On activation function drift and diffusion in large width-depth networks

Adam James Sheppard

July 31, 2024

Abstract

This document proposes a framework to examine the application of numerical approximation methods to nodes on a network to understand how information mutates as it propagates through the network. The framework aims to describe information mutation through input/output pairs of nodes, which could be useful for predicting node outputs. A three-level model is introduced, with each level adding substantial complexity. By measuring the nodes and their interactions, the framework seeks answer when it is possible approximate the trajectory of information evolution under continuous chaining.

It models the information state of each agent on the graph using local "transfer functions" projected into a higher-dimensional space called the "Vertex Function". A "Global Operator" captures the collective network dynamics of the Vertex Function and is related by a nonlinear vector differential equation.

A diagnostic framework is introduced that employs a top-down approach, focusing on the dynamics of a vector differential equation and subsequently working down to the information state level. The aim of this proposal is to the framework and then use it to explore stochastic dynamics arising in large width-depth networks.

1 Review

I shall conduct a brief literature review on Feature Learning in Deep Neural Networks, which is part of the wider effort to explain Artificial Neural Network Dynamics since the phenomena in the title is derived from this. I shall get to why signal propagation in Neural Networks is connected this phenomenon and hopefully convince you that this is an interesting problem. I will then set up a preliminary framework used to study this from a functional analytic and operator theoretic perspective. - though this framework will be adapted and is general.

Linear Models

In linear models, we choose a function f which we know some things about, as a linear function. We are given a vector $\mathbf{x}$ of some output from f, and some parameters $\boldsymbol{\omega}$ such that

$$f(\mathbf{x}, \boldsymbol{\omega}) = \mathbf{x}^{\mathsf{T}} \boldsymbol{\omega}, \quad \boldsymbol{\omega} \in \mathbb{M}_{r \times c}(\mathbb{R})$$

In statistics we see linear models used in the regression setting. Where $\mathbf{y}$ is a variable we wish to infer some things about using f. We have an assumption that $\mathbf{y}$ is generated according to a linear dependence in $\mathbf{x}$ with some additive noise $\boldsymbol{\epsilon}$

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\omega}^*) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \in \mathbb{R}^c$$

$\boldsymbol{\omega}^*$ is an unknown parameter value that we seek to approximate. This is achieved by casting the problem in a least squares setting and finding a Best Linear Unbiased Estimator $\boldsymbol{\omega}^*_{LS}$ where $\mathbb{E}\left[\boldsymbol{\omega}^*_{LS}\right] = \boldsymbol{\omega}^*$, then by the Gauss-Markov Theorem showing that the BLUE estimate minimises the variance of any other estimator such that $\mathbb{V}\left[\boldsymbol{\omega}^*_{LS}\right] < \mathbb{V}\left[\boldsymbol{\omega}^*\right]$.

1.1 Non Parametric Models

Kernel Methods

If we wish to model non-linear relationships, a natural extension is using a feature mapping $\phi : \mathcal{X} \to \mathcal{H}$ where $\mathcal{H}$ is a Hilbert Space.

Definition 1. A Kernel $\mathcal{K} \in \mathbb{R}$ is a measure. It describes the similarity of two inputs $\mathbf{x}$ and $\mathbf{x}'$. When $\mathcal{K}$ is large then two inputs are similar. When $\mathcal{K}$ is small then the two inputs are not similar. There is a indeed a gray area since the kernel is defined over the real numbers.

Definition 2. A positive definite kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathcal{H}$ is a symmetric real-values function such that, for any collection of values $\{\mathbf{x}\}_{i=1}^n$ for $n > 0$, then a matrix $K \in \mathbb{M}_{n\times n}(\mathbb{R})$ where

$$K = \begin{pmatrix} \mathcal{K}(x_1, x_1) & \mathcal{K}(x_1, x_2) & \cdots & \mathcal{K}(x_1, x_n) \\ \mathcal{K}(x_2, x_1) & \mathcal{K}(x_2, x_2) & \cdots & \mathcal{K}(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}(x_n, x_1) & \mathcal{K}(x_n, x_2) & \cdots & \mathcal{K}(x_n, x_n) \end{pmatrix}$$

is positive semi-definite.

Theorem 1. *Two function f and g in a Hilbert Space $\mathcal{H}\left(L^2\left([a,b]\right)\right)$ are considered similar (or linearly independent) if and only if there exists a scalar $\lambda \in \mathbb{R}$ such that*

$$f(x) = \lambda g(x)$$

or equivalently

$$\langle f, g \rangle = ||f||||g||$$

Theorem 2. *(Moore-Aronszajn) For any kernel $\mathcal{K}$, there exists a feature map ϕ with domain $\mathcal{H}$ such that*

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

or in other words. Any complete, separable metric space $\mathcal{X}$ can be embedded into $\mathcal{H}$ while preserving the distance metric d such that

$$d(\mathbf{x}, \mathbf{x}') = \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|$$

The ability to embed data into a Hilbert space means that kernel methods can be used to implicitly map data into this space, enabling linear algorithms to work on data that is not linearly separable in its original space. The theorem assures that any complex, nonlinear transformation of data can be effectively managed by mapping it to a Hilbert space where linear methods can be applied. This is particularly useful in high-dimensional data analysis. [The details of the proof and construction of the kernel $\mathcal{K}$ and it's connections to ridge regression are omitted.]

