain the information, it is not a properly finalised report.

2 Structure of the Reference Paper

Their paper is separated in to two parts. The first part deals with developing the theoretical framework of, and the second part applies this framework on transformers. The theoretical framework is an EFT that expands up $\mathcal{O}\left(\frac{1}{n}\right)$, where n is the width of the model.

To begin with the first part, which they have split up into four parts: 0) they define what a transformer is by identifying several components, 1) they look at one and the variance of the weights and biases at initialization, 2) they introduce the neural tangent kernel (NTK) for two types of back propagation (SGD and AdamW) and 3) they provide a forward equation of the NTK at initialization.

In both the determining of the 1) variance and 3) NTK forward equation, they aim at all the different components staying at $\mathcal{O}(1)$ with respect to the width n, such that it doesn't explode.

3 Defining a Transformer

The transformer architecture definition is sketched in figure 1. The figure starts with a data sample δ from a dataset $\mathcal{D}$. From there all its components are numbered and named. Each component will be explained in the following subesections. The output of each component is also indicated in the algorithm, unless specifically indicated the output is of layer $l \in \{2, L-1\}$ with L the number of the final(/deepest) layer. Note that the $\bigoplus$ is a commonly used symbol for addition.

Finally, each sub section starts with the input and output which is elaborated on where necessary. Furthermore, the input and output will also specify the relevant indices that need to be tracked while performing the action, e.g. $f^{(\ell)}_{\delta,t,i} = f_t$ will mean that only the index t of all the indices will be relevant in the component. However, all the indices could be restored (in all) wherever desired. With the implicit assumption that constants and weights never track the token index t unless specified otherwise.

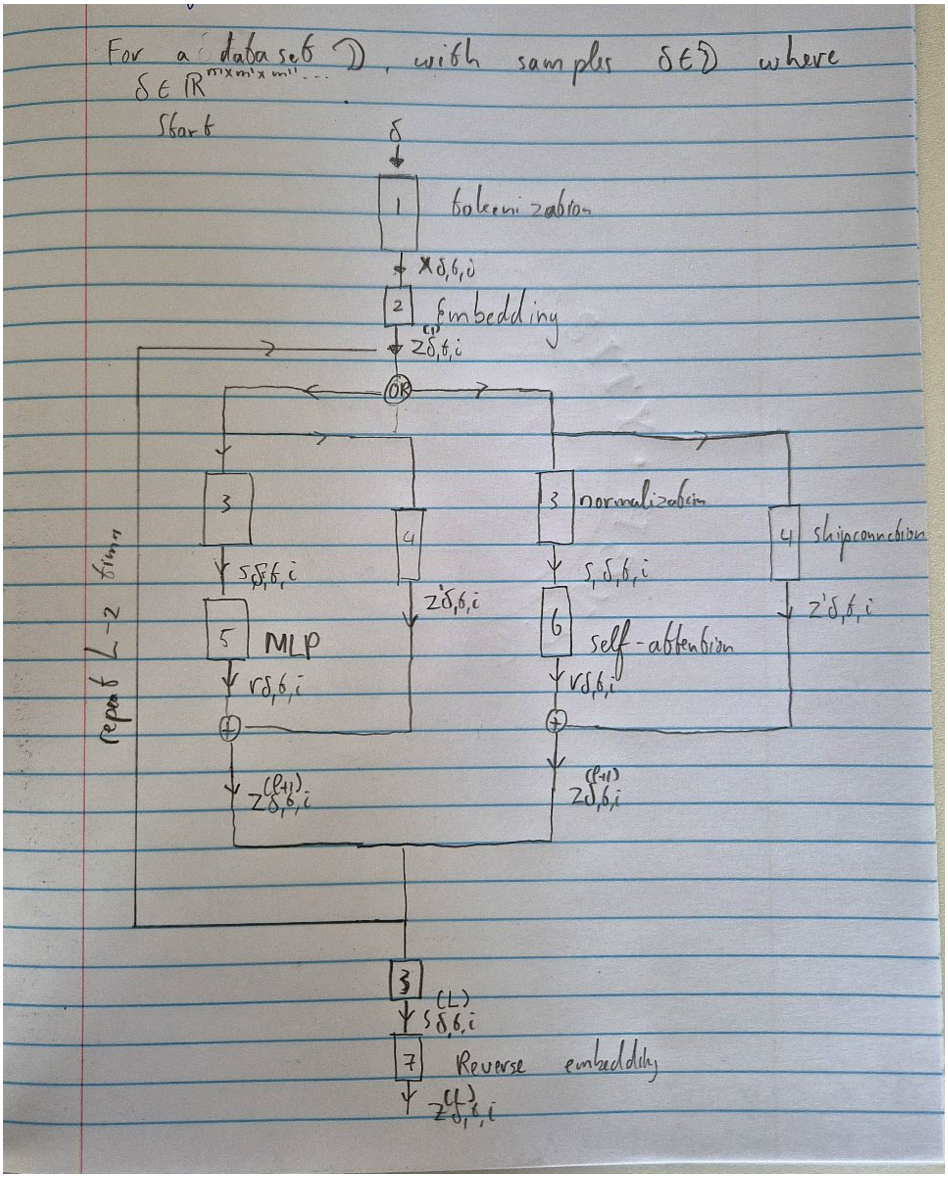


Figure 1: A sketch of a transformer, all its components will be explained per numbered section. The output of each component is also indicated in the algorithm, unless specifically indicated the output is of layer $l \in \{2, L-1\}$ with L the number of the final(/deepest) layer. Note that the $\bigoplus$ is a commonly used symbol for addition.

3.1 Tokenisation

Given a dataset $\mathcal{D}$, let a sample from the dataset be $\delta \in \mathcal{D}$ with $\delta \in \mathbb{R}^{m \times m' \times m'' \cdots}$. The first step is to map the samples δ to a matrix, such that a sample after this mapping has the shape $\delta \in \mathbb{R}^{n_t \times n_{in}}$ and a sample δ has the shape $[x_0, ..., x_t, ..., x_{nt}]$ where $x_{t,i} \in \mathbb{R}^{n_{in}}$, not that the order of the samples has to be preserved.In short, the output of tokenisation is a vector $x_{\delta,t,i}$.

As a remark, note that there is some ambiguity with respect to the word token. In the literature it is also common to refer to individual elements form $\mathbb{R}^{n_{in}}$ as tokens, which is not to be confused with being a token x_t at position t.

3.2 Embedding

Input: $x_{\delta,t,i}$
Output:

$$z^{(\ell)}_{\delta,t,i} = b_{t,i} + \sum_{j=1}^{n'} w_{ij} x_{\delta,t,j} \tag{3.1}$$

where $i \in [1, ..., n]$.
This represents the embedding, where the input $x_{\delta,t,i}$ is transformed using the weight w_{ij} and bias $b_{t,i}$, resulting in the embedding vector $z^{(1)}_{\delta,t,i}$. The bias $b_{t,i}$ in this case is supposed to take into account (some) positional information of the token.

3.3 Normalization

Input: $z^{(\ell-1x)}_{\delta,t,i} = z_i$
Output:

$$s^{(\ell)}_{\delta,t,i} = s_i = \gamma_i [\frac{z_i - \frac{1}{n}\sum_{j=1}^{n} z_j}{\sqrt{\frac{1}{n}\sum_{j=1}^{n} z_j^2 - \left(\frac{1}{n}\sum_{j=1}^{n} z_j\right)^2 + \epsilon}}] + \beta_i, \tag{3.2}$$

Where $\gamma_{i=1}$ and $\beta_i = 0$ in the notation. I'm not a 100% sure, but I think β actually functions as some kind of bias since normalization also happens after each transformation while there is no explicit bias present at those steps in the reference paper.

In this step, s_i represents the normalized version of z_i, where the input is scaled by subtracting the mean and dividing by the standard deviation.

- Mean of z_i:

$$\frac{1}{n}\sum_{i=1}^{n} z_i \tag{3.3}$$

- Variance of z_i:

$$z_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n} z_i\right)^2 \tag{3.4}$$

The small constant ϵ is added to the denominator to avoid division by zero.

3.4 Skip Connection

Input: $z^{(\ell)}_{\delta,t,i} = z_i$
Output:

$$z^{(\text{skip})}_{\delta,t,i} = \mathcal{E}_i \cdot z_i, \tag{3.5}$$

where $\mathcal{E}_i \in [0, 1]$. In this operation, the output z' is generated by applying an element-wise multiplication of z_i with $\mathcal{E}_i$. This operation passes information from earlier layers forward through the network, facilitating gradient flow and improving learning efficiency in deeper models.

3.5 MLP

Inputs: $s^{(\ell)}_{\delta,t,i} = s_i$
Output:

$$r^{(\ell)}_{\delta,t,i} = r_i = \sum_{j=1}^{Mn} X_{ij}\rho(w_j(s_j)), \tag{3.6}$$

where M is an integer, ρ is an activation function, and

$$w_i = \sum_{i=1}^{n} W_{ij}s_j, \tag{3.7}$$

with $i = 1, \ldots, Mn$. Notice that there is no bias.

The scaling up via M can be seen as an a single layer increase to a wider neural network, which is then again decreased to the common 'embedded' space of size n.

As a side note, this should approach large with more easily. Apparently a common choice for $M = 4$.

3.6 Self-Attention

Input: $s^{(\ell)}_{\delta,t,i} = s_{t,i}$
Output:

$$r^{(\ell)}_{\delta,t,i} = r_{t,i} = \sum_{h=1}^{n_h}\sum_{t'=1}^{n_t}\Omega^h_{tt'}(s)\sum_{j=1}^{n} E^h_{ij}s_{t',j}, \tag{3.8}$$

where n_h is the number of 'heads', n_t is the size of the token space, $\Omega^h_{tt'}$ is the query-key dot product, $n_c := \frac{n}{n_h}$ must be an integer and is known as the number of channels, and E^h_{ij} is an encoder-decoder matrix, which is defined as $E^h_{ij} = \sum_{c=1}^{n_c} U^h_{ic}V^h_{cj}$, where U^h_{ic} projects to the token space and V^h_{cj} is known as the value matrix.

Before diving into the specific definitions, we can notice that the term on the r.h.s. of the sums acts as a linear MLP with zero bias. While the matrix E^h_{ij} is originally defined as a (sort of) encoder-decoder via the matrices V^h_{cj} and U^h_{ic}, that respectively act as a weight scaling down from width n to width n_c and scaling up to width n again, this could in principal be hidden, in the notation, as just a 'regular' weight matrix going from width n to n. However, it is important to note that while hiding this encoder-decoder in the notation it is a potential point where the infinite width limit doesn't hold. This MLP per token t' is then weighted via the query-key dot product over the token index t, i.e. it is again a linear MLP but now working on the token t instead of the element i. So in short, each token represented as $s_{t'}$ is multiplied with a weight E^h_{ij} while a another weight $\Omega^h_{tt'}$ runs over the token index.

The number of heads just repeats all of the above this and could perhaps be ignored since the reason/impact is not yet clear, especially since $n_c := \frac{n}{h}$. In short, we do indeed seem to have an MLP on this level, although a more 'complex' one.

The query-key dot product is defined as

$$\Omega^h_{tt'}(s) = \rho_{t,t'}\left(< s_t, s_{t'} >_Q\right), \tag{3.9}$$

where $\rho_{t,t'}$ is an activation function that can also depend on t and t', the inner product is defined as

$$< s_t, s_{t'} >_Q = \frac{1}{\sqrt{n_c}}\sum_{c=1}^{n_c}\sum_{i_1,i_2=1}^{n} Q^h_{ci_1}K^h_{ci_2}s_{t,i_1}s_{t',i_2}, \tag{3.10}$$

where the subscript Q indicates that the inner-product is over 'query-key' space, n_c is the number of channels, $Q^h_{ci_1}$ the query matrix, and $K^h_{ci_2}$ the key matrix. When again looking beyond the names they can just be seen as rotating, or weighting, two (different) inputs and calculating their dot product.

In this review we simplify this even further by using

$$< s_t, s_{t'} >_Q = \sum_{i_1,i_2=1}^{n} Q^h_{i_1i_2}s_{t,i_1}s_{t',i_2}, \tag{3.11}$$

with the same reasoning as for the encoder-decoder weight E_{ij}.

The activation function $\rho_{t,t'}$ can in principle be chosen as any activation function ρ, in the paper they choose a soft-max. The unique part of this activation function $\rho_{t,t'}$ is that it can take into account the order of the tokens t, known as masking, i.e.

$$\rho_{t,t'}(x) = \Theta(t - t')\rho(x) \tag{3.12}$$

However, the dependence on the tokens t, t' can also be ignored. When ignoring the tokens, it is called a biderectional activation function, but this is of course just a regular activation function.

A use case in which the token dependent masking is proven to be useful is in the case of language in which only words prior to the current word should (in general) have an impact.

3.7 Reverse Embedding

Input: $s^{(L)}_{\delta,t,i} = s_{t,i}$
Output:

$$z^{(L)}_{\delta,t,i} = z_{t,i} = b_{t,i} + \sum_{j=1}^{n_t} w_{ji} s_{t,j} \tag{3.13}$$

where $i \in [1, ..., n']$, and n' is the size of what the vector the input has been mapped to, see tokenisation. This represents the return to the token space and is again just a linear MLP (with a transposed embedding matrix) with (potentially) non-zero bias.

4 Summary Formalism Kenway

This section provides a condensed summary of Richard Kenway's formalism.[1] The two main ingredients seem to be the generating function $\mathcal{P}$ and the the neural-tangent kernel (NTK) $\mathcal{H}_2$. If these objects satisfy some assumptions and some properties, the rest of Richards results follow, e.g. truncation of the NTK hierarchy and the NTK time-dependence.

