Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Learning Dynamics of Linear Neural Networks

Supervisor:
Dr. Pedro Mediano

Author:
Nicolas Anguita

Second Marker:
Dr. Tolga Birdal

June 19, 2024

Abstract

In Machine Learning, the full extent of how weight initialization and network structure influence the network's learning properties is still not fully understood. We aim to advance this understanding by enhancing the knowledge of learning dynamics in linear neural networks, with a particular focus on λ-Balanced networks. We derive exact solutions for both aligned and unaligned λ-balanced networks that are interpretable and numerically stable. Our findings demonstrate the equivalence of deep λ-balanced network dynamics to shallow network dynamics for large values of $|\lambda|$. Furthermore, our results show that large values of $|\lambda|$ lead the network to learn in the Lazy Regime, while low values of $|\lambda|$ correspond to the Rich Learning Regime. This work also extends the applications of these analytical dynamics to continual learning, deriving an exact expression for the forgetting rate of a λ-balanced network.

This research is significant because it provides a deeper theoretical understanding of the relationship between weight initialization, network structure, and a network's learning regime. The exact solutions offered in this study allow us to determine the analytical dynamics of the network's representations and the network's Neural Tangent Kernel, furthering our understanding of Rich and Lazy Learning. Additionally, the insights gained from this study can inform the design of better weight initialization methods, ultimately leading to improved performance and reliability of neural networks in practical settings. Understanding the transition between Rich and Lazy Learning Regimes can help enhance neural network training, making this research relevant for advancing both theoretical and applied aspects of Machine Learning.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Pedro Mediano, for his invaluable guidance, encouragement, and insightful feedback throughout the course of this research. His expertise and support have been instrumental in the successful completion of this work. I am especially grateful to him for allowing me to self-propose the topic, which provided me with the opportunity to explore my interests in depth.

I am also deeply grateful to Clemenine Dominé from the Gatsby Computational Neuroscience Unit for her collaboration and support in this thesis. Her insights and feedback have significantly enriched this work.

Additionally, I would like to thank Alexandra Proca for providing feedback on my thesis and her overall assistance and collaboration. Her contributions and suggestions have been greatly appreciated.

Finally, I would like to express my heartfelt appreciation to my parents for their unwavering support, understanding, and encouragement throughout my academic journey. I am deeply grateful for everything they have done for me.

Contents

Chapter 1

Introduction

1.1 Motivation

Deep Learning Theory has become crucial in advancing Artificial Intelligence (AI) by providing deeper insights into neural network mechanisms and guiding the development of more effective models. Understanding these foundations is vital as AI increasingly integrates into various domains, necessitating robust, efficient, and explainable systems.

Mathematical approaches to AI offer rigorous frameworks for analyzing and understanding models, enabling predictions about behavior under various conditions and ensuring desired outcomes. However, these models can be complex and computationally challenging, often relying on simplifying assumptions that may not apply in real-world scenarios. On the other hand, engineering approaches focus on practical implementation, leading to adaptable and effective solutions through empirical performance and real-world feedback. Combining the strengths of both approaches, Deep Learning Theory has significantly improved AI model performance and explainability, fostering the development of trustworthy AI systems ([27]).

Deep Learning Theory has notably enhanced model performance by providing a structured understanding of neural networks, influencing the creation of better training algorithms and architectures. Insights into network architecture and optimization techniques have led to more efficient training processes and improved generalization to unseen data. For example, viewing neural network training through dynamical systems and optimal control perspectives has improved hyperparameter tuning and training efficiency. Understanding stochastic dynamics has facilitated implicit regularization, enhancing model performance ([28]).

In terms of explainability and safety, Deep Learning Theory has developed methods to interpret decisions made by deep learning models, contributing to their trustworthiness and widespread adoption. The Information Bottleneck Theory ([41]), for example, offers insights into how information is processed through network layers, aiding in understanding decision-making processes. Additionally, theoretical foundations have informed strategies to enhance robustness against adversarial attacks, ensuring the reliability and safety of AI systems in high-stakes applications ([7]).

In recent years, mechanistic interpretability has emerged as a key area of research within AI, aiming to elucidate how and why neural networks make decisions. This field intersects with AI safety, emphasizing the need to develop models that are not only powerful but also transparent and aligned with human values. Mechanistic interpretability efforts strive to decode the inner workings of AI systems, ensuring they operate reliably and predictably, which is essential for their safe deployment in critical applications ([32]).

Additionally, research has explored biologically plausible neural networks, which aim to mimic the learning processes observed in biological systems. A significant finding in this area is that different neural systems, whether biological or artificial, can converge to learn similar representations when exposed to comparable stimuli. This convergence is a topic of great interest in neuroscience and AI, as discussed in the NeurIPS 2023 workshop "UniReps: Unifying Representations in Neural

Models",([1]) which examines how similar representations arise and the implications for model fusion and reuse. Furthermore, research ([39]) has shown that biologically plausible neural networks can achieve comparable performance to traditional models, suggesting that understanding these similarities can lead to more robust and efficient AI systems.

Bridging AI with neuroscience, artificial neural networks (ANNs) draw inspiration from the brain's hierarchical information processing. Research has shown parallels between deep learning architectures and brain functions, such as the similarity between convolutional neural networks (CNNs) and the visual cortex's processing stages ([19]). This intersection extends to cognitive processes, with deep learning models simulating decision-making, memory, and learning, reflecting efficient human learning and adaptation ([25]).

The integration of AI and neuroscience reveals profound insights into how neural mechanisms can inform AI design and vice versa. For instance, ([34]) discusses how neuroscience findings guide the development of AI systems and highlights the bidirectional benefits of this interplay. Additionally, ([14]) emphasizes that while AI has historically been inspired by brain function, recent advances in AI are now offering new hypotheses and tools for understanding the brain, showcasing a symbiotic relationship between the fields . Furthermore, comprehensive reviews underscore that neural dynamics, learning algorithms, and computational models derived from brain studies are crucial for advancing AI technologies ([37]).

Recent advances have further integrated neuroscience principles into deep learning, enhancing AI capabilities and understanding brain functions. Developments in biologically plausible learning algorithms, such as unsupervised Hebbian learning, provide insights into brain representation formation ([30]). Neuro-inspired AI models, using naturalistic sensory signals, have solved complex real-world problems, highlighting the close relationship between AI and human cognition ([34]).

Feature learning remains a critical area in deep learning, enabling models to process complex data effectively. However, the exact mechanisms are not fully understood, with factors like network architecture and training regimen playing significant roles ([43]). Understanding Rich and Lazy Learning regimes, where networks undergo significant or minimal parameter changes respectively, is essential for optimizing feature learning and enhancing neural network performance ([6]).

Studying analytical solutions, particularly in linear neural networks, provides valuable insights into learning dynamics, enhancing model interpretability and guiding efficient training algorithm development ([27]).Linear networks, despite their simplicity, offer a tractable model for deriving exact solutions, informing the design of more complex architectures ([36]). These insights bridge the gap between empirical success and theoretical understanding, advancing both artificial and biological neural system comprehension.

1.2 Contributions

This research primarily builds upon and aims to extend the findings of two papers ([5], [23]) which broadly investigate the effects of network structure and weight initialisation on the learning regimes and learning dynamics of (linear [5]) neural networks.

Specifically, this research focuses on deriving exact solutions for learning dynamics in a more general setting and exploring their applications to continual learning. The code used in this project is available upon request. Below is a detailed outline of the contributions:

1. Exact Solutions for Learning Dynamics

- **Aligned λ-Balanced Networks**:
  - Derived exact solutions for the dynamics of aligned λ-Balanced networks.
- **Unaligned λ-Balanced Networks of Equal Dimensions**:

  - Provided numerically stable, interpretable solutions for learning and representation dynamics of unaligned λ-Balanced networks with equal input-output dimensions.
  - These solutions can be utilised to study the effect of network structure and weight initialisation to neural network training in a number of ways, including: alignment dynamics, gating , and continual learning ([5], [37])

- **Progress in Solutions for Unequal Dimensions**:
  - Made advancements towards exact solutions for unaligned λ-Balanced networks with unequal input-output dimensions.

Applications

- **Transition from Rich to Lazy Learning Regimes**:
  - Analyzed the effect of λ on the network's transition between Rich and Lazy learning regimes. Concluded that a larger magnitude of λ moves the network into the Lazy learning regime.
- **Impact of Initial Weight Configurations**:
  - Identified the influence of λ on the convergence and representational similarity matrices of a λ-Balanced network and concluded that a larger magnitude of λ increases speed of learning and makes a deep (3 layer) network behave increasingly more like a shallow (2 layer) network.
- **Continual Learning**:
  - Developed a framework for applying derived solutions to continual learning tasks, validated through simulations. Derived an expression for the rate of forgetting of a linear network, proposed next steps.
- **Preliminary Investigations into Non-Linear Networks**:
  - Conducted preliminary investigations into generalizing the derived solutions to ReLU and Tanh networks.
  - Showed that the approximation of the Balanced Coefficient λ^* of a ReLU network's weights is preserved through training.

Chapter 2

Background

2.1 Theoretical Background

2.1.1 A review of Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a fundamental theorem in linear algebra with numerous applications in data analysis, signal processing, and more. The SVD of a matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is given by:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{2.1}$$

Where:

- $\mathbf{U} \in \mathbb{R}^{n_1 \times n_1}$ is an orthonormal matrix;
- $\mathbf{S} \in \mathbb{R}^{n_1 \times n_2}$ is a diagonal matrix with non-negative real numbers on the diagonal;
- $\mathbf{V} \in \mathbb{R}^{n_2 \times n_2}$ is an orthonormal matrix.

In another form, the SVD can be written as:

- $\mathbf{U} \in \mathbb{R}^{n_1 \times n_1}$ is an orthonormal matrix,
- $\mathbf{S} \in \mathbb{R}^{n_1 \times n_2}$ is a diagonal matrix with non-negative real numbers on the diagonal,
- $\mathbf{V} \in \mathbb{R}^{n_2 \times n_2}$ is an orthonormal matrix.

Where s_α are the singular values, and $\mathbf{u}_\alpha$ and $\mathbf{v}_\alpha$ are the left and right singular vectors respectively.

The Singular Value Decomposition can be visualized as a transformation that decomposes the original matrix $\mathbf{X}$ into three simpler matrices that capture the essential features of the data. The process can be described as follows:

1. $\mathbf{V}^T$: Rotates the data to align with the principal axes.
2. $\mathbf{S}$: Scales the data along the principal axes.
3. $\mathbf{U}$: Rotates the scaled data back to the original coordinate system.

The SVD thus provides a way to understand the structure of the data, identify patterns, and reduce dimensionality while preserving essential information.

We say the matrices $\mathbf{X}$, $\mathbf{Y}$ are **aligned** if they have the same singular vectors, i.e., $\mathbf{X} = \mathbf{U}\mathbf{S}_X\mathbf{V}^T$, $\mathbf{Y} = \mathbf{U}\mathbf{S}_Y\mathbf{V}^T$.

Suppose $\mathbf{XY} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ is the Singular Value Decomposition of the product of two matrices $\mathbf{X}$, $\mathbf{Y}$. Then there exists an orthonormal matrix $\mathbf{R}$ such that:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{R}^T, \quad \mathbf{Y} = \mathbf{R}\mathbf{S}\mathbf{V}^T \tag{2.2}$$

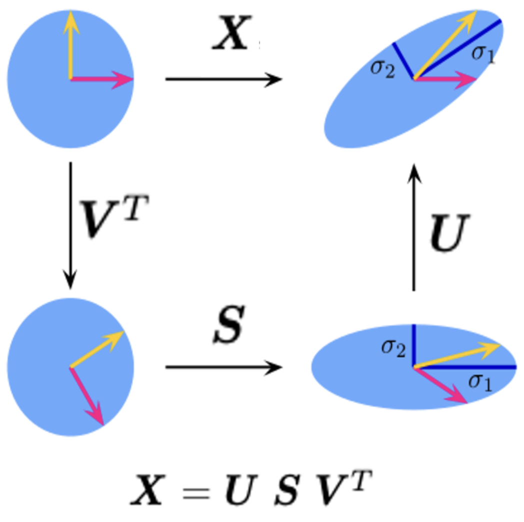


Figure 2.1: Singular Value Decomposition (SVD) Process ([18]). The figure shows how a matrix X is decomposed into three matrices U, S, and V^T, capturing the essential features of the data through rotation, scaling, and another rotation.

Suppose $n_1 > n_2$. Then we can write $\boldsymbol{U} := (\tilde{\boldsymbol{U}}\ \tilde{\boldsymbol{U}}_\perp)$ with $\tilde{\boldsymbol{U}} \in \mathbb{R}^{n_2 \times n_2}$ and $\tilde{\boldsymbol{U}}_\perp)$ consists of the remaining singular vectors to complete the basis. In addition, we write $\boldsymbol{V} := (\tilde{\boldsymbol{V}}\ \tilde{\boldsymbol{V}}_\perp)$ with $\tilde{\boldsymbol{V}} \in \mathbb{R}^{n_2 \times n_2}$ and $\tilde{\boldsymbol{V}}_\perp)$ is a matrix of zeros.

Suppose $n_1 < n_2$. Then we can write $\boldsymbol{U} := (\tilde{\boldsymbol{U}}\ \tilde{\boldsymbol{U}}_\perp)$ with $\tilde{\boldsymbol{U}} \in \mathbb{R}^{n_1 \times n_1}$ and $\tilde{\boldsymbol{U}}_\perp)$ is a matrix of zeros. In addition, we write $\boldsymbol{V} := (\tilde{\boldsymbol{V}}\ \tilde{\boldsymbol{V}}_\perp)$ with $\tilde{\boldsymbol{V}} \in \mathbb{R}^{n_2 \times n_2}$ and $\tilde{\boldsymbol{V}}_\perp)$ consists of the remaining singular vectors to complete the basis.

2.2 Previous Work on Linear Network Dynamics

2.2.1 Experimental Setup

The experimental setup detailed in this section is inspired by the ones in ([36], [5]). We consider a supervised learning regression task containing P training pairs $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1\ldots P}$. Our aim is to predict the output vectors $\mathbf{y}_n \in \mathbb{R}^{n_o}$ from the input vectors $\mathbf{x}_n \in \mathbb{R}^{n_i}$ by the use of an n-layer linear neural network, where the output $\mathbf{Y}$ is expressed as:

$$\mathbf{Y} = \mathbf{W}_{n-1}\mathbf{W}_{n-2}\ldots\mathbf{W}_1\mathbf{X}$$

The neural network will be trained using full batch gradient descent utilizing a learning rate η (or equivalently, a time constant $\tau = 1/\eta$) on the Mean Squared Error (MSE) loss.

$$\mathcal{L}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{1}{2P}\left\|\hat{\boldsymbol{Y}} - \boldsymbol{Y}\right\|_F^2 \tag{2.3}$$

$$= \frac{1}{2P}\left\|\boldsymbol{W}_{n-1}\boldsymbol{W}_{n-2}\cdots\boldsymbol{W}_1\boldsymbol{X} - \boldsymbol{Y}\right\|_F^2 \tag{2.4}$$

The input and output correlation matrices of the dataset are defined as:

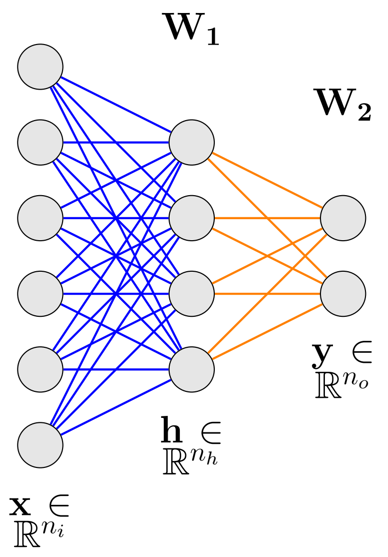
$$\tilde{\boldsymbol{\Sigma}}^{xx} = \frac{1}{P}\sum_{n=1}^{P}\bar{x}_n\bar{x}_n^T \in \mathbb{R}^{N_i \times N_i} \quad \text{and} \quad \tilde{\boldsymbol{\Sigma}}^{yx} = \frac{1}{P}\sum_{n=1}^{P}\bar{y}_n\bar{x}_n^T \in \mathbb{R}^{N_o \times N_i} \tag{2.5}$$

In this text, we will assume whitened inputs, that is:

$$\tilde{\boldsymbol{\Sigma}}^{xx} = \mathbb{I} \tag{2.6}$$

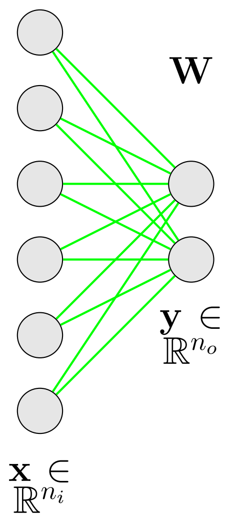
It has been shown ([36]) that under these conditions the network will converge to the global minimum, characterised by $(\mathbf{W}_{n-1}\mathbf{W}_{n-2}\dots\mathbf{W}_1) = \mathbf{\Sigma}^{yx}$. However, our research interest extends beyond mere convergence; it encompasses the dynamics of the learning process as well as the internal representations formed within the network during training. To gain a comprehensive understanding, we examine two distinct scenarios: a three-layer (deep) neural network represented by $\mathbf{Y} = \mathbf{W}_2\mathbf{W}_1\mathbf{X}$ and a two-layer (shallow) neural network represented by $\mathbf{Y} = \mathbf{W}\mathbf{X}$.

It is important to note the dimensions of each of the weight matrices: $\mathbf{W}_1 \in \mathbb{R}^{n_h \times N_i}$ and $\mathbf{W}_2 \in \mathbb{R}^{n_o \times n_h}$, $\mathbf{W} \in \mathbb{R}^{n_o \times n_i}$ where N_h is the number of neurons in the hidden layer. The gradient descent process commences with initial weights represented by $\mathbf{W}_2(0)$ and $\mathbf{W}_1(0)$, $\mathbf{W}(0)$.



$\mathbf{W_1}$
$\mathbf{W_2}$
$\mathbf{y} \in \mathbb{R}^{n_o}$
$\mathbf{h} \in \mathbb{R}^{n_h}$
$\mathbf{x} \in \mathbb{R}^{n_i}$

Figure 2.2: Diagram of a 3-layer (Deep) neural network.



$\mathbf{W}$
$\mathbf{y} \in \mathbb{R}^{n_o}$
$\mathbf{x} \in \mathbb{R}^{n_i}$

Figure 2.3: Diagram of a 2-layer (Shallow) neural network.

2.2.2 Gradient Flow

The gradient descent update equations are given by ([35], [36]):

$$\boldsymbol{W_1}^{(i+1)} \leftarrow \boldsymbol{W_1}^{(i)} - \eta(\boldsymbol{W_2}^{(i)})^T(\boldsymbol{W_2}^{(i)}\boldsymbol{W_1}^{(i)}\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{X}^T \tag{2.7}$$

$$\boldsymbol{W_2}^{(i+1)} \leftarrow \boldsymbol{W_2}^{(i)} - \eta(\boldsymbol{W_2}^{(i)}\boldsymbol{W_1}^{(i)}\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{X}^T(\boldsymbol{W_1}^{(i)})^T \tag{2.8}$$

Hence

$$\frac{1}{\eta}(\boldsymbol{W_1}^{(i+1)} - \boldsymbol{W_1}^{(i)}) = -(\boldsymbol{W_2}^{(i)})^T(\boldsymbol{W_2}^{(i)}\boldsymbol{W_1}^{(i)}\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{X}^T \tag{2.9}$$

$$\frac{1}{\eta}(\boldsymbol{W_2}^{(i+1)} - \boldsymbol{W_2}^{(i)}) = -(\boldsymbol{W_2}^{(i)}\boldsymbol{W_1}^{(i)}\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{X}^T(\boldsymbol{W_1}^{(i)})^T \tag{2.10}$$

As the learning rate $\eta \to 0$ these difference equations approach the following differential equations ([35], [36]):

$$\tau\frac{d}{dt}\boldsymbol{W_1} = (\boldsymbol{W_2})^T\left(\tilde{\mathbf{\Sigma}}^{yx} - \boldsymbol{W_2}\boldsymbol{W_1}\tilde{\mathbf{\Sigma}}^{xx}\right), \tag{2.11}$$

$$\tau\frac{d}{dt}\mathbf{W_2} = \left(\tilde{\mathbf{\Sigma}}^{yx} - \mathbf{W_2}\mathbf{W_1}\tilde{\mathbf{\Sigma}}^{xx}\right)(\mathbf{W_1})^T, \tag{2.12}$$

When the learning rate is sufficiently small, the differential equations above are a reasonable approximation for the real dynamics of the weights through learning in the neural network. Since in practice a very small learning rate is used, we are interested in finding the exact solutions to the dynamics of these differential equations. Gradient flow is a useful approximation for understanding neural network learning dynamics, as it allows us to express the gradient descent process as a set of differential equations ([36])

2.2.3 Balanced Condition

Definition 2.2.1 (Definition of λ-Balanced property ([36], [31])) *The weights $\boldsymbol{W_1}, \boldsymbol{W_2}$ are λ-Balanced if and only if there exists a **Balanced Coefficient** $\lambda \in \mathbb{R}$ such that:*

$$B(\boldsymbol{W_1}, \boldsymbol{W_2}) = \boldsymbol{W_2}^T \boldsymbol{W_2} - \boldsymbol{W_1} \boldsymbol{W_1}^T = \lambda \mathbb{I} \tag{2.13}$$

*where B is called the **Balanced Computation**. For $\lambda = 0$ we have **Zero Balanced**.*

Theorem 2.2.1 (Balance Condition Persists Through Training) *Suppose at Initialisation*

$$\boldsymbol{W_2(0)}^T \boldsymbol{W_2(0)} - \boldsymbol{W_1(0)} \boldsymbol{W_1(0)}^T = \lambda \mathbb{I} \tag{2.14}$$

Then for all $t \geq 0$

$$\boldsymbol{W_2(t)}^T \boldsymbol{W_2(t)} - \boldsymbol{W_1(t)} \boldsymbol{W_1(t)}^T = \lambda \mathbb{I} \tag{2.15}$$

Proof 1 (Proof of 2.2.1 ([36])) *Consider:*

$$\begin{aligned}
\tau \frac{d}{dt} \left[\boldsymbol{W_2}(t) \boldsymbol{W_2}(t)^T - \boldsymbol{W_1}(t) \boldsymbol{W_1}(t)^T \right] &= \left(\tau \frac{d}{dt} \boldsymbol{W_2}(t) \right) \boldsymbol{W_2}(t)^T + \boldsymbol{W_2}(t) \left(\tau \frac{d}{dt} \boldsymbol{W_2}(t) \right)^T \\
&- \left(\tau \frac{d}{dt} \boldsymbol{W_1}(t) \right) \boldsymbol{W_1}(t)^T - \boldsymbol{W_1}(t) \left(\tau \frac{d}{dt} \boldsymbol{W_1}(t) \right)^T \\
&= \boldsymbol{W_1}(t) \left(\tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{W_2}(t) \boldsymbol{W_1}(t) \tilde{\boldsymbol{\Sigma}}^{xx} \right)^T \boldsymbol{W_2}(t) \boldsymbol{W_2}(t) \\
&+ \boldsymbol{W_2}(t) \left(\tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{W_2}(t) \boldsymbol{W_1}(t) \tilde{\boldsymbol{\Sigma}}^{xx} \right) \boldsymbol{W_1}(t) \boldsymbol{W_1}(t)^T \\
&- \boldsymbol{W_2}(t)^T \left(\tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{W_2}(t) \boldsymbol{W_1}(t) \tilde{\boldsymbol{\Sigma}}^{xx} \right) \boldsymbol{W_1}(t) \\
&- \boldsymbol{W_1}(t) \left(\tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{W_2}(t) \boldsymbol{W_1}(t) \tilde{\boldsymbol{\Sigma}}^{xx} \right)^T \boldsymbol{W_2}(t) \\
&= \boldsymbol{0}
\end{aligned}$$

Definition 2.2.2 (Aligned Weights) *Let $\boldsymbol{Y} = \boldsymbol{W}_2(t) \boldsymbol{W}_1(t) X$ be a three-layer linear network. We say the initial weights $\boldsymbol{W}_2(t) \boldsymbol{W}_1(t)$ are **aligned** if they share the same singular vectors as the task $\boldsymbol{\Sigma}^{yx}$.*

i

Definition 2.2.3 (Input and Output Network Representations ([5])) *Let $\boldsymbol{Y} = \boldsymbol{W}_2(t) \boldsymbol{W}_1(t) X$ be a three layer linear network. We define the Input Network Representations at time-step t as $\boldsymbol{W}_1(t)^T \boldsymbol{W}_1(t)$. We define the Output Network Representations as time-step t as $\boldsymbol{W}_2(t) \boldsymbol{W}_2(t)^T$*

2.2.4 Existing Analytical Solutions for Network Dynamics

Linear network learning dynamics have been shown to qualitatively resemble those in the nonlinear case ([36]). Previous research by ([36])has derived analytical solutions for the learning dynamics of aligned, zero-Balanced networks of arbitrary depth.
Additionally, there is a substantial body of work that provides analytical solutions for the learning dynamics of more complex networks with a single output ([24]). While this research significantly contributes to the field, it does not fully address the complexities inherent in networks with multiple outputs.
More recently, ([5]) have derived analytical solutions for the learning and representation dynamics of unaligned, zero-Balanced three-layer networks with unequal input-output dimensions. These analytical results are particularly valuable for studying the network's learning regime and understanding its behavior in various contexts.