Gaussian Processes

While kernel methods can be thought of as allowing one to construct and optimise predictors over a rich space of functions, in some cases we may wish to work with a probability distribution of functions. Gaussian Processes (GPs) [18] can be thought of as an probabilistic extension of kernel methods to give some quantification of uncertainty in predictions and do inference over these functions.

GPs are stochastic processes $f_{\mathcal{X}}$ over the input space $\mathcal{X}$ satisfying the property that the joint distribution of any finite collection of random variables from f is multivariate Gaussian distribution. A Gaussian Process $f_{\mathcal{X}}$ depends on the kernel $\mathcal{K}$ and function $\mu : \mathcal{X} \to \mathbb{R}$ describing the mean, where

$$f_{\mathcal{X}} \sim \mathcal{GP}(\mu, \mathcal{K})$$

where the kernel $\mathcal{K}$ determines the covariance over all inputs. To put this together we have for any collection of inputs $\{\mathbf{x}_i\}_{i=1}^n$ then

$$\{f_{\mathcal{X}}(\mathbf{x}_i)\}_{i=1}^n \sim \mathcal{N}(\mu, \mathcal{K})$$

we can think of f as establishing a distribution over random functions on $\mathcal{X}$, with kernel $\mathcal{K}$ controlling the likelihood associated to different functions.

Both kernel methods and gaussian processes can be used to construct predictive functions without some fixed parameter space. Gaussian processes can be used in Bayesian statistics to model a prior belief function using the GP $f_{\mathcal{X}}$ and then updating out belief of $f_{\mathcal{X}}$ given some observed input data $\mathbf{x}$ and output $\mathbf{y}$. It provides a structured way to account for uncertainty in estimates [14] making it a natural and popular consideration. However, due to the intractability of doing full bayesian inference in high dimensional settings and large parameter spaces then approximate inference is sought. [6]. However, the appealing idea about GPs is that they analytic in the case of regression problems and a conditional probability distribution given data pairs $f_{\mathcal{X}} | \{x_i, y_i\}$ is analytic [details omitted].

Both Kernel methods and Gaussian Processes are non-parametric methods that construct predictive functions where parameter space is, rather, an infinite set.

1.2 Parametric Models (Neural Networks)

As the Deep Learning community has applied Neural Networkss to increasingly many tasks and data types in recent yers, the range of possibilities for how to construct the feature mapping ϕ has grown too...a lot.

Here we will consider feedforward network structures in this project. We consider some nonlinear function f and the depth of the network L and input $\mathbf{x}$ that is propagated through the network recursively. While we know the number of layers, the exact functions mapping between them are unknown, they can be thought of as feature maps ϕ_l, however we call them hidden layers h_l. We can model the propagation of $\mathbf{x}$ as

$$\begin{aligned} h_0(\mathbf{x}) &= \mathbf{x} \\ h_l(\mathbf{x}) &= f\left(h_{l-1}(\mathbf{x}), \boldsymbol{\omega}_l\right) \\ &= h_L^\intercal \left(h_{l-1}^\intercal \left(\cdots h_1^\intercal (\mathbf{x}) \boldsymbol{\omega}_2\right) \boldsymbol{\omega}_L\right) \boldsymbol{\omega}_{L+1} \end{aligned}$$

where $\boldsymbol{\omega} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{L+1}\}$ are trainable parameters. Now we can apply this to the Multi-Layer Perceptron [15] to provide a very basic neural network architecture. This will form the example used throughout the project. The MLP is a fully-connected feedforward network of different hidden layers that are iteratively composed. We define a depth L and width d_l for each later $l \leq L$. Then again we have

$$h_l(\mathbf{x}) = \alpha\left(h_{l-1}^\intercal(\mathbf{x}) \mathbf{W}_l + \mathbf{b}_l\right)$$