Note that in this section the notation is used that $f^{(\ell)} = f$ and $f^{(\ell-1)} = f'$. Furthermore

4.1 The Generating Function

The generation function is defined as

$$\mathcal{P}(\eta) := \int d\phi \frac{e^{-S(\phi)+\eta\phi}}{Z}, \tag{4.1}$$

where $S(\phi)$ is the action of layer ℓ such that

$$\frac{e^{-S(\phi)}}{Z} := p(\phi|x), \tag{4.2}$$

and

$$Z = \int d\phi e^{-S(\phi)}. \tag{4.3}$$

The first assumption is that all the layers have the same quartic action (up to $\mathcal{O}(\frac{1}{n})$),

$$S(\phi) = \frac{1}{2}\phi\phi g^{-1} - \frac{1}{8n}(\phi\phi)^T v(\phi\phi), \tag{4.4}$$

where g and v are the couplings that still need to be determined. For such a quartic interaction the generating function has the form

$$\mathcal{P}(\eta) = (1 + \frac{1}{2}\eta\eta\frac{K_1}{n} + \frac{1}{8n}(\eta\eta)V(\eta\eta))e^{\frac{1}{2}\eta\eta K_0} + \mathcal{O}\left(\frac{1}{n^2}\right), \tag{4.5}$$

where K_0, K_1 and V are 'coupling' constants that still need to be determined.

The forward equation of the generating function has, independent of the action, the form

$$\begin{aligned}\mathcal{P}(\eta) &= \int d\phi e^{\phi\eta} \int d\phi' p(\phi|\phi') p(\phi'|x) \\ &= \int d\phi e^{\phi\eta} p(\phi|\partial\eta') \mathcal{P}(\eta')|_{\eta'=0}.\end{aligned} \tag{4.6}$$

If it can be shown that the generating forward equation results in a generating function of a quartic action (4.5) or that the forward equation 4.6 reduces to a shape (?),

$$\mathcal{P}(\eta) = e^{\frac{1}{2}\eta\eta G(\partial\eta')} \mathcal{P}(\eta')|_{\eta'=0}, \tag{4.7}$$

where G determines the quadratic coupling in the action, then the assumption that all the actions are the same in each layer (i.e. a quartic interaction) is self consistent. Such that the generating function can be used without a reference to the underlying network architecture, and it is just assumed that the quartic action gives a proper model of the ℓth layer.

5 Proposed Path

To be sure to understand every step of the paper I will try to replicate their results in some parts. The general aim would be to understand all the components that are different with respect to the MLP case:

1. Normalisation: $s^{(\ell)}_{\delta,t,i}$
2. Self-Attention: $\Omega^{(\ell)}_{\delta,t,i}$
3. Encoder-Decoder: E^h_{ij}
4. Skip-connection: $z^{(\text{skip})}_{\delta,t,i}$

Note that the primary attention will go to 1) that is required to enter each self-attention or MLP block and 2) the Self-Attention mechanism that is the novel component of the transformer. Note that for 3) we could choose to use a matrix E^h_{ij} directly instead of the use of an encoder-decoder, and for 4) we could set the weight of skip-connection $\mathcal{E}_i = 0$. For the time being we will concern ourselves with such a simplified transformer.

The approach will then be to first ensure all the steps in the paper are thoroughly understood with respect to at least 1) and 2). This means walking through their calculations, which will mostly not be recorded here.

The second step is up for debate. One possible way will be to use the FT formalism as developed 'in-house' to first tackle the case where the layers, besided the embedding start and end, just consist out of the normalisation and an MLP. The second step would be to have a look at layers consiting out of a normalisation and a self-attention (block). The third step would be to combine all these layers. A fourth perhaps to introduce the skip-connection, and a fifth to introduce the encoder. Another step would be to also ignore the normalisation 1), and just consider an NN consisting out of self-attention blocks starting from an embedding.

In the en

6 Walking Through the Paper

6.1 Normalisation Section 1.B Seems Wrong

This section is crucial since both the MLP and attention (block) are expressed in its result. Despite the result being confirmed by Richards derivation, the derivation provided in the paper seems to break down.

In this section of the paper the authors look at the correlation function between two elements from the normalisation layer s_i. The start from equation 1.46,

$$\mathbb{E}[\frac{1}{n}\sum_i s_i s_i] = \frac{\frac{1}{n}\sum_{i=1} z_i^2 - (\frac{1}{n}\sum_{j=1}^n z_j)^2}{\sqrt{\frac{1}{n}\sum_{j=1}^n z_j^2 - \left(\frac{1}{n}\sum_{j=1}^n z_j\right)^2 + \epsilon}} \tag{6.1}$$

Note that they have dropped the index for both the data δ and token t. When trying to walk through their steps in equation 1.46-1.52 it would have been better to keep them since it does matter that there are possibly two different elements $s_{t,i}$ and $s_{t',i}$, which they correctly reintroduce in equation 1.53.

The central claim of this section is that the expectation

$$\mathbb{E}[\hat{\Delta G}^p \hat{\nabla G}^q] = \mathcal{O}\left(\frac{1}{n}\right), \tag{6.2}$$

where $\hat{\Delta G}$ is a measure for fluctuations and $\hat{\nabla G}$ for the mean. The authors use these new variables to rewrite the above equation 6.1, and using the result 6.2 to discard the sums and get equation 1.53 (the same as 1.36).

$$F^{(\ell)}_{(\delta_1,t_1)(\delta_2,t_2)} = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n s^{(\ell)}_{\delta_1,t_1,i} s^{(\ell)}_{\delta_2,t_2,i}\right] = \frac{G^{(\ell)}_{(\delta_1,t_1)(\delta_2,t_2)}}{\sqrt{G^{(\ell)}_{(\delta_1,t_1)(\delta_1,t_1)+\epsilon}}\sqrt{G^{(\ell)}_{(\delta_2,t_2)(\delta_2,t_2)}+\epsilon}} + \mathcal{O}\left(\frac{1}{n}\right), \tag{6.3}$$

where

$$\delta_{i_1,i_2} G^{(\ell)}_{(\delta_1,t_1)(\delta_2,t_2)} = \mathbb{E}[z^{(\ell)}_{\delta_1,t_1,i_1} z^{(\ell)}_{\delta_2,t_2,i_2}] \tag{6.4}$$

which is the same variance G that is also part of the 'in-house' formalism with some extra labels. Since F is encountered time and time again in the rest of the paper the result is quite crucial. However, when walking through their proof of their result 6.2, it seems to break down. Important to note is that Richard in the notes he has shared, does also find the same expression for F so they might be correct in the rest of their paper, it is just the proof of this sub result that doesn't work.

They provide their proof in a derivation in their equation 1.52, which does break down on explicitly checking one of their assumptions in the step

$$\begin{aligned} E[\hat{\Delta G}^p \hat{\nabla G}^q] &= \sum_{r=0}^p \binom{p}{r} (-G)^{p-r} \frac{1}{n^{r+2q}} \sum_{i_1,\ldots,i_r=1}^n \sum_{j_1,\ldots,j_{2q}=1}^n \mathbb{E}[z_{i_1}^2 \ldots z_{i_r}^2 z_{j_1}\ldots z_{j_{2q}}] \\ &= \sum_{r=0}^p \binom{p}{r} (-G)^{p-r} \frac{n(n-1)\ldots(n-r-2q+1)}{n^{r+2q}} \mathbb{E}[z_1^2 \ldots z_r^2 z_{r+1}\ldots z_{r+2q}] + O(\frac{1}{n}). \end{aligned} \tag{6.5}$$

In this step they have assumed that have correctly identified the leading order term. However, upon careful examination the leading order term turns out to be something else. When just focusing on the relevant sums:

$$\sum_{i_1,\ldots,i_r=1}^n \sum_{j_1,\ldots,j_{2q}=1}^n z_{i_1}^2 \ldots z_{i_r}^2 z_{j_1}\ldots z_{j_{2q}} = \left[\binom{n}{r+2q}(z_1^2\ldots z_r^2 z_{r+1}\ldots z_{r+2q}) + \mathcal{O}\left(\binom{n}{r+2q-1}\right)\right], \tag{6.6}$$

where $\mathcal{O}\left(\binom{n}{r+2q+1}\right)$ terms are not possible given that z_i^2 terms will never recombine into $z_i z_j$, while if $z_i z_j$ recombines into z_i^2 this would mean a decrease of 1 total combinations, and where

$$\frac{1}{n^{r+2q}}\binom{n}{r+2q} = \frac{n(n-1)\ldots(n-r-2q+1)}{n^{r+2q}(r+2q)!} = \frac{1}{(r+2q)!}(1+\mathcal{O}\left(\frac{1}{n}\right)), \tag{6.7}$$

and

$$\frac{1}{n^{r+2q}}\binom{n}{r+2q-1} = \frac{n(n-1)\ldots(n-r-2q+2)}{n^{r+2q}(r+2q)!} = \mathcal{O}\left(\frac{1}{n}\right), \tag{6.8}$$

such that

$$\frac{1}{n^{r+2q}}\sum_{i_1,\dots,i_r=1}^{n}\sum_{j_1,\dots,j_{2q}=1}^{n} z_{i_1}^2\dots z_{i_r}^2 z_{j_1}\dots z_{j_{2q}} = \left[\frac{1}{(r+2q)!}(z_1^2\dots z_r^2 z_{r+1}\dots z_{r+2q}) + \mathcal{O}\left(\frac{1}{n}\right)\right], \tag{6.9}$$

The difference between the result in the paper and the one here is a factor of $\frac{1}{(r+2q)!}$, which when skipping to the final part of the derivation would interfere with their use of the binominal formula $(x+y)^p = \sum_r^p \binom{p}{r} x^r y^{p-r}$ in the resulting

$$G^p \sum_r^p \binom{p}{r} 1^r (-1)^{p-r} 0^q \frac{1}{(r+2q)!} + \mathcal{O}\left(\frac{1}{n}\right), \tag{6.10}$$

due to extra factor being dependent on r. Note that it would also be impossible to use inequalities such as $\leq$ to get rid of the $\frac{1}{(r+2q)!}$ since there is a $(-1)^{p-r}$.

Furthermore, note that in the above, z_i was treated as coming from the same token. When also accounting for possibly two different tokens the number of possible orderings in the double sum would be $\binom{n}{r}\binom{n}{2q} = \frac{n^{r+2q}}{(2q)!r!} + \mathcal{O}\big(n^{r+2q-1}\big)$. Using this would not change the breakdown of the derivation, since there would still be a factor dependent on r.

To summarize, their derivation does not seem to work such that they do not have a justification for F, eq. 6.3.

7 Novel Work: Multi-Head Self-Attention Only

This section serves as a collection of the novel work being done in this project. In the end this didn't go beyond the multi-head self attention (MHSA) block.

7.1 Deriving the Generator Function

In this subsection I assume that the network solly consists of self-attention layers, such that each layer ℓ has as input $r^{\ell-1}_{\delta,t,i}$ and as output $r^{\ell}_{\delta,t,i}$. The aim of this section will be to identify whether or not it would be possible to apply Richards formalism as a whole to the transformers. In line with the conclusion in the subsection on the generating function 4.1, I will try to identify the 'correspondence' by looking at the forward equation and see if a shape of 4.7 roles out.

We thus start with the assumption that the generating function is given by

$$\mathcal{P}(\eta) = \int d\phi \frac{e^{-S(\phi)+\eta\phi}}{Z}, \tag{7.1}$$

where S is a quartic action

$$S(\phi) = \frac{1}{2}\phi\phi g^{-1} \tag{7.2}$$

In that case the forward equation

$$\mathcal{P}(\eta) = \int d\phi e^{\phi\eta} \int d\phi' p(\phi|\phi') p(\phi'|x), \tag{7.3}$$

where we only need to find an expression for $p(\phi|\phi')$, since we can solve fo $p(\phi'|x)$ using the generating function. Explicitly writing out the first term gives

$$\begin{aligned} p(\phi|\phi') &= \int d\theta p(\theta) p(r|r',\theta) \\ &= \int dE dQ p(E) p(Q) \prod_{\delta,t,i} \delta\left(r_{\delta,t,i} - \sum_{h,t,l} \Theta(t-t')\rho(\sum_{jk} r'_{tj} Q^h_{jk} r'_{t'k}) E^h_{il} r'_{\delta t' l}\right) \\ &= \int dE dQ p(E) p(Q) \delta\left(r - \Theta(t-t')\rho(r'_t Q r'_{t'}) E r'_{t'}\right), \end{aligned} \tag{7.4}$$

where on the final line the indices have been suppressed and only the relevant ones remain. Important to notice is that one of the weights, Q, is in the the activation function ρ. This situation is exactly what is avoided by using the generating function in the MLP case. Therefore a straight forward replication, meaning the use of an action that is only dependent on the the previous layer via ϕ' doesn't seem applicable.

Note that this difficulty is obscured in the notation of 3.8 by 'hiding' the activation function in the matrix. Perhaps it is therefore a slightly over simplified notation, or it could point at some simplification that I'm perhaps overlooking when doing the calculations.