	0-Balanced	**λ-Balanced**
Aligned	Unequal input/output: [38]	Unequal input/output: This Thesis
Not aligned	Equal input/output: [12] Unequal input/output: [5]	Equal input/output: This Thesis

Table 2.1: Summary of alignment and balance conditions

Below is a table visualising results in the area of analytical learning dynamics of neural networks: We present the derivation for the learning dynamics of an Aligned 0-Balanced Network together with the main result for didactic purposes:

Theorem 2.2.2 ***[Dynamics of Aligned 0-Balanced Network]***
To solve for the dynamics of $\boldsymbol{W}_1, \boldsymbol{W}_2$ over time, we decompose the input-output correlations through the singular value decomposition (SVD) (this proof is paraphrased from [38]),

$$\tilde{\boldsymbol{\Sigma}}^{yx} = \boldsymbol{USV}^T = \sum_{\alpha=1}^{N_1} s_\alpha \bar{u}^\alpha (\bar{v}^\alpha)^T, \tag{2.16}$$

and then change variables to $\overline{\boldsymbol{W}}_1, \overline{\boldsymbol{W}}_2$ where $\boldsymbol{W}_1 = \boldsymbol{R}\overline{\boldsymbol{W}}_1\boldsymbol{V}^T$ and $\boldsymbol{W}_2 = \boldsymbol{U}\overline{\boldsymbol{W}}_2\boldsymbol{R}^T$

$$\tau \frac{d}{dt}(\boldsymbol{R}\overline{\boldsymbol{W}}_1\boldsymbol{V}^T) = \boldsymbol{R}\overline{\boldsymbol{W}}_2\boldsymbol{U}^T(\tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{U}\overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{V}^T\tilde{\boldsymbol{\Sigma}}^{xx}), \tag{2.17}$$

$$\begin{aligned} \tau \frac{d}{dt}\overline{\boldsymbol{W}}_1 &= \overline{\boldsymbol{W}}_2\boldsymbol{U}^T(\boldsymbol{USV}^T - \boldsymbol{U}\overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{V}^T)\boldsymbol{V} && (2.18)\\ &= \overline{\mathbf{W}}_2(\boldsymbol{S} - \overline{\mathbf{W}}_2\overline{\mathbf{W}}_1), && (2.19)\end{aligned}$$

$$\tau \frac{d}{dt}(\boldsymbol{U}\overline{\boldsymbol{W}}_2\boldsymbol{R}^T) = (\tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{U}\overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{V}^T\tilde{\boldsymbol{\Sigma}}^{xx})\boldsymbol{V}^T\overline{\boldsymbol{W}}_1\boldsymbol{R}^T, \tag{2.20}$$

Here we use the whitened inputs assumption:

$$\begin{aligned} \tau \frac{d}{dt}\overline{\boldsymbol{W}}_2 &= \boldsymbol{U}^T(\boldsymbol{USV}^T - \boldsymbol{U}\overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{V}^T)\boldsymbol{V}\overline{\boldsymbol{W}}_1^T && (2.21)\\ &= (\boldsymbol{S} - \overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1)\overline{\boldsymbol{W}}_1^T, && (2.22)\end{aligned}$$

Consider $c_\alpha = \overline{\mathbf{W}}1^{\alpha\alpha}$ and $d\alpha = \overline{\mathbf{W}}_2^{\alpha\alpha}$ as the α^{th} diagonal elements of the first and second matrices. These elements represent the strength of mode α transmitted through the input-to-hidden and hidden-to-output weights, respectively. The scalar dynamics are given by

$$\tau \frac{d}{dt}c_\alpha = d_\alpha(s_\alpha - c_\alpha d_\alpha), \tag{2.23}$$

$$\tau \frac{d}{dt}d_\alpha = c_\alpha(s_\alpha - c_\alpha d_\alpha) \tag{2.24}$$

$$\tau \frac{d}{dt}a_\alpha = (c_\alpha^2 + d_\alpha^2)(s - a_\alpha) \tag{2.25}$$

$$\tau \frac{d}{dt}a_\alpha = (2a_\alpha)(s - a_\alpha) \tag{2.26}$$

$$t = \frac{\tau}{2}\int_{a^0}^{a^f} \frac{da}{a(s-a)} = \frac{\tau}{2s}\ln\frac{a^f(s-a^0)}{a^0(s-a^f)} \tag{2.27}$$

The complete learning trajectory can be determined by solving for a_f, resulting in

$$a_\alpha(t) = \frac{s_\alpha e^{2s_\alpha t/\tau}}{e^{2s_\alpha t/\tau} - 1 + s_\alpha/a_\alpha^0}. \tag{2.28}$$

$$\mathbf{W}_2(t)\mathbf{W}_1(t) = U\overline{\mathbf{W}}_2(t)\overline{\mathbf{W}}_1(t)V^T = UA(t)V^T. \tag{2.29}$$

This result implies that an Aligned 0-Balanced Network remains aligned through training, and the singular values are independently of each other (as the differential equation is decoupled). These two properties hold for learning dynamics of Aligned Networks of any depth ([36]).

Note that both alignement and λ remain constant through training in Linear Networks. Hence we introduce the terms λ-Balanced Network and Aligned λ-Balanced Network to signify networks with λ-Balanced weights and Aligned λ-Balanced weights respectively.

Solution for Shallow Not Aligned Network

Theorem 2.2.3 (Gradient Flow Dynamics of a 2 Layer (Shallow) Linear Network) *Consider the 2 layer linear network $Y = WX$. The gradient flow equation for W is given by:*

$$W(t) = \tilde{\mathbf{\Sigma}}^{yx} + (\boldsymbol{W(0)} - \tilde{\mathbf{\Sigma}}^{yx})e^{-\frac{t}{\tau}}$$

This result was established in ([36]). We omit the proof.

Solution for Not Aligned 0-Balanced Network

Theorem 2.2.4 ***[Gradient Flow Dynamics as a Matrix Riccati equation with known solution]***

Define

$$\boldsymbol{Q}(t) = \begin{bmatrix} \boldsymbol{W}_1^T(t) \\ \boldsymbol{W}_2(t) \end{bmatrix}$$

Let $\boldsymbol{F}$ be an $(n_h + n_o) \times (n_h + n_i)$ matrix such that:

$$\tau \frac{d}{dt}(\boldsymbol{QQ}^T) = \boldsymbol{FQQ}^T + \boldsymbol{QQ}^T\boldsymbol{F} - (\boldsymbol{QQ}^T)^2.$$

For all t and some $\tau \in \mathbf{R}$
The equation above is a Matrix Riccati equation. Assuming further that $\mathbf{Q}(0)$ has full rank, the equation has unique solution given by:

$$\boldsymbol{QQ}^T(t) = e^{\boldsymbol{F}\frac{t}{\tau}}\boldsymbol{Q}(0)\left[\boldsymbol{I} + \frac{1}{2}\boldsymbol{Q}(0)^T\left(e^{\boldsymbol{F}\frac{t}{\tau}}\boldsymbol{F}^{-1}e^{\boldsymbol{F}\frac{t}{\tau}} - \boldsymbol{F}^{-1}\right)\boldsymbol{Q}(0)\right]^{-1}\boldsymbol{Q}(0)^T e^{\boldsymbol{F}\frac{t}{\tau}}$$

Let $\tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T = \mathbf{\Sigma}^{yx}$, $\boldsymbol{USV}^T = \boldsymbol{W}_2(0)\boldsymbol{W}_1(0)$ be the Singular Value Decomposition of the task and initial weights Respectively.

We define the following assumptions ([5]):

Assumption 2.2.1 *Define $\mathbf{B} = U^T\widetilde{U} + V^T\widetilde{V}$ and $\mathbf{C} = U^T\widetilde{U} - V^T\widetilde{V}$. $\mathbf{B}$ is non-singular.*

Assumption 2.2.2 *The input data is whitened, that is $\tilde{\Sigma}^{xx} = \mathbf{I}$*

Assumption 2.2.3 *The network's weight matrices are zero-Balanced at the beginning of training.*

Assumption 2.2.4 *The input-output correlation of the task and the initial state of the network function have full rank, that is $rank(\tilde{\mathbf{\Sigma}}^{xy}) = rank(\mathbf{W}_2(0)\mathbf{W}_1(0)) = N_i = N_o$. This implies that the network is not bottlenecked, i.e. $N_h \geq min(N_i, N_o)$*

Theorem 2.2.5 (Numerically stable Fukumizu equation) ***Theorem 3.1*** *Under the assumptions of whitened inputs, zero-Balanced weights, full rank, and $\boldsymbol{B}$ non-singular 3.1, the temporal dynamics of $\boldsymbol{QQ}^T$ are*

$$\boldsymbol{QQ}^T(t) = \boldsymbol{Z}\left[4e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\boldsymbol{B}^{-1}\boldsymbol{S}^{-1}(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}} + \left(\boldsymbol{I} - e^{-2\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\right)\widetilde{\boldsymbol{S}}^{-1}\right. \tag{2.30}$$

$$\left. - e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\boldsymbol{B}^{-1}\boldsymbol{C}\left(e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}} - \boldsymbol{I}\right)\widetilde{\boldsymbol{S}}^{-1}\boldsymbol{C}^T(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\right.$$

$$+\frac{t}{\tau}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\boldsymbol{B}^{-1}\left(\boldsymbol{V}^T\widetilde{\boldsymbol{V}}_\perp\widetilde{\boldsymbol{V}}_\perp^T\boldsymbol{V}+\boldsymbol{U}_\perp^T\widetilde{\boldsymbol{U}}_\perp^T\boldsymbol{U}\right)(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\boldsymbol{Z}^T$$

$$+4e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\boldsymbol{B}^{-1}\left(\boldsymbol{V}^T\widetilde{\boldsymbol{V}}_\perp\widetilde{\boldsymbol{V}}_\perp^T\boldsymbol{V}+\boldsymbol{U}_\perp^T\widetilde{\boldsymbol{U}}_\perp^T\boldsymbol{U}\right)(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\widetilde{\boldsymbol{S}}^{-1}\boldsymbol{Z}^T$$

with

$$\boldsymbol{Z}=\begin{bmatrix}\widetilde{\boldsymbol{V}}\left(\boldsymbol{I}-e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\boldsymbol{C}^T(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\right)+2\widetilde{\boldsymbol{V}}_\perp\widetilde{\boldsymbol{V}}_\perp^T\boldsymbol{V}(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\\ \widetilde{\boldsymbol{U}}\left(\boldsymbol{I}+e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\boldsymbol{C}^T(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\right)+2\widetilde{\boldsymbol{U}}_\perp\widetilde{\boldsymbol{U}}_\perp^T\boldsymbol{U}(\boldsymbol{B}^T)^{-1}e^{-\widetilde{\boldsymbol{S}}\frac{t}{\tau}}\end{bmatrix}. \tag{2.31}$$

During the rest of the report, we will continue to assume assumptions (**2.2.1, 2.2.4**). Assumption **2.2.2** will also be maintained in some cases.

2.3 Existing Applications of this Work

2.3.1 The Neural Tangent Kernel (NTK)

The Neural Tangent Kernel (NTK) ([17]) is a concept that has garnered significant attention in the field of deep learning. It provides a new perspective on understanding the behavior of neural networks, particularly in the regime of wide networks. The NTK framework allows us to analyze how neural networks learn by studying the dynamics of the training process through the lens of kernel methods.The sturcutre of this section is inspired by ([42]).

Consider a neural network $f(\mathbf{x};\theta)$ where $\mathbf{x}\in\mathbb{R}^n$ represents the input and θ represents the parameters of the network. The Jacobian matrix J of the network is defined as:

$$J=\frac{\partial f}{\partial \mathbf{x}}\in\mathbb{R}^{m\times n} \tag{2.32}$$

where m is the output dimension. The gradient of the function with respect to the parameters θ can be expressed as:

$$\nabla_\theta f=J^\top\in\mathbb{R}^{n\times m} \tag{2.33}$$

The NTK, K, is defined as a kernel function $K:\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}$ which measures the similarity between the gradients of the neural network with respect to its parameters:

$$K(\mathbf{x},\mathbf{x}';\theta)=\nabla_\theta f(\mathbf{x};\theta)^\top\nabla_\theta f(\mathbf{x}';\theta) \tag{2.34}$$

During training, the parameters θ of the neural network are updated using gradient descent. The update rule can be written as:

$$\frac{d\theta}{dt}=-\nabla_\theta L(\theta) \tag{2.35}$$

where $L(\theta)$ is the loss function. Using the NTK, the evolution of the function $f(\mathbf{x};\theta)$ during training can be described by the differential equation:

$$\frac{df(\mathbf{x};\theta)}{dt}=-K(\mathbf{x},\mathbf{x};\theta)\nabla_\theta L(\theta) \tag{2.36}$$

For an infinitely wide neural network, the NTK K remains constant during training. This leads to a linearization of the training dynamics, allowing for a more tractable analysis.

The NTK framework reveals that, under certain conditions, wide neural networks can be approximated by kernel methods. This insight bridges the gap between neural networks and classical machine learning algorithms, providing a theoretical foundation for understanding why deep networks perform well in practice.

Moreover, the NTK can be used to analyze the generalization capabilities of neural networks. By studying the properties of the kernel, researchers can gain insights into how well the network will perform on unseen data.

2.3.2 Expressing NTK in and Network Representational Similarity Matrices in terms of Network Representations

We define task-relevant representational similarity matrices (RSM) ([21]), which are the kernel matrix $\phi(x)^T\phi(x')$, of the neural representations within the hidden layer ([5])([21]) as:

$$\mathrm{RSM}_I = \mathbf{X}^T\mathbf{W}_1^T\mathbf{W}_1(t)\mathbf{X},$$
$$\mathrm{RSM}_O = \mathbf{Y}^T\left(\mathbf{W}_2\mathbf{W}_2^T(t)\right)^+\mathbf{Y},$$

In [5] the authors show that the NTK of a Linear Neural Network can be expressed in the following form:

$$\mathrm{NTK} = \mathbf{I}_{N_o}\otimes\mathbf{X}^T\mathbf{W}_1^T\mathbf{W}_1(t)\mathbf{X} + \mathbf{W}_2\mathbf{W}_2^T(t)\otimes\mathbf{X}^T\mathbf{X} \tag{2.37}$$

2.3.3 Feature Learning and Rich and Lazy Learning

Feature Learning and Learning Regimes in Neural Networks and the Brain

Feature learning is the backbone of deep learning, enabling models to understand and process complex data. Features are representations or patterns extracted from raw data during the learning process, ranging from low-level edges in images to high-level abstract concepts. The automatic learning and extraction of useful features is a key factor in the success of deep learning, allowing for complex tasks such as image recognition, natural language processing, and speech recognition to be performed more effectively than traditional machine learning methods reliant on manually crafted features ([4]).

Despite this success, the exact mechanisms by which features are learned remain an active area of research ([3]). Feature learning involves adjusting network weights during training, but the specific processes through which certain patterns are identified and enhanced while others are suppressed are not fully understood. Multiple mechanisms, including alignment, disalignment, and rescaling within network layers, influence this process and are affected by factors such as network architecture, data nature, and learning regimes ([43]).

The concepts of Rich and Lazy learning regimes are crucial for understanding feature learning in neural networks. In the Rich (feature learning) regime, significant changes occur during training, allowing the network to learn complex, high-dimensional features. This regime involves dynamic parameter evolution and the development of intricate internal representations. In contrast, the Lazy (kernel learning) regime, associated with the Neural Tangent Kernel (NTK) ([17], [13]), involves minimal parameter changes, with the network acting more like a fixed feature extractor where weights are only slightly adjusted, and learning dynamics are largely linear ([6]).

Understanding the relationship between network structure, weight initialization, and these learning regimes is essential for advancing knowledge of feature learning. The transition between Rich and Lazy regimes depends on factors such as network width, learning rates, and initialization scale. Analytical frameworks, such as those provided in ([43]), help study these transitions in minimal finite-width models, showing that feature learning mechanisms like alignment are present only in the Rich phase and absent in the Lazy phase.

In the brain, both Rich and Lazy learning strategies are likely utilized depending on the task and the region involved. Early sensory areas may adopt Lazy learning to maintain flexibility, while higher-order areas may employ Rich learning to optimize for specific tasks. This balance allows the brain to process a wide range of information efficiently while developing specialized capabilities for critical functions. Studies modeling brain learning processes with artificial neural networks provide insights into how different brain regions balance these strategies, with higher-order cortical areas adapting to task demands and early sensory areas maintaining general representations ([8], [10], [9], [11]).

Understanding these dynamics informs both artificial neural network design and the functioning of biological neural circuits, leading to more efficient learning algorithms and enhanced comprehension of brain function and adaptability. The study of Rich and Lazy learning regimes offers valuable insights into developing more effective and interpretable deep learning models.

2.3.4 Continual Learning

We can formally establish the context of continual learning in a three-layer (deep) linear network as outlined in existing literature ([5]):

Consider the scenario of training a two-layer deep linear network on a sequence of tasks $\mathcal{T}_a, \mathcal{T}_b, \mathcal{T}_c, \ldots$, each with corresponding correlation functions $\mathcal{T}_a = \tilde{\mathbf{\Sigma}}_a^{yx}, \mathcal{T}_b = \tilde{\mathbf{\Sigma}}_b^{yx}$. Subsequently, the full batch loss of the i-th task at any training time point i

$$\mathcal{L}_i = \frac{1}{2P} \left\| \mathbf{W}_2 \mathbf{W}_1 \mathbf{X}_i - \mathbf{Y}_i \right\|_F^2$$

After completing the training process until convergence for task $\mathcal{T}_j$, we understand that the network function is $\mathbf{W}_2 \mathbf{W}_1 = \tilde{\mathbf{U}} \tilde{\mathbf{S}} \tilde{\mathbf{V}}^T = \tilde{\mathbf{\Sigma}}_j^{yx}$.

Utilizing the assumption of whitened inputs, the entire batch loss for the ith task is

$$\begin{aligned}
\mathcal{L}_i(\mathcal{T}_j) &= \frac{1}{2P} \left\| \tilde{\mathbf{\Sigma}}_j^{yx} \mathbf{X}_i - \mathbf{Y}_i \right\|_F^2 \\
&= \frac{1}{2P} \operatorname{Tr}\left(\left(\tilde{\mathbf{\Sigma}}_j^{yx} \mathbf{X}_i - \mathbf{Y}_i \mid \right) \left(\tilde{\mathbf{\Sigma}}_j^{yx} \mathbf{X}_i - \mathbf{Y}_i \mid \right)^T \right) \\
&= \frac{1}{2P} \operatorname{Tr}\left(\tilde{\mathbf{\Sigma}}_j^{yx} \mathbf{X}_i \mathbf{X}_i^T \tilde{\mathbf{\Sigma}}_j^{yx^T} \right) - \frac{1}{P} \operatorname{Tr}\left(\tilde{\mathbf{\Sigma}}_j^{yx} \mathbf{X}_i \mathbf{Y}_i^T \right) + \frac{1}{2P} \operatorname{Tr}\left(\mathbf{Y}_i \mathbf{Y}_i^T \right) \\
&= \frac{1}{2} \operatorname{Tr}\left(\tilde{\mathbf{\Sigma}}_j^{yx} \tilde{\mathbf{\Sigma}}_j^{yx^T} \right) - \operatorname{Tr}\left(\tilde{\mathbf{\Sigma}}_j^{yx} \tilde{\mathbf{\Sigma}}_i^{yx^T} \right) + \frac{1}{2} \operatorname{Tr}\left(\tilde{\mathbf{\Sigma}}_i^{yy} \right) \\
&= \frac{1}{2} \operatorname{Tr}\left(\left(\tilde{\mathbf{\Sigma}}_j^{yx} - \tilde{\mathbf{\Sigma}}_i^{yx} \right) \left(\tilde{\mathbf{\Sigma}}_j^{yx} - \tilde{\mathbf{\Sigma}}_i^{yx} \right)^T - \tilde{\mathbf{\Sigma}}_i^{yx} \tilde{\mathbf{\Sigma}}_i^{yx^T} \right) + \frac{1}{2} \left(\tilde{\mathbf{\Sigma}}_i^{yy} \right) \\
&= \frac{1}{2} \left\| \tilde{\mathbf{\Sigma}}_j^{yx} - \tilde{\mathbf{\Sigma}}_i^{yx} \right\|_F^2 \underbrace{- \frac{1}{2} \operatorname{Tr}\left(\tilde{\mathbf{\Sigma}}_i^{yx} \tilde{\mathbf{\Sigma}}_i^{yx^T} \right) + \frac{1}{2} \left(\tilde{\mathbf{\Sigma}}_i^{yy} \right)}_{c}.
\end{aligned}$$

Hence, the extent of forgetting, denoted as $\mathcal{F}$ for task $\mathcal{T}_i$ during training on task $\mathcal{T}_k$ subsequent to training the network on task $\mathcal{T}_j$, specifically, the relative change in loss, is entirely dictated by the similarity structure among tasks.

$$\begin{aligned}
\mathcal{F}_i(\mathcal{T}_j, \mathcal{T}_k) &= \mathcal{L}_i(\mathcal{T}_k) - \mathcal{L}_i(\mathcal{T}_j) \\
&= \frac{1}{2} \left\| \tilde{\mathbf{\Sigma}}_k^{yx} - \tilde{\mathbf{\Sigma}}_i^{yx} \right\|_F^2 + c - \frac{1}{2} \left\| \tilde{\mathbf{\Sigma}}_j^{yx} - \tilde{\mathbf{\Sigma}}_i^{yx} \right\|_F^2 - c \\
&= \frac{1}{2} \left(\left\| \tilde{\mathbf{\Sigma}}_k^{yx} - \tilde{\mathbf{\Sigma}}_i^{yx} \right\|_F^2 - \left\| \tilde{\mathbf{\Sigma}}_j^{yx} - \tilde{\mathbf{\Sigma}}_i^{yx} \right\|_F^2 \right).
\end{aligned}$$

It is important to note that the amount of forgetting is a function of the weight trajectories, finding analytical solutions for the weight trajectories will enable us to solve for trajectories of forgetting as well.

Chapter 3

Exact Solutions for Learning Dynamics

3.1 Random Weight Initialisations and λ-Balanced Property

Throughout this work, we assume that initial weights are λ-Balanced. However, in practice, weights are not initialized with that goal in mind. Usually, a weight matrix $\mathbf{W}$ is initialized with some random distribution centered around 0, with variance inversely proportional to the number of layers on which $\mathbf{W}$ has a direct effect ([15], [26], [16]). In this section, we show that many common initialization techniques lead to λ-Balanced weights in expectation. Furthermore, as the size of a network tends to infinity, these random weights are λ-Balanced in probability.

We do this by first finding the expectation and variance of the balance computation for two adjacent weight matrices, $\mathbf{W}_{i+1}$ and $\mathbf{W}_i$, initialized under a normal distribution with zero mean. Subsequently, we describe how network structure and size can impact the expectation and variance of the balance computation.

Theorem 3.1.1 ***[Random Weight Initialization Leads to Balanced Condition]*** *Consider a fully connected neural network with L layers. Each layer has n_i neurons, and the weights of each layer $\boldsymbol{W_i}$ is a matrix of dimension (n_i, n_{i+1}). The matrix $\boldsymbol{W_i} = (w^i_{n,m})$ where $w^i_{n,m} \sim \mathcal{N}(0, \sigma_i^2)$ where σ_i^2 is decided based on the initialization technique. Then the following follow for all $i \in [1, L-1]$:*

1. $\mathbb{E}\left[\boldsymbol{W^T_{i+1}W_{i+1}} - \boldsymbol{W_iW_i^T}\right] = (n_{i+2}\sigma^2_{i+1} - n_i\sigma_i^2)\mathbb{I}$

2. *$Var\left[\boldsymbol{W^T_{i+1}W_{i+1}} - \boldsymbol{W_iW_i^T}\right] = (n_{i+2}\sigma^4_{i+1} + n_i\sigma_i^4)\mathbb{B}$, where $\mathbb{B}$ is a square matrix with fours across the diagonal and ones everywhere else.*

Proof 2 (Proof of Theorem 3.1.1)

$$\text{Let } \boldsymbol{W_{i+1}} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n_{i+2}} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n_{i+2}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_{i+1},1} & w_{n_{i+1},2} & \cdots & w_{n_{i+1},n_{i+2}} \end{pmatrix} \tag{3.1}$$
$$= \begin{pmatrix} \overline{w}_1 & \overline{w}_2 & \dots & \overline{w}_{n_{i+2}} \end{pmatrix}$$

with $\overline{w}_j = (w_{1,j}, w_{2,j}, \dots, w_{n_{i+1},j})^T$.

Then,

$$\boldsymbol{W_{i+1}^T W_{i+1}} = \begin{pmatrix} \overline{w}_1^T \\ \overline{w}_2^T \\ \vdots \\ \overline{w}_{n_{i+2}}^T \end{pmatrix} \begin{pmatrix} \overline{w}_1 & \overline{w}_2 & \cdots & \overline{w}_{n_{i+2}} \end{pmatrix}$$
$$= \begin{pmatrix} \langle \overline{w}_1, \overline{w}_1 \rangle & \langle \overline{w}_1, \overline{w}_2 \rangle & \cdots & \langle \overline{w}_1, \overline{w}_{n_{i+2}} \rangle \\ \langle \overline{w}_2, \overline{w}_1 \rangle & \langle \overline{w}_2, \overline{w}_2 \rangle & \cdots & \langle \overline{w}_2, \overline{w}_{n_{i+2}} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \overline{w}_{n_{i+2}}, \overline{w}_1 \rangle & \langle \overline{w}_{n_{i+2}}, \overline{w}_2 \rangle & \cdots & \langle \overline{w}_{n_{i+2}}, \overline{w}_{n_{i+2}} \rangle \end{pmatrix}$$

Now, consider $\langle \overline{w}_i, \overline{w}_j \rangle$ *with* $i \neq j$,

$$\begin{aligned} \langle \overline{w}_i, \overline{w}_j \rangle &= \sum_{k=1}^{n_{i+2}} w_{k,i} w_{k,j} \\ \mathbb{E}\left[\langle \overline{w}_i, \overline{w}_j \rangle\right] &= \mathbb{E}\left[\sum_{k=1}^{n_{i+2}} w_{k,i} w_{k,j}\right] \\ &= \sum_{k=1}^{n_{i+2}} \mathbb{E}[w_{k,i} w_{k,j}] \\ &= \sum_{k=1}^{n_{i+2}} \mathbb{E}[w_{k,i}]\mathbb{E}[w_{k,j}] = 0 \quad \textit{(by independence)} \\ Var\left[\langle \overline{w}_i, \overline{w}_j \rangle\right] &= \mathbb{E}\left[\langle \overline{w}_i, \overline{w}_j \rangle^2\right] - \left[\mathbb{E}\left[\langle \overline{w}_i, \overline{w}_j \rangle\right]\right]^2 \\ &= \mathbb{E}\left[\left(\sum_{k=1}^{n_{i+2}} w_{k,i} w_{k,j}\right)^2\right] \\ &= \mathbb{E}\left[\sum_{k=1}^{n_{i+2}} w_{k,i}^2 w_{k,j}^2 + 2\sum_{k=1}^{n_{i+2}} \sum_{l>k} w_{k,i} w_{k,j} w_{l,i} w_{l,j}\right] \\ &= \sum_{k=1}^{n_{i+2}} \mathbb{E}[w_{k,i}^2 w_{k,j}^2] + 2\sum_{k=1}^{n_{i+2}} \sum_{l>k} \mathbb{E}[w_{k,i}]\mathbb{E}[w_{k,j}]\mathbb{E}[w_{l,i}]\mathbb{E}[w_{l,j}] \\ &= \sum_{k=1}^{n_{i+2}} \mathbb{E}[w_{k,i}^2]\mathbb{E}[w_{k,j}^2] \\ &= (n_{i+2})\sigma_{i+1}^4 \end{aligned}$$

Similarly, consider $\langle \overline{w}_i, \overline{w}_i \rangle$:

$$
\begin{aligned}
\langle \overline{w}_i, \overline{w}_i \rangle &= \sum_{k=1}^{n_{i+2}} w_{k,i}^2 \\
E\left[\langle \overline{w}_i, \overline{w}_i \rangle\right] &= E\left[\sum_{k=1}^{n_{i+2}} w_{k,i}^2\right] = n_{i+2} E\left[w_{k,i}^2\right] = n_{i+2}\sigma_{n_{i+1}}^2 \\
Var\left[\langle \overline{w}_i, \overline{w}_i \rangle\right] &= E\left[\left(\langle \overline{w}_i, \overline{w}_i \rangle\right)^2\right] - E\left[\langle \overline{w}_i, \overline{w}_i \rangle\right]^2 \\
&= E\left[\left(\sum_{k=1}^{n_{i+2}} w_{k,i}^2\right)^2\right] - n_{i+2}^2\sigma_{n_{i+1}}^4 \\
&= E\left[\left(\sum_{k=1}^{n_{i+2}} w_{k,i}^2\right)^2\right] - n_{i+2}^2\sigma_{n_{i+1}}^4 \\
&= E\left[\sum_{k=1}^{n_{i+2}} w_{k,i}^4 + 2\sum_{k=1}^{n_{i+2}} \sum_{l=k+1}^{n_{i+2}} w_{k,i}^2 w_{l,i}^2\right] - n_{i+2}^2\sigma_{n_{i+1}}^4 \\
&= \sum_{k=1}^{n_{i+2}} E\left[w_{k,i}^4\right] + 2\sum_{k=1}^{n_{i+2}} \sum_{l=k+1}^{n_{i+2}} E\left[w_{k,i}^2\right] E\left[w_{l,i}^2\right] - n_{i+2}^2\sigma_{n_{i+1}}^4 \\
&= n_{i+2}(3\sigma_{n_{i+1}}^4) + (n_{i+2}^2 - n_{i+2})\sigma_{n_{i+1}}^4 - n_{i+2}^2\sigma_{n_{i+1}}^4 \\
&= 4n_{i+2}\sigma_{n_{i+1}}^4
\end{aligned}
$$

Hence

$$E\left[\boldsymbol{W_{i+1}^T W_{i+1}}\right] = \left(n_{i+2}\sigma_{i+1}^2\right)\mathbb{I}$$

$$Var\left[\boldsymbol{W_{i+1}^T W_{i+1}}\right] = 4\left(n_{i+2}\right)\sigma_{i+1}^4\mathbb{B}$$

For the case for $\boldsymbol{W_i}$, notice we can express $\boldsymbol{W_i W_i^T}$ as $(\boldsymbol{W_i^T})^T(\boldsymbol{W_i^T})$. Hence we can use the proof above, with $\boldsymbol{W'_{i+1}} = \boldsymbol{W_i^T}$. In this case the matrix $\boldsymbol{W'_{i+1}}$ has shape (n_i, n_{i+1}), and each element of the matrix has variance σ_i^2. We have:

$$E\left[\boldsymbol{W_i W_i^T}\right] = n_i\sigma_i^2\mathbb{I}$$

$$Var\left[\boldsymbol{W_i W_i^T}\right] = n_i\sigma_i^4\mathbb{B}$$

By assumption, $\boldsymbol{W_i}, \boldsymbol{W_{i+1}}$ are independent. Hence $Cov(\boldsymbol{W_i}, \boldsymbol{W_{i+1}}) = 0$. We can use this property together with linearity of expectation:

$$E\left[\boldsymbol{W_{i+1}^T W_{i+1} - W_i^T W_i}\right] = \left(n_{i+2}\sigma_{i+1}^2 - n_i\sigma_i^2\right)\mathbb{I}$$

$$Var\left[\boldsymbol{W_{i+1}^T W_{i+1} - W_i^T W_i}\right] = \left(n_{i+2}\sigma_{i+1}^4 + n_i\sigma_i^4\right)\mathbb{B}$$

This completes the proof.