Where $\alpha : \mathbb{R} \to \mathbb{R}$ is the activation, α is a nonlinear function that is applied element-wise on h_l. The matrix $\mathbf{W} \in \mathbb{M}_{d_{l-1} \times d_l}(\mathbb{R})$ is a trainable weight matrix and bias $\mathbf{b}_l \in \mathbb{R}^{d_l}$. We can bunch the parameters $\mathbf{W}$and $\mathbf{b}$ into the parameter space $\boldsymbol{\omega} = \{\mathbf{W}, \mathbf{b}\}$. Some popular choices for α are:

$$\alpha = \max\{0, x_i\}$$
$$\alpha = \frac{1}{1+\exp(-x_i)}$$

For some weight variance $\sigma^2_{\mathbf{W}}$ and bias variance $\sigma^2_{\mathbf{b}}$ we have that

$$\mathbf{W}_l^{(0)} \overset{\text{i.i.d}}{\sim} \mathcal{N}\left(0, \frac{\sigma^2_{\mathbf{W}}}{\dim(d_{l-1})}\right)$$
$$\mathbf{b}_l^{(0)} \overset{\text{i.i.d}}{\sim} \mathcal{N}\left(0, \sigma^2_{\mathbf{b}}\right)$$

as per [7] and is necessary to preserve the variance of the pre-activations. Now by letting the feature mappings

$$\phi := h_l$$

i.e. be defined as the outputs from the hidden layers of the network we have a construction of the feature mapping ϕ given by a composition of linear and nonlinear functions. However, when wanting to find the best $\boldsymbol{\omega}$ then we have just transformed a previously convex optimisation problem into a non-convex one. We no longer can tractably find a closed form (or analytical) solution. In neural network communities the parameters $\boldsymbol{\omega}$ are typically initialised randomly and drawn from some probability measure supported on the space of all parameters Ω and then iteratively updated with some gradient-based local optimisation method [3] to minimise an abstract "loss-function", like a least-squares objective function but these differ across network architectures. In fact, crucial to a neural networks success is scalability that changes linearly due to efficient gradient-based optimisation methods. Given the non-convexity in the parameter space, the initialisation of networks becomes an important consideration in determining the final performance of a trained model.

1.3 Some recent results and implication

Understanding feature learning in DNNs involves exploring both linear and non-linear models. Kernel methods and GPs offer non-parametric approaches, while NNs provide a flexible parametric framework. The dynamics of signal propagation central to explaining how features are learned and optimised within these networks. The initial conditions and optimisation strategies significantly impact the final performance, highlighting the importance of initialisation and training methods in neural networks. It is now pertinent to look at what happens in large width limits and how these are connected to the kernel $\mathcal{K}$ and thus the feature map ϕ.

Random Feature Maps

Kernels and Gaussian Processes are computationally expensive, an $n \times n$ matrix inverses in linear models and then when applied to posterior calculations require a large portion of memory $\mathcal{O}(n^2)$ and time $\mathcal{O}(n^3)$. The scalability of these methods is questionable. Some optimisations of gaussian processes are to use random features which should approximate kernels $\mathcal{K}$ via the Moore-Aronszajn, instead of in a Hilbert space $\mathcal{H}$, but in finite dimensional real vector space $\mathbb{R}^m$, $m < \dim(\mathcal{H})$ with $\mathcal{K}(\mathbf{x}, \mathbf{x}') \approx \left\|\hat{\phi}(\mathbf{x})\right\| \left\|\hat{\phi}(\mathbf{x}')\right\|$, where $\hat{\phi}$ is some approximation to the feature map ϕ using data $\{\mathbf{x}_i\}_{i=1}^n$. As the dimension is increased then in the limit these random feature approximations to kernels become exact.

Random features models are parametric approximations to kernel-based methods that become exact as the parameter dimension increases to infinity. Neural Networks have a comparable situation under certain assumptions and size of the parameter space, that as the width in a neural network increases to infinity we get a connection (or more correctly an equivalence) with kernel/Gaussian process based methods. Limiting properties of models can inform one about the behaviours of finite models.

We have known about a connection between kernel methods, Gaussian processes and neural networks since 1996 with the paper [17]. Neal found that if one initialises an multi layer perceptron with independent and identically distributed Gaussian weights then the function that the neural network produces converges, in distribution, to a GP in large width.

There have been several attempts to view deep neural networks through the lens of Gaussian processes and kernels. However, of particular note is the work by Lee et al. (2018) [8], who developed the framework for Neural Network Gaussian Processes (NNGPs). This framework describes a resulting GP and its conjugate kernel by connecting the stochastic gradient descent (SGD) training of neural networks with the posterior of the NNGP.

For a network with L hidden layers, as the width of each layer tends to infinity, the network can be seen as drawing from a Gaussian process. The kernel of this GP depends critically on the choice of activation function, the architecture of the network, and the weight initialisations used. For instance, with a ReLU activation function, the resulting GP kernel is distinct from that with a Tanh activation function. This relationship underscores the importance of these factors in defining the properties of the induced GP and highlights the versatility of the NNGP framework in modeling various types of data and learning tasks.