A possible hope might be that instead of straight away trying to integrate out all the weights, it might somehow turn out to be more doable when evaluating the term inside the activation function, $q^h_{\delta,t,t'} = \sum_{jk} r'_{\delta tj} Q^h_{jk} r'_{\delta t'k}$, separately. To that end we could decompose the transition matrix from r' to r as

$$p(r|r') = \int dup(q|r')p(r|q,r'), \tag{7.5}$$

and evaluate the two probabilities independently at first. Doing so results in

$$\begin{aligned} p(q|r') &= \int dQp(Q) \prod_{h\delta t_1 t_2} \delta(q^h_{\delta t_1 t_2} - \sum_{ij} r'_{\delta t_1 i} Q^h_{ij} r'_{\delta t_2 j}) \\ &= \frac{1}{((2\pi C_q)^{2n_t+d}|R|)^{\frac{n_h}{2}}} e^{-\frac{1}{2c_q} qR^{-1}q}, \end{aligned} \tag{7.6}$$

where c_q is the variance of the weights Q, $qR^{-1}q = \sum_{\delta_1\delta_2,t_1t_2t_3t_4,h} q^h_{\delta_1 t_1 t_2} R^{-1}_{\delta_1\delta_2 t_1 t_2 t_3 t_4} q^h_{\delta_2 t_3 t_4}$ and

$$R_{\delta_1\delta_2 t_1 t_2 t_3 t_4}(r') = \sum_{i,j} r'_{\delta_1 t_1 i} r'_{\delta_1 t_2 j} r'_{\delta_2 t_3 i} r'_{\delta_2 t_4 j}. \tag{7.7}$$

For a similar problem with all derivation steps see A.1. In addition,

$$\begin{aligned} p(r|q,r') &= \int dEp(E) \prod_{\delta t_1 i} \delta(r_{\delta t_1 i} - \sum_{h,t_2,j} \rho^h_{\delta t_1 t_2} E^h_{ij} r'_{\delta t_2 j}) \\ &= \frac{1}{((2\pi c_e)^{nt+d}|A|)^{\frac{n}{2}}} e^{-\frac{1}{2c_e} rA^{-1}r}, . \end{aligned} \tag{7.8}$$

On the first line $\rho^h_{\delta t_1 t_2} = \Theta(t_1 - t_2)\rho(q^{h_1}_{\delta t_1 t_2})$. On the second line c_e is the variance of the weights E, $rA^{-1}r = \sum_{\delta_1,\delta_2,t_1,t_2,i} r_{\delta_1 t_1 i} A^{-1}_{\delta_1\delta_2 t_1 t_2} r_{\delta_2 t_2 i}$ and

$$A_{\delta_1\delta_2 t_1 t_2}(q,r') = \sum_{j,t_3,t_4,h} \Theta(t_1 - t_3)\rho(q^h_{\delta_1 t_1 t_3}) r'_{\delta_1 t_3 j} \Theta(t_2 - t_4)\rho(q^h_{\delta_2 t_2 t_4}) r'_{\delta_2 t_4 j}, \tag{7.9}$$

Where J is a dxd matrix of ones. Note that $R_{t_1t_2t_3t_4}$ is symmetric inside and when switching the pairs t_1t_2 and t_3t_4. In addition, note that $A_{\delta t_1 t_2 i}$ is symmetric in t_1 and t_2. These just confirm that they are proper variances, but as far as I can tell do not offer any further exploitations.

Combining the above gives

$$\int (\prod_{h,t_1,t_2} dq^h_{t1t2}) \frac{1}{((2\pi C_q)^{2n_t}|R|)^{\frac{h}{2}}} e^{-\frac{1}{2c_q} qR^{-1}q} \frac{1}{((2\pi c_e)^{nt+d}|A|)^{\frac{n}{2}}} e^{-\frac{1}{2c_e} rA^{-1}r}. \tag{7.10}$$

This again looks like it needs a complete the square to find its solution with respect to u. However, to do so we would still run into the anticipated issue where it is unclear what to do with unknown activation function ρ that encapsulates u, i.e. we might not be able to do complete the square if $\rho(u) \neq au + b$, which it almost never is. On top of this, this expression is now hiding in the inverse of the matrix A. In short the anticipated break down does seem to happen.

The ambiguity of what to do with the weight Q in the activation function does indeed remain however, such that another approach seems needed.

This does highlight why the approach works so well in the MLP: the layers are separated via an activation function. However in this self attention block there are activation functions inside each layer, i.e. weights and the previous result a used both inside and outside the activation function.[1]

A solution turned out to be a direct use of generating functions. In the following I will suppress all index notation since it doesn't seem relevant for this derivation. Begin with a more general definition of the generation function

$$\mathcal{P}_y(\eta) = \int dy p(y|x) e^{\eta y}, \tag{7.11}$$

where y is a dummy 'field' and x is the starting dataset.

Then we start by defining a generation function for $q = rQr$,

$$\begin{aligned}
\mathcal{P}_q(\eta) &= \int dq p(q|x) e^{\eta q} \\
&= \int dq e^{\eta q} \int dr' p(q|r') p(r'|x) \\
&= \int dq e^{\eta q} \int dr' p(q|r') p(r'|x) e^{\eta' r'}|_{\eta'=0} \\
&= \int dq e^{\eta q} p(q|\partial') \int dr' p(r'|x) e^{\eta' r'}|_{\eta'=0} \\
&= \int dq e^{\eta q} p(q|\partial\eta') \mathcal{P}_{r'}(\eta')|_{\eta'=0} \\
&= \int dq e^{\eta q} \frac{1}{((2\pi C_q)^{2n_t}|R|)^{\frac{h}{2}}} e^{-\frac{1}{2c_q} q R^{-1} (\partial\eta') q} \mathcal{P}_{r'}(\eta')|_{\eta'=0} \\
&= e^{\frac{c_q}{2} \eta R(\partial\eta')\eta} \mathcal{P}_{r'}(\eta')|_{\eta'=0},
\end{aligned} \tag{7.12}$$

where I have used line by line: 1) standard probability theory, 2) inserted a factor of 1, 3) used 'the derivative trick', 4) the definition of the generator (eq. 7.11), 5) the result of the transition matrix from r' to q from earlier in equation 7.6 and 6) finally performed completing the square and the Gaussian integral. The final result is thus

$$\mathcal{P}_q(\eta) = e^{\frac{c_q}{2} \eta\eta R(\partial\eta')} \mathcal{P}_{r'}(\eta')|_{\eta'=0}. \tag{7.13}$$

We continue with the generating function for r,

$$\begin{aligned}
\mathcal{P}_r(\eta) &= \int dr e^{\eta r} p(r|x) \\
&= \int dr e^{\eta r} \int dq dr' p(r|q,r') p(q|r') p(r'|x) \\
&= \int dr e^{\eta r} \int dq dr' p(r|q,r') p(q|r') p(r'|x) e^{\gamma q + \eta' r'}|_{\gamma,\eta'=0} \\
&= \int dr e^{\eta r} p(r|\partial\gamma, \partial\eta') \int dq p(r'|\partial\eta') e^{\gamma q} \int dr' p(r|x) e^{\eta' r'}|_{\gamma,\eta'=0} \\
&= \int dr e^{\eta r} p(r|\partial\gamma, \partial\eta') e^{\frac{c_q}{2} \gamma\gamma R(\partial\eta')} \mathcal{P}_{r'}(\eta')|_{\gamma,\eta'=0} \\
&= \int dr e^{\eta r} \frac{1}{((2_e)^{nt}|A|)^{\frac{nd}{2}}} e^{-\frac{1}{2c_e} r A^{-1} (\partial\gamma,\partial\eta') r} e^{\frac{c_q}{2} \gamma\gamma R(\partial\eta')} \mathcal{P}_{r'}(\eta')|_{\gamma,\eta'=0} \\
&= e^{\frac{c_e}{2} \eta\eta A(\partial\gamma,\partial\eta')} e^{\frac{c_q}{2} \gamma\gamma R(\partial\eta')} \mathcal{P}_{r'}(\eta')|_{\gamma,\eta'=0},
\end{aligned} \tag{7.14}$$

[1] As a side note, the MLP definition in the transformer might seem to run into a similar issue at first since the output $r = X\rho(Wr')$ where the indices have been suppressed. However, upon closer examination it is just a shift of what is considered the output of the previous layer, i.e. redefine $r' = Wr'$ and we're back in the regular MLP. The difference would just be combination of weights in the previous layer as opposed to just one. This again the encoder-decoder effect of increasing the size and decreasing it again. I'm curious if the addition of all these extra weights adds some information or if this is just a redundancy. And ofcourse the question again, how to show this.

where I have used line by line: 1) again what standard probability theory, 2) inserted a factor of 1, 3) used the derivatives trick, 4) recognized that we have encountered an expression from the generating function $\mathcal{P}_q$ derivation and put in the resulting 7.13, 5) used the explicit form of the transition matrix r', q to r as found in equation 7.8 and 6) finally performed again a Gaussian integral after completing the square.

In short the result is

$$\mathcal{P}_r(\eta) = e^{\frac{c_e}{2}\eta\eta A(\partial\gamma,\partial\eta')} e^{\frac{c_q}{2}\gamma\gamma R(\partial\eta')} \mathcal{P}_{r'}(\eta')|_{\gamma,\eta'=0}. \tag{7.15}$$

When continuing with this simplified, index suppressed form we can use the assumption that each layer of the self-attention block r is drawing from a gaussian distribution, i.e.

$$p(r|x) = \frac{e^{-\frac{1}{2}G^{-1}r^2}}{\sqrt{2\pi|G|}}, \tag{7.16}$$

where G is a covariant matrix. Using this the generating function for r can be written as

$$\mathcal{P}_r(\eta) = \int dr \frac{e^{-\frac{1}{2}G^{-1}r^2+\eta r}}{\sqrt{2\pi|G|}} = e^{\frac{1}{2}G\eta^2} = 1 + G\eta\eta + \mathcal{O}\big(G^2\big), \tag{7.17}$$

such that if we plug this in into the forward of equation 7.15 and only consider terms proportional to η^2 we get

$$\begin{aligned}
G &= c_e A(\partial\gamma,\partial\eta') e^{\frac{c_q}{2}\gamma\gamma R(\partial\eta')} e^{\frac{1}{2}G'\eta'\eta'}|_{\gamma,\eta'=0} \\
&= c_e \rho(\partial\gamma_1)\partial\eta'_1\rho(\partial\gamma_2)\partial\eta'_2 e^{\frac{c_q}{2}\gamma\gamma R(\partial\eta')} e^{\frac{1}{2}G'\eta'\eta'}|_{\gamma,\eta'=0}, \\
&= c_e \rho(\partial\gamma_1)\rho(\partial\gamma_2) e^{\frac{c_q}{2}\gamma\gamma R(\partial\eta')}\partial\eta'_1\partial\eta'_2 e^{\frac{1}{2}G'\eta'\eta'}|_{\gamma,\eta'=0}, \\
&= c_e \rho(\partial\gamma_1)\rho(\partial\gamma_2) e^{\frac{c_q}{2}\gamma\gamma R(\partial\eta')} 2(G'_{12} + G'_1\eta' + G'_2\eta') e^{\frac{1}{2}G'\eta'\eta'}|_{\gamma,\eta'=0}, \\
&= 2c_e G'_{12} \rho(\partial\gamma_1)\rho(\partial\gamma_2) e^{\frac{c_q}{2}\gamma\gamma R(\partial\eta')} e^{\frac{1}{2}G'\eta'\eta'}|_{\gamma,\eta'=0}.
\end{aligned} \tag{7.18}$$

In the first line we have inserted the definition of A (eq. 7.8) with suppressed indices (such that also the step function can be ignored) and we have introduced a short hand numerical index notation to (pictorially) track the relevant contractions, i.e. with respect to objects in the exponents. To evaluate the expression further we need to take the derivatives. In the second step we move the derivatives with respect to η' to the right, which we can do since R only consists out of partial derivatives (eq. 7.6), and partial derivatives commute (for sufficiently smooth functions). In the third step we take the derivatives and use that G, as a variance, must be symmetric. In the final step we use that R consists out of even derivatives with respect to η' such that a power series expansion in the terms of $e^{\frac{c_q}{2}\gamma\gamma R(\partial\eta')}$ with respect to $\eta' e^{G'\eta'\eta'}$ would always result in a remaining η' term. Since we will set $\eta' = 0$ at the end, we can do so now for these terms since they would drop out in the end anyway.

At this point we're stuck however at two fronts. To see this we can use the pictorial index notation to make R more explicit in 7.18 to get

$$G = 2c_e G'_{12} \rho(\partial\gamma_1)\rho(\partial\gamma_2) e^{\frac{c_q}{2}\gamma\gamma\partial\eta'_3\partial\eta'_4\partial\eta'_5\partial\eta'_6} e^{\frac{1}{2}G'\eta'\eta'}|_{\gamma,\eta'=0}. \tag{7.19}$$

The first point in equation 7.19 where we're sort of stuck is trying to take the derivatives with respect to η'. This would require an expansion of e. I have written this out and taken the derivatives explicitly for a 'k-th' term f_k such that there is a forward equation

$$f_k e^{G\eta\eta} = \partial\eta_4\partial\eta_3\partial\eta_2\partial\eta_1 f_{k-1} e^{G\eta\eta} = [g(G,\partial\eta^4 f_{k-1},\partial\eta^2 f_{k-1}) + \eta^2 h(G,\partial\eta^2 f_{k-1}) + \eta^4 l(G) + \mathcal{O}\Big(\eta^{(\text{odd})}\Big)] e^{G\eta\eta}, \tag{7.20}$$

where the functions g, h and l could be provided explicitly, but they important part is their dependence on derivatives of f_{k-1} to η, for which the indices have been suppressed. When setting $\eta = 0$ at the very end all the terms proportional to *eta* will be 0. Such that only function g would remain. However, since g depends on derivatives we would need to perform them before setting η to 0. Since the derivatives in g are only even, all the odd terms would in the end drop out. The terms $\partial\eta^4 f_{k-1}$ would resolve nicely, however the terms

$\partial\eta^2 f_{k-1}$ would again depend on f_{k-2}, and so for until $f_0 = 1$. As a result each term f_k has a non-trivial summation of combinations such that it becomes highly non-trivial to resume the terms.