In neural network training, proper weight initialization is crucial for ensuring stable gradients during backpropagation, which helps to avoid issues such as vanishing and exploding gradients. The goal of weight scaling is to maintain appropriate variance across layers, enabling efficient and effective learning ([15]). The weights are typically initialized to be random and centered around 0 to break symmetry and ensure that different neurons learn different features.

Some of the most commonly used initialization methods are detailed below:

- **LeCun Initialization ([26])**: Weights are initialized using a normal distribution with a mean of 0 and a variance of $\frac{1}{n_i}$, where n_i is the number of input units in the layer. Mathematically, the weights w are drawn from $\mathcal{N}(0, \frac{1}{n_i})$.

- **Glorot Initialization ([15])**: Weights are initialized using a normal distribution with a mean of 0 and a variance of $\frac{2}{n_i+n_{i+1}}$, where n_i is the number of input units and n_{i+1} is the number of output units. This method balances the variance between layers with different widths. Mathematically, the weights w are drawn from $\mathcal{N}(0, \frac{2}{n_i+n_{i+1}})$.

- **He Initialization ([16])**: Weights are initialized using a normal distribution with a mean of 0 and a variance of $\frac{2}{n_i}$, where n_i is the number of input units in the layer. This method is particularly suited for layers with ReLU activation functions. Mathematically, the weights w are drawn from $\mathcal{N}(0, \frac{2}{n_i})$.

- **Scaled Initialization ([33])**: Weights are initialized using a normal distribution with a mean of 0 and a variance of $\frac{\alpha_i}{n_i}$, where n_i is the number of input units in the layer and α_i is a parameter specific to each layer. Mathematically, the weights w are drawn from $\mathcal{N}(0, \frac{\alpha_i}{n_i})$.

These initialization methods help ensure that the network starts with weights that facilitate stable and efficient learning, avoiding the common pitfalls of poorly initialized neural networks.

Using (3.1.1), we can calculate the respective Expectations and Variances of the Balanced Computation under the different initialisations:

Initialization	$\mathrm{Var}(w^{i+1}_{n,m})$	$\mathrm{Var}(w^{i}_{n,m})$	$\mathbb{E}[\boldsymbol{W}^T_{i+1}\boldsymbol{W}_{i+1} - \boldsymbol{W}_i\boldsymbol{W}^T_i]$	$\mathrm{Var}[\boldsymbol{W}^T_{i+1}\boldsymbol{W}_{i+1} - \boldsymbol{W}_i\boldsymbol{W}^T_i]$
LeCun	$\frac{1}{n_{i+1}}$	$\frac{1}{n_i}$	$\left(\frac{n_{i+2}}{n_{i+1}} - 1\right)\mathbb{I}$	$\left(\frac{n_{i+2}}{n^2_{i+1}} + \frac{1}{n_i}\right)\mathbb{B}$
Glorot	$\frac{2}{n_{i+1}+n_{i+2}}$	$\frac{2}{n_{i+1}+n_i}$	$2\left(\frac{n_{i+2}}{n_{i+1}+n_{i+2}} - \frac{n_i}{n_{i+1}+n_i}\right)\mathbb{I}$	$(n_{i+2}\left(\frac{2}{n_{i+1}+n_{i+2}}\right)^2 + n_i\left(\frac{2}{n_i+n_{i+1}}\right)^2)\mathbb{B}$
He	$\frac{2}{n_{i+1}}$	$\frac{2}{n_i}$	$2\left(\frac{n_{i+2}}{n_{i+1}} - 1\right)\mathbb{I}$	$4\left(\frac{n_{i+2}}{n^2_{i+1}} + \frac{1}{n_i}\right)\mathbb{B}$
Scaled	$\frac{\alpha^2_{i+1}}{n_{i+1}}$	$\frac{\alpha^2_i}{n_i}$	$\left(\frac{n_{i+2}}{n_{i+1}}\alpha^2_{i+1} - \alpha^2_i\right)\mathbb{I}$	$(n_{i+2}\left(\frac{\alpha^2_{i+1}}{n_{i+1}}\right)^2 + n_i\left(\frac{\alpha^2_i}{n_i}\right)^2)\mathbb{B}$

Table 3.1: Comparison of Variance and Expectation of Balanced Computation for Different Weight Initializations

(**Table 3.1**) shows that for the above initialisations the Balanced Computation of the weight pair will result in λ-Balanced weights for some λ. The table also details how network structure will influence the value of λ for different initialisation techniques.

The figure below shows a numerical example of how the Balanced computation would like like for initialising weights with LeCun, He, Scaled and Glorot initialisations using $n_i = 40, n_{i+1} = 200, n_{i+2} = 80$. With these numbers of nodes in each layer one can appreciate how the Balanced Computation on the weights is visually similar to a scaled identity matrix.

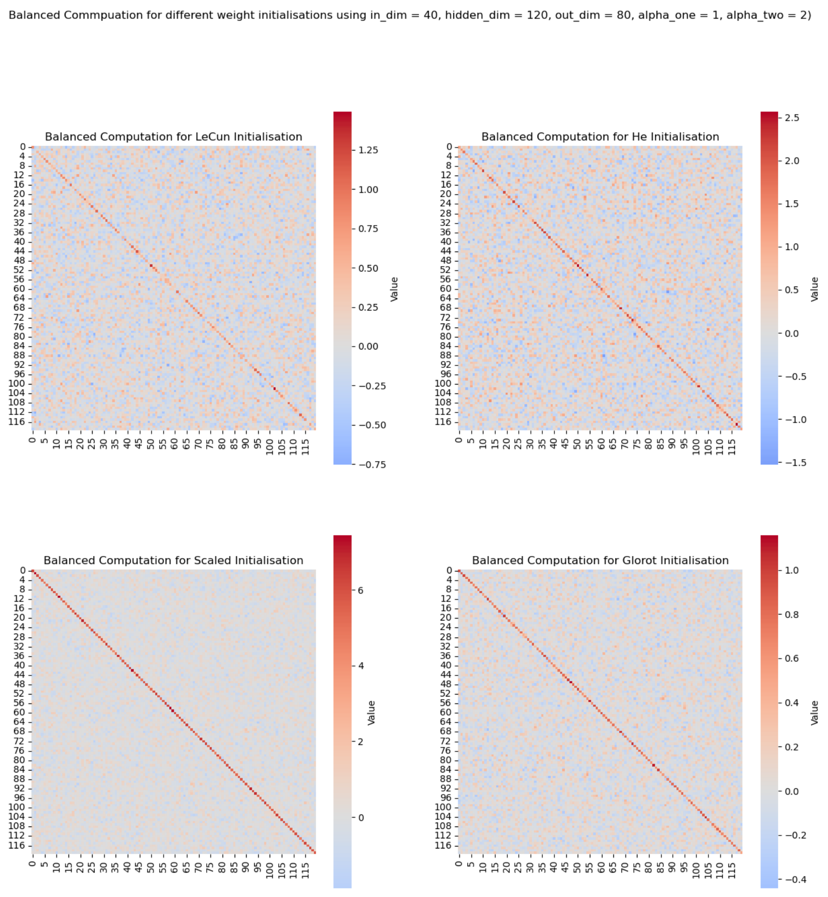


Figure 3.1: Figure 3.1: Balanced computation for different weight initializations. The figure compares LeCun, He, Scaled, and Glorot initializations, showing how the balanced computation on the weights visually resembles a scaled identity matrix.

State of the art models such as ([22], [29]) use more than 10,000 nodes in each hidden layer. This implies that if we wished to perform some mathematical analysis on these models, assuming the initial random weights are λ-Balanced would be a very close depiction to reality. In addition, these models often have a different number of nodes per layer so understanding the effect of the relationship between n_i, n_{i+1} and n_{i+2} is crucial.

Specifically, we aim to understand how changes in the relative width of layers $i, i+1, i+2$ affect the Balanced Computation: suppose $n_{i+1} = kn_i$ for some $k \in \mathbb{R}^+$ and $n_{i+2} = rn_i$ for some $r \in \mathbb{R}^+$. Then we can express above table in terms of k, r and , n_i only:

Initialization	$\mathrm{Var}(w^{i+1}_{n,m})$	$\mathrm{Var}(w^{i}_{n,m})$	$\mathbb{E}[\boldsymbol{W}_{i+1}^T\boldsymbol{W}_{i+1} - \boldsymbol{W}_i\boldsymbol{W}_i^T]$	$\mathrm{Var}[\boldsymbol{W}_{i+1}^T\boldsymbol{W}_{i+1} - \boldsymbol{W}_i\boldsymbol{W}_i^T]$
LeCun	$\frac{1}{kn_i}$	$\frac{1}{n_i}$	$\left(\frac{r}{k} - 1\right)\mathbb{I}$	$\frac{1}{n_i}\left(\frac{r}{k^2} + 1\right)\mathbb{B}$
Glorot	$\frac{2}{n_i(k+r)}$	$\frac{2}{n_i(k+1)}$	$2\left(\frac{r}{k+r} - \frac{1}{k+1}\right)\mathbb{I}$	$\frac{4}{n_i}\left(\frac{r}{(r+k)^2} + \frac{1}{(k+1)^2}\right)\mathbb{B}$
He	$\frac{2}{kn_i}$	$\frac{2}{n_i}$	$2\left(\frac{r}{k} - 1\right)\mathbb{I}$	$\frac{4}{n_i}\left(\frac{r}{k^2} + 1\right)\mathbb{B}$
Scaled	$\frac{\alpha_{i+1}^2}{kn_i}$	$\frac{\alpha_i^2}{n_i}$	$\left(\frac{r}{k}\alpha_{i+1}^2 - \alpha_i^2\right)\mathbb{I}$	$\frac{1}{n_i}\left(\frac{r\alpha_{i+1}^2}{k^2} + \alpha_i^2\right)\mathbb{B}$

Table 3.2: Comparison of Variance and Expectation for Different Initializations

From (**Table 3.2**) we can observe that as the number of nodes in each layer tends to infinity,while the ratios between the number of nodes in each layer (r, k) are maintained, the variance of the

Balanced Computation tends to zero. Hence the Balanced Computation converges in probability to λ-Balanced weights.

Further, (**Table 3.2**) shows that we a larger value of r will lead to a higher value of λ in every one of the initialisations displayed. Moreover, a larger k has the opposite effect. In addition, we can observe that in some initialisations there are limits as to what value λ can take (such as in LeCun $\lambda \geq 0$).

Some special cases of r and k are interesting to consider to gain an intuition of how changing these values influences the Balancedness of the Weights. In the table below, we consider the cases:

1. $r = k$: the three layers of the network have the same number of nodes.
2. $r \to 0$, k fixed: $n_i >> n_{i+2}, n_{i+1}$, the first of the three layers is much larger than the other two layers .
3. $r \to \infty$, k fixed: $n_{i+2} >> n_i, n_{i+1}$, the last of the three layers is much larger than the other two layers (k is fixed).
4. $k \to 0$, r fixed: the middle layer is exceedingly small, $n_{i+1} << n_i, n_{i+2}$
5. $k \to \infty$, r fixed: the middle layer is much bigger than the other two layers, $n_{i+1} >> n_i, n_{i+2}$
6. $r = \frac{\alpha_i^2}{\alpha_{i+1}^2}$: the inner and outer layers have a ratio proportional to the scale factors of each weight layer. This case is important for Scaled initialisation.

Initialization	$r = k$	$r \to 0$	$r \to \infty$	$k \to \infty$	$k \to 0$	$r = \frac{\alpha_i^2}{\alpha_{i+1}^2} k$
LeCun	0	$-\mathbb{I}$	$\frac{r}{k}\mathbb{I}$	$-\mathbb{I}$	$\frac{r}{k}\mathbb{I}$	-
Glorot	0	$2\mathbb{I}$	$2\mathbb{I}$	0	$-\frac{2}{k+1}\mathbb{I}$	-
He	0	$-2\mathbb{I}$	$2\left(\frac{r}{k}\right)\mathbb{I}$	$-2\mathbb{I}$	$2\left(\frac{r}{k}\right)\mathbb{I}$	-
Scaled	$(\alpha_{i+1}^2 - \alpha_i^2)\mathbb{I}$	$-\alpha_i^2\mathbb{I}$	$\frac{r}{k}\alpha_{i+1}^2\mathbb{I}$	$-\alpha_i^2\mathbb{I}$	$\frac{r}{k}\alpha_{i+1}^2\mathbb{I}$	0

Table 3.3: Comparison of Variance and Expectation under Different Conditions

From the table above one can appreciate the impact network structure can have on the Balanced Computation of the weights of each layer. One can also see that there are many cases when the Balanced computation does not equal 0, both in the limit of r, k and not in the limit.

We have showed that although the Balanced property is only preserved in Linear Networks, it occurs at initialisation for large networks which utilise some of the most common weight initialisation techniques.

These findings provide motivation to better understand the relation between the Balanced Computation of a Network, its structure and the regime it will learn in. If we are able to understand the relation between λ-Balanced weights and Rich and Lazy Learning in Linear Networks, one might be able to approximate these results to the nonlinear case.

A possible future application might be the ability to heavily influence a network's learning regime by altering the relative width of its layers, its activation functions or weight initialisation techniques used for each layer.

In order to better understand the effects of λ on the learning dynamics and learning regime of the network, we first study aligned λ-Balanced networks.

3.2 Exact Solutions for Aligned λ-Balanced Network

In this section we consider a Linear Network with λ-Balanced initial weights, which are aligned to the task. Previous work ([38]) has shown that weights will remain aligned during training, and each singular value of the task will be learned independently.

Thus the aligned case is a good opportunity to gain intuition on the effect of λ on learning. There has been previous work ([40]) which has derived equivalent dynamics as the ones shown in this report. However these results where arrived to independently, and they are used in a novel way to better understand the impact of λ on learning dynamics.

Theorem 3.2.1 ***[Dynamics of output in Aligned λ-Balanced Linear Network]*** *Consider a three layer linear network with starting weights which are aligned to the dataset and λ-Balanced with non-zero λ. We perform gradient descent with learning rate $\frac{1}{\tau}$ Let $a_\alpha(t)$ be the dynamics of the α singular value of the matrix $W_2(t)W_1(t)$, where the singular values are sorted in descending order. Then:*

$$a_\alpha(t) = \frac{\lambda \frac{K(C_{0,\alpha}e^{K\frac{t}{\tau}}-1)-\lambda(C_{0,\alpha}e^{K\frac{t}{\tau}}+1)}{2s(C_{0,\alpha}e^{K\frac{t}{\tau}}+1)}}{1-(\frac{K(C_{0,\alpha}e^{K\frac{t}{\tau}}-1)-\lambda(C_{0,\alpha}e^{K\frac{t}{\tau}}+1)}{2s(C_{0,\alpha}e^{K\frac{t}{\tau}}+1)})^2} \tag{3.2}$$

With

$$K = \sqrt{\lambda^2 + 4s_\alpha^2}$$

$$C_{0,\alpha} = \frac{K + \lambda + \frac{4a_\alpha(0)s\, sgn(\lambda)}{\sqrt{\lambda^2+4a_\alpha(0)^2}+\lambda}}{K - \lambda - \frac{4a_\alpha(0)s\, sgn(\lambda)}{\sqrt{\lambda^2+4a_\alpha(0)^2}+\lambda}}$$

Proof 3 (Proof of 3.2.1) *We start with the differential equation describing the evolution of the singular values of each weight from ([36]).*

Our starting point stems from ([36]), where the authors derive exact dynamics for aligned 0-Balanced weights.

$$\tau\frac{d}{dt}c_\alpha = d_\alpha(s_\alpha - c_\alpha d_\alpha),$$

$$\tau\frac{d}{dt}d_\alpha = c_\alpha(s_\alpha - c_\alpha d_\alpha)$$

$$\tau\frac{d}{dt}a_\alpha = (c_\alpha^2 + d_\alpha^2)(s - a_\alpha)$$

In ([36]) the authors assume 0-Balanced weights, that is $d_\alpha^2 - c_\alpha^2 = 0$. Here we assume λ-Balanced weights, that is $d_\alpha^2 - c_\alpha^2 = \lambda$.

$$\tau\frac{d}{dt}c_\alpha = d_\alpha(s_\alpha - c_\alpha d_\alpha),$$
$$\tau\frac{d}{dt}d_\alpha = c_\alpha(s_\alpha - c_\alpha d_\alpha),$$
$$\tau\frac{d}{dt}a_\alpha = (c_\alpha^2 + d_\alpha^2)(s - a_\alpha),$$

$$\begin{aligned}
d_\alpha^2 - c_\alpha^2 &= \lambda, \\
(d_\alpha^2 + c_\alpha^2) &= (d_\alpha^2 - c_\alpha^2) + 4d_\alpha^2 c_\alpha^2, \\
(d_\alpha^2 + c_\alpha^2)^2 &= (d_\alpha^2 - c_\alpha^2)^2 + 4d_\alpha^2 c_\alpha^2, \\
&= (d_\alpha^2 - c_\alpha^2)^2 + 4a_\alpha^2, \\
&= \lambda^2 + 4a_\alpha^2.
\end{aligned}$$

*We can substitute this result into (**3.4**) to obtain:*

$$\tau \frac{da_\alpha(t)}{dt} = \sqrt{\lambda^2 + 4a_\alpha(t)^2}\,(s - a_\alpha(t)) \tag{3.3}$$

$$d_\alpha^2 - c_\alpha^2 = \lambda \tag{3.4}$$

$$(d_\alpha^2 + c_\alpha^2) = (d_\alpha^2 - c_\alpha^2) + 4d_\alpha^2 c_\alpha^2 \tag{3.5}$$

$$(d_\alpha^2 + c_\alpha^2)^2 = (d_\alpha^2 - c_\alpha^2)^2 + 4d_\alpha^2 c_\alpha^2 \tag{3.6}$$

$$= (d_\alpha^2 - c_\alpha^2)^2 + 4a_\alpha^2 \tag{3.7}$$

$$= \lambda^2 + 4a_\alpha^2 \tag{3.8}$$

*We can substitute this result into (**3.4**) to obtain:*

$$\tau \frac{da_\alpha(t)}{dt} = \sqrt{\lambda^2 + 4a_\alpha(t)^2}\,(s - a_\alpha(t)) \tag{3.9}$$

Assume $\lambda \neq 0$. Then we can make the substitution

$$a_\alpha(t) := \frac{\lambda}{2}\sinh(\theta) \tag{3.10}$$

$$\tau \frac{d}{dt}\left[\frac{\lambda}{2}\sinh(\theta)\right] = \sqrt{\lambda^2 + 4\left(\frac{\lambda}{2}\sinh(\theta)\right)^2}\left(s - \frac{\lambda}{2}\sinh(\theta)\right)$$

$$\tau \frac{\lambda}{2}\cosh(\theta)\frac{d\theta}{dt} = |\lambda|\cosh(\theta)\left(s - \frac{\lambda}{2}\sinh(\theta)\right)$$

$$\tau \frac{d\theta}{dt} = sgn(\lambda)\,(2s - \lambda\sinh(\theta)) \tag{3.11}$$

$$\int_{\theta_0}^{\theta_t} \frac{d\theta}{2s - \lambda\sinh(\theta)} = \int_0^t \frac{sgn(\lambda)\,dt'}{\tau}$$

$$\left[\frac{2\tanh^{-1}\left(\frac{1+2s\tanh(\frac{\theta}{2})}{\sqrt{\lambda^2+4s^2}}\right)}{\sqrt{\lambda^2+4s^2}}\right]_{\theta_0}^{\theta_t} = \frac{sgn(\lambda)\,t}{\tau}$$

With $\theta_t = \theta(t), \theta_0 = \theta(0)$. Using the fact that

$$\tanh^{-1}(x) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right) \tag{3.12}$$

$$\frac{1}{\sqrt{\lambda^2+4s^2}}\left[\ln\left(\frac{\sqrt{\lambda^2+4s^2}+\lambda+2s\tanh\left(\frac{\theta}{2}\right)}{\sqrt{\lambda^2+4s^2}-\lambda-2s\tanh\left(\frac{\theta}{2}\right)}\right)\right]_{\theta_0}^{\theta_t} = \frac{sgn(\lambda)t}{\tau}$$

$$\frac{1}{\sqrt{\lambda^2+4s^2}}\left[\ln\left(\frac{\sqrt{\lambda^2+4s^2}+\lambda+2s\tanh\left(\frac{\theta_t}{2}\right)}{\sqrt{\lambda^2+4s^2}-\lambda-2s\tanh\left(\frac{\theta_t}{2}\right)}\right)-\ln\left(\frac{\sqrt{\lambda^2+4s^2}+\lambda+2s\tanh\left(\frac{\theta_0}{2}\right)}{\sqrt{\lambda^2+4s^2}-\lambda-2s\tanh\left(\frac{\theta_0}{2}\right)}\right)\right] = \frac{sgn(\lambda)t}{\tau}$$

Let

$$K = \sqrt{\lambda^2+4s^2}, \quad C = \frac{K+\lambda+2s\tanh\left(\frac{\theta_0}{2}\right)}{K-\lambda-2s\tanh\left(\frac{\theta_0}{2}\right)} \tag{3.13}$$

$$\frac{1}{K}\left[\ln\left(\frac{K+\lambda+2s\tanh\left(\frac{\theta_t}{2}\right)}{K-\lambda-2s\tanh\left(\frac{\theta_t}{2}\right)}\right)-\ln(C)\right] = \frac{sgn(\lambda)t}{\tau}$$

$$\frac{K+\lambda+2s\tanh\left(\frac{\theta_t}{2}\right)}{K-\lambda-2s\tanh\left(\frac{\theta_t}{2}\right)} = Ce^{\frac{sgn(\lambda)Kt}{\tau}} \tag{3.14}$$

$$K+\lambda+2s\tanh\left(\frac{\theta_t}{2}\right) = \left(K-\lambda-2s\tanh\left(\frac{\theta_t}{2}\right)\right)Ce^{\frac{sgn(\lambda)Kt}{\tau}}$$

$$2s\tanh\left(\frac{\theta_t}{2}\right)\left(1+Ce^{\frac{sgn(\lambda)Kt}{\tau}}\right) = K\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}-1\right)-\lambda\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}+1\right)$$

$$\tanh\left(\frac{\theta_t}{2}\right) = \frac{K\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}-1\right)-\lambda\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}+1\right)}{2s\left(1+Ce^{\frac{sgn(\lambda)Kt}{\tau}}\right)}$$

$$\theta = 2\tanh^{-1}\left(\frac{K\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}-1\right)-\lambda\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}+1\right)}{2s\left(1+Ce^{\frac{sgn(\lambda)Kt}{\tau}}\right)}\right) \tag{3.15}$$

Since $a = \frac{\lambda}{2}\sinh(\theta)$*, we can use the identity* $\sinh\left(2\tanh^{-1}(x)\right) = \frac{2x}{1-x^2}$.

$$a_t = \frac{\lambda}{2}\sinh\left(2\tanh^{-1}\left(\frac{K\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}-1\right)-\lambda\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}+1\right)}{2s\left(1+Ce^{\frac{sgn(\lambda)Kt}{\tau}}\right)}\right)\right) \tag{3.16}$$

$$a_t = \lambda\left(\frac{\left(\frac{K\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}-1\right)-\lambda\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}+1\right)}{2s\left(1+Ce^{\frac{sgn(\lambda)Kt}{\tau}}\right)}\right)}{1-\left(\frac{K\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}-1\right)-\lambda\left(Ce^{\frac{sgn(\lambda)Kt}{\tau}}+1\right)}{2s\left(1+Ce^{\frac{sgn(\lambda)Kt}{\tau}}\right)}\right)^2}\right) \tag{3.17}$$

Further, since

$$\theta = \sinh^{-1}\left(\frac{2a}{\lambda}\right)$$

Using the identity

$$\tanh\left(\frac{1}{2}\sinh^{-1}(x)\right) = \frac{x}{\sqrt{1+x^2}+1} \tag{3.18}$$

we have

$$\tanh\left(\frac{\theta}{2}\right) = \frac{\frac{2a}{\lambda}}{\sqrt{1+\left(\frac{2a}{\lambda}\right)^2}+1}$$

Simplifying,

$$\tanh\left(\frac{\theta}{2}\right) = \frac{2a\, sgn(\lambda)}{\sqrt{\lambda^2+4a^2}+\lambda}$$

So,

$$C = \frac{K+\lambda+\frac{4a_0 s\, sgn(\lambda)}{\sqrt{\lambda^2+4a_0^2}+\lambda}}{K-\lambda-\frac{4a_0 s\, sgn(\lambda)}{\sqrt{\lambda^2+4a_0^2}+\lambda}} \tag{3.19}$$

The derived equation above describes the evolution of the singular values of the weights $W_2(t)W_1(t)$. By ([36]) the weights remain alighed through training. Hence we have a complete description of the evolution of the output of the network through training. The equation can be applied to equal input output dimensions as well as unequal input output dimensions by performing the compact Singular Value Decomposition of the data.