Corollary 1. *(NNGP kernel convergence of MLP feature map inner-products). Consider a MultiLayer Perceptron Model, that is a linear function of feature mappings*

$$\begin{aligned}\phi &= \alpha \circ \left(h_{l-1}^{\mathsf{T}}(\cdot)\mathbf{W}_l + \mathbf{b}_l\right)\\ &= \alpha \circ g_L\end{aligned}$$

Then, for any $\mathbf{x}$ *and* $\mathbf{x}'$*, then the normalised inner product.*

$$\frac{1}{m}\left\langle \phi\left(\mathbf{x}\right), \phi\left(\mathbf{x}'\right)\right\rangle = \frac{1}{m}\sum_{k=1}^{m} \alpha\left(g_l(\mathbf{x})\right)_k \alpha\left(g_l(\mathbf{x}')\right)_k$$

has a deterministic kernel limit, $\mathcal{K}_{\text{det}}$ *for* $k \to \infty$

Then from this it is possible to show that for some output function f of a wide neural network at initialisation converges in distribution to

$$f \sim \mathcal{GP}(0, \mathcal{K}_{\text{det}})$$

by writing

$$\begin{aligned}f(\mathbf{x}) &= \left[\left(\alpha \circ g_L\right)^{\mathsf{T}}(\mathbf{x})\right]\mathbf{W}_L + \mathbf{b}_L\\ &= \left(h_{l-1}^{\mathsf{T}}(\mathbf{x})\mathbf{W}_l + \mathbf{b}_l\right)\mathbf{W}_L + \mathbf{b}_L\end{aligned}$$

Yang in 2019 proved this result i.e. the output of a wide feed-forward network converges to a Gaussian process, this was proved in proposition G.4 of the paper [19]. In the context of Yang's result, the convergence of the output to a Gaussian process, given by the convergence of the second moment, means that all the second moments of the networks output converge to a Gaussian process. Convergence of second moments typically implies convergence in the L_2 sense. So we have

$$\mathbb{E}[f] \to \mu$$

and

$$\mathrm{Cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] \to \mathcal{K}\left(\mathbf{x}, \mathbf{x}'\right)$$

and so

$$\lim_{n\to\infty}\left\{\mathbb{E}\left[f_n(\mathbf{x}) - \mu(\mathbf{x})\right]^2\right\} \to \mathcal{K}_{\mathrm{det}}$$

which is the deterministic variance of the GP. So it was found that. A wide NN function converges in distribution to a GP, with kernel that is defined by the NNGP kernel limit of last layer feature map inner products.

1.4 Signal Propagation in wide Deep Neural Networks

Efforts have been made to characterise and improve the behaviour of the NNGP kernel, particularly the impact of large depth in a wide DNN. When the depth of the network increases, certain issues arise such as the vanishing or exploding gradient problem. This line of work, known in the literature as Signal Propagation, aims to understand and mitigate these issues to ensure stable and efficient training of deep networks. Signal Propagation was first coined in papers [1, 13]. Signal Propagation studies focus on maintaining the balance of signal variance throughout the layers to prevent the loss of information and ensure that the gradient-based optimisation remains effective.

We now look at how one improves the behaviour of the neural network induced Gaussian Process kernel function when, as well as large width we look at large depth networks. In signal propagation we are interested in "Initialisation-time behaviour".

Initialisation-time behaviour refers to the characteristics and performance of a neural network at the very beginning, right after the network's weights and biases have been initialised but before any training has taken place. Proper initialization is critical because it can significantly influence the training dynamics and the ultimate performance of the neural network. Here are the key aspects of initialisation-time behaviour: Specifically the initialisation time dynamics of $\mathcal{K}_{\mathrm{det}}$ given by

$$\frac{1}{m}\left\langle \phi\left(\mathbf{x}\right), \phi\left(\mathbf{x}'\right)\right\rangle$$

as l varies. All limits are given by a function q_l that depends on the layer and input-data. We want:

$$\frac{1}{m}\left\langle \phi\left(\mathbf{x}\right), \phi\left(\mathbf{x}'\right)\right\rangle \to q_l(\mathbf{x}, \mathbf{x}')$$

in distribution, as $m \to \infty$, the width of the network. For example in a MultiLayer Perceptron we have that

$$g_{l+1} \sim \mathcal{GP}\left(0, \sigma^2_{\mathbf{W}} q_l + \sigma^2_{\mathbf{b}}\right)$$

and thus we can write q_{l+1} recursively in terms of q_l. Assuming that we initialise weights according to

$$\mathbf{W}^{(0)}_l \overset{\text{i.i.d}}{\sim} \mathcal{N}\left(0, \frac{\sigma^2_{\mathbf{W}}}{\dim(d_{l-1})}\right), \quad \mathbf{b}^{(0)}_l \overset{\text{i.i.d}}{\sim} \mathcal{N}\left(0, \sigma^2_{\mathbf{b}}\right)$$

with $\dim(d_{l-1})$ refers to the dimensionality of the output from the $(l-1)$-th layer of a neural network, then it can be shown that

$$q_{l+1}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_f\left[\alpha(f(\mathbf{x}))\alpha(f(\mathbf{x}'))\right]$$

with input-layer

$$q_0(\mathbf{x}, \mathbf{x}') = \frac{1}{\dim(\mathbf{x})}\sigma^2_{\mathbf{W}}\mathbf{x}^{\mathsf{T}}\mathbf{x}' + \sigma^2_{\mathbf{b}}.$$

The Neural Tangents library allows analytic computation of infinite-width NN kernels for such appropriate activation functions [12].