Another way to see this is by 'using' Wicks theorem and noting that we get multiples of four point correlators, i.e. 4, 8, 12, etc., such that we know we get sums of combinations of pairs. In addition we also know that these pairs will be combinations of (powers of) the 16 possible combinations that can be made with the 4 different types of indices. However, this knowledge doesn't seem to provide an edge on the problem as far as I can estimate. This because the first term, f_1 will only contain a subset of the possible combinations (excluding for instance G'_{ii}), such that a possible resummation would at the very least involve a combination of two functions. Google searches also do not result in any known solutions, but I do know that for some reason my google searches algorithm is not favorable for very specific math questions. So feel free to double check.

In short, we cannot take the derivatives with respect to η' such that we need to revert back to the Gaussian distribution with respect to η'. For the steps see below.

The second point where we're (seemingly) stuck in equation 7.19 is at taking the derivatives with respect to γ. To do this we need the expansion of the activation function ρ. There are four 'types' of activation functions: ambiguous how to expand (RELU), uncommon (linear), unfamiliar (softplus, SWISH,GELU) or it has only odd terms when expanded around 'x=0' (tanh, erf, sigmoid). Analogous to the previous argument, odd derivatives with respect to γ in the even exponent will vanish in the end when setting $\gamma = 0$. Alternatively you could choose to expand both activation functions at the same time, which would make allot of terms even. However, how to resum this is far from trivial. As a result we cannot get R 'out of the exponent'. The reason this is not an apparent issue in the MLP case is that ρ is taken into the expectation value, i.e. for a general form

$$\rho(\partial\eta_1)\rho(\partial\eta_2)e^{\frac{1}{2}G\eta\eta}|_{\eta=0} = \int d\phi\rho(\phi)\rho(\phi)\frac{e^{-\frac{1}{2}G^{-1}\phi\phi}}{\sqrt{2\pi|G|}}. \tag{7.21}$$

Which is just complete the square, Gaussian integration and the derivatives trick in reverse order. Note that the derivatives trick here works because there is only one term to which the derivative is taken, i.e. $e^{\phi\eta}$.

Interestingly enough this encountered 'issue' seems to align with 'industry standards', because with respect to self-attention blocks apparently the GELU is quite a common activation function. The RELU-like activation functions all have a linear term and even powers when expanding around 0. Although I still see issues due to this linear term remaining proportional to γ and a combination of the two activation functions would maybe again result in an impossible to resum summation, it is something to have a look at.

To continue however with the trivial case of a linear activation function. In that case equation **??** would simplify to

$$\begin{aligned} G &= 2c_e G'_{12}\partial\gamma_1\partial\gamma_2 e^{\frac{c_q}{2}\gamma\gamma\partial\eta'_3\partial\eta'_4\partial\eta'_5\partial\eta'_6}e^{\frac{1}{2}G'\eta'\eta'}|_{\gamma,\eta'=0} \\ &= c_q c_e G'_{12}(\partial\eta'_3\partial\eta'_4\partial\eta'_5\partial\eta'_6)_{12}e^{\frac{1}{2}G'\eta'\eta'}|_{\eta'=0} \\ &= 4c_q c_e G'_{12}(G'_{56}G'_{34} + G'_{53}G'_{46} + G'_{54}G'_{63})_{12}, \end{aligned} \tag{7.22}$$

where we have clearly run into the limitation of the index notation. Note that in this case, we have been able to resolve all the derivatives and don't need to revert to Gaussian integrals.

In the case where we cannot expand the activation function, we would also not be able to resolve the derivatives with respect to γ. As a result we would get

$$\begin{aligned} G &= 2c_e G'_{12}\int d\phi' d\phi \frac{1}{2\pi\sqrt{|G'R(\phi')|}}\rho(\phi_1)\rho(\phi_2)e^{-\frac{c_q}{2}R^{-1}(\phi')\phi^2 + G'^{-1}\phi'^2} \\ &= 2c_e G'_{12} < \rho_1\rho_2 >_{G'R}, \end{aligned} \tag{7.23}$$

where we have defined an 'expectation value'[2]

$$< \rho_1\rho_2 >_{G'R} = \int d\phi' d\phi \frac{1}{2\pi\sqrt{|G'R(\phi')|}}\rho(\phi_1)\rho(\phi_2)e^{-\frac{c_q}{2}R^{-1}(\phi')\phi^2 + G'^{-1}\phi'^2}. \tag{7.24}$$

[2]Maybe this notation would obscure the actual complications of the a bit too much?

In any case, it is clear that this also has to be performed with proper index tracking. A preliminary thought however is that we do significantly diverge from the paper (equation 1.65) in that we do always get a forward equation (seemingly) proportional to the previous variance. As is also remarked in the deep learning book [3] this becomes unstable relatively quickly when the layers increase, i.e.

$$G^{(\ell)} = c^{\ell} E_{\ell} G^{(0)}, \tag{7.25}$$

where c is a constant and E_ℓ is the 'expectation value'. Such tat in the case $c \neq 1$, G could become trivially small or large when getting deeper into the network. However, do note that the paper also uses a normalisation layer so that perhaps this phenomenon is avoided by the normalisation layer. In addition, it is not a simple c. In any case, it is a diversion from the MLP case where this dependence wasn't present in the first place. Perhaps a more minimalist transformer architecture would hence only need normalisaiton for the self attention block? In any case, after the steps are performed with the index notation it would be good to include the normalisation.

Finding the forward equation with the proper indices for an unspecified activation function ρ (eq. 7.23) gives

$$G = G_{\delta_1\delta_2 t_1 t_2 ij} = c_e \delta_{ij} \sum_{t_3 t_4, h, k} G'_{\delta_1\delta_2 t_3 t_4 kk} < \rho^h_{\delta_1 t_1 t_3} \rho^h_{\delta_2 t_2 t_4} >_{G'R}, \tag{7.26}$$

where $\rho^h_{\delta t_1 t_2} = \Theta(t_1 - t_2)\rho(q^h_{\delta t_1 t_2})$. Note that to have variance G independent of the number of data points, tokens, head or neurons a division by respectively n_t^2, n_h, n is needed. Suggesting that in this case $c_e \propto \frac{1}{n}$ when just focusing on the number of neurons. When taking into account all possible invariances $c_e \propto \frac{1}{n n_h}$ and $\Omega_{tt'} \propto \frac{1}{t}\Theta(t - t')$.

In the case the activation function is linear the equation simplifies to

$$\begin{aligned} G = G_{\delta_1\delta_2 t_1 t_2 ij} = 2c_q c_e \delta_{ij} \sum_{t_3 t_4, h, klm} & \Theta(t_1 - t_3)\Theta(t_2 - t_4) G'_{\delta_1\delta_2 t_3 t_4 kk} \\ & \left[G'_{\delta_1\delta_1 t_1 t_3 lm} G'_{\delta_2\delta_2 t_2 t_4 lm} + G'_{\delta_1\delta_2 t_1 t_2 ll} G'_{\delta_1\delta_2 t_3 t_4 mm} + G'_{\delta_1\delta_2 t_1 t_4 lm} G'_{\delta_1\delta_2 t_3 t_2 ml} \right]. \end{aligned} \tag{7.27}$$

Note that to have variance G independent of the number of data points, tokens, head or neurons a division by respectively n_t^2, n_h, n^3 is needed. Suggesting that in this case $c_q \propto \frac{1}{n^2}$ when just focusing on the number of neurons. When taking into account all possible invariances $c_q \propto \frac{1}{n^2}$. Note that in the case where $c_q \propto \frac{1}{n^2}$ the first and third G' in the brackets will drop out, since they will first have to 'resolve' a delta function due to the mixed l and m indices at the end. Such that for those terms they actually scale with $\mathcal{O}(n)$ instead of $\mathcal{O}\big(n^2\big)$ like the second term in the brackets. Hence when focusing on the leading order contribution

$$\begin{aligned} G = G_{\delta_1\delta_2 t_1 t_2 ij} &= 2c_q c_e \delta_{ij} \sum_{t_3 t_4, h, klm} \Theta(t_1 - t_3)\Theta(t_2 - t_4) G'_{\delta_1\delta_2 t_3 t_4 kk} G'_{\delta_1\delta_2 t_1 t_2 ll} G'_{\delta_1\delta_2 t_3 t_4 mm} \\ &= 2c_q c_e \delta_{ij} n^3 n_h G'_{\delta_1\delta_2 t_1 t_2 ll} \sum_{t_3 t_4} \Theta(t_1 - t_3)\Theta(t_2 - t_4) G'_{\delta_1\delta_2 t_3 t_4 kk} G'_{\delta_1\delta_2 t_3 t_4 mm}. \end{aligned} \tag{7.28}$$

Finally note that this is super sensitive to the variance G since there is a third power dependence on the variance of the previous layer. To complete the forward equation we need to have a $G^{(1)}$. Assuming that the input data vector x_i enters the neural network via a linear term such that

$$p(r^{(1)}|x) = \int dW p(W) \prod_{\delta,t,i} (r^{(1)}_{\delta t i} - \sum_j W_{ij} x_{\delta t j}), \tag{7.29}$$

where $i \in \{1, .., n\}$ and $j \in \{1, .., n_{in}\}$, then the variance of the first layer

$$G^{(1)} = \frac{c_W}{2} G^{(1)}_{\delta_1\delta_2 t_1 t_2 ij} = \frac{c_W}{2} \delta_{ij} \sum_k x_{\delta_1 t_1 k} x_{\delta_2 t_2 k}. \tag{7.30}$$

Note that to be independent from the input dimension, $c_w \propto \frac{1}{n_{in}}$.

As a final remark, as noticed at the start it is impossible to get a Gaussian expression straight away from the previous input. This of course doesn't argue in favour for the assumption the self-attention block as being

properly described by a gaussian. This is again seen when trying to find the expresion for the variance of the first layer in case of a NN of purely self-attention blocks. The usual route, in case of a linear activation function, start from

$$p(r^{(1)}|x) = \int dEdQp(E)p(Q)\prod_{\delta t_1 i}\delta(r^{(1)}_{\delta t_1 i} - \sum_{h,t_2,jkl}\Theta(t_1-t_2)x_{\delta t_1 k}Q^h_{kl}x_{\delta t_2 l}E^h_{ij}x_{\delta t_2 j}), \tag{7.31}$$

when first performing complete the square with respect to Q^h_{kl} and the Q dependence can integrated out. However, in this completion of the square we get a squared $\sum_{ij}E^h_{ij}$ such that when collecting by E^h_{ij} terms we have

$$-\frac{1}{c_e}\sum_{ij}E^{h^2}_{ij} - \sum_{ijmn}E^h_{ij}B_{ijmn}(\Lambda,x)E^h_{mn} = -\sum_{ijmn}E^h_{ij}(\frac{1}{c_e}\delta_{im}\delta_{jm} + B_{ijmn}(\Lambda,x))E^h_{mn}, \tag{7.32}$$

where B_{ijmn} depends quadratically on Λ and on sums of x. As far as I can tell this will only allow for solving when the whole middle part can be diagonalized. However in doing so, the Λ^2 inside B_{ijmn} will be taken out of the exponential and appears in front of the exponent, i.e.

$$p(r^{(1)}|x) = c\int d\Lambda\frac{1}{|B(\Lambda,x)|}e^{i\Lambda x}, \tag{7.33}$$

where c is some constant and the index notation has been suppressed. This resulting form I would not now know how to analyse this further, at least the complete the square trick seems to be at its end here.

8 Numerical Experiment Method

For the numerical experiment of the multi-head self-attention with a linear activation function we implement the variance forward equations (FE)7.28 with the first layer variance given by 7.30. This is for the neural network (NN) defined as

$$\begin{aligned} o^{th}\text{-layer} : r_{\delta ti} &= \sum_{j}^{n_{in}} W_{ij}x_{\delta tj}, \\ \ell^{th}\text{-layer} : r_{\delta ti} &= \sum_{h=1}^{n_h}\sum_{t'=1}^{n_t}\Omega^h_{\delta tt'}(r')\sum_{j=1}^{n}E^h_{ij}r'_{\delta t',j}, \end{aligned} \tag{8.1}$$

where $r'_{\delta ti} \in \mathbb{R}^{d\times n_t\times n}$ is the output of the previous layer, $x_{\delta ti} \in \mathbb{R}^{d\times n_t\times n_{in}}$ is the tokenised data input form $\delta \in \mathcal{D}$ with $|\mathcal{D}| = d$ and

$$\Omega^h_{\delta tt'} = \Theta(t-t')\sum_{ij}r'_{\delta ti}Q^h_{ij}r'_{\delta t'j}, \tag{8.2}$$

with Θ as the heaviside step function. Furthermore the weights W, Q, E are (per layer) initialized from a gaussian distribution with variance c_w, c_q and c_e respectively. Both the neural network (NN) and the forward equations are implemented in python [4].