Below is an example of the empirical and analytical dynamics plotted against each other for $\lambda = 0.5$. From (**Figure 3.2**) we can observe that for a small value of λ each singular value can have very different dynamics. However, the dynamics of each singular value are completely captured by our analytical solution.

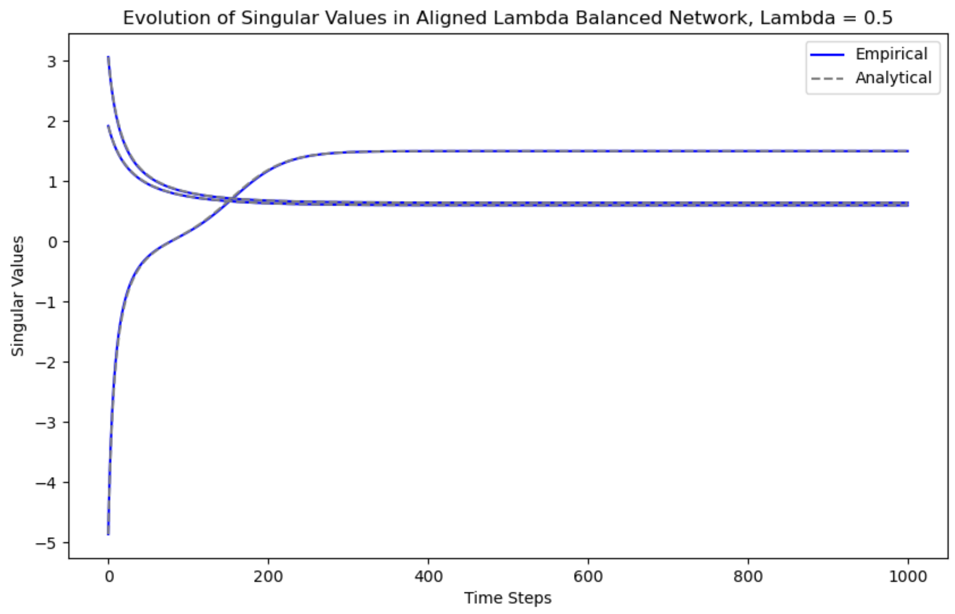


Figure 3.2: Empirical and analytical dynamics plotted against each other for $\lambda = 0.5$. The figure demonstrates how the dynamics of each singular value are captured by the analytical solution

From (Equation 3.2) one can observe that $a_\alpha(t)$ evolves with the curve e^{Kt}, with $K = \sqrt{\lambda^2 + 4s_\alpha^2}$. Consequently, as the magnitude of λ increases, the rate of learning will increase as well. For large values of $|\lambda|$ it can be shown that $K \approx |\lambda|$. Hence, for large values of $|\lambda|$ $a_\alpha(t)$ evolves with the curve $e^{|\lambda|t}$

Additionally, the equation for λ-Balanced weights can be complex and difficult to interpret. To develop an intuitive understanding of the impact of λ on learning, we will examine two specific cases: $\lambda = 0$ and the limit as $|\lambda| \to \infty$.

Theorem 3.2.2 ***[Rate of Learning of Aligned λ Balanced Weights for extreme values of λ]*** *For $\lambda = 0$, the singular values are learned according to sigmoidal dynamics. However, as $|\lambda| \to \infty$ the singular values are learned according to exponential dynamics.*

Proof 4 (Proof of Theorem 3.2.2) *For $\lambda = 0$:*

$$\tau \frac{d}{dt} a_\alpha = 2a_\alpha(t) \left(s_\alpha - a_\alpha(t)\right)$$

From [38], this has the solution

$$a_\alpha(t) = \frac{s_\alpha e^{\frac{2s_\alpha t}{\tau}}}{e^{\frac{2s_\alpha t}{\tau}} - 1 + \frac{s_\alpha}{a_\alpha(0)}}.$$

which clearly depicts a sigmoidal trajectory.
On the other hand, as $|\lambda| \to \infty$:

$$\begin{aligned}
\tau \frac{d}{dt} a_\alpha &= \sqrt{\lambda^2 + 4a_\alpha(t)^2} \left(s_\alpha - a_\alpha(t)\right), \\
\tau \frac{d}{dt} a_\alpha &\approx |\lambda| \left(s_\alpha - a_\alpha(t)\right), \\
a_\alpha(t) &= s_\alpha + (a_\alpha(0) - s_\alpha)e^{-|\lambda| t/\tau},
\end{aligned}$$

which clearly follows an exponential trajectory.

From ([38]), in a shallow network the singular values of the task are learned according to the equation:

$$b_\alpha(t) = s_\alpha + (b_\alpha(0) - s_\alpha)e^{-\frac{t}{\tau}} \tag{3.20}$$

If we perform the substitution $u = |\lambda| t$ one can observe that the dynamics of $a_\alpha(u)$ are the same as the ones of $b_\alpha(t)$: $a_\alpha(t)$ is a horizontal compression by $|\lambda|$ of the dynamics of $b_\alpha(t)$. The more λ increases in magnitude, the more a deep network behaves like a shallow network in terms of learning dynamics. We have shown that modifying initial weights can have a similar effect on network learning dynamics as removing the network's hidden layer. One might ask the question if this similarity also holds in the case of internal representations and learning regimes. We will answer this question later on in the text.

The expression derived above offers valuable insights into the unaligned case and the impact of λ on learning dynamics. However, this solution leaves several important questions unanswered: How does the value of λ influence the alignment process of the weights in a linear network? How do internal representations evolve with varying λ in a linear network? Can we identify task-agnostic weight initializations that affect a network's learning regime? Addressing these questions is crucial for a comprehensive understanding of the role of λ in learning dynamics.

To answer these questions we must derive an analytical solution for the outputs and representations of an unaligned λ-Balanced network.

3.3 Exact Solutions for Unaligned λ-Balanced Network of Equal Dimensions

We have seen that the aligned setting can provide a number of insights into the impact of weight initialisation on Neural Networks.

Previous work has shown that small random weights undergo a "Silent Alignment Process" ([2]) in which during the first steps of training, the weights become fully aligned to the task. Hence the dynamics outlined in the previous section are a good approximation for small random weights. However, we desire an expression that allows us to study how the weights of a network become aligned, as well as a solution that also works for large weights.

To gain further insight into the learning paradigms of these networks, it is necessary to derive a solution for the unaligned case. We aim for a solution that is numerically stable so that it can closely resemble empirical dynamics even for large values of t and λ. The solution must also be interpretable so that we can easily understand how the network learns in terms of the Singular Value Decomposition of the data, therefore better understanding the alignment process of the weights.

The solution provided below builds on previous work from ([5]) which derived exact solutions for unequal input output zero-Balanced network. The first step of the proof was derived independently, but ([40]) shows an equivalent result with different notation.

Theorem 3.3.1 ***[QQt(t) dynamics for λ-Balanced weights for $n_i = n_o$]***
Let

$$\boldsymbol{Q}(t) = \begin{bmatrix} \boldsymbol{W}_1^T(t) \\ \boldsymbol{W}_2(t) \end{bmatrix}$$

With $\boldsymbol{W}_1(t), \boldsymbol{W}_2(t)$ λ-Balanced weights. Then:

1. *The matrix*

$$\boldsymbol{F} = \begin{bmatrix} -\frac{\lambda}{2}\mathbb{I} & \tilde{\boldsymbol{\Sigma}}^{yxT} \\ \tilde{\boldsymbol{\Sigma}}^{yx} & \frac{\lambda}{2}\mathbb{I} \end{bmatrix}$$

Satisfies Fukumizu's condition and hence the Fukumizu Matrix Riccatti equation for $\boldsymbol{QQ}^T(t)$ has a unique solution given in (2.2.4).

2. *We can diagonalize F as*

$$\boldsymbol{F} = \frac{1}{2}\begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A}-\boldsymbol{XA}) & \tilde{\boldsymbol{V}}(\boldsymbol{XA}+\boldsymbol{A}) \\ \tilde{\boldsymbol{U}}(\boldsymbol{A}-\boldsymbol{XA}) & -\tilde{\boldsymbol{U}}(\boldsymbol{A}+\boldsymbol{XA}) \end{pmatrix} \begin{pmatrix} \sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} & 0 \\ 0 & -\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A}-\boldsymbol{XA}) & \tilde{\boldsymbol{V}}(\boldsymbol{XA}+\boldsymbol{A}) \\ \tilde{\boldsymbol{U}}(\boldsymbol{A}-\boldsymbol{XA}) & -\tilde{\boldsymbol{U}}(\boldsymbol{A}+\boldsymbol{XA}) \end{pmatrix}^T$$

, where $\tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^{\boldsymbol{T}} = \tilde{\Sigma}^{yx}$ and

$$\boldsymbol{X} = \frac{\sqrt{\lambda^2\mathbb{I} + 4\tilde{\boldsymbol{S}}^2} - 2\tilde{\boldsymbol{S}}}{\lambda}$$

$$\boldsymbol{A} = \frac{1}{\sqrt{1 + \boldsymbol{X}^2}}$$

are diagonal matrices.

3. *We can rewrite the Fukumizu Matrix Riccatti equation for $\boldsymbol{QQ^T(t)}$ as:*

$$\mathbf{QQ}^T(t) = \mathbf{Z}[4e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}\mathbf{B}^{-1}\left(\mathbf{B}^T\right)^{-1}e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} + \left(\mathbf{I} - e^{-2\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}\right)\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}}^{-1}$$
$$- e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}\mathbf{B}^{-1}\mathbf{C}\left(e^{-2\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}}\frac{t}{\tau}} - \mathbf{I}\right)\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}}^{-1}\mathbf{C}^T\left(\mathbf{B}^T\right)^{-1}e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}]\mathbf{Z}^T$$

with

$$\mathbf{Z} = \begin{bmatrix} \tilde{\mathbf{V}}\left((\boldsymbol{A}-\boldsymbol{XA}) - (\boldsymbol{A}+\boldsymbol{XA})e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}\mathbf{C}^T(\mathbf{B}^T)^{-1}e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}\right) \\ \tilde{\mathbf{U}}\left((\boldsymbol{A}+\boldsymbol{XA}) + (\boldsymbol{A}-\boldsymbol{XA})e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}\mathbf{C}^T(\mathbf{B}^T)^{-1}e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}}\right) \end{bmatrix}$$

$$\boldsymbol{B} = \boldsymbol{S_2}\boldsymbol{U}^T\tilde{\boldsymbol{U}}(\boldsymbol{A} + \boldsymbol{X}\boldsymbol{A}) + \boldsymbol{S_1}\boldsymbol{V}^T\tilde{\boldsymbol{V}}(\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A})$$

$$\boldsymbol{C} = \boldsymbol{S_2}\boldsymbol{U}^T\tilde{\boldsymbol{U}}(\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}) - \boldsymbol{S_1}\boldsymbol{V}^T\tilde{\boldsymbol{V}}(\boldsymbol{A} + \boldsymbol{X}\boldsymbol{A})$$

$$\boldsymbol{W}_2(0) = \boldsymbol{U}\boldsymbol{S}_2\boldsymbol{R}^T, \quad \boldsymbol{W}_1(0) = \boldsymbol{R}\boldsymbol{S}_1\boldsymbol{V}^T, \quad \boldsymbol{\Sigma}^{yx} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T$$

Note that $\boldsymbol{Q}\boldsymbol{Q}^{\boldsymbol{T}}(t)$ is calculated using Cholesky decomposition.

Proof 5 (Proof of 3.3.1) *1. Proof of First Statement:*

We introduce the variables

$$\mathbf{Q} = \begin{bmatrix} \mathbf{W}_1^T \\ \mathbf{W}_2 \end{bmatrix} \quad \text{and} \quad \mathbf{Q}\mathbf{Q}^T = \begin{bmatrix} \mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2^T \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T \end{bmatrix}. \tag{3.21}$$

We compute the time derivative

$$\tau\frac{d}{dt}(\mathbf{Q}\mathbf{Q}^T) = \tau\begin{bmatrix} \frac{d\mathbf{W}_1^T}{dt}\mathbf{W}_1 + \mathbf{W}_1^T\frac{d\mathbf{W}_1}{dt} & \frac{d\mathbf{W}_1^T}{dt}\mathbf{W}_2 + \mathbf{W}_1^T\frac{d\mathbf{W}_2}{dt} \\ \frac{d\mathbf{W}_2}{dt}\mathbf{W}_1 + \mathbf{W}_2\frac{d\mathbf{W}_1}{dt} & \frac{d\mathbf{W}_2^T}{dt}\mathbf{W}_2 + \mathbf{W}_2^T\frac{d\mathbf{W}_2}{dt} \end{bmatrix}. \tag{3.22}$$

Using equations 18 and 19, we compute the four quadrants separately giving

$$\tau\left(\frac{d\mathbf{W}_1^T}{dt}\mathbf{W}_1 + \mathbf{W}_1^T\frac{d\mathbf{W}_1}{dt}\right) = \tag{3.23}$$

$$= (\Sigma^{yx} - \mathbf{W}_2\mathbf{W}_1)^T\mathbf{W}_2\mathbf{W}_1 + \mathbf{W}_1^T\mathbf{W}_2^T(\Sigma^{yx} - \mathbf{W}_2\mathbf{W}_1) \tag{3.24}$$

$$= (\Sigma^{yx})^T\mathbf{W}_2\mathbf{W}_1 + \mathbf{W}_1^T\mathbf{W}_2^T\Sigma^{yx} - \mathbf{W}_1^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1 - (\mathbf{W}_2\mathbf{W}_1)^T\mathbf{W}_2\mathbf{W}_1 \tag{3.25}$$

$$= (\Sigma^{yx})^T\mathbf{W}_2\mathbf{W}_1 + \mathbf{W}_1^T\mathbf{W}_2^T\Sigma^{yx} - \mathbf{W}_1^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_1^T\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_1 - \lambda\mathbf{W}_1^T\mathbf{W}_1, \tag{3.26}$$

$$\tau\left(\frac{d\mathbf{W}_1^T}{dt}\mathbf{W}_2^T + \mathbf{W}_1^T\frac{d\mathbf{W}_2^T}{dt}\right) = \tag{3.27}$$

$$= (\Sigma^{yx} - \mathbf{W}_2\mathbf{W}_1)^T\mathbf{W}_2\mathbf{W}_2^T + \mathbf{W}_1^T\mathbf{W}_1(\Sigma^{yx} - \mathbf{W}_2\mathbf{W}_1)^T \tag{3.28}$$

$$= (\Sigma^{yx})^T\mathbf{W}_2\mathbf{W}_2^T + \mathbf{W}_1^T\mathbf{W}_1(\Sigma^{yx})^T - \mathbf{W}_1^T\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_2^T - \mathbf{W}_1^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_2^T, \tag{3.29}$$

$$\tau\left(\frac{d\mathbf{W}_2}{dt}\mathbf{W}_1 + \mathbf{W}_2\frac{d\mathbf{W}_1}{dt}\right) = \tag{3.30}$$

$$= (\Sigma^{yx} - \mathbf{W}_2\mathbf{W}_1)\mathbf{W}_1^T\mathbf{W}_1 + \mathbf{W}_2\mathbf{W}_2^T(\Sigma^{yx} - \mathbf{W}_2\mathbf{W}_1) \tag{3.31}$$

$$= \Sigma^{yx}\mathbf{W}_1^T\mathbf{W}_1 + \mathbf{W}_2\mathbf{W}_2^T\Sigma^{yx} - \mathbf{W}_2\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_1, \tag{3.32}$$

$$\tau\left(\frac{d\mathbf{W}_2}{dt}\mathbf{W}_2^T + \mathbf{W}_2\frac{d\mathbf{W}_2^T}{dt}\right) = \tag{3.33}$$

$$(\tilde{\Sigma}^{yx} - \mathbf{W}_2\mathbf{W}_1)\mathbf{W}_1^T\mathbf{W}_2^T + \mathbf{W}_2\mathbf{W}_1(\tilde{\Sigma}^{yx} - \mathbf{W}_2\mathbf{W}_1)^T \tag{3.34}$$

$$= \tilde{\Sigma}^{yx}\mathbf{W}_1^T\mathbf{W}_2^T + \mathbf{W}_2\mathbf{W}_1(\tilde{\Sigma}^{yx})^T - \mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_2^T - \mathbf{W}_2\mathbf{W}_1(\mathbf{W}_2\mathbf{W}_1)^T \tag{3.35}$$

$$= \tilde{\Sigma}^{yx}\mathbf{W}_1^T\mathbf{W}_2^T + \mathbf{W}_2\mathbf{W}_1(\tilde{\Sigma}^{yx})^T - \mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_2^T - \mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_2^T \tag{3.36}$$

$$= \tilde{\Sigma}^{yx}\mathbf{W}_1^T\mathbf{W}_2^T + \mathbf{W}_2\mathbf{W}_1(\tilde{\Sigma}^{yx})^T - \mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_2^T - \mathbf{W}_2\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_2^T + \lambda\mathbf{W}_2\mathbf{W}_2^T. \tag{3.37}$$

Defining

$$\mathbf{F} = \begin{bmatrix} -\frac{\lambda}{2}I & (\tilde{\Sigma}^{yx})^T \\ \tilde{\Sigma}^{yx} & \frac{\lambda}{2}I \end{bmatrix}, \tag{3.38}$$

the gradient flow dynamics of $\mathbf{QQ}^T(t)$ *can be written as a differential matrix Riccati equation*

$$\tau\frac{d}{dt}(\mathbf{QQ}^T) = \mathbf{FQQ}^T + \mathbf{QQ}^T\mathbf{F} - (\mathbf{QQ}^T)^2. \tag{3.39}$$

We write $\tau\frac{d}{dt}(\mathbf{QQ}^T)$ *for completeness*

$$\begin{aligned}\tau\frac{d}{dt}(\mathbf{QQ}^T) = &\begin{bmatrix}-\frac{\lambda}{2} & (\tilde{\Sigma}^{yx})^T \\ \tilde{\Sigma}^{yx} & \frac{\lambda}{2}\end{bmatrix}\begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix} + \begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}^T\begin{bmatrix}-\frac{\lambda}{2} & (\tilde{\Sigma}^{yx})^T \\ \tilde{\Sigma}^{yx} & \frac{\lambda}{2}\end{bmatrix} \\ &- \begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}^2\end{aligned} \tag{3.40}$$

$$\begin{aligned}= &\begin{bmatrix}-\frac{\lambda}{2} & (\tilde{\Sigma}^{yx})^T \\ \tilde{\Sigma}^{yx} & \frac{\lambda}{2}\end{bmatrix}\begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix} + \begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}^T\begin{bmatrix}-\frac{\lambda}{2} & (\tilde{\Sigma}^{yx})^T \\ \tilde{\Sigma}^{yx} & \frac{\lambda}{2}\end{bmatrix} \\ &- \begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}\begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}\end{aligned} \tag{3.41}$$

$$\begin{aligned}= &\begin{bmatrix}-\frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_1 + (\tilde{\Sigma}^{yx})^T\mathbf{W}_2\mathbf{W}_1 & -\frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_2 + (\tilde{\Sigma}^{yx})^T\mathbf{W}_2\mathbf{W}_2^T \\ \tilde{\Sigma}^{yx}\mathbf{W}_1^T\mathbf{W}_1 + \frac{\lambda}{2}\mathbf{W}_2\mathbf{W}_1 & \tilde{\Sigma}^{yx}\mathbf{W}_1^T\mathbf{W}_2^T + \frac{\lambda}{2}\mathbf{W}_2\mathbf{W}_2^T\end{bmatrix} \\ &+ \begin{bmatrix}-\frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_1 + \mathbf{W}_1^T\mathbf{W}_1(\tilde{\Sigma}^{yx})^T & \frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_2 + \mathbf{W}_1^T\mathbf{W}_2(\tilde{\Sigma}^{yx})^T \\ -\frac{\lambda}{2}\mathbf{W}_2^T\mathbf{W}_1 + \mathbf{W}_2\mathbf{W}_1(\tilde{\Sigma}^{yx})^T & \frac{\lambda}{2}\mathbf{W}_2\mathbf{W}_2^T + \mathbf{W}_2\mathbf{W}_2^T(\tilde{\Sigma}^{yx})^T\end{bmatrix} \\ &- \begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}\begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}\end{aligned} \tag{3.42}$$

$$\begin{aligned}= &\begin{bmatrix}-\frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_1 + (\tilde{\Sigma}^{yx})^T\mathbf{W}_2\mathbf{W}_1 & -\frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_2 + (\tilde{\Sigma}^{yx})^T\mathbf{W}_2\mathbf{W}_2^T \\ \tilde{\Sigma}^{yx}\mathbf{W}_1^T\mathbf{W}_1 + \frac{\lambda}{2}\mathbf{W}_2\mathbf{W}_1 & \tilde{\Sigma}^{yx}\mathbf{W}_1^T\mathbf{W}_2^T + \frac{\lambda}{2}\mathbf{W}_2\mathbf{W}_2^T\end{bmatrix} \\ &+ \begin{bmatrix}-\frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_1 + \mathbf{W}_1^T\mathbf{W}_1(\tilde{\Sigma}^{yx})^T & \frac{\lambda}{2}\mathbf{W}_1^T\mathbf{W}_2 + \mathbf{W}_1^T\mathbf{W}_2(\tilde{\Sigma}^{yx})^T \\ -\frac{\lambda}{2}\mathbf{W}_2^T\mathbf{W}_1 + \mathbf{W}_2\mathbf{W}_1(\tilde{\Sigma}^{yx})^T & \frac{\lambda}{2}\mathbf{W}_2\mathbf{W}_2^T + \mathbf{W}_2\mathbf{W}_2^T(\tilde{\Sigma}^{yx})^T\end{bmatrix} \\ &- \begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_1 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{W}_2^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_2 + \mathbf{W}_1^T\mathbf{W}_2\mathbf{W}_2^T\mathbf{W}_2 \\ \mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_1 + \mathbf{W}_2\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T\mathbf{W}_2 + \mathbf{W}_2\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_2^T\end{bmatrix}\end{aligned} \tag{45}$$

The four quadrants of 3.39 are equivalent to equations 3.26, 3.29, 3.32, and 3.37 respectively. Assuming that $\mathbf{Q}(0)$ *is full rank, the continuous differential equation (41) has a unique solution for all* $t \geq 0$

$$\mathbf{QQ}^T(t) = e^{\mathbf{F}\frac{t}{\tau}}\mathbf{Q}(0)\left[\mathbf{I} + \frac{1}{2}\mathbf{Q}(0)^T\left(e^{\mathbf{F}\frac{t}{\tau}}\mathbf{F}^{-1}e^{\mathbf{F}\frac{t}{\tau}} - \mathbf{F}^{-1}\right)\mathbf{Q}(0)\right]^{-1}\mathbf{Q}(0)^Te^{\mathbf{F}\frac{t}{\tau}}. \tag{3.43}$$

2. *Proof of second statement:*

$$\mathbf{F} = \begin{pmatrix}-\frac{\lambda}{2}\mathbb{I} & (\tilde{\mathbf{\Sigma}}^{yx})^T \\ \tilde{\mathbf{\Sigma}}^{yx} & \frac{\lambda}{2}\mathbb{I}\end{pmatrix} \tag{3.44}$$

$$\mathbf{F} = \frac{1}{2}\begin{pmatrix}\tilde{\boldsymbol{V}} & \tilde{\boldsymbol{V}} \\ \tilde{\boldsymbol{U}} & -\tilde{\boldsymbol{U}}\end{pmatrix}\begin{pmatrix}\tilde{\boldsymbol{S}} & \frac{\lambda}{2}\mathbf{I} \\ \frac{\lambda}{2}\mathbf{I} & -\tilde{\boldsymbol{S}}\end{pmatrix}\begin{pmatrix}\tilde{\boldsymbol{V}} & \tilde{\boldsymbol{V}} \\ \tilde{\boldsymbol{U}} & -\tilde{\boldsymbol{U}}\end{pmatrix}^T \tag{3.45}$$

Since $\tilde{S}$, $\frac{\lambda}{2}\mathbf{I}$ *are diagonal matrices, we can diagonalize the block diagonal matrix*

$$\begin{pmatrix}\tilde{\boldsymbol{S}} & \frac{\lambda}{2}\mathbb{I} \\ \frac{\lambda}{2}\mathbb{I} & -\tilde{\boldsymbol{S}}\end{pmatrix} \tag{3.46}$$

as if it were a 2×2 *matrix.*

First, we diagonalize the 2×2 *matrix:*

$$\begin{pmatrix} \tilde{s} & \frac{\lambda}{2} \\ \frac{\lambda}{2} & -\tilde{s} \end{pmatrix} = \begin{pmatrix} a & xa \\ -xa & a \end{pmatrix} \begin{pmatrix} \sqrt{\tilde{s}^2 + \frac{\lambda^2}{4}} & 0 \\ 0 & -\sqrt{\tilde{s}^2 + \frac{\lambda^2}{4}} \end{pmatrix} \begin{pmatrix} a & xa \\ -xa & a \end{pmatrix}^T \tag{3.47}$$

with $x = \frac{\sqrt{\lambda^2 + 4\tilde{s}^2} - 2\tilde{s}}{\lambda}$, $a = \frac{1}{\sqrt{1+x^2}}$.
We then generalize this result to 2×2 *block diagonal matrices:*

$$\begin{pmatrix} \tilde{\boldsymbol{S}} & \frac{\lambda}{2}\mathbb{I} \\ \frac{\lambda}{2}\mathbb{I} & -\tilde{\boldsymbol{S}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{XA} \\ -\boldsymbol{XA} & \boldsymbol{A} \end{pmatrix} \begin{pmatrix} \sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} & 0 \\ 0 & -\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{XA} \\ -\boldsymbol{XA} & \boldsymbol{A} \end{pmatrix}^T \tag{3.48}$$

With $\tilde{X}$ *a diagonal matrix:*

$$\tilde{\boldsymbol{X}} = \frac{\sqrt{\frac{\lambda^2}{4}\mathbb{I} + \tilde{\boldsymbol{S}}^2} - \tilde{\boldsymbol{S}}}{\lambda}, \quad \boldsymbol{A} = \frac{1}{\sqrt{1 + \boldsymbol{X}^2}} \tag{3.49}$$

Hence

$$\mathbf{F} = \frac{1}{2} \begin{pmatrix} \tilde{\boldsymbol{V}} & \tilde{\boldsymbol{V}} \\ \tilde{\boldsymbol{U}} & -\tilde{\boldsymbol{U}} \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{XA} \\ -X\boldsymbol{A} & \boldsymbol{A} \end{pmatrix} \begin{pmatrix} \sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} & 0 \\ 0 & -\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{XA} \\ -\boldsymbol{XA} & \boldsymbol{A} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{V}} & \tilde{\boldsymbol{V}} \\ \tilde{\boldsymbol{U}} & -\tilde{\boldsymbol{U}} \end{pmatrix}^T \tag{3.50}$$

$$\mathbf{F} = \mathbf{O}\Lambda\mathbf{O}^T \tag{3.51}$$

$$\mathbf{O} = \frac{1}{\sqrt{2}} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A} - \boldsymbol{XA}) & \tilde{\boldsymbol{V}}(\boldsymbol{A} + \boldsymbol{XA}) \\ \tilde{\boldsymbol{U}}(\boldsymbol{A} + \boldsymbol{XA}) & -\tilde{\boldsymbol{U}}(\boldsymbol{A} - \boldsymbol{XA}) \end{pmatrix} \tag{3.52}$$

$$\Lambda = \begin{pmatrix} \sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} & 0 \\ 0 & -\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} \end{pmatrix} \tag{3.53}$$

This completes the proof.