In studying the evolution of neural network behaviour, particularly as depth increases, it becomes crucial to understand the dynamics of certain degeneracies that can arise during initialisation. This understanding helps in designing initialisation schemes that ensure stable and efficient training. In particular the covariance function q_l, if this value grows or shrinks excessively with depth, it indicates instability in the network. When q_l rapidly grows with depth, it suggests that the pre-activation norms are exploding. This can lead to numerical instability and gradient explosion during training. If q_l shrinks with depth, the pre-activation norms are vanishing, leading to the vanishing gradient problem where gradients become too small to effect meaningful weight updates. This is mitigated with $1/\text{dim}(d_{l-1})$. However, one more subtle point is the correlation $\text{corr}\,[q_l]$. Specifically, how the correlation of q_l evolves across depth layers over time. If

$$0 \leq \lim_{l \to \infty} \text{corr}\,[q_l] \leq 1$$

and is independent of $\mathbf{x}$ and $\mathbf{x}'$. It suggests that the network is not preserving meaningful distinctions between different inputs. Then we are not gaining any meaningful information. So we can not measure distance in the features, which means we can not measure distance in the input. The network's features lose their ability to reflect the relative distances and distinctions of the input space. This means that the learned feature representations do not provide useful information about the input geometry. This means we get a failure to train altogether, an overfit, which in turn leads to a failure to generalise. All from the wrong initialisation [16, 8, 11]. Martens et al. (2021) [11] argue and provide empirical evidence that having well-behaved correlation and covariance functions is sufficient for a DNN to undergo rapid training. Their work underscores the importance of maintaining stable signal propagation to achieve effective learning and model performance. This is called *correlation collapse*. Essentially, the rate of convergence to collapsed correlation function is determined through the choice of activation function α, as well as the choice of weight and bias variances. At worst, convergence to the collapsed state $0 \leq \lim_{l\to\infty} \text{corr}\,[q_l] \leq 1$ happens in $\mathcal{O}(\exp(l))$ time. In the Edge of Chaos work [5][2, 13] the rate of convergence can be slowed to sub-exponential growth and some activation functions α.

The depth of neural networks is often considered crucial for achieving optimal performance, as evidenced by the success of deep architectures like VGGNet (Simonyan and Zisserman, 2014) and ResNet (He et al., 2015). However, several studies have highlighted the pitfalls associated with simply increasing depth without incorporating specific architectural tools.

Poole et al. (2016), Schoenholz et al. (2017), and Martens et al. (2021) use the recursive kernel behaviour $\mathbb{E}_f\,[\alpha(f(\mathbf{x}))\alpha(f(\mathbf{x}'))]$ to show that naively stacking layers leads to degenerate correlation functions as the network depth increases to infinity. This degeneration agrees with the observation that deeper neural networks tend to be harder to train (Glorot and Bengio, 2010; Nielsen, 2015). While deep neural networks have demonstrated significant success, simply increasing depth without considering the underlying architectural design and dynamical systems analysis can lead to degenerate correlation functions and training difficulties. Innovations such as skip connections, normalization layers, and careful initialisation strategies are crucial for maintaining stable signal propagation and ensuring effective learning.

Recent studies have advanced our understanding of signal propagation in deep neural networks by considering different limiting behaviors. Li et al. (2022) [9] introduced a novel approach by analysing the joint width-and-depth limit, as opposed to the traditional width-first-then-depth limit. Their work demonstrated that instead of following a deterministic recursion, the Neural Network Gaussian Process (NNGP) kernel at deep layers is governed by a stochastic differential equation. This approach, using similarly transformed activation functions, reveals the stochastic nature of signal propagation when both the width and depth of the network grow simultaneously. (I find rather interesting - what would the evolution of the pdf of this SDE look like? (FPE), steady states w.r.t depth/width?)

In a related vein, Hayou (2022) [4] explored the infinite-depth limit for finite-width residual networks at initialisation. His findings indicate that individual neuron activations in these networks evolve according to a zero-drift diffusion process over depth. This diffusion process, characterised by its stochastic properties, provides insights into how activations spread and interact as the network depth increases, even when the width remains finite.

Lou et al. (2022) [10] examine the role of signal propagation for understanding the feature alignment phenomenon that occurs during training.