The experiment aims to determine under which conditions the assumption that S is a quartic interaction (eq. 7.2). To study if this is the case we will compare the correlations functions as a result of the NN and FE since the FE have been derived using this assumption. For the NN the corelation functions are determined over the ensemble of networks. Such that for layer ℓ the corelation functions are determined as

$$< r^{(\ell)^n}_{\delta_1 t_1 i}r^{(\ell)^m}_{\delta_2 t_2 j} >= \frac{1}{N_{net}}\sum_{N_i}^{N_{net}} r^{(\ell,N_i)^n}_{\delta_1 t_1 i}r^{(\ell,N_i)^m}_{\delta_2 t_2 j}, \tag{8.3}$$

where N_{net} is the number of networks, N_i is the i^{th} network, n and m in this case are integers that act as powers and $r^{(\ell)}_{\delta ti}$ is 'measured' in a realised network. For the FE the correlation functions are calculated via the generating function

$$< r^{(\ell)^n}_{\delta_1 t_1 i}r^{(\ell)^m}_{\delta_2 t_2 j} >= \partial\eta^{(\ell)^n}_{\delta_1 t_1 i}\partial\eta^{(\ell)^m}_{\delta_2 t_2 j}e^{\frac{1}{2}G\eta\eta}|_{\eta=0}, \tag{8.4}$$

where the variance G follows from the FE (eqs. 7.28 and 7.30).

Given the FE we know what the correlations should be. Such that we will determine the correlation functions up to the fourth order with the expectations:

- $< r^{(\ell)}_{\delta ti} >= 0$
- $< r^{(\ell)^2}_{\delta ti} >= G^{(\ell)}_{\delta\delta ttii}$
- $< r^{(\ell)}_{\delta_1 t_1 i} r^{(\ell)}_{\delta_2 t_2 j} >= \delta_{ij} G^{(\ell)}_{\delta_1\delta_2 t_1 t_2 ij}$
- $< r^{(\ell)^3}_{\delta ti} >= 0$
- Also look a the fourth order?

A possible extension could be to also look at summed out indices, e.g.

$$< r^2_{ti} >= \frac{1}{d} \sum_{\delta} < r^2_{\delta ti} > . \tag{8.5}$$

According to existing literature the relevant parameters to vary are the width n and the depth ℓ where large n, i.e. wide, and small ℓ, i.e. shallow, networks are supposed to be Gaussian. In addition, to make networks independent from the number of neurons, heads or tokens we could also scale our variances. Following the result at the end of the previous section 7.1 this would mean

$$\begin{aligned} c_w &\propto \frac{1}{n_{in}} \\ c_q &\propto \frac{1}{n^2} \\ c_e &\propto \frac{1}{n n_h}. \\ \Omega_{tt'} &\propto \frac{1}{t}\Theta(t-t') \end{aligned} \tag{8.6}$$

It is important to note that in the case of large n (or n_t) the width will decrease very quickly and you might run into numerically challenging regions. This is perhaps a good reason to split up the weight c_q into two different weights, since then each weight could be drawn from a distribution with variance $c \propto \frac{1}{n}$. In addition, note that it might make more sense to normalise Ω from 8.2 with respect to the number of entries in the row (which is 1 in the first row, and n_t in the final row given we use a heaviside step function).

Finally, to begin I will just calculate the expectations values for a given $r_{\delta ti}$. To be consistent I will choose to evaluate r_{000}. Before delving into 'uncharted' territory I will first have a look at known cases. The case in which we are expecting to get a 'known' results is that of a small and deep network, i.e. n is small and ℓ is large. In this limit it is expected that the NN is not described by a gaussian. Do also note, that we introduce the self attention block at layer 2, so we will need at least 2 layers to see the effect of the self attention block. Therefore, as another first sanity check we should compare to the better known linear MLP case. For the linear MLP the forward equation is given by

$$G_{\delta_1\delta_2 ij} = \frac{C_w}{2}\delta_{ij} \sum_k G'_{\delta_1\delta_2 kk} = \frac{C_w n}{2}\delta_{ij} G'_{\delta_1\delta_2}, \tag{8.7}$$

where (obviously) $c_w \propto \frac{1}{n}$ and (less obvious) the first layer expression is equal to the self-attention NN from eq. 7.30 without the token-t index.

9 Numerical Experiment Results

This section gives a summary of the numerical results. There are three parameters that are varied, the width n, the number of tokens n_t, and the the number of heads n_h. In addition, when varying the width n a

comparison (and sanity check) against the MLP is made. The full results can also be generated using the code on github [4].

As a summary: Note that the order of performing the comparisons was:

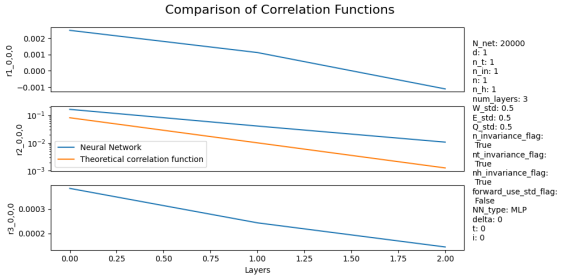
- Comparing to an MLP for varying widths n, interesting: the MLP is better described by the theory when using the std instead of variance. This is not true for the MHSA.
- Varying the number of headsn_h: Nothing to note.
- Varying the width n more elaborate: The initial observations when comparing to the MLP remain true. Increasing n does increase the variance for the MHSA from almost 0 to near MLP levels. For the MLP the order of magnitude remains almost the same. Perhaps even higher n would be interesting.
- Varying the number of tokens n_t: The conlusion seems to be that the nuber of tokens doesn't really seem to impact the (non-)gaussianity. In a way this is suprprising, given that I would expect it to be a source of non-gaussianity. However, on the otherhand, the MHSA doesn't show Gaussian behaviour when there is just one token. Sucht that the weighting, by virtue of using the previous layer output already seems to impact gaussianity, independent of the number of tokens.

Finally, do note that this hyperparameter search could be done more widely (and better) with hyperopt [5] if a measure (such as variance agreement) is provided.

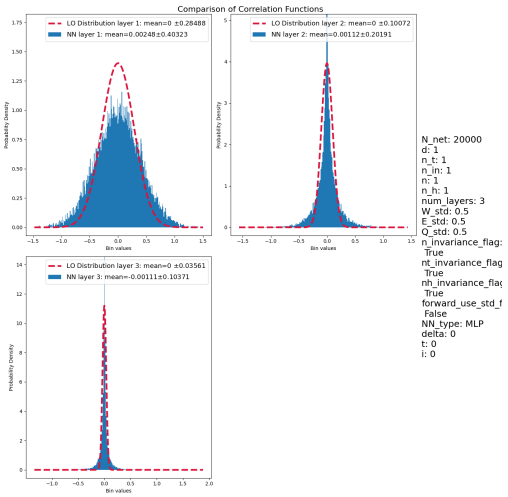
9.1 Varying the Width

This section looks at varying the width n. Since the MLP can also vary in the width n, we analyse both the MLP and MHSA results. The section contains a selection from the figures from the neural network (NN) and forward equation (FE), the full selection can be generated using the code on github [4]. I compared for $n = [1, 10, 20, 50, 100, 200, 500]$ and calculated the forward equation (i) with the variance and with the standard deviation (ii). Do note, that the standard deviation should be worse since the theory uses the variance c.

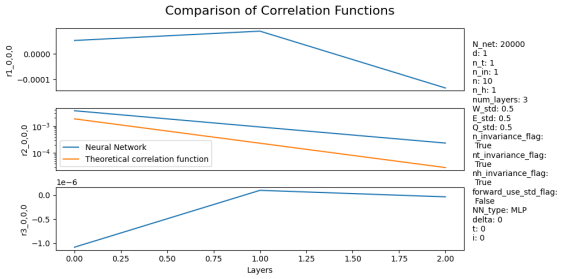
MLP



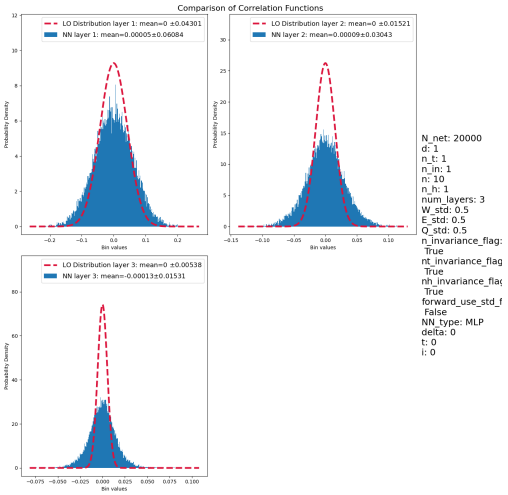
(a) $n = 1$ correlation functions



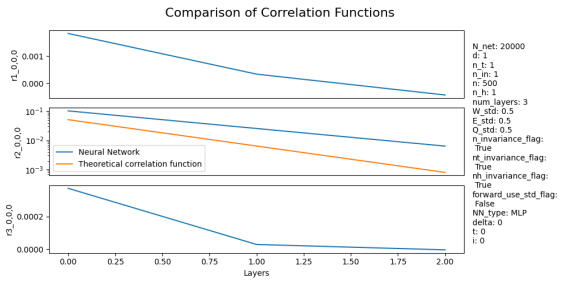
(b) $n = 1$ histogram



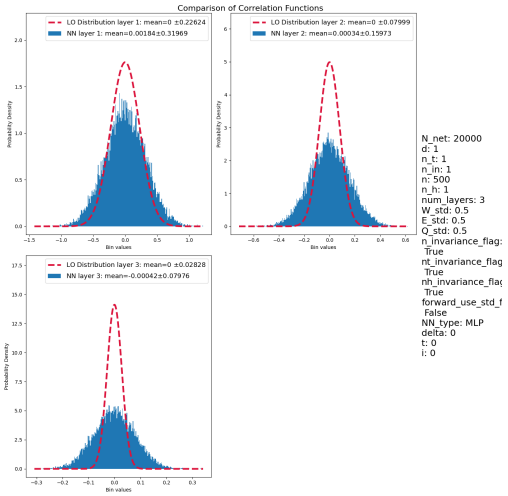
(c) $n = 10$ correlation functions



(d) $n = 10$ histogram

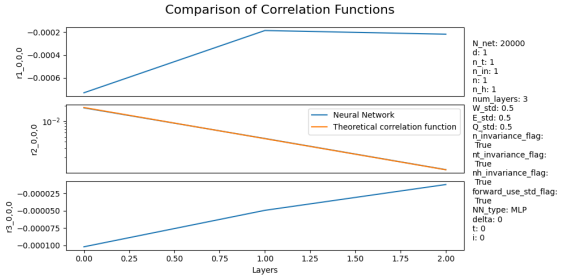


(e) $n = 500$ correlation functions

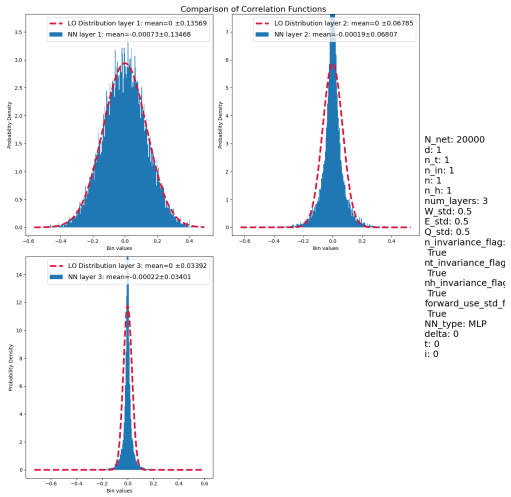


(f) $n = 500$ histogram

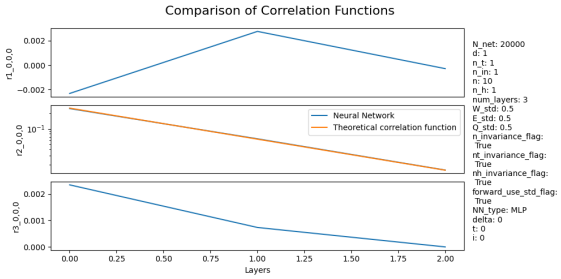
Figure 2: Correlation functions and histograms for the MLP per width n where the FE uses the variance



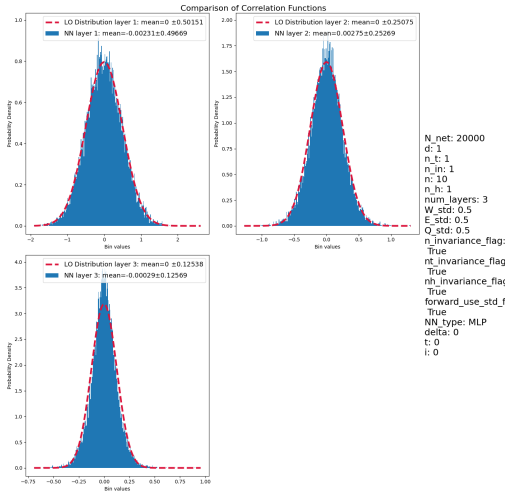
(a) $n = 1$ correlation functions



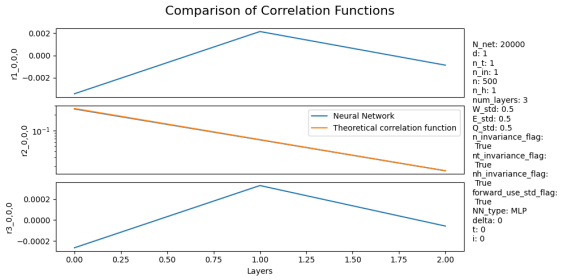
(b) $n = 1$ histogram



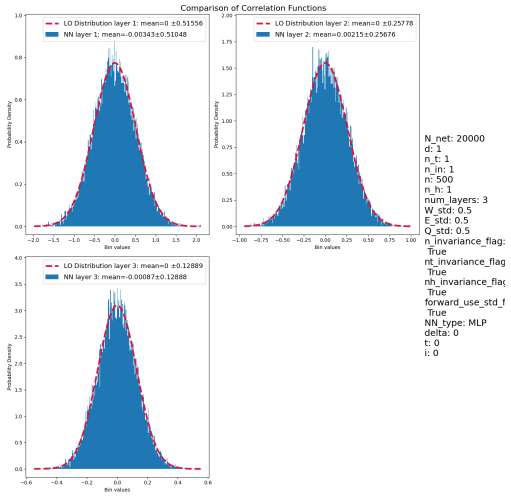
(c) $n = 10$ correlation functions



(d) $n = 10$ histogram



(e) $n = 500$ correlation functions



(f) $n = 500$ histogram

Figure 3: Correlation functions and histograms for the MLP per width n where the FE (erronously) uses the standard deviation

1 Neuron Edge Case Expectation: Non-Gaussian, since as far away as possible from the infinite width limit.

i. The edge case doesn't seem to be working very well, since the picture remains the same: the variance of the NN is always higher than the FE. However, when looking at the visual comparison, the thinner NN seems more spiked than the later wide NNs, e.g., vs $n = 500$.

ii. Although the variance always aligns well, as can be seen in the correlation plot, the visual histogram plots improve with increasing depth when n increases. In the edge case, the second and third layers look more like a delta function. In this erroneous case, we thus see the expectation of the infinite width coming to better expression.