3. *Proof of third statement:*

$$\boldsymbol{O}^T\boldsymbol{Q}(0) = \frac{1}{\sqrt{2}} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A} - \boldsymbol{XA}) & \tilde{\boldsymbol{V}}(\boldsymbol{A} + \boldsymbol{XA}) \\ \tilde{\boldsymbol{U}}(\boldsymbol{A} + \boldsymbol{XA}) & -\tilde{\boldsymbol{U}}(\boldsymbol{A} - \boldsymbol{XA}) \end{pmatrix}^T \begin{pmatrix} \boldsymbol{W}_1^T \\ \boldsymbol{W}_2 \end{pmatrix} \tag{3.54}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A} - \boldsymbol{XA}) & \tilde{\boldsymbol{V}}(\boldsymbol{A} + \boldsymbol{XA}) \\ \tilde{\boldsymbol{U}}(\boldsymbol{A} + \boldsymbol{XA}) & -\tilde{\boldsymbol{U}}(\boldsymbol{A} - \boldsymbol{XA}) \end{pmatrix}^T \begin{pmatrix} \boldsymbol{VS}_1\boldsymbol{R}^T \\ \boldsymbol{US}_2\boldsymbol{R}^T \end{pmatrix} \tag{3.55}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} (\boldsymbol{A} - \boldsymbol{XA})\tilde{\boldsymbol{V}}^T\boldsymbol{VS}_1\boldsymbol{R}^T + (\boldsymbol{A} + \boldsymbol{XA})\tilde{\boldsymbol{U}}^T\boldsymbol{US}_2\boldsymbol{R}^T \\ (\boldsymbol{A} + \boldsymbol{XA})\tilde{\boldsymbol{V}}^T\boldsymbol{VS}_1\boldsymbol{R}^T - (\boldsymbol{A} - \boldsymbol{XA})\tilde{\boldsymbol{U}}^T\boldsymbol{US}_2\boldsymbol{R}^T \end{pmatrix} \tag{3.56}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} (\boldsymbol{A} - \boldsymbol{XA})\tilde{\boldsymbol{V}}^T\boldsymbol{VS}_1 + (\boldsymbol{A} + \boldsymbol{XA})\tilde{\boldsymbol{U}}^T\boldsymbol{US}_2 \\ (\boldsymbol{A} + \boldsymbol{XA})\tilde{\boldsymbol{V}}^T\boldsymbol{VS}_1 - (\boldsymbol{A} - \boldsymbol{XA})\tilde{\boldsymbol{U}}^T\boldsymbol{US}_2 \end{pmatrix} \boldsymbol{R}^T \tag{3.57}$$

$$\boldsymbol{O}e^{\boldsymbol{\Lambda}t/\tau} = \frac{1}{\sqrt{2}} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A} - \boldsymbol{XA}) & \tilde{\boldsymbol{V}}(\boldsymbol{A} + \boldsymbol{XA}) \\ \tilde{\boldsymbol{U}}(\boldsymbol{A} + \boldsymbol{XA}) & -\tilde{\boldsymbol{U}}(\boldsymbol{A} - \boldsymbol{XA}) \end{pmatrix} \begin{pmatrix} e^{\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} & 0 \\ 0 & e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \end{pmatrix} \tag{3.58}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} & \tilde{\boldsymbol{V}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \\ \tilde{\boldsymbol{U}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} & -\tilde{\boldsymbol{U}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \end{pmatrix} \tag{3.59}$$

Let

$$\boldsymbol{B} = \boldsymbol{S}_2 \tilde{\boldsymbol{U}}^T \tilde{\boldsymbol{U}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) + \boldsymbol{S}_1 \boldsymbol{V}^T \tilde{\boldsymbol{V}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) \tag{3.60}$$

$$\boldsymbol{C} = \boldsymbol{S}_2 \tilde{\boldsymbol{U}}^T \tilde{\boldsymbol{U}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) - \boldsymbol{S}_1 \boldsymbol{V}^T \tilde{\boldsymbol{V}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) \tag{3.61}$$

$$\boldsymbol{O}^T \boldsymbol{Q}(0) = \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{B}^T \\ -\boldsymbol{C}^T \end{pmatrix} \boldsymbol{R}^T \tag{3.62}$$

$$= \frac{1}{2} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} & \tilde{\boldsymbol{V}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \\ \tilde{\boldsymbol{U}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} & -\tilde{\boldsymbol{U}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \end{pmatrix} \begin{pmatrix} \boldsymbol{B}^T \\ -\boldsymbol{C}^T \end{pmatrix} \boldsymbol{R}^T \tag{3.63}$$

$$= \frac{1}{2} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \boldsymbol{B}^T - \tilde{\boldsymbol{V}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \boldsymbol{C}^T \\ \tilde{\boldsymbol{U}}(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \boldsymbol{B}^T + \tilde{\boldsymbol{U}}(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \boldsymbol{C}^T \end{pmatrix} \boldsymbol{R}^T \tag{3.64}$$

$$= \frac{1}{2} \begin{pmatrix} \tilde{\boldsymbol{V}} \left[(\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) - (\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \boldsymbol{C}^T \boldsymbol{B}^{-T} e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \right] \\ \tilde{\boldsymbol{U}} \left[(\boldsymbol{A}+\boldsymbol{X}\boldsymbol{A}) + (\boldsymbol{A}-\boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \boldsymbol{C}^T \boldsymbol{B}^{-T} e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \right] \end{pmatrix} \boldsymbol{B}^T e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}\mathbb{I}}\frac{t}{\tau}} \boldsymbol{R}^T \tag{3.65}$$

Now,

$$\left[\boldsymbol{I} + \frac{1}{2} \boldsymbol{Q}(0)^T \left(e^{\boldsymbol{F}\frac{t}{\tau}} \boldsymbol{F}^{-1} e^{\boldsymbol{F}\frac{t}{\tau}} - \boldsymbol{F}^{-1} \right) \boldsymbol{Q}(0) \right]^{-1} \tag{3.66}$$

$$= \left[\boldsymbol{I} + \frac{1}{2} \boldsymbol{Q}(0)^T \left(\boldsymbol{O} e^{\boldsymbol{\Lambda}\frac{t}{\tau}} \boldsymbol{\Lambda}^{-1} \boldsymbol{O}^T - \boldsymbol{O} \boldsymbol{\Lambda}^{-1} \boldsymbol{O}^T \right) \boldsymbol{Q}(0) \right]^{-1} \tag{3.67}$$

$$= \left[\boldsymbol{I} + \frac{1}{2} \boldsymbol{Q}(0)^T \boldsymbol{O} \left(e^{\boldsymbol{\Lambda}\frac{t}{\tau}} \boldsymbol{\Lambda}^{-1} - \boldsymbol{\Lambda}^{-1} \right) \boldsymbol{O}^T \boldsymbol{Q}(0) \right]^{-1} \tag{3.68}$$

$$\left[\boldsymbol{I} + \frac{1}{2} \boldsymbol{Q}(0)^T \boldsymbol{O} \left(e^{\boldsymbol{\Lambda}\frac{t}{\tau}} \boldsymbol{\Lambda}^{-1} - \boldsymbol{I} \right) \boldsymbol{\Lambda}^{-1} \boldsymbol{O}^T \boldsymbol{Q}(0) \right]^{-1} \tag{3.69}$$

$$= \left[\boldsymbol{I} + \frac{1}{2} \boldsymbol{Q}(0)^T \boldsymbol{O} e^{\boldsymbol{\Lambda}\frac{t}{\tau}} \boldsymbol{\Lambda}^{-1} \boldsymbol{O}^T \boldsymbol{Q}(0) - \frac{1}{2} \boldsymbol{Q}(0)^T \boldsymbol{O} \boldsymbol{\Lambda}^{-1} \boldsymbol{O}^T \boldsymbol{Q}(0) \right]^{-1} \tag{3.70}$$

$$= \left[\boldsymbol{I} + \frac{1}{4} \left[\boldsymbol{B} - \boldsymbol{C} \right] \begin{pmatrix} e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}}\frac{t}{\tau}} & 0 \\ 0 & e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}}\frac{t}{\tau}} \end{pmatrix} \begin{pmatrix} \boldsymbol{B}^T \\ -\boldsymbol{C}^T \end{pmatrix} \right]^{-1} \tag{3.71}$$

$$= \left[\boldsymbol{I} + \frac{1}{4} \left[\boldsymbol{B} - \boldsymbol{C} \right] \begin{pmatrix} e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}}\frac{t}{\tau}} & 0 \\ 0 & e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}}\frac{t}{\tau}} \end{pmatrix} \begin{pmatrix} \boldsymbol{B}^T \\ -\boldsymbol{C}^T \end{pmatrix} \right]^{-1} \tag{3.72}$$

$$= \left[\boldsymbol{I} + \frac{1}{4} \left[\boldsymbol{B} - \boldsymbol{C} \right] \begin{pmatrix} e^{\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}}\frac{t}{\tau}} & 0 \\ 0 & e^{-\sqrt{\tilde{\boldsymbol{S}}^2+\frac{\lambda^2}{4}}\frac{t}{\tau}} \end{pmatrix} \begin{pmatrix} \boldsymbol{B}^T \\ -\boldsymbol{C}^T \end{pmatrix} \right]^{-1} \tag{3.73}$$

$$= \left[\boldsymbol{I} + \frac{1}{4} [\boldsymbol{B} - \boldsymbol{C}] \begin{pmatrix} e^{\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} & 0 \\ 0 & e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \end{pmatrix} \begin{pmatrix} \boldsymbol{B}^T \\ -\boldsymbol{C}^T \end{pmatrix} \right]^{-1} \tag{3.74}$$

$$\begin{aligned} = \Bigg[& \boldsymbol{I} + \frac{1}{4} \left(\boldsymbol{B} e^{\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} (\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}})^{-1} \boldsymbol{B}^T - \boldsymbol{C} e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} (\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}})^{-1} \boldsymbol{C}^T \right) \\ & - \left(\boldsymbol{B} (\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}})^{-1} \boldsymbol{B}^T + \boldsymbol{C} (\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}})^{-1} \boldsymbol{C}^T \right) \Bigg]^{-1} \end{aligned} \tag{3.75}$$

$$= \left[\boldsymbol{I} + \frac{1}{4} \left(\boldsymbol{B} \left(e^{\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} - \boldsymbol{I} \right) (\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}})^{-1} \boldsymbol{B}^T - \boldsymbol{C} \left(e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} - \boldsymbol{I} \right) (\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}})^{-1} \boldsymbol{C}^T \right) \right]^{-1} \tag{3.76}$$

So, Final form:

$$\begin{aligned} & \Big[\tilde{\boldsymbol{V}} (\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}) - (\boldsymbol{X}\boldsymbol{A} + \boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \boldsymbol{C} \boldsymbol{B}^T e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \\ & \tilde{\boldsymbol{U}} (\boldsymbol{X}\boldsymbol{A} + \boldsymbol{A}) - (\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}) e^{\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \boldsymbol{C} \boldsymbol{B}^T e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \Big] \\ & \Big[4 e^{\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \boldsymbol{B}^T \boldsymbol{A}(0) \boldsymbol{B} e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \\ & + \left(\boldsymbol{I} - e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \right) \left(\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \right)^{-1} \boldsymbol{C} \boldsymbol{B}^T \left(e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} - \boldsymbol{I} \right) \left(\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \right)^{-1} \Big]^{-1} \\ & \Big[\tilde{\boldsymbol{V}} (\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}) - (\boldsymbol{X}\boldsymbol{A} + \boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \boldsymbol{C} \boldsymbol{B}^T e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \\ & \tilde{\boldsymbol{U}} (\boldsymbol{X}\boldsymbol{A} + \boldsymbol{A}) - (\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}) e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \boldsymbol{C} \boldsymbol{B}^T e^{-\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}} \frac{t}{\tau}} \Big]^T \end{aligned} \tag{3.77}$$

This completes the proof.

We have hence derived an expression for the dynamics of the $QQ^T(t)$ matrix which is numerically stable and in terms of the Singular Value Decomposition of the initial weights and the data. This expression is a generalisation of ([38])'s expression for an aligned zero-Balanced network, ([5])'s expression for an unaligned zero-Balanced network and the previous expression for an aligned λ-Balanced network.

It is important to notice that as $\lambda \to 0$ $A \to \mathbb{I}$ and $XA \to 0$, and $S_1 = S_2 = \sqrt{S}$, so we are left with the exact expression given in ([5]) paper.

This expression is very powerful because it gives access to exact dynamics of the Loss, Network Output, Internal Representations and Neural Tangent Kernel of the network, among other quantities. An example of these exact dynamics is shown below:

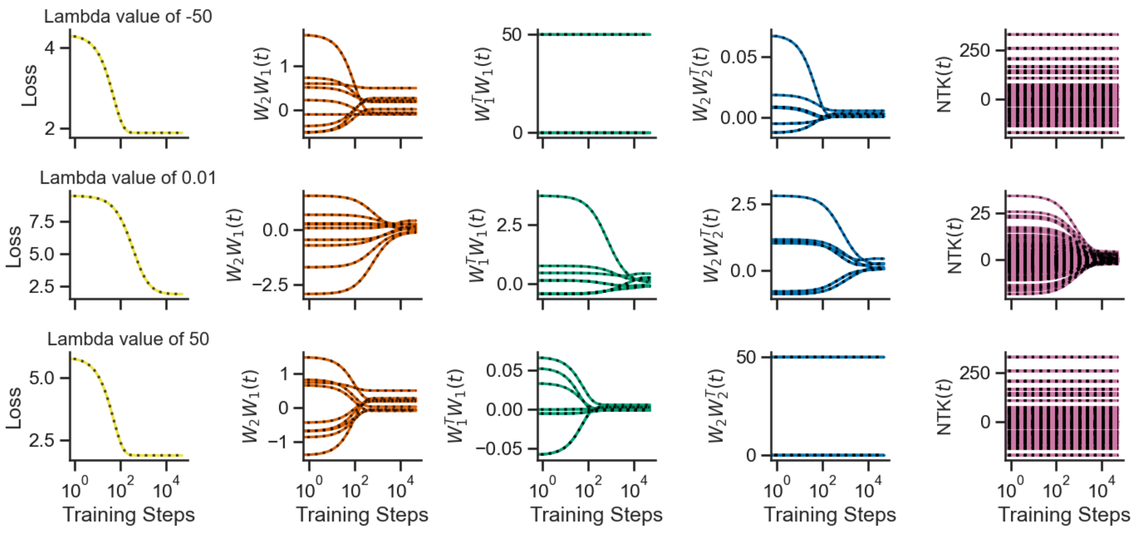


Figure 3.3: Comparison of analytical and empirical dynamics for varying values of λ. The figure shows that the analytical solution matches empirical results for both large positive and negative values of λ, as well as values close to 0.

(**Figure 3.3**) shows that the analytical solution matches empirical results for both large negative and positive values of λ as well as λ close to 0. Note that at $\lambda = 0$ the expression is not defined. However in the limit as $\lambda \to 0$ our expression matches the one from ([5]). It is important to note that for a large negative λ the input representations ($\boldsymbol{W}_1(t)^T\boldsymbol{W}_1(t)$) remain almost constant. In addition, the output representations ($\boldsymbol{W}_2(t)\boldsymbol{W}_2(t)^T$) are much smaller in magnitude than the corresponding output representations for $\lambda \approx 0$. Similarly, for large positive λ the output representations remain almost constant and the input representations are smaller in magnitude than the corresponding input representations for $\lambda \approx 0$. In addition, the NTK remains almost constant for $|\lambda|$ large. In later chapters we will develop a theory to explain these results.

All the derivations for exact dynamics in this report make the gradient flow assumption: the network's learning rate is infinitesimally small. As mentioned in the background, the gradient flow assumption is extremely useful for understanding network dynamics. However, as the learning rate of our model increases, we will expect the model to behave less closely to our analytical solution. We are interested in how the accuracy of our expression changes as the learning rate increases. We are also interested in how the accuracy of the analytical solution changes with network size.

To answer this question an experiment was designed with the following setup: networks of sizes $5, 10, 15$ and 25, a constant λ value of 3 and learning rates logarithmically spaced from 10^{-4} to 10^{-2} are initialised. Each network is trained for 200 steps, and at each step, the difference between the relative deviation of the analytical loss from the empirical loss is calculated. These deviations are then averaged to obtain an Average Deviation value for this trial. For each (Network Size, Learning Rate) combination the same trial is done 40 times, obtaining random λ-Balanced weights and training data each time. The Average Deviations of each trial are then Averaged again, grouped by Network Size.

Below (**Figure 3.4**) is a graph with the results of the experiment. We notice that the average deviation of the network increases with learning rate but remains below 0.05 for learning rates below 0.01.

As the learning rate increases the average deviation does so as well. This corresponds to our intuition of the analytical solution being more and more precise as the empirical setup is closer to the gradient flow assumption. In addition, the average deviation is larger for larger networks. This is due to the fact that the error of the (lack of) gradient flow is amplified in large networks, as there are many more computations being performed.

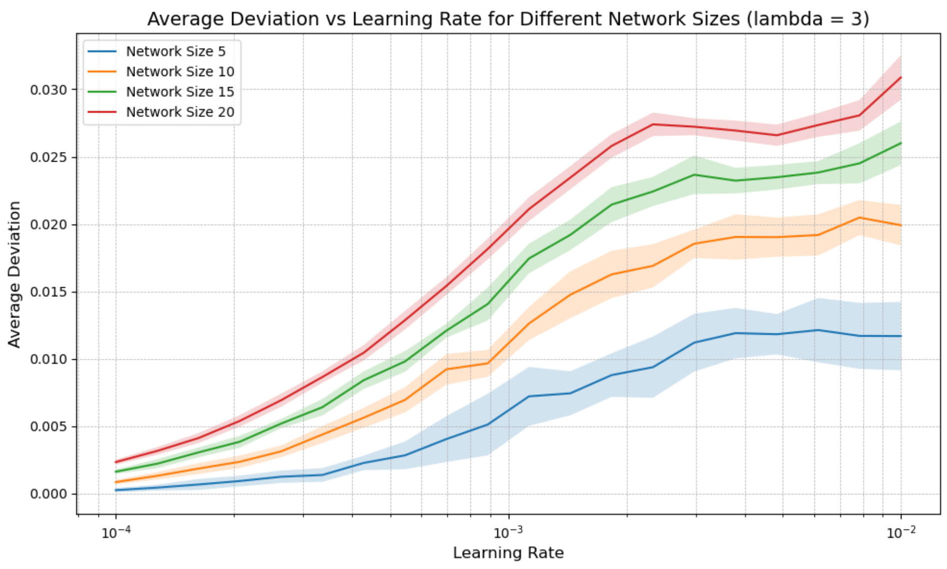


Figure 3.4: Graph showing the accuracy of the analytical solution for varying learning rates with λ fixed. The average deviation increases with the learning rate, indicating the precision of the analytical solution decreases as the empirical setup deviates from the gradient flow assumption.

From (3.3.1), note that the analytical solution makes use of $e^{-K\frac{t}{\tau}}$, with $K = \sqrt{\tilde{S}^2 + \frac{\lambda^2}{4}}$. As λ increases in magnitude, K is closer and closer to $|\lambda|/2$. Hence the exponential resembles the shape $e^{-(\frac{|\lambda|}{\tau})t} = e^{-\frac{t}{\tau'}}$ with $\tau' = \frac{\tau}{\lambda}$. Define $\eta'\frac{1}{\tau'}$ to be the "Real Learning Rate" of the network. We can see that increasing the magnitude of λ increases the Real Learning Rate of the network.

Hence we would expect that for a fixed learning rate, increasing λ will increase the Real Learning Rate and hence make our empirical network farther away from the gradient flow assumption, hence the analytical and empirical dynamics will yield different results. A similar experiment was run to confirm this property, this time varying Network Size and λ while keeping a fixed learning rate.

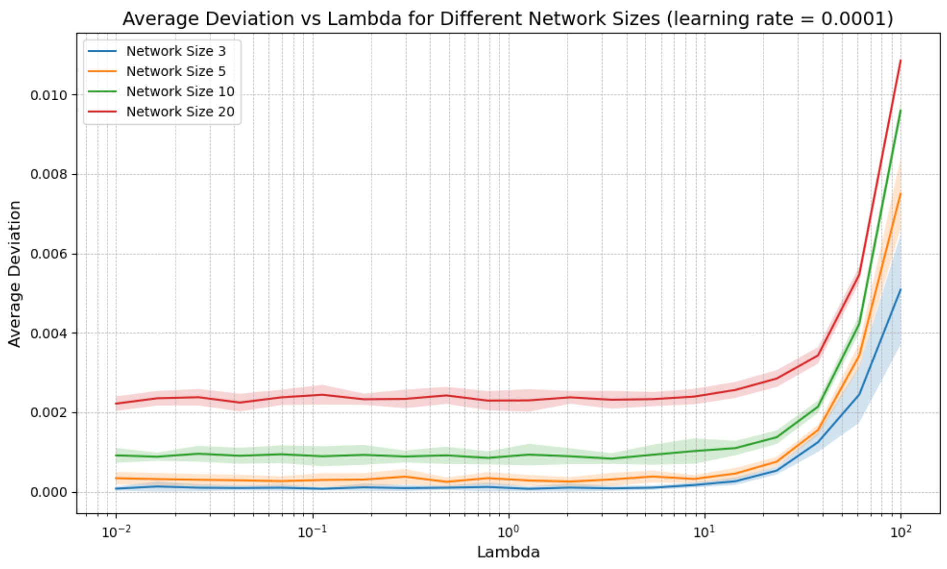


Figure 3.5: Graph showing the accuracy of the analytical solution for varying λ values with the learning rate fixed. The figure highlights how increasing λ affects the empirical network's deviation from the gradient flow assumption.

Following this idea, we can see that the quantity that truly controls the gradient flow assumption is the Real Learning Rate $\eta' = \frac{\lambda}{\tau}$. According to this theory, if the Real Learning Rate is kept constant, the analytical and empirical dynamics should not diverge for different combinations of λ and τ. A similar experiment as previously outlined was performed to confirm this hypothesis:

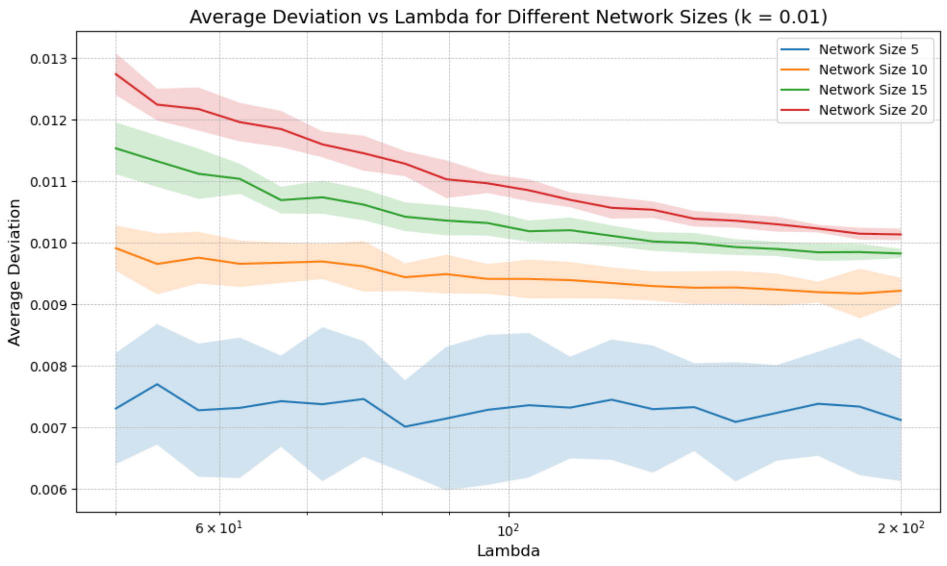


Figure 3.6: Analytical and empirical dynamics for varying λ values with a constant ratio of λ to learning rate. The figure demonstrates that keeping the real learning rate constant results in similar dynamics across different λ and τ combinations.

Various observations can be made from (**Figure 3.6**): Firstly, as we keep the Real Learning Rate constant while varying λ and τ, the average deviation remains almost constant across different values. For larger networks, the error slightly decreases as λ increases. Additionally, the variability of the average deviation across trials is lower for networks of higher dimensions, although the average deviation itself is higher. This aligns with the intuition about the variance of large sample averages: larger networks imply more computations (samples), so the average error will have a lower variance.

Below is a summary o the key findings and limitations of the work shown in this section.

1. We have reformulated a solution to Fukumizu's equation in the λ-Balanced equal input output case in terms of singular values of the data and initial weights.
2. The solution is extremely accurate for small learning rates relative to λ.
3. With this solution we can obtain dynamics of output, internal representations and NTK throughout training.
4. By taking the limit $t \to \infty$, one can derive the output at convergence of the network very intuitively due to the vanishing exponentials, as well as the representations and NTK at convergence and how this depends on λ.
5. The equation shows how singular values and singular vectors are learned, which can be used for further study in the alignment process of linear neural networks.
6. However, this equation only works for equal input output dimensions. In order to gain a better understanding of how the relationship between network structure and λ affects dynamics we must derive an equation for unequal input output dimensions.

3.4 Progress in Exact Solutions for Unaligned λ-Balanced Network of Unequal Dimensions

In this section, we explore the advancements in deriving exact solutions for unaligned λ-Balanced networks with unequal input and output dimensions. The idea is to develop an analytical solution that works just as well as the one given in (3.3.1) but for unequal input output dimensions.

We begin by presenting some theorems setting some restrictions on the possible combinations of network dimensions and λ values. Next, we show a diagonalisation for the F matrix with unequal input outputs.