An in depth discussion on the Neural Tangent Kernel will also need to be discussed since the NTK offers a valid mathematical regime to characterise entire training trajectories of wide deep neural networks of any architecture or depth. The nets behaviour reduces to kernel learning dictated by two closely related kernels, the NNGP and the NTK, which govern the NN's output at initialisation and training dynamics respectively. As well as criticism therein.

2 The Proposed Framework

It is aimed to explore the interactions of neurons with the proposed framework, and adapt it as needed.

2.1 Mathematical Structure

Suppose we have a time varying graph $G(t) = (V(t), E(t))$ where $V(t)$ represents agents on the graph, and $E(t)$ represents connections between the agents. Here time is continuous, $t \in \mathbb{R}^+$. The manipulation of information between agents v_i and v_j as it diffuses through a network is described by transfer functions ψ_i that operates on some information θ_i .

Information should encompass any data or signals being transmitted and processed by an agent. Here, the information θ_i and θ_j is contained in a set Θ, the set of all possible information on G.

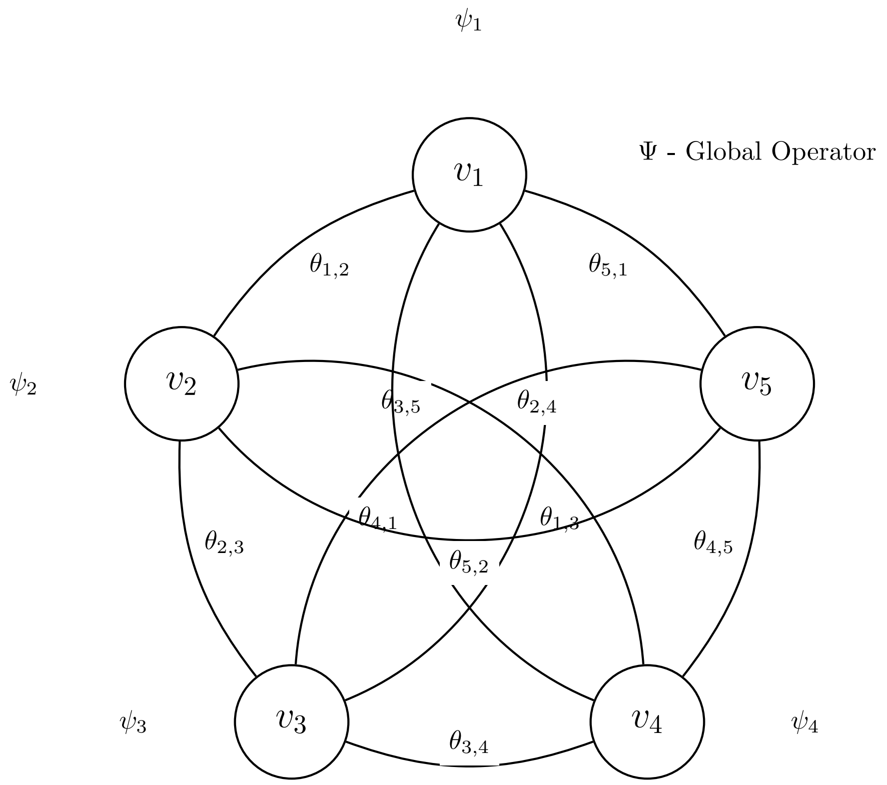
A transfer function ψ_i describes the change of information from θ_i to θ_j contained in the agents $v_i \in V$ to other connected agents $v_j \in V$.

Each transfer function ψ_i is an element an of a vertex function $\psi \in \Theta^{|V|}$ on the graph G. The vertex function shall correspond to a vector function $\psi(\boldsymbol{\theta}) = \big(\psi_1(\theta_1), ..., \psi_{|V|}(\theta_{|V|})\big)^{\mathsf{T}}$. This essentially projects information exchange into a higher dimensional space $\Theta^{|V|}$. This function should tell us how information itself changes over time.

An operator Ψ acts as a unifying law to control information transformations given collectively by the vertex function ψ and provides values for all information states in a network over time. The operator Ψ lives in a function space and maps between other function spaces

$$\Psi : \Theta^{|V|} \times G \to \Theta^{|V'|}$$

this operator will operate on the vertex function to tell us how the agents change in processing and distributing information. The reason $\Theta^{|V'|}$ is chosen here is to emphasise that the networks structure may change over time. $\Theta^{|V'|}$ is a Hilbert or Banach space, and functional mappings between them are the focus - The graph topology can be time-varying due to factors like node mobility, varying transmission power, or environmental changes affecting connectivity.



This diagram visualises what a process being discussed here might look like. After the overall goal of the project has been discussed a diagram detailing how the mathematical description shall be decode this diagram.