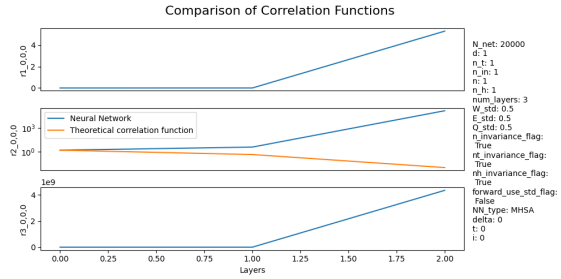
Increasing the Number of Neurons per Layer Expectation: The closer it gets to the infinite width limit, the closer it should get to a Gaussian.

i. The variance stays the same, i.e., the NN always has a larger variance than the FE. However, there is a change in the deeper layers from a spiked form where the FE peak is too low, to a broader NN distribution where the FE peak is too high. This switch happens when leaving the $n = 1$ regime.

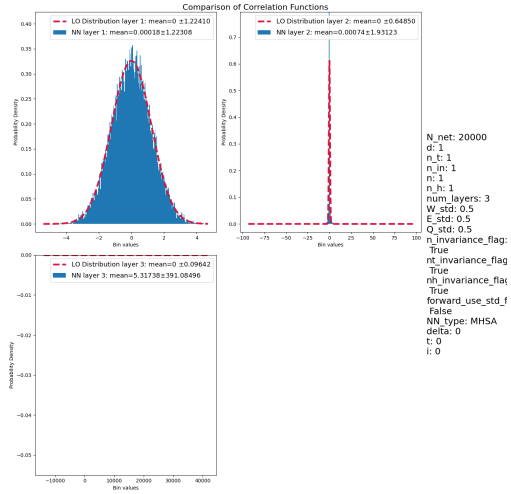
ii. The variance always remains a good match. However, the visual histograms have a higher peak for the NNs at lower n, while as n increases, the peaks of the NN distribution and the FE align better and better until they almost perfectly align in the $n = 500$ case.

Deeper in the Layers This one is uncertain, but based on a comment by Richard, the layers should increase linearly. As can be seen in both the variance plots (R^2 correlation), this is indeed the case (albeit with a negative slope).

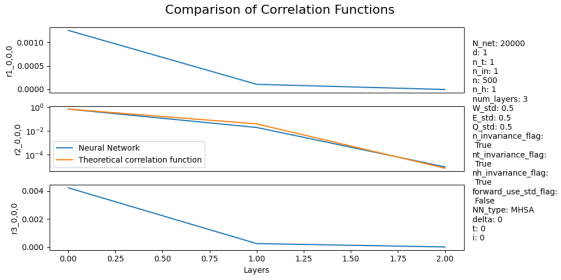
MHSA



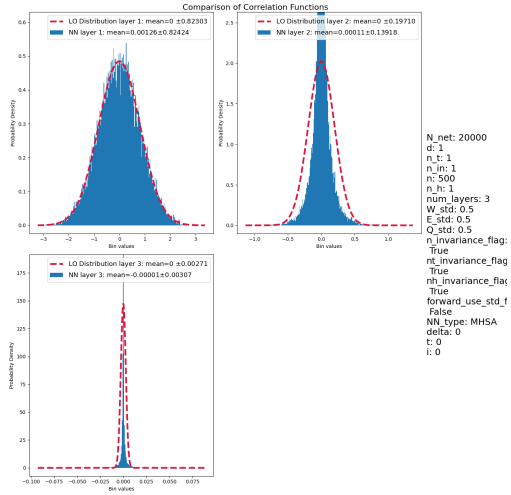
(a) $n = 1$ correlation functions



(b) $n = 1$ histogram

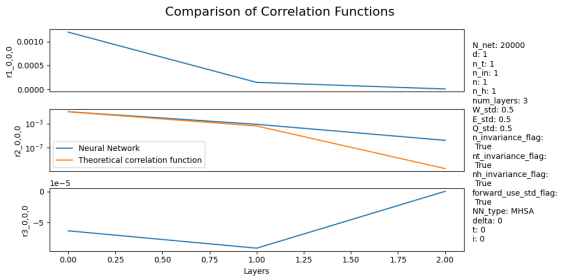


(c) $n = 500$ correlation functions

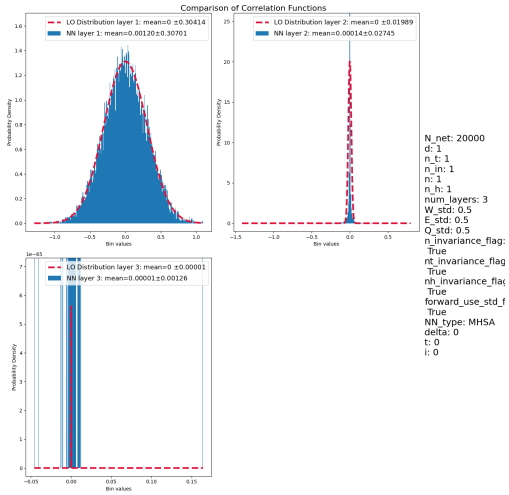


(d) $n = 500$ histogram

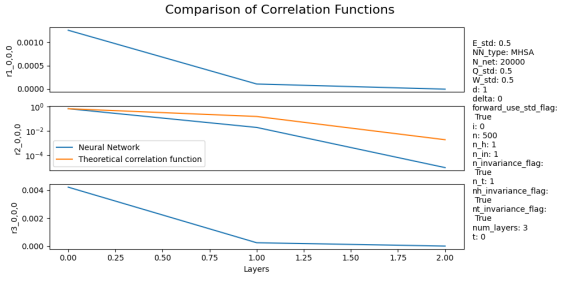
Figure 4: Correlation functions and histograms for the MHSA per width n where the FE uses the variance



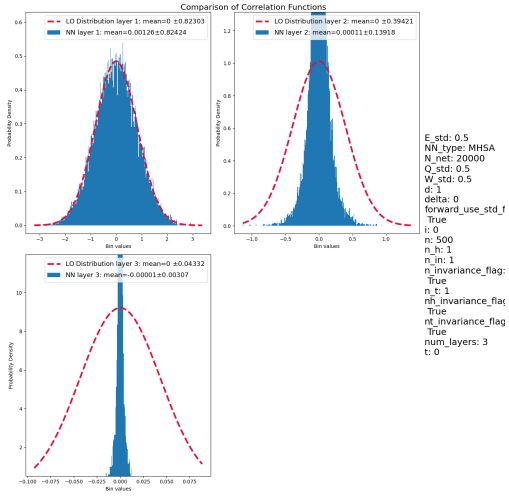
(a) $n = 1$ correlation functions



(b) $n = 1$ histogram



(c) $n = 500$ correlation functions



(d) $n = 500$ histogram

Figure 5: Correlation functions and histograms for the MHSA per width n where the FE (erronously) uses the standard deviation

1 Neuron Edge Case Expectation: Non-Gaussian, since as far away as possible from the infinite width limit.

i. In the edge case where we have 1 neuron per layer, we indeed get the largest difference between the NN and FE.

ii. The edge case shows among the largest differences, but there are other networks that come a bit closer.

Increasing the Number of Neurons per Layer Expectation: The closer it gets to the infinite width limit, the closer it should get to a Gaussian. Since the number of tokens is set to 1, the Ω matrix that weighs tokens should thus (naively) not do much. From the MLP comparison, we would also expect some better alignment with increasing n (at least in the erroneous standard deviation case).

i. The variance remains fairly close (at the very least the same order of magnitude, but mostly quite close). As n increases, the variance increases. As the layers increase, the variance spread mostly also increases, but not necessarily. In addition, from the histogram plots, it can be seen that the NN and FE peaks get lower as n increases. However, the image that the NN distribution is more strongly peaked than the FE remains. It would be interesting to increase the number of neurons even more to see if it does indeed approach the infinite width limit. Note that doing an extra $n = 1000$ doesn't significantly improve compared to $n = 500$. However, this is all for $t = 1$.

ii. As opposed to the MLP case, this always performs slightly worse in terms of matching the distribution, both in terms of variance matching and the visual distributions. It does, however, show the same trends.

Parameter	**Value**
E_std	0.5
NN_type	MHSA
N_net	20000
Q_std	0.5
W_std	0.5
d	1
delta	0
i	0
n	20
n_in	1
n_invariance_flag	true
n_t	1
n_h	1
nh_invariance_flag	true
nt_invariance_flag	true
num_layers	3
t	0

Table 1: Parameter settings varying n.

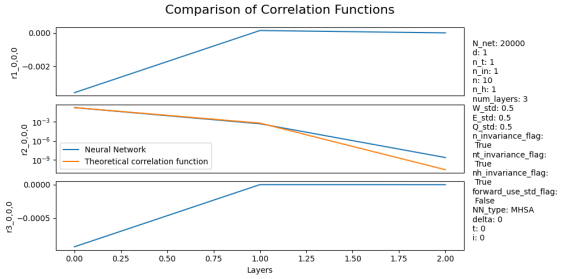
9.2 Varying the Number of Heads

This section analyses the variation of the number of heads. Varied n_h from 1 to 30. No patterns where found when varying the number of heads.

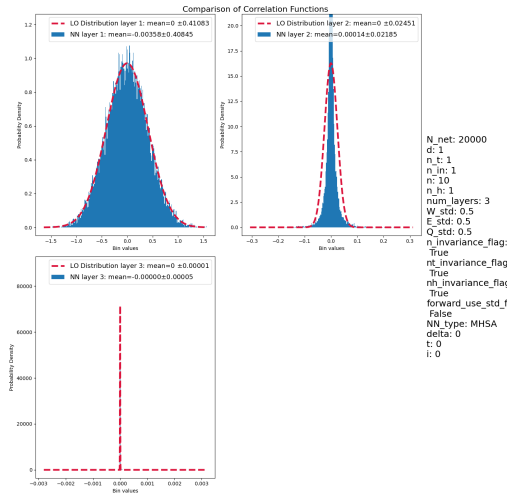
9.3 Varying the Number of Tokens

Parameter	**Value**
E_std	0.5
NN_type	MHSA
N_net	20000
Q_std	0.5
W_std	0.5
d	1
delta	0
forward_use_std_flag	false
i	0
n	20
n_in	1
n_invariance_flag	true
n_t	1
nh_invariance_flag	true
nt_invariance_flag	true
num_layers	4
t	0

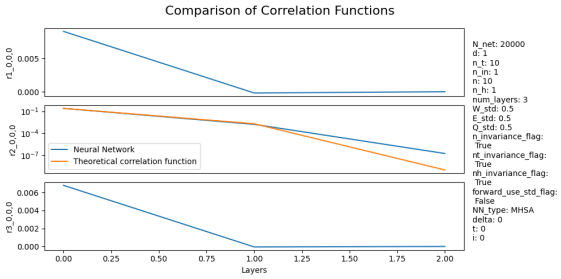
Table 2: Parameter settings varying n_h.