Theorem 3.4.1 ***[Relationship between λ and network dimensions]*** *Let $\boldsymbol{W}_2, \boldsymbol{W}_1$ be λ-Balanced weights of dimensions (n_o, n_h) and (n_h, n_i) respectively with $\lambda \neq 0$. Then:*

1. $\min(n_i, n_o) \leq n_h \leq \max(n_i, n_o)$
2. $n_o > n_i \implies \lambda > 0$, $n_o < n_i \implies \lambda < 0$

Proof 6 (Proof of Theorem 3.4.1) *We have $n_h \geq \min(n_i, n_o)$ by assumption since the network is not bottlenecked.*
Suppose $n_h > \min(n_i, n_o)$. Let $\boldsymbol{USV}^T = \boldsymbol{W}_1$.
We can write $\boldsymbol{W}_2 = \boldsymbol{U}_2\boldsymbol{S}_2\boldsymbol{R}^T$, $\boldsymbol{W}_1 = \boldsymbol{R}\boldsymbol{S}_1\boldsymbol{V}_1^T$ with $\boldsymbol{S}_2, \boldsymbol{S}_1$ of dimensions (n_o, n_h) and (n_h, n_i) respectively. Since $n_h > \max(n_o, n_i)$:

$$\boldsymbol{S}_2 = \begin{pmatrix} \boldsymbol{S}_2' & 0 \end{pmatrix}$$

where $\boldsymbol{S}_2'$ is a diagonal matrix of size (n_o, n_o).

$$\boldsymbol{S}_1 = \begin{pmatrix} \boldsymbol{S}_1' \\ 0 \end{pmatrix}$$

where $\boldsymbol{S}_1'$ is a diagonal matrix of size (n_i, n_i).
We know:

$$\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T = \lambda\boldsymbol{I}$$

$$\boldsymbol{R}\boldsymbol{S}_2^T\boldsymbol{S}_2\boldsymbol{R}^T - \boldsymbol{R}\boldsymbol{S}_1\boldsymbol{S}_1^T\boldsymbol{R}^T = \lambda\boldsymbol{I}$$

$$\boldsymbol{S}_2^T\boldsymbol{S}_2 - \boldsymbol{S}_1\boldsymbol{S}_1^T = \lambda\boldsymbol{I}$$

$$\begin{pmatrix} \boldsymbol{S}_2'^2 & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} \boldsymbol{S}_1'^2 & 0 \\ 0 & 0 \end{pmatrix} = \lambda\boldsymbol{I}$$

The left-hand side of the equation will have zeros on the diagonal, so $\lambda = 0$. We have a contradiction so (1) must be true.
Now suppose $\min(n_i, n_o) \leq n_h \leq \max(n_i, n_o)$.
Suppose $n_o > n_i$
Similarly, we can write:

$$\boldsymbol{S}_2 = \begin{pmatrix} \boldsymbol{S}_2' \\ 0 \end{pmatrix}$$

with $\boldsymbol{S}_2'$ a diagonal matrix of shape (n_h, n_h).

$$\boldsymbol{S}_1 = \begin{pmatrix} \boldsymbol{S}_1' & 0 \end{pmatrix}$$

with $\boldsymbol{S}_1'$ a diagonal matrix of shape (n_i, n_i).

$$\boldsymbol{S}_2^T\boldsymbol{S}_2 - \boldsymbol{S}_1\boldsymbol{S}_1^T = \lambda\boldsymbol{I}$$

$$\boldsymbol{S}_2'^2 - \begin{pmatrix} \boldsymbol{S}_1'^2 & 0 \\ 0 & 0 \end{pmatrix} = \lambda\boldsymbol{I}$$

Since some elements of the diagonal will contain only squared terms from $\boldsymbol{S}_2'^2$ and not $\boldsymbol{S}_1'^2$, they will be non-negative. Since $\lambda \neq 0$, $\lambda > 0$.
Suppose $n_i > n_o$.

$$\boldsymbol{S}_2 = \begin{pmatrix} \boldsymbol{S}'_2 & 0 \end{pmatrix}$$

with $\boldsymbol{S}'_2$ a diagonal matrix of shape (n_o, n_o).

$$\boldsymbol{S}_1 = \begin{pmatrix} \boldsymbol{S}'_1 \\ 0 \end{pmatrix}$$

with $\boldsymbol{S}'_1$ a diagonal matrix of shape (n_i, n_i).

$$\boldsymbol{S}_2^T \boldsymbol{S}_2 - \boldsymbol{S}_1 \boldsymbol{S}_1^T = \lambda \boldsymbol{I}$$

$$\begin{pmatrix} \boldsymbol{S}'^2_2 & 0 \\ 0 & 0 \end{pmatrix} - \boldsymbol{S}'^2_1 = \lambda \boldsymbol{I}$$

Since some elements of the diagonal will contain only terms from $-\boldsymbol{S}'^2_1$ and not $\boldsymbol{S}'^2_2$, they will be negative. Since $\lambda \neq 0$, $\lambda < 0$.

Proof 7

In order to produce a numerically stable version of Fukumizu's equation, we must first diagonalise F in terms of the singular value decomposition of the task (as in [5]) and λ. As expected, the first singular values of F are the same as when F has equal input output dimensions, but there are some extra singular values corresponding to λ.

Theorem 3.4.2 ***[Diagonalisation of F for unequal input-output dimensions]***
Let

$$\boldsymbol{F} = \begin{bmatrix} -\frac{\lambda}{2}\mathbb{I} & \tilde{\boldsymbol{\Sigma}}^{yxT} \\ \tilde{\boldsymbol{\Sigma}}^{yx} & \frac{\lambda}{2}\mathbb{I} \end{bmatrix} \tag{3.78}$$

Then

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{O} & \boldsymbol{M} \end{pmatrix} \begin{pmatrix} \sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} & 0 & 0 \\ 0 & -\sqrt{\tilde{\boldsymbol{S}}^2 + \frac{\lambda^2}{4}\mathbb{I}} & 0 \\ 0 & 0 & sgn(n_o - n_i)\frac{\lambda}{2} \end{pmatrix} \begin{pmatrix} \boldsymbol{O}^T \\ \boldsymbol{M}^T \end{pmatrix} \tag{3.79}$$

With

$$\boldsymbol{O} = \frac{1}{\sqrt{2}} \begin{pmatrix} \tilde{\boldsymbol{V}}(\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}) & \tilde{\boldsymbol{V}}(\boldsymbol{A} + \boldsymbol{X}\boldsymbol{A}) \\ \tilde{\boldsymbol{U}}(\boldsymbol{A} + \boldsymbol{X}\boldsymbol{A}) & -\tilde{\boldsymbol{U}}(\boldsymbol{A} - \boldsymbol{X}\boldsymbol{A}) \end{pmatrix}, \quad \textit{and}$$
$$\boldsymbol{M} = \begin{pmatrix} \tilde{\boldsymbol{V}}_\perp \\ \tilde{\boldsymbol{U}}_\perp \end{pmatrix} \tag{3.80}$$

Proof 8 (Proof Sketch of 3.4.2) *We offer a proof sketch of the theorem.*
As in [5], it is clear that $\boldsymbol{O}\boldsymbol{M} = 0$. Suppose $n_o > n_i$ (the proof for $n_i > n_o$ is similar in nature). Then $\tilde{\boldsymbol{V}}_\perp \neq 0, \tilde{\boldsymbol{U}}_\perp = 0$.
Now, we can write

$$\boldsymbol{F} = \begin{pmatrix} -\frac{\lambda}{2}\mathbb{I} & (\tilde{\boldsymbol{V}}\,\tilde{\boldsymbol{V}}_\perp)\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T \\ \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}(\tilde{\boldsymbol{V}}\,\tilde{\boldsymbol{V}}_\perp)^T & \frac{\lambda}{2}\mathbb{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}' & \tilde{\boldsymbol{V}}_\perp\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T \\ \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}_\perp^T & \frac{\lambda}{2}\mathbb{I} \end{pmatrix} \tag{3.81}$$

With

$$\boldsymbol{F}' = \begin{pmatrix} -\frac{\lambda}{2}\mathbb{I} & \tilde{\boldsymbol{V}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T \\ \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T & \frac{\lambda}{2}\mathbb{I} \end{pmatrix} \tag{3.82}$$

Hence

$$\begin{pmatrix} \boldsymbol{O}^T \\ \boldsymbol{M}^T \end{pmatrix} \boldsymbol{F} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{M} \end{pmatrix} = \begin{pmatrix} \boldsymbol{O}^T \\ \boldsymbol{M}^T \end{pmatrix} \begin{pmatrix} \boldsymbol{F}' & \tilde{\boldsymbol{V}}_\perp\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T \\ \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}_\perp^T & \frac{\lambda}{2}\mathbb{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{M} \end{pmatrix} \tag{3.83}$$

$$= \begin{pmatrix} \boldsymbol{O}\boldsymbol{F}'\boldsymbol{O}^T & 0 \\ 0 & \frac{\lambda}{2}\mathbb{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Lambda} & 0 \\ 0 & \frac{\lambda}{2}\mathbb{I} \end{pmatrix} \tag{3.84}$$

Which is a diagonal matrix as required.
Now suppose $n_o < n_i$. Then $\tilde{\boldsymbol{V}}_\perp = 0, \tilde{\boldsymbol{U}}_\perp \neq 0$.

$$\boldsymbol{F} = \begin{pmatrix} -\frac{\lambda}{2}\mathbb{I} & \tilde{\boldsymbol{V}}\tilde{\boldsymbol{S}}(\tilde{\boldsymbol{U}}\,\tilde{\boldsymbol{U}}_\perp)^T \\ (\tilde{\boldsymbol{U}}\,\tilde{\boldsymbol{U}}_\perp)\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}T & \frac{\lambda}{2}\mathbb{I} \end{pmatrix} = \begin{pmatrix} -\frac{\lambda}{2}\mathbb{I} & \tilde{\boldsymbol{V}}_\perp\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T \\ \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}_\perp^T & \boldsymbol{F}' \end{pmatrix} \tag{3.85}$$

Hence

$$\begin{pmatrix} \boldsymbol{O}^T \\ \boldsymbol{M}^T \end{pmatrix} \boldsymbol{F} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{M} \end{pmatrix} = \begin{pmatrix} \boldsymbol{O}^T \\ \boldsymbol{M}^T \end{pmatrix} \begin{pmatrix} -\frac{\lambda}{2}\mathbb{I} & \tilde{\boldsymbol{V}}_\perp\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T \\ \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}_\perp^T & \boldsymbol{F}' \end{pmatrix} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{M} \end{pmatrix} \tag{3.86}$$

$$= \begin{pmatrix} -\frac{\lambda}{2}\mathbb{I} & 0 \\ 0 & \boldsymbol{O}\boldsymbol{F}'\boldsymbol{O}^T \end{pmatrix} = \begin{pmatrix} -\frac{\lambda}{2}\mathbb{I} & 0 \\ 0 & \boldsymbol{\Lambda} \end{pmatrix} \tag{3.87}$$

Which is a diagonal matrix as specificed. This comlpetes the proof.

To better test different potential numerically stable analytical solutions we developed functions to generate λ-Balanced aligned and unaligned weights using (4.1.1, covered later in the text). This are shown on the next page (**Algorithm 1, Algorithm 2**)
Future work involves producing a derivation to ([5]) in order to produce an expression with negative exponential. This will equation will be a powerful tool to understanding the relation between network structure, λ, and the network's dynamics and learning regime.

Algorithm 1 Get λ-Balanced

```
1: function GET_LAMBDA_BALANCED($\lambda$, $in\_dim$, $hidden\_dim$, $out\_dim$, $\sigma = 1$)
2:     if $out\_dim > in\_dim$ and $\lambda < 0$ then
3:         raise Exception('Lambda must be positive if out_dim > in_dim')
4:     end if
5:     if $in\_dim > out\_dim$ and $\lambda > 0$ then
6:         raise Exception('Lambda must be positive if in_dim > out_dim')
7:     end if
8:     if $hidden\_dim < \min(in\_dim, out\_dim)$ then
9:         raise Exception('Network cannot be bottlenecked')
10:    end if
11:    if $hidden\_dim > \max(in\_dim, out\_dim)$ and $\lambda \neq 0$ then
12:        raise Exception('hidden_dim cannot be the largest dimension if lambda is not 0')
13:    end if
14:    $W_1 \leftarrow \sigma \cdot$ random normal matrix$(hidden\_dim, in\_dim)$
15:    $W_2 \leftarrow \sigma \cdot$ random normal matrix$(out\_dim, hidden\_dim)$
16:    $[U, S, Vt] \leftarrow \text{SVD}(W_2 \cdot W_1)$
17:    $R \leftarrow$ random orthonormal matrix$(hidden\_dim)$
18:    $S2_{equal\_dim} \leftarrow \sqrt{\left(\sqrt{\lambda^2 + 4 \cdot S^2} + \lambda\right)/2}$
19:    $S1_{equal\_dim} \leftarrow \sqrt{\left(\sqrt{\lambda^2 + 4 \cdot S^2} - \lambda\right)/2}$
20:    if $out\_dim > in\_dim$ then
21:        $S2 \leftarrow \begin{bmatrix} S2_{equal\_dim} & 0 \\ 0 & \sqrt{\lambda} \cdot I_{hidden\_dim - in\_dim} \end{bmatrix}$
22:        $S1 \leftarrow \begin{bmatrix} S1_{equal\_dim} & 0 \\ 0 & 0 \end{bmatrix}$
23:    else if $in\_dim > out\_dim$ then
24:        $S1 \leftarrow \begin{bmatrix} S1_{equal\_dim} & 0 \\ 0 & \sqrt{-\lambda} \cdot I_{hidden\_dim - out\_dim} \end{bmatrix}$
25:        $S2 \leftarrow \begin{bmatrix} S2_{equal\_dim} & 0 \\ 0 & 0 \end{bmatrix}$
26:    end if
27:    $init\_W_2 \leftarrow U \cdot S2 \cdot R^T$
28:    $init\_W_1 \leftarrow R \cdot S1 \cdot Vt$
29:    return $(init\_W_1, init\_W_2)$
30: end function
```

Algorithm 2 Get λ-Balanced Aligned

```
1: function GET_LAMBDA_BALANCED_ALIGNED($\lambda$, $in\_dim$, $hidden\_dim$, $out\_dim$, $X$, $Y$,
   $\sigma = 1$)
2:     $[U, \_, Vt] \leftarrow \text{SVD}(Y \cdot X^T)$
3:     $[W_1, W_2] \leftarrow$ get_lambda_balanced$(\lambda, in\_dim, hidden\_dim, out\_dim)$
4:     $[U\_, \_, Vt\_] \leftarrow \text{SVD}(W_2 \cdot W_1)$
5:     $init\_W_2 \leftarrow U \cdot U\_^T \cdot W_2$
6:     $init\_W_1 \leftarrow W_1 \cdot Vt\_^T \cdot Vt$
7:     return $(init\_W_1, init\_W_2)$
8: end function
```

Chapter 4

Applications

4.1 λ and the Transition from the Lazy to the Rich Regime

The Lazy and Rich regimes are defined by the dynamics of the Neural Tangent Kernel of the network. Lazy learning occurs when the NTK is constant, Rich learning occurs when it is not. ([8])

The NTK intuitively measures the movement of the network representations through training. As shown in ([5]), in specific experimental setup, we can calculate the NTK of the network in terms of the internal representations in a straightforward way:

$$\text{NTK} = \mathbf{I}_{N_o} \otimes \mathbf{X}^T \mathbf{W}_1^T \mathbf{W}_1(t) \mathbf{X} + \mathbf{W}_2 \mathbf{W}_2^T(t) \otimes \mathbf{X}^T \mathbf{X} \tag{4.1}$$

In order to better understand the effect of λ on NTK dynamics, we first prove some theorems involving the Singular Values of the λ-Balanced weights, and the representations of a λ-Balanced network.

Theorem 4.1.1 ***[Relationship between λ and Singular Values of weights]***
Let $\boldsymbol{W}_2, \boldsymbol{W}_1$ be λ-Balanced weights, $\boldsymbol{W}_2\boldsymbol{W}_1 = \boldsymbol{USV}^T$.
We can write $\boldsymbol{W}_2 = \boldsymbol{US}_2\boldsymbol{R}^T$, $\boldsymbol{W}_1 = \boldsymbol{RS}_1\boldsymbol{V}^T$ with

$$\boldsymbol{S}_2 = \begin{pmatrix} \left(\frac{\sqrt{\lambda^2\mathbb{I}+4\boldsymbol{S}^2}+\lambda\mathbb{I}}{2}\right)^{\frac{1}{2}} & 0 \\ 0 & \sqrt{|\lambda|}\mathbb{I}_{\max(0,n_o-n_i)} \end{pmatrix}, \quad \boldsymbol{S}_1 = \begin{pmatrix} \left(\frac{\sqrt{\lambda^2\mathbb{I}+4\boldsymbol{S}^2}-\lambda\mathbb{I}}{2}\right)^{\frac{1}{2}} & 0 \\ 0 & -\sqrt{|\lambda|}\mathbb{I}_{\max(0,n_o-n_i)} \end{pmatrix} \tag{4.2}$$

$\boldsymbol{R}$ is an orthonormal matrix.

Proof 9 (Proof of 4.1.1) *We prove the case $n_i \leq n_o$. The proof for $n_o \leq n_i$ follows the same structure. Let $\boldsymbol{USV}^T = \boldsymbol{W}_2(t)\boldsymbol{W}_1(t)$ be the Singular Value Decomposition of the product of the weights at training step t. We will use $\boldsymbol{W}_2 = \boldsymbol{W}_2(t), \boldsymbol{W}_1 = \boldsymbol{W}_1(t)$ as a shorthand.*

By properties of Singular Value Decomposition, we can write $\boldsymbol{W}_2 = \boldsymbol{US}_2\boldsymbol{R}^T, \boldsymbol{W}_1 = \boldsymbol{RS}_1\boldsymbol{V}^T$, where $\boldsymbol{R}$ is an orthonormal matrix and $\boldsymbol{S}_2, \boldsymbol{S}_1$ are diagonal (possibly rectangular) matrices.

The Balanced property states that $\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T = \lambda\mathbb{I}$. We know this holds for any t since this is a conserved quantity in linear networks.

Hence

$$\boldsymbol{RS}_2^T\boldsymbol{S}_2\boldsymbol{R}^T - \boldsymbol{RS}_1\boldsymbol{S}_1\boldsymbol{R}^T = \lambda\mathbb{I} \tag{4.3}$$

$$\boldsymbol{S}_2^T\boldsymbol{S}_2 - \boldsymbol{S}_1\boldsymbol{S}_1 = \lambda\mathbb{I} \tag{4.4}$$

The matrices $\boldsymbol{S}_2, \boldsymbol{S}_1$ have shapes (n_o, n_h), (n_h, n_i) respectively. We introduce the diagonal matrices $\hat{\boldsymbol{S}}_2, \hat{\boldsymbol{S}}_1$ of shape (n_i, n_i) and $\boldsymbol{S}_2^*$ of shape $(n_o - n_i, n_o - n_i)$ such that:

$$\boldsymbol{S}_1 = \begin{pmatrix} \hat{\boldsymbol{S}}_1 \\ 0 \end{pmatrix}, \quad \boldsymbol{S}_2 = \begin{pmatrix} \hat{\boldsymbol{S}}_2 & 0 \\ 0 & \boldsymbol{S}_2^* \\ 0 & 0 \end{pmatrix} \tag{4.5}$$

Hence

$$\begin{pmatrix} \hat{\boldsymbol{S}}_2^2 & 0 \\ 0 & \boldsymbol{S}_2^{*2} \end{pmatrix} - \begin{pmatrix} \hat{\boldsymbol{S}}_1^2 & 0 \\ 0 & 0 \end{pmatrix} = \lambda \mathbb{I} \tag{4.6}$$

From the equation above and the fact that $\hat{\boldsymbol{S}}_1 \hat{\boldsymbol{S}}_2 = \boldsymbol{S}$ we derive that:

$$\hat{\boldsymbol{S}}_2 = \left(\frac{\sqrt{\lambda^2 \mathbb{I} + 4\boldsymbol{S}^2} + \lambda \mathbb{I}}{2} \right)^{\frac{1}{2}}, \quad \hat{\boldsymbol{S}}_1 = \left(\frac{\sqrt{\lambda^2 \mathbb{I} + 4\boldsymbol{S}^2} - \lambda \mathbb{I}}{2} \right)^{\frac{1}{2}}, \quad \boldsymbol{S}_2^* = \sqrt{\lambda} \mathbb{I} \tag{4.7}$$

Hence

$$\boldsymbol{W}_2 = \boldsymbol{U} \begin{pmatrix} \left(\frac{\sqrt{\lambda^2 \mathbb{I} + 4\boldsymbol{S}^2} + \lambda \mathbb{I}}{2} \right)^{\frac{1}{2}} & 0 \\ 0 & \sqrt{\lambda} \mathbb{I}_{\max(0, n_o - n_i)} \end{pmatrix} \boldsymbol{R}^T, \quad \boldsymbol{W}_1 = \boldsymbol{R} \begin{pmatrix} \left(\frac{\sqrt{\lambda^2 \mathbb{I} + 4\boldsymbol{S}^2} - \lambda \mathbb{I}}{2} \right)^{\frac{1}{2}} & 0 \\ 0 & -\sqrt{-\lambda} \mathbb{I}_{\max(0, n_o - n_i)} \end{pmatrix} \boldsymbol{V}^T \tag{4.8}$$

As required.

We have thus shown that we can completely determine the singular values of the λ-Balanced weights $\boldsymbol{W}_2, \boldsymbol{W}_1$ and the singular values of their product **at any point in training**. This is a very important result since it allows us to reason about the effect of λ on representations and consequently the Neural Tangent Kernel of the network. From (4.1.1) the relationship between λ and Network Representations is clear:

Theorem 4.1.2 ***[Relationship between λ and Network Representations]***
Consider a λ-Balanced network training on data $\boldsymbol{\Sigma}^{yx} = \tilde{\boldsymbol{U}} \tilde{\boldsymbol{S}} \tilde{\boldsymbol{V}}^T$.

1. *After convergence, the internal representations will be*

$$\boldsymbol{W}_2 \boldsymbol{W}_2^T = \tilde{\boldsymbol{U}} \begin{pmatrix} \frac{\sqrt{\lambda^2 \mathbb{I} + 4\boldsymbol{S}^2} + \lambda \mathbb{I}}{2} & 0 \\ 0 & \lambda \mathbb{I}_{\max(n_o, n_i)} \end{pmatrix} \tilde{\boldsymbol{U}}^T, \quad \boldsymbol{W}_1^T \boldsymbol{W}_1 = \tilde{\boldsymbol{V}} \begin{pmatrix} \frac{\sqrt{\lambda^2 \mathbb{I} + 4\boldsymbol{S}^2} - \lambda \mathbb{I}}{2} & 0 \\ 0 & -\lambda \mathbb{I}_{\max(n_i, n_o)} \end{pmatrix} \tilde{\boldsymbol{V}}^T$$

2. *Further, as $\lambda \to \infty$*

$$\boldsymbol{W}_2 \boldsymbol{W}_2^T = \lambda \mathbb{I}, \quad \boldsymbol{W}_1^T \boldsymbol{W}_1 = \frac{1}{\lambda} \tilde{\boldsymbol{V}} \tilde{\boldsymbol{S}}^2 \tilde{\boldsymbol{V}}^T$$

   As $\lambda \to -\infty$

$$\boldsymbol{W}_2 \boldsymbol{W}_2^T = -\frac{1}{\lambda} \tilde{\boldsymbol{U}} \tilde{\boldsymbol{S}}^2 \tilde{\boldsymbol{U}}^T, \quad \boldsymbol{W}_1^T \boldsymbol{W}_1 = -\lambda \mathbb{I}$$

It is important to note that if we let $\hat{U}\hat{S}\hat{V}^T \boldsymbol{W}_2(t) \boldsymbol{W}_1(t)$ be the Singular Value Decomposition of the product of the weights at training step t, we can apply the same logic substituting $\hat{U}\hat{S}\hat{V}^T$ for $\tilde{U}\tilde{S}\tilde{V}^T$.

Proof 10 (Proof of (4.1.2)) *(1) is clear from (4.1.1).*
For (2) we use the Taylor expansion of $f(x) = \sqrt{1 + x^2}$:

$$\frac{\sqrt{\lambda^2 \mathbb{I} + 4\tilde{\boldsymbol{S}}^2} + \lambda \mathbb{I}}{2} = \frac{|\lambda| \sqrt{1 + \left(\frac{2\tilde{\boldsymbol{S}}}{\lambda} \right)^2} + \lambda \mathbb{I}}{2} \tag{4.9}$$

$$= \frac{|\lambda| \left(1 + \left(\frac{2\boldsymbol{S}}{\lambda} \right)^2 + O(\lambda^{-4}) \right) + \lambda \mathbb{I}}{2} = \frac{|\lambda| + \lambda}{2} + \frac{\boldsymbol{S}^2}{|\lambda|} + O(\lambda^{-3}) \tag{4.10}$$

Hence

$$\lim_{\lambda\to\infty}\frac{\sqrt{\lambda^2\mathbb{I}+4\tilde{\boldsymbol{S}}^2}+\lambda\mathbb{I}}{2}=\lambda\mathbb{I},\quad \lim_{\lambda\to-\infty}\frac{\sqrt{\lambda^2\mathbb{I}+4\tilde{\boldsymbol{S}}^2}+\lambda\mathbb{I}}{2}=\frac{\boldsymbol{S}^2}{|\lambda|}=-\frac{\boldsymbol{S}^2}{\lambda} \tag{4.11}$$

Similarly,

$$\frac{\sqrt{\lambda^2\mathbb{I}+4\tilde{\boldsymbol{S}}^2}-\lambda\mathbb{I}}{2}=\frac{|\lambda|-\lambda}{2}+\frac{\boldsymbol{S}^2}{|\lambda|}+O(\lambda^{-3}) \tag{4.12}$$

$$\lim_{\lambda\to\infty}\frac{\sqrt{\lambda^2\mathbb{I}+4\tilde{\boldsymbol{S}}^2}-\lambda\mathbb{I}}{2}=\frac{\boldsymbol{S}^2}{\lambda},\quad \lim_{\lambda\to-\infty}\frac{\sqrt{\lambda^2\mathbb{I}+4\tilde{\boldsymbol{S}}^2}-\lambda\mathbb{I}}{2}=\frac{\boldsymbol{S}^2}{|\lambda|}=-\lambda\mathbb{I} \tag{4.13}$$

Since $\tilde{\boldsymbol{U}},\tilde{\boldsymbol{V}}$ are independent of λ:

$$\lim_{\lambda\to\pm\infty}\boldsymbol{W}_2\boldsymbol{W}_2^T=\tilde{\boldsymbol{U}}\left(\lim_{\lambda\to\pm\infty}\boldsymbol{S}_2\right)\tilde{\boldsymbol{U}}^T \tag{4.14}$$

$$\lim_{\lambda\to\pm\infty}\boldsymbol{W}_1^T\boldsymbol{W}_1=\tilde{\boldsymbol{V}}\left(\lim_{\lambda\to\pm\infty}\boldsymbol{S}_1\right)\tilde{\boldsymbol{V}}^T \tag{4.15}$$

Substituting in the limit expressions for $\boldsymbol{S}_1,\boldsymbol{S}_2$, we can see that the results for (2) follow. This completes the proof.