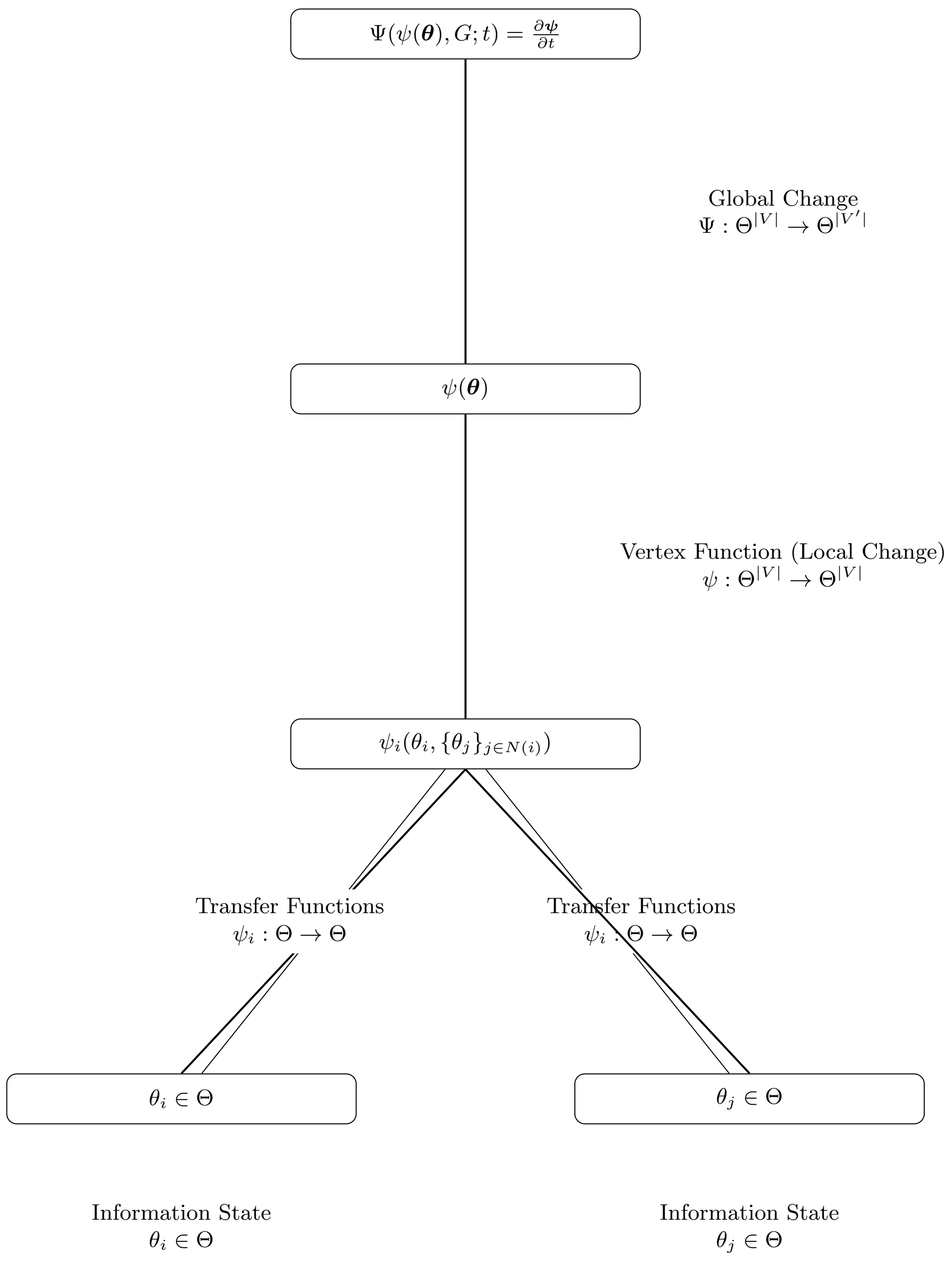
2.2 Overarching Goal

The aim is to build a framework to study the (non)linear-vector-differential equation,

$$\Psi(\psi(\boldsymbol{\theta}), G; t) = \frac{\partial \boldsymbol{\psi}(\boldsymbol{\theta}(t), t)}{\partial t}$$

Where each transfer function ψ_i itself is a dynamical system living on a network G and $\Psi : \Theta^{|V|} \to \Theta^{|V'|}$. Furthermore, Ψ is a non-autonomous system i.e. non-relativistic mechanical systems subject to time-dependent transformations and see what forms it takes with respect to the topics discussed above.

A simplified diagram of the three level model is provided below and it should help with visualising the high level structure used throughout this document and the way I think about this problem.