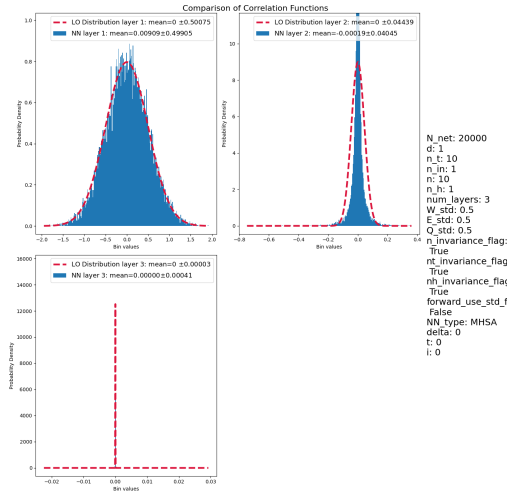
(a) $n_t = 1$ correlation functions



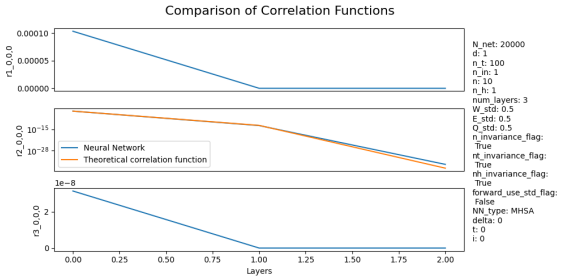
(b) $n_t = 1$ histogram



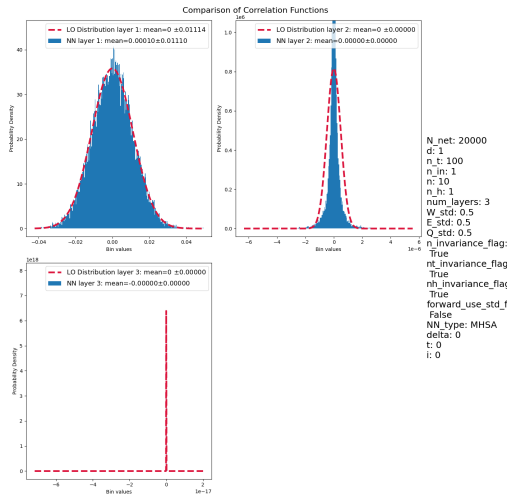
(c) $n_t = 10$ correlation functions



(d) $n_t = 10$ histogram



(e) $n_t = 100$ correlation functions



(f) $n_t = 100$ histogram

Figure 6: Correlation functions and histograms for varying n_t per width $n = 10$

This section analyses the variation in the number of tokens n_t. I compared for $n_t = [1, 10, 20, 50]$ with $n = 50$ and $n_t = [1, 10, 20, 50, 100]$ with $n = 1$ due to hardware limits. Furthermore, I compare with (i) and without (ii) invariance with respect to n_t.

$n = 50$ and $n = 10$ seem from the n results to be a reasonable width while at the same time allowing variation in t more widely within the hardware bands. Note that $n = 1$ is not a good choice since this shows the most deviating behavior with respect to the other n (see the n comparison).

Number of Neurons $n = 10$

Edge Case?, $n_t = 1$

i. There doesn't really seem to be an edge case. Both the spread and order of the variance differ without a clear pattern.

ii. There doesn't really seem to be an edge case. Both the spread and order of the variance differ without a clear pattern.

A comparison between the invariant and non-invariant cases is therefore difficult to make.

Increasing n_t Same as above.

Number of Neurons $n = 50$

Edge Case?, $n_t = 1$ Same as above.

Increasing n_t **for Varying** n Same as above.

So the conclusion seems to be that the number of tokens doesn't really seem to impact the (non-)Gaussianity. In a way, this is surprising, given that I would expect it to be a source of non-Gaussianity. However, on the other hand, the MHSA doesn't show Gaussian behavior when there is just one token. Such that the weighting, by virtue of using the previous layer output, already seems to impact Gaussianity, independent of the number of tokens.

Parameter	**Value**
E_std	0.5
NN_type	MHSA
N_net	20000
Q_std	0.5
W_std	0.5
d	1
delta	0
i	0
n_in	1
n_invariance_flag	true
n_h	1
nh_invariance_flag	true
nt_invariance_flag	true
num_layers	3
t	0

Table 3: Parameter settings varying n.

9.4 Surprising MLP Result

In addition to the analysis done for the MLP in section 9.1 on varying the width n, this section showcases the impact of erroneously using the standard deviation for c by looking at deeper layers (maximum of 9). When looking at the probability from which a weight is drawn,

$$p(W) \propto e^{-\frac{1}{2c}W^2}, \tag{9.1}$$

this would mean that c would be (erroneously) set as the standard deviation instead of the variance in the forward equation. As can be seen in the figures the erroneous use of the standard deviation continues to

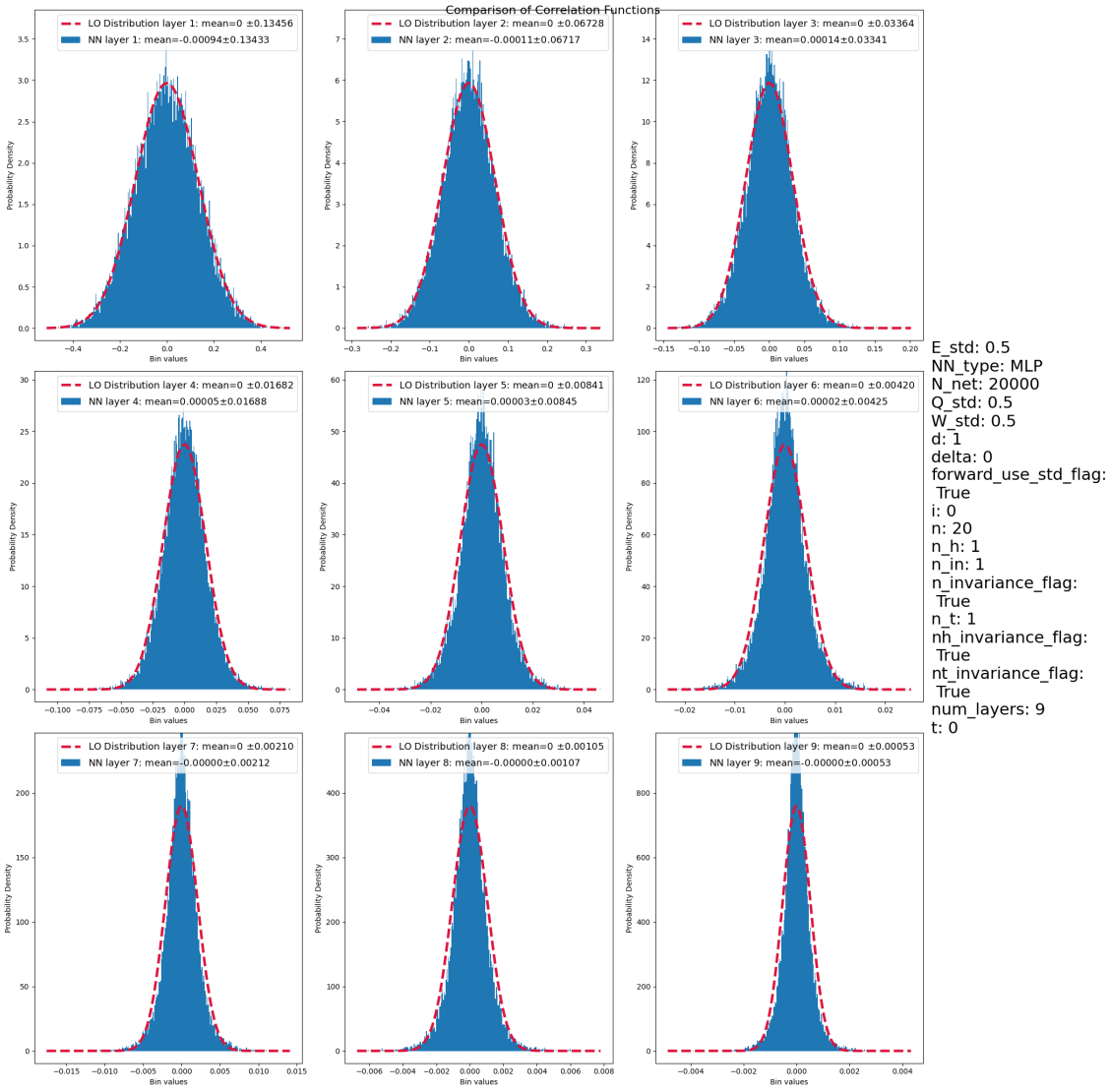
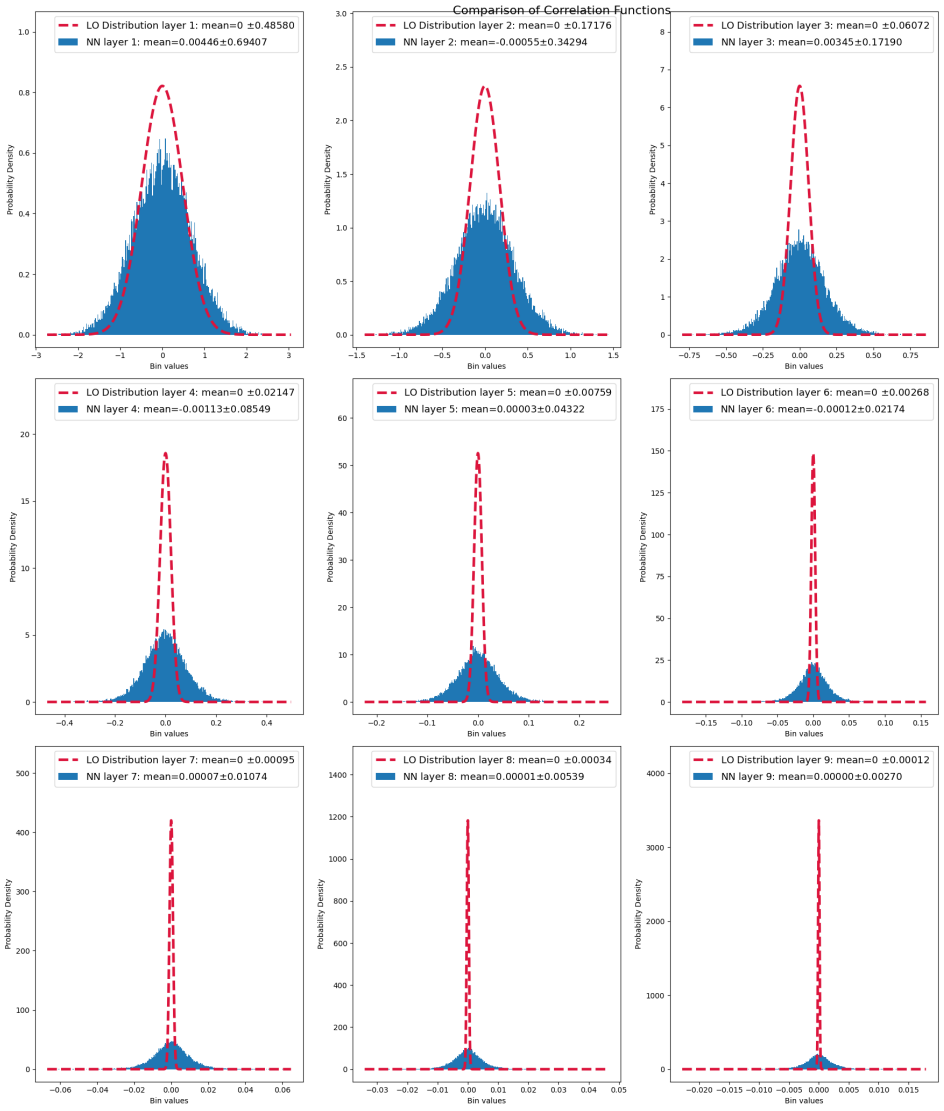


Figure 7: MLP with erroneous std usage in the FE

describe the MLP numerical results much better. I don't have a readily answer to explain why this is the case, and due to time constraints I'm unable to explore this further. I note it down, since it is an interesting result. See the legend for the layer number, but the count is from left to right 1-9.



E_std: 0.5
NN_type: MLP
N_net: 20000
Q_std: 0.5
W_std: 0.5
d: 1
delta: 0
forward_use_std_flag: False
i: 0
n: 20
n_h: 1
n_in: 1
n_invariance_flag: True
n_t: 1
nh_invariance_flag: True
nt_invariance_flag: True
num_layers: 9
t: 0

Figure 8: MLP with correct variance usage in the FE

10 Conclusion

The project aimed at gaining a better understanding of the transformer. Due to the complexity this narrowed down to what seemed the most novel (from the EFT) perspective in the transfomer: the multi-head self-attention (MHSA) component. At first a theoretical understanding of the MHSA was gained by studying the reference paper that claimed the MHSA could be described as gaussian. The most important intermediate result for that claim has been shown to be wrong. However, to understand if the claim itself was wrong we continued with a theoretical exploration using the EFT formalism developed by Richard Kenway. Using this formalism we focused purely on the network at intitalisation. The most important part of this formalism is finding the forward equation from layer to layer when assuming that the previous layer is drawn from a gaussian. We did find it for the MHSA at leading order with respect to the width n. The forward equation does contain a 4^{th} order derivative, which is an indication there could very well be a relevant leading order quadratic term. If this is the case, the gaussian assumtion doesn't hold. In addition, we could not find the forward equation when going from the input to the first hidden layer. This also seems an indication that something more complex than gaussians is going on. To further confirm the theoretical framework we compared it to numerical results. Since the first layer cannot be calculated if it is a MHSA block, we used the (known) multi-linear perceptron (MLP) as is also done in transformers. Furhtermore, to simplify the calculations we used a linear activation function with zero bias. From the results it also seems that the MHSA isn't gaussian even in the 'easy' case of the linear MLP. This work thus seems to indicate that the MHSA isn't well described by gaussians.

As an additional note, in the process of ensuring the implementation was performed properly, we (accidentally) encountered that the linear MLP is better described by the theory when using the standard deviation instead of the variance in the theoretical calculations. We did not investigate this further, but since it is easy to try out in implementations (i.e. change one value to its square root or vice versa) it should be easy to explore in future research.

A Derivation of Transition Matrix

Before continuing into derivations with respect to transformers, it would be good to be confident in the sharpness of my abilities. For that reason, I aim at re-deriving earlier results, to get up to speed with the nuances (which their seem to be).

For some weird reason I do not seem to be able to retrieve the same result for the derivation of the book, or Richard. Given that via the book I do end up at a result in the first place, I will write that one out. Hopefully this will make it easy to detect the error.