The fact that as $|\lambda|\to\infty$ one of the network representation converges to a scaled identity matrix, while the other converges to zero, is a very interesting result. Intuitively, this implies that the representations move increasingly less as λ grows in magnitude. Hence we would expect the Neural Tangent Kernel of the network to vary increasingly less as we increase $|\lambda|$. One might wonder if the NTK would even converge to zero. This is in fact the case and it is shown in the next theorem. We can use (4.1.2) to derive some results for the dynamics of the NTK of a λ-Balanced Network:

Theorem 4.1.3 ***[Relationship between λ and the Neural Tangent Kernel of the network]*** *Consider a linear network training on data $\boldsymbol{\Sigma}^{yx}=\tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T$, with initial weights $\boldsymbol{W}_2(0),\boldsymbol{W}_1(0)$ such that $\boldsymbol{W}_2(0)\boldsymbol{W}_1(0)=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ and $\boldsymbol{W}_2(0),\boldsymbol{W}_1(0)$ are λ-Balanced ($\boldsymbol{S}$ does not depend on λ).*

Let $t\geq 0$ be an arbitrary point in training. At this point let $\boldsymbol{W}_2(t)\boldsymbol{W}_1(t)=\boldsymbol{U}^*\boldsymbol{S}^*\boldsymbol{V}^{*T}$. Then

1. *For any $\lambda\in\mathbf{R}$:*

$$\begin{aligned}NTK(0)=\mathbb{I}_{n_o}\otimes\boldsymbol{X}^T\boldsymbol{V}\begin{pmatrix}\frac{\sqrt{\lambda^2\mathbb{I}+4\boldsymbol{S}^{*2}}-\lambda\mathbb{I}}{2} & 0\\ 0 & -\lambda\mathbb{I}_{\max(n_i,n_o)}\end{pmatrix}\boldsymbol{V}^T\boldsymbol{X}\\ +\boldsymbol{U}\begin{pmatrix}\frac{\sqrt{\lambda^2\mathbb{I}+4\boldsymbol{S}^{*2}}+\lambda\mathbb{I}}{2} & 0\\ 0 & \lambda\mathbb{I}_{\max(n_o,n_i)}\end{pmatrix}\boldsymbol{U}^T\otimes\boldsymbol{X}^T\boldsymbol{X}\end{aligned} \tag{4.16}$$

$$\begin{aligned}NTK(t)=\mathbb{I}_{n_o}\otimes\boldsymbol{X}^T\boldsymbol{V}^*\begin{pmatrix}\frac{\sqrt{\lambda^2\mathbb{I}+4\boldsymbol{S}^{*2}}-\lambda\mathbb{I}}{2} & 0\\ 0 & -\lambda\mathbb{I}_{\max(n_i,n_o)}\end{pmatrix}\boldsymbol{V}^{*T}\\ +\boldsymbol{U}^*\begin{pmatrix}\frac{\sqrt{\lambda^2\mathbb{I}+4\boldsymbol{S}^{*2}}+\lambda\mathbb{I}}{2} & 0\\ 0 & \lambda\mathbb{I}_{\max(n_o,n_i)}\end{pmatrix}\boldsymbol{U}^{*T}\otimes\boldsymbol{X}^T\boldsymbol{X}\end{aligned} \tag{4.17}$$

2. *As $\lambda\to\infty$:*

$$NTK(t)-NTK(0)\to\frac{1}{\lambda}\left(\mathbb{I}_{n_o}\otimes\boldsymbol{X}^T\boldsymbol{V}^*\boldsymbol{S}_2^{*2}\boldsymbol{V}^{*T}\boldsymbol{X}-\mathbb{I}_{n_o}\otimes\boldsymbol{X}^T\boldsymbol{V}\boldsymbol{S}_2^2\boldsymbol{V}^T\boldsymbol{X}\right)\to 0 \tag{4.18}$$

3. *As $\lambda\to-\infty$:*

$$NTK(t)-NTK(0)\to\frac{1}{\lambda}\left(\boldsymbol{U}\boldsymbol{S}_2^2\boldsymbol{U}^T\otimes\boldsymbol{X}^T\boldsymbol{X}-\boldsymbol{U}^*\boldsymbol{S}_2^{*2}\boldsymbol{U}^{*T}\otimes\boldsymbol{X}^T\boldsymbol{X}\right)\to 0 \tag{4.19}$$

Proof 11 (Proof of 4.1.3) *(1) Follows by substituting the expressions for the network representations in terms of λ derived in (4.1.2) from ([5])'s expression for the NTK of a linear network.*

Similarly, (2) follows from substituting the limit expressions for the network representations and the fact that the Kronecker product is linear in both arguments.

The theorem above demonstrates that as $|\lambda| \to \infty$, the Neural Tangent Kernel (NTK) of a λ-Balanced Network remains constant. **This indicates that the network operates in the Lazy regime throughout all training steps.**

This finding is significant as it highlights the impact of weight initialization on learning regimes. Specifically, initializing a Linear Network with a high $|\lambda|$ will result in Lazy Learning. We have thus discovered a method to exert complete control over the Learning Regime of the Linear Network. Depending on the application, either Lazy Learning (e.g., when the task benefits from a more stable and less adaptive model) or Rich Learning (e.g., when the task requires the model to adapt and learn complex patterns) might be preferred. The ability to dictate the learning regime purely based on initialising weights in a specific **task agnostic** manner is a powerful tool for optimizing neural network performance.

4.2 λ and the Transition to Shallow Learning Dynamics

In theorem (3.2.2) we noted that as $|\lambda| \to \infty$ a λ-Balanced aligned deep network behaves like a shallow network in terms of its learning dynamics. In this section we generalise these results to λ-Balanced unaligned networks, both in terms of learning dynamics and network representations.

To compare network representations, we must first define network representations for a shallow network:

Theorem 4.2.1 *[Expression for Representations in a Shallow Network at Convergence]*
The representations of a shallow network at convergence are given by:

$$\boldsymbol{W}^T\boldsymbol{W} = \tilde{\boldsymbol{V}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T, \quad \boldsymbol{W}\boldsymbol{W}^T = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T \tag{4.20}$$

Proof 12 (Proof of 4.2.1) *By ([38]), this network converges to the global minimum, given by $\boldsymbol{W} = \boldsymbol{Y}\boldsymbol{X}^T$ ($\boldsymbol{X}$ is whitened). Hence $\boldsymbol{W} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T$.*

We can see that the representations of a shallow network are **task specific**. In addition, we can observe that the representations are constantly changing. Consequently, one can reason that the network's NTK will change through training so the network learns in the Rich Regime.

We further investigate the similarities between λ-Balanced deep networks for $|\lambda|$ large and shallow networks by deriving their learning dynamics:

Theorem 4.2.2 ***[Gradient Flow Dynamics of a Shallow Linear Network]***

Consider a shallow linear network $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$ with whitened inputs and Mean Squared Error Loss. The gradient flow dynamics of $\boldsymbol{W}(t)$ can be expressed as:

$$\boldsymbol{W}(t) = \tilde{\boldsymbol{\Sigma}}^{yx} + (\boldsymbol{W}(0) - \tilde{\boldsymbol{\Sigma}}^{yx})e^{-\frac{t}{\tau}} \tag{4.21}$$

Proof 13 (Proof of 4.2.2) *From previous work ([36]), the gradient flow differential equation for the weights of a shallow network is given by:*

$$\tau\frac{d}{dt}\boldsymbol{W}(t) = \tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{W}(t) \tag{4.22}$$

Define $\boldsymbol{X}(t) = \tilde{\boldsymbol{\Sigma}}^{yx} - \boldsymbol{W}(t)$. Then

$$\begin{aligned} \tau\frac{d}{dt}\boldsymbol{X}(t) &= -\boldsymbol{X}(t), \\ \boldsymbol{X}(t) &= \boldsymbol{X}(0)e^{-\frac{t}{\tau}} \end{aligned} \tag{4.23}$$

Hence:

$$\tilde{\mathbf{\Sigma}}^{yx} - \boldsymbol{W}(t) = (\tilde{\mathbf{\Sigma}}^{yx} - \boldsymbol{W}(0))e^{-\frac{t}{\tau}} \tag{4.24}$$

$$\boldsymbol{W}(t) = \tilde{\mathbf{\Sigma}}^{yx} + (\boldsymbol{W}(0) - \tilde{\mathbf{\Sigma}}^{yx})e^{-\frac{t}{\tau}} \tag{4.25}$$

As required.

We have shown that unaligned shallow networks evolve exponentially. Interestingly, there is no alignment phase when the network first rotates its weights so that it can then learn singular values independently. Instead both the singular values and singular vectors are learned at the same time (as t increases, the exponential becomes closer to 0, taking $\boldsymbol{W}(t)$ closer to $\tilde{\mathbf{\Sigma}}^{yx}$).

Next, we derive an expression for the dynamics of the output of a λ-Balanced network for large $|\lambda|$:

Theorem 4.2.3 ***[Gradient Flow Dynamics of a Deep Network for Extreme λ Resemble Shallow Network Dynamics]***

Consider a λ-Balanced Deep Linear Network $\boldsymbol{Y} = \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{X}$. As $\lambda \to \pm\infty$, the dynamics of the output $\boldsymbol{W}_2(t)\boldsymbol{W}_1(t)$ are given by:

$$\boldsymbol{W}_2(t)\boldsymbol{W}_1(t) = \tilde{\mathbf{\Sigma}}^{yx} + (\boldsymbol{W}_2(0)\boldsymbol{W}_1(0) - \tilde{\mathbf{\Sigma}}^{yx})e^{-|\lambda|\frac{t}{\tau}} \tag{4.26}$$

Proof 14 (Proof of 4.2.3) *By the product rule of differentiation:*

$$\tau\frac{d}{dt}(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t)) = \tau\frac{d}{dt}(\boldsymbol{W}_2(t))\boldsymbol{W}_1(t) + \boldsymbol{W}_2(t)\tau\frac{d}{dt}(\boldsymbol{W}_1(t)) \tag{4.27}$$

We can substitute the expressions for gradient flow for $\boldsymbol{W}_2(t), \boldsymbol{W}_1(t)$ from ([36]):

$$\begin{aligned}\tau\frac{d}{dt}(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t)) &= (\tilde{\mathbf{\Sigma}}^{yx} - \boldsymbol{W}_2(t)\boldsymbol{W}_1(t))(\boldsymbol{W}_1(t))^T\boldsymbol{W}_1(t) \\ &\quad + \boldsymbol{W}_2(t)(\boldsymbol{W}_2(t))^T(\tilde{\mathbf{\Sigma}}^{yx} - \boldsymbol{W}_2(t)\boldsymbol{W}_1(t))\end{aligned} \tag{4.28}$$

We have previously shown that as $\lambda \to \infty$:

$$\begin{aligned}&\boldsymbol{W}_1(t)^T\boldsymbol{W}_1(t) \in O\left(\frac{1}{\lambda}\right) \to 0, \\ &\boldsymbol{W}_2(t)\boldsymbol{W}_2(t)^T \to |\lambda|\mathbb{I}\end{aligned} \tag{4.29}$$

In addition, as $\lambda \to -\infty$:

$$\begin{aligned}&\boldsymbol{W}_2(t)\boldsymbol{W}_2(t)^T \in O\left(\frac{1}{\lambda}\right) \to 0, \\ &\boldsymbol{W}_1(t)^T\boldsymbol{W}_1(t) \to |\lambda|\mathbb{I}\end{aligned} \tag{4.30}$$

Substituting these results into the previous equation:

$$\lim_{|\lambda|\to\infty} \tau\frac{d}{dt}(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t)) = |\lambda|(\tilde{\mathbf{\Sigma}}^{yx} - \boldsymbol{W}_2(t)\boldsymbol{W}_1(t)) \tag{4.31}$$

This differential equation is equivalent to the gradient flow for the weights of a shallow network, and has a known solution:

$$\boldsymbol{W}_2(t)\boldsymbol{W}_1(t) = \tilde{\mathbf{\Sigma}}^{yx} + (\boldsymbol{W}_2(0)\boldsymbol{W}_1(0) - \tilde{\mathbf{\Sigma}}^{yx})e^{-|\lambda|\frac{t}{\tau}} \tag{4.32}$$

As required.

As in the aligned case, we have shown that or large values of $|\lambda|$ the dynamics of the output $W_2(t)W_1(t)$ of a λ-Balanced Deep Network are equivalent to the dynamics of the output $W(t)$ of a shallow network, compressed horizontally by a factor of λ.

In addition, we have also shown that for large $\lambda \to \infty$ the expression $W_1(t)^T W_1(t)$ corresponds to a scaled version of the internal representation of the shallow network $W(t)W(t)^T$ while $W_2(t)W_2(t)^T$ remains fixed. As $\lambda \to \infty$ the expression $W_2(t)W_2(t)^T$ corresponds to a scaled version of the internal representation of the shallow network $W(t)^T W(t)$ while $W_1(t)^T W_1(t)$ remains fixed.

However, in the previous section we have proved that a λ-Balanced ($|\lambda|$ large) deep network will learn in the Lazy regime. This is an insightful example of when two networks can have similar representations, similar dynamics but still learn in completely different regimes. Crucially, the Deep Network contains two representations, only one of which is task specific. Intuitively, this task specific representation tends to 0 as $\lambda \to \infty$, so in the limit this representation will not be task specific but instead consist of only 0s and not evolve through training.

4.3 Illustrative Example

To showcase some for the results proven so far, we examine the same semantic learning task as in ([5]) where a set of living organisms must be linked to their positions within a hierarchical framework. Below is a description of the task which is paraphrased from this paper:

The representational similarity of the task's input ($\boldsymbol{\tilde{V}\tilde{S}\tilde{V}^T}$) exposes its underlying structure. For instance, the representations of two fish are most alike, somewhat similar to birds, and least similar to plants. Similarly, the representational similarity of the task's target values ($\boldsymbol{\tilde{U}\tilde{S}\tilde{U}^T}$) highlights the main categories into which items are organized.

Consequently, one can deduce that a fish is an animal and a plant is not a bird. Reflecting these structural relationships in internal representations enables the rich regime to generalize in ways that the Lazy regime cannot. Importantly, $QQ^T(t)$ captures the temporal dynamics of the weights' representational similarity, making it possible to analyze whether a network adopts a Rich or Lazy solution.

It is important to note that this task is a **regression task** in which the network learns the correlation structure in the ecosystem. It is inspired by ([36]) where the authors show that both linear networks and more complex networks experience **hierarchical differentiation**. They first learn the difference between a plant and an animal, and progressively learn finer structures.

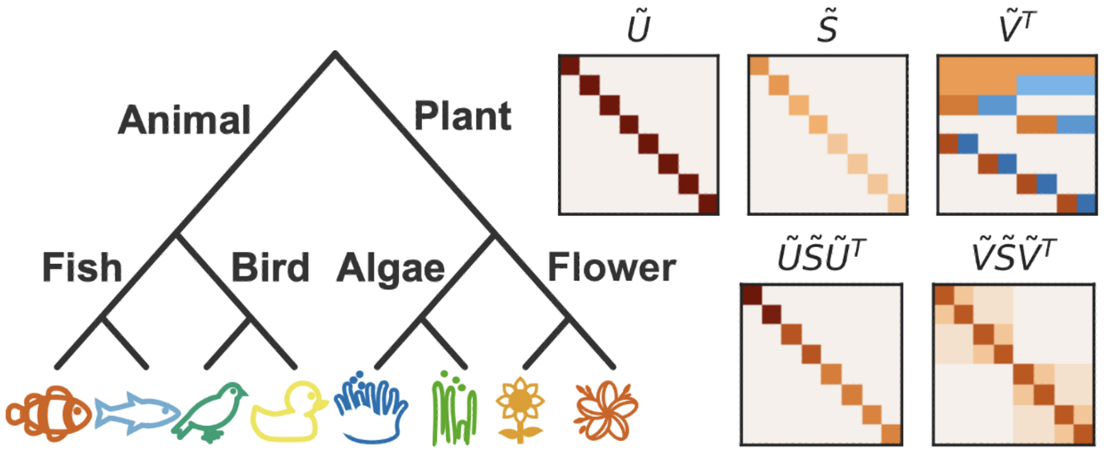


Figure 4.1: Semantic Learning Task (Figure from [5])

(**Figure 4.1**) shows the $\boldsymbol{QQ^T}(t)$ matrices at convergence for different weight initialisations. One can see that the network function in all of these cases is identical (lower left quadrant in each heat-map) but both the input and the output network representations of the Networks vary with initialisation (top left and bottom right quadrants of each heat-map).

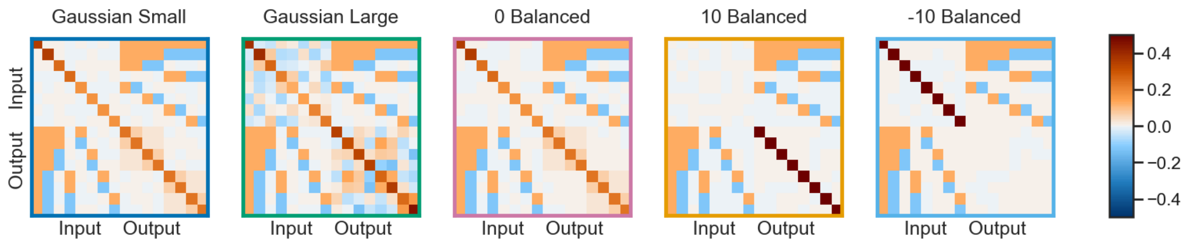


Figure 4.2: $\mathbf{QQ}^{\mathrm{T}}(t)$ matrices at convergence for Gaussian Small, Gaussian Large, 0-Balanced, 10-Balanced, and -10-Balanced initial weights. The figure shows how different initial weights affect the network's input and output representations and their alignment with task structure.

(**Figure 4.2**) shows the $\boldsymbol{QQ^T}(t)$ matrices at convergence for Gaussian Small, Gaussian Large, 0-Balanced, 10-Balanced and -10-Balanced initial weights. We can observe (also shown in [5]) that when training with Gaussian Small or 0 Balanced initial weights the weights' input and output network representations and the task's structure are identical at convergence. Intuitively this is because small weights are approximately 0-Balanced so they have similar behavior ([36]) In contrast when training with Gaussian Large weights the network has converged to a Lazy solution (the input and output network representations are not task specific).

The cases of λ-Balanced weights for $\lambda = 10, -10$ have a different behavior. The singular values of the task are sufficiently small for us to observe limit properties for these values of λ. We can see that in the case for $\lambda = 10$ the output network representation resembles $|\lambda|\mathbb{I}$, while in the case $\lambda = -10$ the input network representation resembles $|\lambda|\mathbb{I}$.

For the case $\lambda = 10$ the input network representation is task specific as shown earlier in the text ($\frac{1}{\lambda}\tilde{V}\tilde{S}\tilde{V}^T$) but the fact that it is scaled by $|\lambda|$ makes all of the values very close to 0. An analogous case occurs for $\lambda = -10$, where the output network representation is task specific but very close to 0 due to its scaling by λ.

Crucially, **only one of the two** network representations is a task specific representation of the data, and as $|\lambda| \to \infty$ none of the network representations will have be task specific. We will then have a Lazy solution to the task.

Hence, **we can think about the parameter λ as a way to determine the richness of the solution** by controlling how task the network's input and output network representations evolve and what value they converge to. We also wish to understand how λ affects loss dynamics and speed of convergence in this setting.

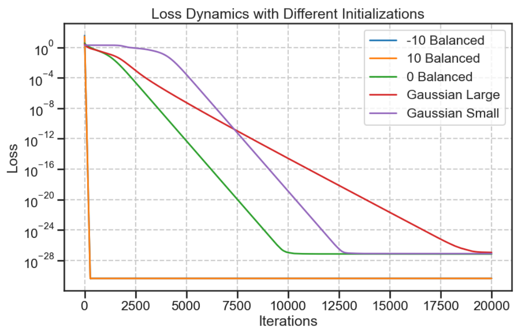


Figure 4.3: Log loss trajectories for different initializations in the semantic learning task. The figure highlights that higher values of $|\lambda|$ lead to faster convergence and that for high $|\lambda|$ values, the log loss trajectories follow a linear pattern until convergence.

(**Figure 4.3**) shows a graph of the Log Loss trajectories for different initialisations. Two insights can be gained from this graph:

1. Higher values of $|\lambda|$ lead to faster convergence of the Loss function. This agrees with the theory of rate of learning governed by the quantity $r = \frac{|\lambda|}{\tau}$ for large $|\lambda|$.
2. For high values of $|\lambda|$ the graph of the log Loss trajectories is a line with constant gradient until convergence. This confirms the theory that for large $|\lambda|$ the network follows exponential dynamics.

To further illustrate the relation between λ and speed of convergence, we plot the number of training iterations for a λ-Balanced network to converge when training in the semantic learning task. We are able to fit an exponential curve through the data for high values of λ ($|\lambda| > 5$), confirming again that the output trajectories λ-Balanced unaligned case are exponential for large $|\lambda|$ and qualitatively similar to the trajectories of a shallow network. For small values of λ the behavior of the network is not exponential but more complex, so the fitted exponential curve shown does not approximate these values correctly (as expected).

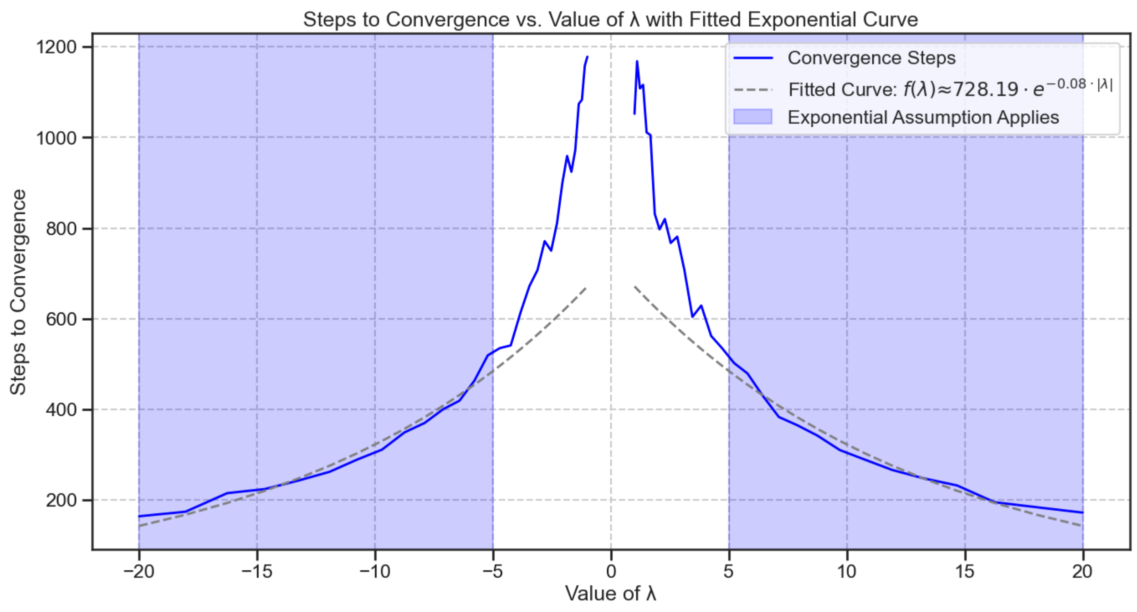


Figure 4.4: Graph showing the number of training steps to convergence for varying λ values in the semantic learning task. The figure illustrates how increasing λreduces the number of iterations needed for convergence, fitting an exponential curve for high λ values.

(**Figure 4.4**) Shows gives some intuition about another interpretation of the λ coefficient: **increasing λ coefficient between the weights of two adjacent layers increases their rate of convergence during training.**

We wish to show how λ affects the degree to which network representations are specific to the task. Previous results ([5]) have shown that when $\lambda = 0$ both network representations will be task specific. We have previously shown that in the limit $|\lambda| \to \infty$ one of the two representations will be task specific while the other one will not. Also, from (4.1.2) we can reason that if λ is close to the singular values of the task, neither of the representations will depict the task as clearly as for $\lambda \to 0, \pm\infty$.

To visualise this relation, we created a measure of how task specific a representation is. Intuitively, the input representation will be task specific if it is a simple relation of $\tilde{\boldsymbol{V}}, \tilde{\boldsymbol{S}}$. Similarly, the output representation will be task specific if it can be expressed simply in terms of $\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{S}}$. In this graph we are not interested in the dampening factor λ that affects λ-Balanced representations. Consequently, we use the cosine similarity metric to assign a Task Specific score to representations. The Task Specific Score TSC of input and output representations is defined as:

$$TSC(\boldsymbol{W}_1^T\boldsymbol{W}_1) = max(f(\boldsymbol{W}_1^T\boldsymbol{W}_1, \tilde{\boldsymbol{V}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T), f(\boldsymbol{W}_1^T\boldsymbol{W}_1, \tilde{\boldsymbol{V}}\tilde{\boldsymbol{S}}^2\tilde{\boldsymbol{V}}^T)) \tag{4.33}$$

$$TSC(\boldsymbol{W}_2\boldsymbol{W}_2^T) = max(f(\boldsymbol{W}_2\boldsymbol{W}_2^T, \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{U}}^T), f(\boldsymbol{W}_2^T\boldsymbol{W}_2, \tilde{\boldsymbol{U}}\tilde{\boldsymbol{S}}^2\tilde{\boldsymbol{U}}^T)) \tag{4.34}$$

Where $f(\boldsymbol{A}, \boldsymbol{B})$ is defined as the cosine similarity between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$

(**Figure 4.5** below) is a graph plotting a measure of the Task Specific Score of each network representation is with varying λ. This graph fits our mathematical theory: both curves peak at $\lambda = 0$. As $\lambda \to \infty$ the Task Specific Score of the input representation tends to its maximum while the Task Specific Score of the output representation tends to its minimum. On the contrary, as $\lambda \to -\infty$ the Task Specific Score of the input representation tends to its minimum while the Task Specific Score of the output representation tends to its maximum.

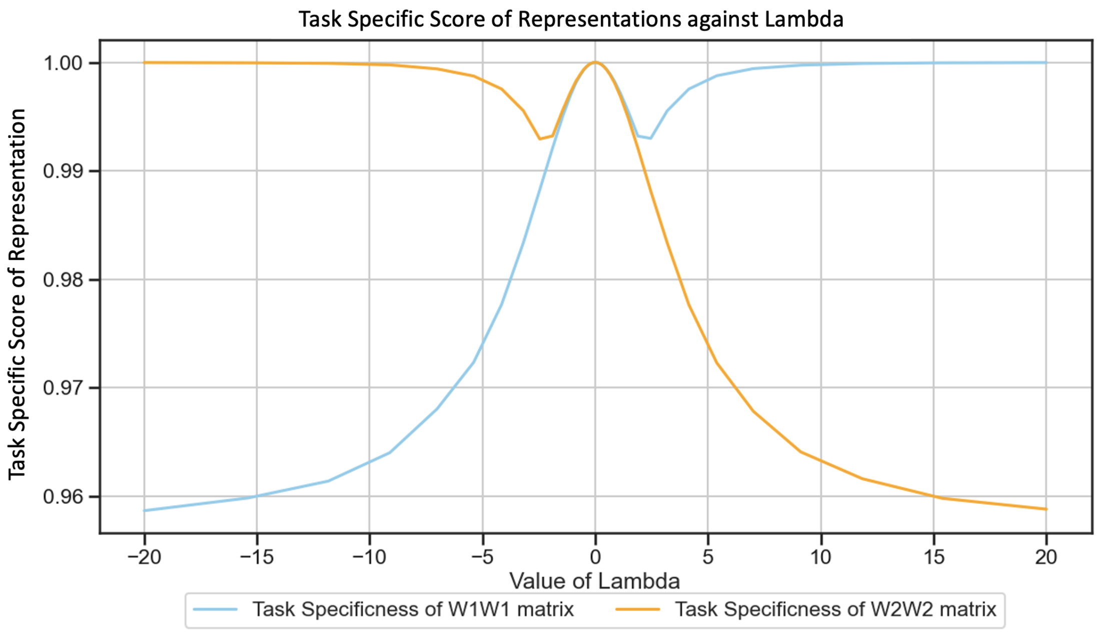


Figure 4.5: Task Specific Score of Network Representations for Varying λ

Examining this semantic learning task is useful for building intuition on the role of λ in λ-Balanced Networks: by increasing $|\lambda|$ we will make the network converge at an exponentially faster rate with the cost of our learning regime and representations becoming lazier. In addition, the sign of λ determines which of the network representations will cease to be a task specific representation. For large $|\lambda|$, $\lambda > 0$ means the output network representation will be less task specific, while $\lambda < 0$ will mean that the input network representation will be less task specific for large λ.

4.4 Continual Learning

Continual Learning and Catastrophic Forgetting have long been an important topic in the field of Artificial Intelligence ([20]). Continual Learning focuses on enabling models to learn continuously form an ongoing stream of data. The primary goal is to integrate new knowledge seamlessly while preserving previously acquired information. The opposite case, when the model significantly losses previously acquired knowledge upon learning new information, is refereed to as Catastrophic Forgetting.