$\Psi(\psi(\boldsymbol{\theta}), G; t) = \frac{\partial \boldsymbol{\psi}}{\partial t}$
Global Change
$\Psi : \Theta^{|V|} \to \Theta^{|V'|}$
$\psi(\boldsymbol{\theta})$
Vertex Function (Local Change)
$\psi : \Theta^{|V|} \to \Theta^{|V|}$
$\psi_i(\theta_i, \{\theta_j\}_{j \in N(i)})$
Transfer Functions
$\psi_i : \Theta \to \Theta$
Transfer Functions
$\psi_i : \Theta \to \Theta$
$\theta_i \in \Theta$
$\theta_j \in \Theta$
Information State
$\theta_i \in \Theta$
Information State
$\theta_j \in \Theta$

The diagram shows how this framework breaks down network information change $\partial_t \theta$ through a three-level structure with distinct mathematical structures. Individual agents possess information states θ_i residing in an information space Θ (this could be $\mathbb{R}, \mathbb{C}$ or some suitable space). Local dynamics are dictated by transfer functions ψ_i, governing state changes at each node in Θ. However, to capture the collective network dynamics, a vertex function ψ is constructed, which coordinates the state transitions across the network. This transformation bridges the gap between the transfer functions $\psi_i : \Theta \to \Theta$ and the vertex function $\psi : \Theta^{|V|} \to \Theta^{|V|}$, so a vector of transfer functions is created $\psi = (\psi_1, ..., \psi_{|V|})$.

Finally, the operator Ψ oversees the evolution of the network's state vector. It is given by the differential equation,

$$\Psi(\psi(\boldsymbol{\theta}), G; t) = \frac{\partial \boldsymbol{\psi}}{\partial t},$$

Realisations of $\psi(\boldsymbol{\theta}(t))$ for $t \in \mathbb{R}^+$ determines the state of each node over time, $\theta_i \in \Theta$ for all t.

If the information processing rules remains constant over time and the network structure is fixed, so $\Psi = \mathbf{I}$, the identity operator, i.e. the agents do not change how they process information over time and graph structure is stationary, then,

$$\begin{aligned}\Psi(\psi(\boldsymbol{\theta}); t) &= \Psi(\psi(\boldsymbol{\theta})) \\ &= \boldsymbol{\psi}(\boldsymbol{\theta})\end{aligned}$$

which simplifies the problem a bit. We can define this new problem as

$$\begin{aligned}\boldsymbol{\theta}(t) &= \boldsymbol{\theta}(0) + \int_0^t \Psi(\psi(\boldsymbol{\theta}(s), s); s) ds \\ &= \boldsymbol{\theta}(0) + \int_0^t \boldsymbol{\psi}(\boldsymbol{\theta}(s), s) ds\end{aligned}$$

which is an integral equation.

$$\Phi\left(\psi(\boldsymbol{\theta})\right) := \boldsymbol{\theta}(t) = \boldsymbol{\theta}(0) + \int_0^t \left(\boldsymbol{\psi} \circ \boldsymbol{\theta}\right)(s) ds$$

and possibly nonlinear Volterra type, of the second kind; where $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$ are unknown functions. This should tell us how information changes with interacting transfer functions. Somehow apply the Banach Fixed Point Theorem? BFP Theorem might be applicable in cases where a contraction mapping can be established, facilitating convergence to a fixed point that represents a stead-state of $\boldsymbol{\theta}$ change across the network.

As one can see this framework is purposefully general and not much exploration of the framework itself in very fruitful, but once this is applied to studying a particular, studying activation function and the stochastic processes that arise it should definitely help with breaking down and understanding a problem and attacking it from a few mathematical lenses, providing a geometric and intuitive feel.

Hope you enjoyed!

References

[1] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances in neural information processing systems*, 29, 2016.

[2] Kaloyan Danovski, Miguel C Soriano, and Lucas Lacasa. Dynamical stability and chaos in artificial neural network trajectories along training. *Frontiers in Complex Systems*, 2:1367957, 2024.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[4] Soufiane Hayou. On the infinite-depth limit of finite-width neural networks. *Transactions on Machine Learning Research*, 2022.

[5] Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63:2585–2619, 2021.

[6] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[8] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[9] Mufan Li, Mihai Nica, and Dan Roy. The neural covariance sde: Shaped infinite depth-and-width networks at initialization. *Advances in Neural Information Processing Systems*, 35:10795–10808, 2022.

[10] Yizhang Lou, Chris E Mingard, and Soufiane Hayou. Feature learning and signal propagation in deep neural networks. In *International Conference on Machine Learning*, pages 14248–14282. PMLR, 2022.

[11] James Martens, Andy Ballard, Guillaume Desjardins, Grzegorz Swirszcz, Valentin Dalibard, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765*, 2021.

[12] Roman Novak, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Fast finite width neural tangent kernel. In *International Conference on Machine Learning*, pages 17018–17044. PMLR, 2022.

[13] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

[14] Frank P Ramsey. Truth and probability. In *Readings in formal epistemology: Sourcebook*, pages 21–45. Springer, 1926.

[15] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[16] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

[17] Christopher KI Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.

[18] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Number 3. MIT press Cambridge, MA, 2006.

[19] Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *Advances in Neural Information Processing Systems*, 32, 2019.