In this derivation I will use the conventions that $f^{(\ell+1)} = f$, $f^{(\ell)} = f'$, and $\rho(\phi_{\delta,i}) = \rho_{\delta,i}$ where ρ is the activation function and

$$\phi_{\delta,i} = b_i + \rho'_{\delta,i}. \tag{A.1}$$

Furthermore I will assume all summations over index i go up to n and the other Latin indices go up to n', i.e. respectively to the $\ell + 1$ layer and ℓ layer.

A.1 Starting From Deep Learning Book

Assuming, the limit has already been taken the δ is given by

$$\delta(f) = \int \frac{d\Lambda}{2\pi} e^{i\Lambda(f)}. \tag{A.2}$$

Then

$$\begin{aligned} p(\phi|\phi') &= \int d\theta p(\theta) p(\phi|\phi', \theta) \\ &= \left(\prod_i \int db_i (2\pi c_b)^{-\frac{1}{2}} e^{-\frac{1}{2c_b} b_i^2}\right) \left(\prod_{i,j} \int dW_{ij} (2\pi c_w)^{-\frac{1}{2}} e^{-\frac{1}{2c_w} W_{ij}^2}\right) \\ &\left(\prod_{\delta,i} \delta(\phi_{\delta,i} - b_i - \sum_j W_{ij} \rho'_{\delta,j})\right) \end{aligned} \tag{A.3}$$

Where the δ function ensures that the outgoing neuron is fully determined by the weights and biases. However, continuing from there, this becomes

$$\begin{aligned} p(\phi|\phi') &= \prod_i [\int db_i (2\pi c_b)^{-\frac{1}{2}} e^{-\frac{1}{2c_b} b_i^2} \left(\prod_j \int dW_{ij}\right) (2\pi c_w)^{-\frac{n'}{2}} e^{-\frac{1}{2c_w} \sum_j W_{ij}^2} \\ &\left(\prod_\delta \int \frac{d\Lambda_{\delta,i}}{2\pi} e^{i\Lambda_{\delta,i}(\phi_{\delta,i} - b_i - \sum_j W_{ij}\rho'_{\delta,j})}\right)] \\ &= \prod_i \left[\alpha \int \frac{db_i}{\sqrt{(2\pi c_b)}} \left(\prod_j \frac{dW_{ij}}{\sqrt{(2\pi c_w)}}\right) \left(\prod_\delta \frac{d\Lambda_{\delta,i}}{2\pi}\right) e^{-\frac{1}{2c_b} b_i^2 - \frac{1}{2c_w} \sum_j W_{ij}^2 + i \sum_\delta \Lambda_{\delta,i}(\phi_{\delta,i} - b_i - \sum_j W_{ij}\rho'_{\delta,j})}\right]. \end{aligned} \tag{A.4}$$

In the first line we have taken out the product over the index i, taken out everything except the dW_{ij} from the product over the index j, which results in a sum in the corresponding exponential e. Furthermore I have used the definition of the δ. In the second line I have combined all the powers of e.

When looking at the equation we can use completing the square, i.e.

$$\int dx e^{-ax^2 + bx + c} = e^{-\frac{b^2}{4a} + c} \sqrt{\frac{\pi}{a}}, \tag{A.5}$$

for both b_i and W_{ij} in the exponent this can be done. For both respectively

$$-\frac{1}{2c_b} b_i^2 - i \sum_\delta \Lambda_{\delta,i} b_i = -\frac{1}{2c_b} (b_i + i c_b \sum_\delta \Lambda_{\delta,i})^2 - \frac{c_b}{2} (\sum_\delta \Lambda_{\delta,i})^2, \tag{A.6}$$

and

$$-\frac{1}{2c_w} W_{ij}^2 - i \sum_\delta \Lambda_{\delta,i} \sum_j W_{ij} \rho'_{\delta,j} = - - \frac{1}{2c_w} (W_{ij}^2 + i c_w \sum_\delta \Lambda_{\delta,i} \rho'_{\delta,j})^2 - \frac{c_w}{2} (\sum_{\delta,j} \Lambda_{\delta,i} \rho'_{\delta,j})^2, \tag{A.7}$$

where shifting the b_i and W_{ij} to account for the phase leaves the measure db_i and dW_{ij} unchanged, such that we can perform a standard Gaussian integral:

$$\begin{aligned} p(\phi|\phi') &= \prod_i [\int \left(\prod_\delta \frac{d\Lambda_{\delta,i}}{2\pi}\right) e^{-\frac{c_b}{2}(\sum_\delta \Lambda_{\delta,i})^2 - \frac{c_w}{2}(\sum_{\delta,j} \Lambda_{\delta,i} \rho'_{\delta,j})^2 - \sum_\delta \Lambda_{\delta,i} \phi_{\delta,i}} \left(\int db_i (2\pi c_b)^{-\frac{1}{2}} e^{-\frac{1}{2c_b} b_i^2}\right) \\ &\left(\prod_j \int \frac{dW_{ij}}{\sqrt{(2\pi c_w)}} e^{-\frac{1}{2c_w} W_{ij}^2}\right)] \\ &= \prod_i \left[\int \left(\prod_\delta \frac{d\Lambda_{\delta,i}}{2\pi}\right) e^{-\frac{c_b}{2}(\sum_\delta \Lambda_{\delta,i})^2 - \frac{c_w}{2}(\sum_{\delta,j} \Lambda_{\delta,i} \rho'_{\delta,j})^2 - \sum_\delta \Lambda_{\delta,i} \phi_{\delta,i}}\right], \end{aligned} \tag{A.8}$$

where I have removed the sum with respect to j when moving the exponent back into the product. In going to the second line I have used that the standard Gaussian integral becomes 1.

Doing the same for $\Lambda_{\delta,i}$ gives

$$-\frac{1}{2}\sum_{\delta_1,\delta_2}\Lambda_{\delta_1,i}\Lambda_{\delta_2,i}G_{\delta_1,\delta_2}-\sum_{\delta}\Lambda_{\delta,i}\phi_{\delta,i}=-\frac{1}{2}\sum_{\delta_1,\delta_2}(\Lambda_{\delta_1,i}-i\sum_{\delta_3}G^{-1}_{\delta_1,\delta_3}\phi_{\delta_3,i})(\Lambda_{\delta_2,i}-i\sum_{\delta_3}G^{-1}_{\delta_2,\delta_3}\phi_{\delta_3,i})G_{\delta_1,\delta_2} - \frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}, \tag{A.9}$$

where

$$G_{\delta_1,\delta_2}=c_b+c_w\sum_j\rho'_{\delta_1,j}\rho'_{\delta_2,j}. \tag{A.10}$$

Assumed it was invertible.

Again shifting out the imaginary part gives

$$p(\phi|\phi')=\prod_i\left[e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}}\int\left(\prod_\delta\frac{d\Lambda_{\delta,i}}{2\pi}\right)e^{-\frac{1}{2}(\sum_{\delta_1,\delta_2}G_{\delta_1,\delta_2}\Lambda_{\delta_1,i}\Lambda_{\delta_2,i}}\right]. \tag{A.11}$$

Now rotating, without consequences $d\Lambda_{\delta,i}\to d(O\Lambda)_{\delta,i}$ for a matrix such that $G_{\delta_1,\delta_2}=ODO^{-1}$, where D is a diagonal matrix with eigen values g_δ and O is also orthogonal, the equation becomes

$$\begin{aligned}
p(\phi|\phi') &= \prod_i\left[e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}}\int\left(\prod_\delta\frac{d\Lambda_{\delta,i}}{2\pi}\right)e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}g_\delta\delta_{\delta_1,\delta_2}\Lambda_{\delta_1,i}\Lambda_{\delta_2,i}}\right]\\
&= \prod_i\left[e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}}\int\left(\prod_\delta\frac{d\Lambda_{\delta,i}}{2\pi}\right)e^{-\frac{1}{2}\sum_\delta g_\delta\Lambda^2_{\delta,i}}\right]\\
&= \prod_i\left[e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}}\int\left(\prod_\delta\frac{d\Lambda_{\delta,i}}{2\pi}e^{-\frac{1}{2}g_\delta\Lambda^2_{\delta,i}}\right)\right]\\
&= \prod_i\left[e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}}\left(\prod_\delta\frac{1}{2\pi}\sqrt{\frac{2\pi}{g_\delta}}\right)\right]\\
&= \prod_i\left[e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}}\left(\prod_\delta\frac{1}{\sqrt{2\pi g_\delta}}\right)\right]\\
&= \prod_i\left[e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1,i}\phi_{\delta_2,i}}\frac{1}{\sqrt{(2\pi)^d|G_{\delta_1,\delta_2}|}}\right]\\
&= e^{-\frac{1}{2}\sum_{\delta_1,\delta_2}G^{-1}_{\delta_1,\delta_2}\phi_{\delta_1}\phi_{\delta_2}}\frac{1}{((2\pi)^d|G_{\delta_1,\delta_2}|)^{\frac{n}{2}}}
\end{aligned} \tag{A.12}$$

Where d is the size of the dataset,. This result differs with respect to the exponent of d over 2π is of by quite margin if the dataset d is large. But this is probably due to a sloppy mistake on the authors side. However, this is also missing in Richards notation. Which, as is in the overleaf, also misses the (explicit) sum $\sum_{\delta_1,\delta_2}$ in the final exponent of e.

A.2 Starting From Richards Point

The δ function is also defined as

$$\delta(f)=\lim_{C\to0}\frac{1}{\sqrt{2\pi C}}e^{-\frac{f^2}{2C}}, \tag{A.13}$$

where $f \in \mathbb{R}$. Such that without taking the limit, i.e. keeping it regularised we would start from

$$\begin{aligned} p(\phi|\phi') &= \int d\theta p(\theta) p(\phi|\phi', \theta) \\ &= \left(\prod_i \int db_i (2\pi c_b)^{-\frac{1}{2}} e^{-\frac{1}{2c_b} b_i^2} \right) \left(\prod_{i,j} \int dW_{ij} (2\pi c_w)^{-\frac{1}{2}} e^{-\frac{1}{2c_w} W_{ij}^2} \right) \\ &\left(\prod_{\delta,i} \frac{1}{2\pi c} e^{(\frac{-1}{2C}\phi_{\delta,i} - b_i - \sum_j W_{ij}\rho'_{\delta,j})^2} \right). \end{aligned} \tag{A.14}$$

Without going through all the steps again, it is just the same tricks as before I end up at

$$p(\phi|\phi') = \prod_i \left[\alpha e^{\sum_{j,k} C_{ij} A^{-1}_{jk} C_{ik} + c_i} \right], \tag{A.15}$$

where $\alpha \propto \frac{1}{\sqrt{|A_{ij}|}}$,

$$C_{ij} = (\frac{2b+c}{c^3} + \frac{1}{c}) \sum_\delta \phi_{\delta,i} \rho'_{\delta,j}, \tag{A.16}$$

and

$$c_i = -\sum_{\delta_1,\delta_2} \phi_{\delta_1,i} \phi_{\delta_2,i} K_{\delta_1\delta_2}, \tag{A.17}$$

and

$$A_{jk} = \frac{\delta_{jk}}{2C_w} + \sum_{\delta_1,\delta_2} \rho'_{\delta_1,j} \rho'_{\delta_2,k} K_{\delta_1\delta_2}, \tag{A.18}$$

where

$$K_{\delta_1\delta_2} = \frac{\delta_{\delta_1\delta_2}}{2c} - \frac{2b+c}{c^3}. \tag{A.19}$$

The key takeaway is that this seems a dead point to me given that I don't know how to calculate the inverse of A_{jk}, such that it is impossible to continue from here. However, even when solving the inverse, I don't readily see how A_{jk} would result in the required G_{jk}.

B Existing Literature

Note that this section is a work in progress, and the papers cited have not been properly read beyond first glances and quick readings.

The normalisation s, could break down the MLP approach according to [6], where it is referred to as the batchnorm, due to the presence of a singularity in the jacobian. [6], does in general seem an interesting read. In addition, the (non-)gaussianty of the self-attention element has been studied in [7].

As a side note, with an eye on the ongoing discussion regarding the comparison between NNPDF and the other groups using Hessian methods, [8] might be illuminating. On a quick glance it seems as if an NN contains a hessian component and another thing (dubbed sensitivity) in the propagator in the infinite width limit.

References

[1] R. Kenway, *Lectures richard kenway* (2024).

[2] E. Dinan, S. Yaida, and S. Zhang, *Effective theory of transformers at initialization* (2023), 2304.02034, URL https://arxiv.org/abs/2304.02034.

[3] D. A. Roberts, S. Yaida, and B. Hanin, *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks* (Cambridge University Press, 2022), ISBN 9781316519332, URL http://dx.doi.org/10.1017/9781009023405.

[4] G. van Seeventer, *code implementation* (2024), URL https://github.com/ggvanseev/EFT-transformers.git.

[5] J. Bergstra, D. Yamins, and D. D. Cox, *Making a science of model search* (2012), 1209.5111, URL https://arxiv.org/abs/1209.5111.

[6] G. Yang, *Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation* (2020), 1902.04760, URL https://arxiv.org/abs/1902.04760.

[7] J. Hron, Y. Bahri, J. Sohl-Dickstein, and R. Novak, *Infinite attention: Nngp and ntk for deep attention networks* (2020), 2006.10540, URL https://arxiv.org/abs/2006.10540.

[8] B. Bordelon and C. Pehlevan, *Dynamics of finite width kernel and prediction fluctuations in mean field neural networks* (2023), 2304.03408, URL https://arxiv.org/abs/2304.03408.