Deriving analytical solutions for dynamics in a continual learning setting for linear networks can provide insights into how to best initialise models in order to diminish forgetting, as well as aid the search of metrics that aid in determining optimal stopping times.

The paper ([5]) shows that since the Balanced Property is conserved through training the analytical solution will remain being correct in a continual learning setting for 0 Balanced Networks. This is also true for λ-Balanced Networks. Below is an example of the empirical and analytical dynamics of a λ-Balanced network in a continual learning setting.

Since the Balanced Property is preserved through training in linear networks, once a λ-Balanced network learns a task its trained weights will also be λ-Balanced. Hence its starting weights for the second task will be λ-Balanced and so will its trained weights. We can see by induction how the network will remain λ-Balanced in a continual learning setting. Since the λ-Balanced assumption always holds, the analytical solutions developed in (2.2.1) can be used for the whole learning process:

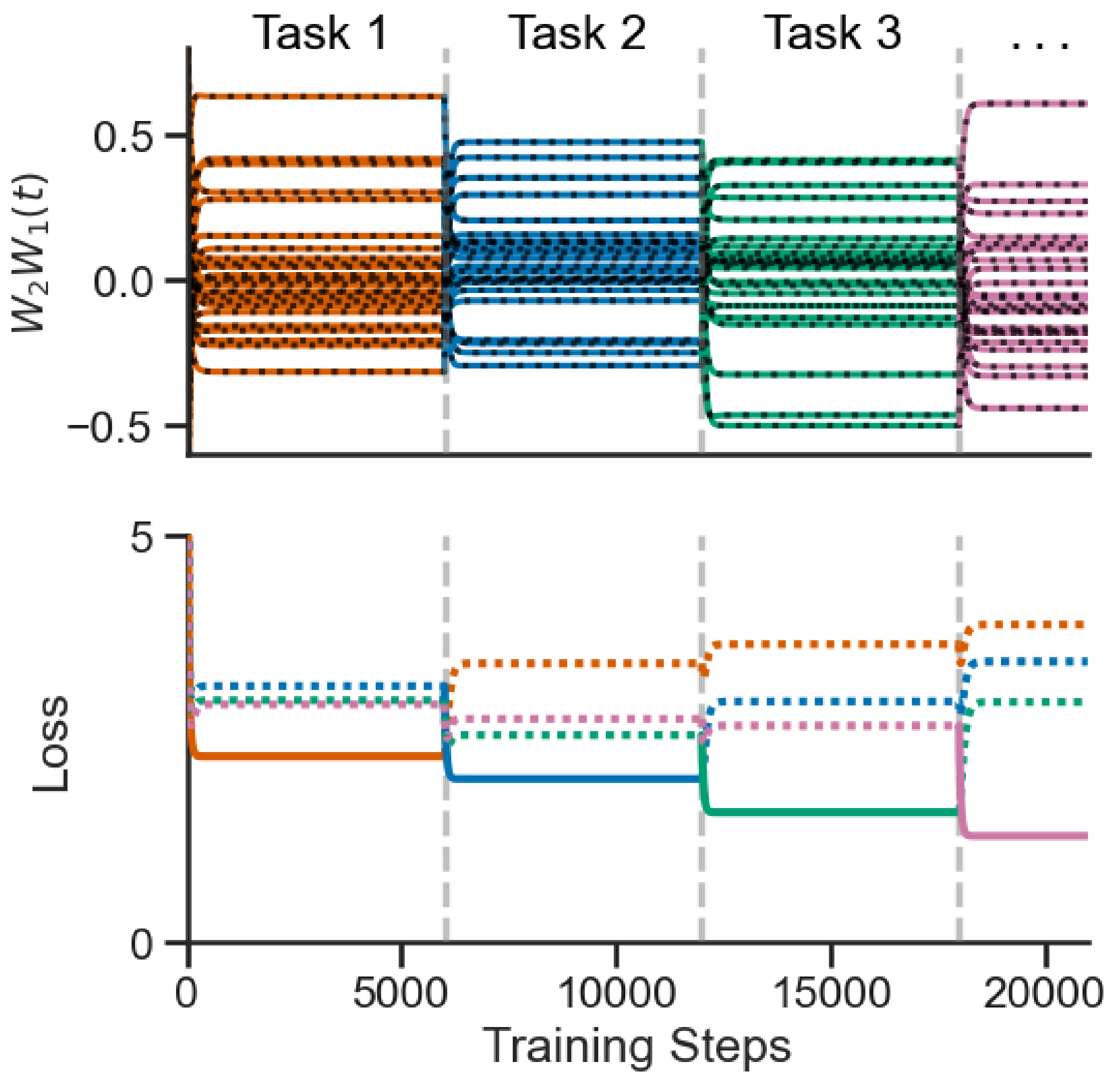


Figure 4.6: Graph showing analytical solutions in a continual learning task. The figure demonstrates how the rate of forgetting and loss are expressed in terms of input-output correlations and network weights, providing insights into optimal stopping strategies.

As described in the background of this report (**Continual Learning** section) it is possible to express the loss function of a linear network in terms of input output correlations and the network's weights. Since the Forgetting metric is defined as the difference of two loss functions, it is can also be defined in terms of input output correlations and the network's weights, thus making these formulae more interpretable.

A possible next step in this approach is to derive analytical expressions for the rate of loss and rate of forgetting of a network. If we can express these in terms of the the input output correlations of different tasks and the network's weights, we will be able to find minimum in the forgetting function and devise optimal stopping strategies based on analytical results. We derive an expression for the rate of forgetting in the next theorem.

Theorem 4.4.1 ***[Expression for Rate of Forgetting in a Network with λ-Balanced initial weights]***
Consider a λ-Balanced linear network. Let $\boldsymbol{\mathcal{L}}_{i,k}(t)$ be the loss on task i when training on task k, and $\boldsymbol{\mathcal{F}}_{i,j,k}(t)$ be the Forgetting on task i when training on task k having just trained on task j.

1. *The rate of **L**oss and the rate of **F**orgetting in the network are the same for all t.*

$$\frac{d}{dt}\boldsymbol{\mathcal{L}}_{i,k}(t) = \frac{d}{dt}\boldsymbol{\mathcal{F}}_{i,j,k}(t) \tag{4.35}$$

2. *We can express the rate of **L**oss as:*

$$\frac{d}{dt}\boldsymbol{\mathcal{L}}_i(t) = \mathrm{Tr}\left(\frac{d}{dt}(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))\,(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))^T - \tilde{\boldsymbol{\Sigma}}^{\boldsymbol{yx}T}\right) \tag{4.36}$$

Proof 15 (Proof of 4.4.1) *Proof of (1):*

$$\boldsymbol{\mathcal{F}}_{i,j,k}(t) = \boldsymbol{\mathcal{L}}_i(\boldsymbol{W}_2\boldsymbol{W}_1(t)) - \boldsymbol{\mathcal{L}}_i(\boldsymbol{T}_j) \tag{4.37}$$

$$= \boldsymbol{\mathcal{L}}_{i,k}(t) - \boldsymbol{\mathcal{L}}_i(\boldsymbol{T}_j) \tag{4.38}$$

$$\frac{d}{dt}\boldsymbol{\mathcal{F}}_{i,j,k}(t) = \frac{d}{dt}\boldsymbol{\mathcal{L}}_{i,k}(t) \tag{4.39}$$

As required.
Proof of (2):
From [5] we know that

$$\boldsymbol{\mathcal{L}}_i(t) = \frac{1}{2}\,\mathrm{Tr}[(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))^T] - \mathrm{Tr}[\boldsymbol{W}_2(t)\boldsymbol{W}_1(t)\tilde{\boldsymbol{\Sigma}}^{\boldsymbol{yx}T}] + \frac{1}{P}\,\mathrm{Tr}[\boldsymbol{Y}\boldsymbol{Y}^T] \tag{4.40}$$

Hence

$$\frac{d}{dt}\boldsymbol{\mathcal{L}}_i(t) = \mathrm{Tr}\left[\frac{d}{dt}(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))^T\right] - \mathrm{Tr}\left[\frac{d}{dt}(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))\tilde{\boldsymbol{\Sigma}}^{\boldsymbol{yx}T}\right] \tag{4.41}$$

$$\frac{d}{dt}\boldsymbol{\mathcal{L}}_i(t) = \mathrm{Tr}\left[\frac{d}{dt}(\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))\left((\boldsymbol{W}_2(t)\boldsymbol{W}_1(t))^T - \tilde{\boldsymbol{\Sigma}}^{\boldsymbol{yx}T}\right)\right] \tag{4.42}$$

As required.

This formulation or the rate of Forgetting is useful since we already posses analytical equations for the rate of change of the weights as well as the weights o the network. We can substitute these expressions in and observe hoe λ influences the rate of loss for example.Below is a graph showing the analytical expression for rate of loss matches the empirical. Note that the rate of loss converges at 0. This is expected since the linear network is known to converge to the global minimum during full batch gradient descent. ([36])

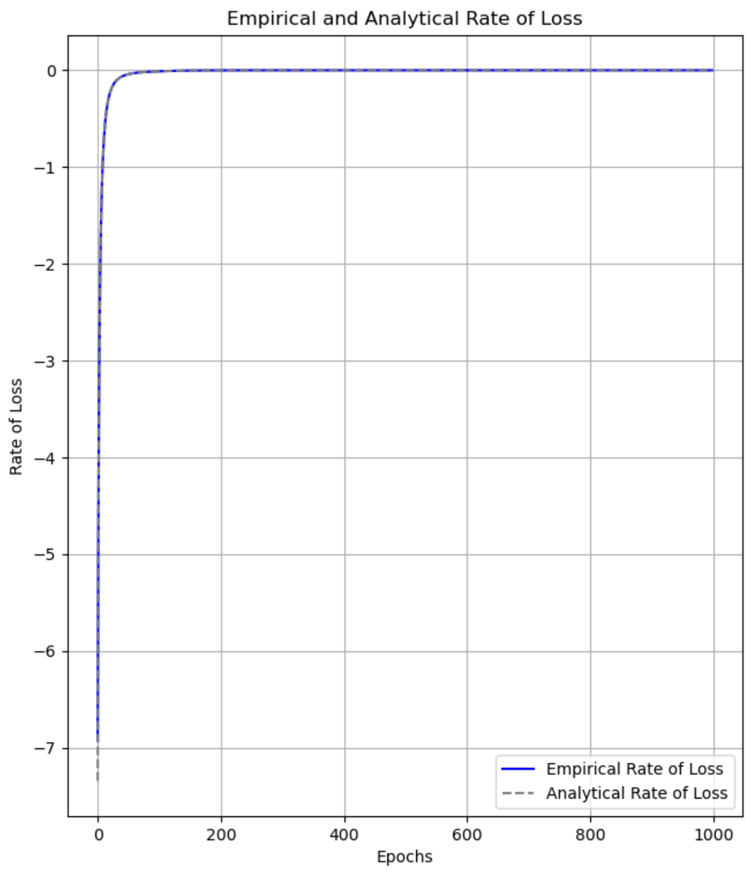


Figure 4.7: Analytical and Empirical Rate of Loss Match Exactly for $\lambda = 5$

4.5 Extensions to the Non-Linear Case

The solutions derived in this text apply only to Linear Networks. Although linear networks' dynamics qualitatively resemble non-linear networks' dynamics ([38]), linear networks ultimately perform linear regression only. Non linear networks possess a much higher amount of expressive power, as they are able to capture complex nonlinear relationships in the data ([4]). Therefore, we are interested in ways to generalise these results to the Non Linear case.

Theorem 4.5.1 ***[Optimal $\boldsymbol{\lambda}^*$ for given weights]***
Let $\boldsymbol{W}_2, \boldsymbol{W}_1$ be the weights of a network. The optimization problem:

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}} \left\| (\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T) - \boldsymbol{\lambda}\boldsymbol{I} \right\|_F^2$$

has the solution $\boldsymbol{\lambda}^* = \mathrm{Tr}(\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T)$.

Proof 16 (Proof of 4.5.1) *Since $\boldsymbol{\lambda}\boldsymbol{I}$ is a diagonal matrix, the optimization problem is equivalent to minimizing $f(\boldsymbol{\lambda})$ with*

$$f(\boldsymbol{\lambda}) = \left\| \mathrm{diag}(\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T) - \boldsymbol{\lambda}\boldsymbol{I} \right\|_F^2 \tag{4.43}$$

Let $\mathrm{diag}(\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T) = \mathrm{diag}(a_1, a_2, \ldots, a_n)$. Then

$$f(\boldsymbol{\lambda}) = \sum_{i=1}^{n}(a_i - \boldsymbol{\lambda})^2 \tag{4.44}$$

We can calculate the derivative of $f(\boldsymbol{\lambda})$ and find when it is equal to 0:

$$f'(\boldsymbol{\lambda}) = 2\sum_{i=1}^{n}(a_i - \boldsymbol{\lambda}) \tag{4.45}$$

$$f'(\boldsymbol{\lambda}^*) = 0 \Leftrightarrow \boldsymbol{\lambda}^* = \frac{1}{n}\sum_{i=1}^{n}(a_i) = \frac{1}{n}\operatorname{Tr}(\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T) \tag{4.46}$$

This concludes the proof.

It has been shown in ([31], [24]) that the quantity $Diag(\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T)$ is invariant through training in a network composed of solely ReLU functions. Hence the quantity $Tr(\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T)$ will also remain invariant. In addition, a network with both ReLU and Linear activations will also have $Tr(\boldsymbol{W}_2^T\boldsymbol{W}_2 - \boldsymbol{W}_1\boldsymbol{W}_1^T)$ as an invariant quantity

We know that the quantity λ^* such that $\lambda^*\mathbb{I}$ is closest to $W_2^TW_2 - W_1W_1^T$ will remain constant through training on a ReLU network. However we are also interested in how this distance changes through training and if there is a way to bound this distance. If there happens to be one, for large $|\lambda^*|$ this distance will become negligible relative to $||\lambda^*\mathbb{I}||$ and hence we will be able to assume that the Balanced property holds through training.

To better understand λ^* and the distance of the Balanced Computation to $\lambda^*\mathbb{I}$, we conducted some preliminary investigations. An experiment was conducted using two random datasets in a continual learning framework. Initially, Balanced weights λ were set up. A ReLU network and a Tanh network were trained sequentially on the two tasks for 2000 iterations or until convergence. After each task reached convergence, λ^* and the distance to $\lambda^*\mathbb{I}$ were calculated. This procedure was repeated 50 times, with results averaged and the variance shown as an error bound. The experiment was conducted for three different values of λ: 10, 0.1, and -10.

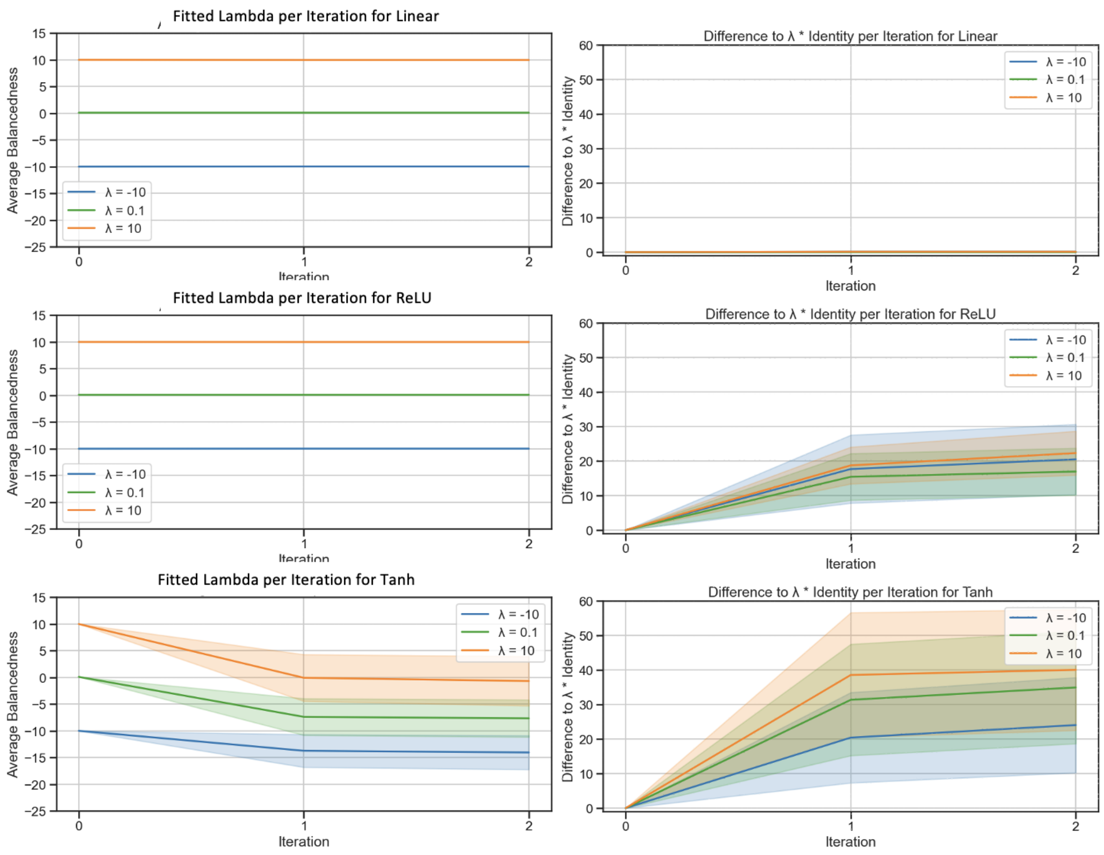


Figure 4.8: Deviation from Balanced Property in ReLU and Tanh Networks

(**Figure 4.8**) both confirms existing theory and suggest possible directions of further research. First of all, the fitted Balanced Coefficient is preserved through training for bot ReLU and Linear networks. In addition, λ^* matches exactly the Balanced Computation of the weights for a linear network. For Tanh network the fitted Balanced coefficient decreases through training. For both ReLU and Tanh networks, the difference between the fitted Balanced coefficient and the Balanced Computation increases after each subsequent task. However, it increases at a lower rate for ReLU networks. It would be interesting to investigate whether the fitted Balanced coefficient converges to some value for Tanh networks or whether it keeps monotonically decreasing without convergence. In addition, a possible stream of investigation would be bounding the distance from the fitted λ^*. If this bound is independent λ^*, we could possible utilise the Balanced Weights assumption if the initial λ is large and if we wish to understand network behavior for limited training steps.

Chapter 5

Ethical Considerations

5.1 Ethical Considerations

This research project is exempt from ethical approval for the following reasons: the analysis does not involve human participants, the data is randomly generated so there are no privacy concerns. Additionally, the analysis does not involve animals. The project is very abstract and does not involve developing countries.

One possible environmental harm is consuming a lot of energy when training the models, but the neural networks we are training are small so will not require much energy to train. Therefore the project does not involve elements that may cause harm to the environment, plants and animals.

This project involves the explainability of neural networks, so I do not foresee any connection to military applications. I also do not see any potential for malevolent abuse from the findings of the project.

The software used in the project is open source, ensuring compliance with legal requirements.

Chapter 6

Conclusion

6.1 Limitations of this Work

While this research advances the understanding of learning dynamics in linear neural networks, it has several limitations. Firstly, it could be argued that the experimental setup is too simplistic. The study primarily focuses on linear neural networks, which, although insightful, do not capture the full complexity of non-liner models commonly used in practice. Another limitation is the assumption of gradient flow. Many machine learning models used in practice utilise adaptive learning rates ([4]). The gradient flow assumption does not suffice to understand the behavior of these models. Furthermore, the study only considers linear networks of at most three layers. to better understand the impact of network structure to learning regimes, we must generalise this theory to deeper networks. In addition, the applications to continual learning are yet to be extended. Although the author believes this is a promising direction, no significant results have been drawn regarding optimal stopping strategies to minimise forgetting.

6.2 Summary of Results and Future Directions

In summary, this research has made significant contributions to the understanding of learning dynamics in neural networks, particularly in the context of Rich and Lazy learning regimes. We derived exact solutions for both aligned and unaligned λ-balanced networks, which are interpretable and numerically stable. Notably, we demonstrated the equivalence of deep λ-balanced network dynamics to shallow network dynamics for large $|\lambda|$. Our study thoroughly analyzed feature learning mechanisms, highlighting the conditions under which a network transitions between Rich and Lazy learning regimes. Specifically, we identified that increasing $|\lambda|$ moves the network into the Lazy learning regime. Additionally, we explored applications of these analytical dynamics in continual learning, deriving an exact expression for the forgetting rate of a λ-balanced linear network.

Looking ahead, several future directions are proposed to build upon this work. One primary direction is extending the derived solutions to non-linear neural networks, which would bridge the gap between theoretical analysis and practical deep learning models. Given that the fitted λ^* is a conserved quantity for ReLU networks, they present a straightforward extension of this work. We propose deriving analytical solutions for the learning dynamics of ReLU networks, which would hold with some degree of error decreasing with λ^*. Another promising research avenue involves developing analytical solutions for deeper networks. While there has been progress for aligned networks ([36]), an analytical solution for deep unaligned networks with more than one output neuron remains elusive (see [24] for the case of one output neuron).

Furthermore, developing robust weight initialization methods that ensure λ-balanced weights across various architectures and datasets would significantly enhance the practical applicability of our theoretical findings. An empirical study could be conducted to initialize a deep linear network with λ-balanced weights, using different λ values for each weight pair. The effect of these varying λ values on the convergence rate of each weight pair during gradient descent could then be measured.

Another promising research direction involves investigating whether similar learning dynamics and representations to those outlined in this report occur in biological brains. This would entail examining if neural activity patterns in the brain, during various learning tasks, exhibit behaviors analogous to the Rich and Lazy learning regimes observed in artificial neural networks. Understanding these similarities would significantly advance the field of computational neuroscience, as it would allow for the testing of theories about biological networks using artificial networks, providing access to precise dynamics with relatively low experimental cost. This cross-disciplinary approach could reveal fundamental principles of learning that are shared between artificial and biological systems, potentially leading to improved neural network designs and deeper insights into brain function.

By addressing these limitations and pursuing the suggested future directions, we can further characterize Rich and Lazy learning and contribute to the development of the theory of feature learning in neural networks. This work ultimately aims to bridge the gap between theoretical insights and practical applications in neural network learning dynamics, enhancing our understanding and capabilities in this field.

Bibliography

[1] Unireps: Unifying representations in neural models, 09 2023.

[2] Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect, 10 2021.

[3] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning, 07 2018.

[4] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2:1–127, 2009.

[5] Lukas Braun, Clémentine Dominé, James Fitzgerald, and Andrew Saxe. Exact learning dynamics of deep linear networks with prior knowledge. *Advances in Neural Information Processing Systems*, 35:6615–6629, 12 2022.

[6] Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Unbalanced optimal transport: Dynamic and kantorovich formulations. *Journal of Functional Analysis*, 274:3090–3123, 06 2018.

[7] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability, 10 2020.

[8] Matthew Farrell, Stefano Recanatesi, and Eric Shea[U+2010]Brown. From lazy to rich to exclusive task representations in neural networks and neural codes. *Current Opinion in Neurobiology*, 83:102780–102780, 12 2023.

[9] Timo Flesch, Jan Balaguer, Ronald Dekker, Hamed Nili, and Christopher Summerfield. Comparing continual task learning in minds and machines. *Proceedings of the National Academy of Sciences*, 115:E10313–E10322, 10 2018.

[10] Timo Flesch, Keno Juechems, Tsvetomira Dumbalska, Andrew Saxe, and Christopher Summerfield. Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, 110:4212–4219, 12 2022.

[11] Timo Flesch, Andrew Saxe, and Christopher Summerfield. Continual task learning in natural and artificial agents. 10 2022.

[12] Kenji Fukumizu. Statistical active learning in multilayer perceptrons. volume 11, pages 17–26, 2000.

[13] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020:113301, 11 2020.

[14] Samuel J Gershman. What have we learned about artificial intelligence from studying the brain? *Biological cybernetics*, 118, 02 2024.

[15] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. page 249–256. proceedings.mlr.press, JMLR Workshop and Conference Proceedings, 03 2010.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 12 2015.

[17] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. volume 31. Neural Information Processing Systems, Curran Associates, Inc., 2018.

[18] Georg Johann. Singular value decomposition, 12 2020.

[19] Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS Computational Biology*, 10:e1003915, 11 2014.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwińska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[21] Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2, 2008.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84–90, 05 2012.

[23] Daniel Kunin, Allan Raventós, Clémentine Dominé, Feng Chen, David Klindt, Andrew Saxe, and Surya Ganguli. Get rich quick: exact solutions reveal how unbalanced initializations promote rapid feature learning, 06 2024.

[24] Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel L. K. Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics, 03 2021.

[25] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 11 2016.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Xuhong Li, Haoyi Xiong, Xingjian Li, Xuanyu Wu, Xiao Zhang, Ji Liu, Jiang Bian, and Dejing Dou. Interpretable deep learning: interpretation, interpretability, trustworthiness, and beyond. *Knowledge and Information Systems*, 64:3197–3234, 09 2022.

[28] Guan-Horng Liu and Evangelos A. Theodorou. Deep learning theory review: An optimal control and dynamical systems perspective, 09 2019.

[29] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hongping He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Qiang Niu, Dingang Shen, Tianming Liu, and Bao Ge. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, 1, 04 2023.

[30] Adam H. Marblestone, Greg Wayne, and Konrad P. Kording. Toward an integration of deep learning and neuroscience, 09 2016.

[31] Sibylle Marcotte, Remi Gribonval, and Gabriel Peyré. Abide by the law and follow the flow: conservation laws for gradient flows, 12 2023.

[32] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[33] Shahryar Rahnamayan and Gary Wang. Toward effective initialization for large-scale search spaces. 01 2009.

[34] Blake A. Richards, Timothy P. Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J. Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W. Lindsay, Kenneth D. Miller, Richard Naud, Christopher C. Pack, Panayiota Poirazi, Pieter Roelfsema, João Sacramento, Andrew Saxe, Benjamin Scellier, Anna C. Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P. Kording. A deep learning framework for neuroscience. *Nature Neuroscience*, 22:1761–1770, 10 2019.

[35] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 10 1986.

[36] Andrew Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *openreview.net*, 12 2013.

[37] Andrew Saxe, Stephanie Nelli, and Christopher Summerfield. If deep learning is the answer, then what is the question? *arXiv:2004.07580 [q-bio]*, 04 2020.

[38] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116:11537–11546, 05 2019.

[39] Martin Schrimpf, Jonas Kubilius, Ha Hong, Najib J. Majaj, Rishi Rajalingham, Elias B. Issa, Kohitij Kar, Pouya Bashivan, Jonathan Prescott-Roy, Franziska Geiger, Kailyn Schmidt, Daniel L. K. Yamins, and James J. DiCarlo. Brain-score: Which artificial neural network for object recognition is most brain-like? 09 2018.

[40] Salma Tarmoun, Guilherme Franca, Benjamin D. Haeffele, and Rene Vidal. Understanding the dynamics of gradient flow in overparameterized linear models. page 10153–10161, 07 2021.

[41] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle.

[42] Lilian Weng. Some math behind neural tangent kernel. *Lil'Log*, Sep 2022.

[43] Xiangxiang Xu and Lizhong Zheng. Neural feature learning in function space *. *Journal of Machine Learning Research*, 25:1–76, 2024.