THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Controlled Descent Training

Viktor Andersson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 855

Controlled Descent Training

Viktor Andersson

Technical Report No. 1111-111X
ISSN 3.1415-9265
This thesis has been prepared using LaTeX.

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden, April855

Abstract

In this work, a novel and model-based artificial neural network (ANN) training method is developed supported by optimal control theory. The method augments training labels in order to robustly guarantee training loss convergence and improve training convergence rate. Dynamic label augmentation is proposed within the framework of gradient descent training where the convergence of training loss is controlled. First, we capture the training behavior with the help of empirical Neural Tangent Kernels (NTK) and borrow tools from systems and control theory to analyze both the local and global training dynamics (e.g. stability, reachability). Second, we propose to dynamically alter the gradient descent training mechanism via fictitious labels as control inputs and an optimal state feedback policy. In this way, we enforce locally $\mathcal{H}_2$ optimal and convergent training behavior. The novel algorithm, *Controlled Descent Training* (CDT), guarantees local convergence. CDT unleashes new potentials in the analysis, interpretation, and design of ANN architectures. The applicability of the method is demonstrated on standard regression and classification problems.

Keywords: Label augmentation, gradient descent training, neural tangent kernel, optimal control, convergent learning, label selection.

List of Publications

This thesis is based on the following publications:

[A] **Viktor Andersson**, Balázs Varga, Vincent Szolnoky, Andreas Syrén, Rebecka Jörnsten, Balázs Kulcsár, "Controlled descent training". Submitted to the *International Journal of Robust and Nonlinear Control*, Mar. 2023.

Acknowledgments

First of all I would like to express my sincerest gratitude to my supervisors Balázs Kulscár and Rebecka Jörnsten. Your experience, knowledge, and guidance have been of utmost importance for this research which would have been impossible without your unwavering support. I would also like to thank Balázs Varga who has greatly assisted me during my research along with Vincent Szolnoky. Finally, I would like to thank Andreas Syrén who was the main catalyst for this research idea.

I gratefully acknowledge the support of the project OCTON 1, 2 at Chalmers University of Technology. This work was supported in part by the Transport Area of Advance, at Chalmers University of Technology. Moreover, the project was carried out in collaboration with and is supported by Centiro Solutions, a logistics software company based in Sweden.

Acronyms

ODE:	Ordinary Difference Equation
ANN:	Artificial Neural Network
CDT:	Controlled Descent Training
LQR:	Linear Quadratric Control (-ler)
NTK:	Neural tangent kernel

Contents

Abstract i

List of Papers iii

Acknowledgements v

Acronyms v

1 Introduction and Motivation 1

1.1 Machine Learning and Deep Learning 1

Fully-connected neural networks 2

Convolutional neural networks 2

Recurrent neural networks 3

Generative adversarial networks 3

Transformer 4

1.2 Reinforcement Learning 5

Online 5

Offline 5

Deep Reinforcement learning 5

1.3 Control Theory 6

Optimal control 6

ML in control theory . . . 7
1.4 NTK . . . 8
Indirect . . . 9
Direct . . . 9
1.5 Contributions: . . . 10
1.6 Thesis outline . . . 10
1.7 Ethics . . . 11
1.8 Notations . . . 11

2 Neural Network training dynamics **13**
2.1 Neural Tangent Kernel . . . 13
2.2 Local and global ANN training dynamics . . . 14
2.3 Controlled ANN training dynamics with label augmentation . . . 15
2.4 Analytic properties of discrete-time training dynamics . . . 16
Boundedness of the training dynamics . . . 16
Reachability . . . 18

3 Locally Optimal Control of ANN training dynamics **21**
3.1 Controlled Descent Training . . . 21
The CDT algorithm . . . 24

4 Numerical performance and benchmarking **25**
4.1 Regression . . . 26
4.2 Classification . . . 36

5 Summary of included papers **39**
5.1 Paper A . . . 39

6 Concluding Remarks and Future Work **41**
6.1 Concluding remarks . . . 41
Future work . . . 42

7 Appendecies **45**
7.1 Existence and uniqueness of solution . . . 45
7.2 Local training dynamics . . . 46
7.3 Lagrange error bounds for local training dynamics . . . 46
7.4 Loss Linearization . . . 47
7.5 Examples of equilibrium points . . . 48

7.6 Boundedness for common losses 49
7.7 Boundedness of the global dynamics 51
7.8 The DARE equation 52
7.9 Initialization of the ANN 52

References **55**

CHAPTER 1

Introduction and Motivation

1.1 Machine Learning and Deep Learning

In recent years, machine learning (ML) and deep learning methods have begun to significantly impact society. Deep learning and artificial neural network (ANN) models have made strides in solving complicated problems like image recognition [1] and highly advanced regression problems [2]. Moreover, generative deep learning ANN models like the popular transformer and generative adversarial networks (GANs) are able to generate text and images indistinguishable from human examples. The recent GPT-4 model, based on the transformer architecture is able to pass a wide array of exams ranging from the uniform bar exam for newly examined lawyers to advanced placement physics and statistics [3].

A neural network, inspired by the neurological pathways in a biological brain, is a network of interconnected nodes. Each node consists of a simple linear function and an often nonlinear activation function. The nodes form a network with a certain width (number of nodes per layer) and depth(number of layers). At their core ANNs are universal function approximators. With sufficient data and model complexity ANNs are able to approximate any com-

plex and highly non-linear input-output relationships.

In general, ANNs are trained to fit the data using smooth loss functions which capture the models adherence to the observed relationship. The loss function is backpropagated through the network and parameters updated with gradient descent-based methods [4] like mini-batch gradient descent. Mini-batch gradient descent divides the data into mini-batches stochastically [5], making the training process parallelizable and scalable with data size. Many different ANN architectures exist with varying use cases. In the following sections, a few particularly impactful architectures are discussed.

Fully-connected neural networks

Early iterations of ANNs, often referred to as dense or fully connected neural networks connected each node to all nodes in the previous layer. The dense or fully-connected architecture is often used for solving regression and tabular data problems like time-series forecasting [6]. Recent practical applications of dense ANNs include efficiency of water desalination estimation based on solar thermal energy [7] and estimating compressive strength of CO2 concrete [8]. However, dense ANNs need to learn local spatial relationships in the input implicitly with many examples. This makes solving image-based tasks possible but challenging.

Convolutional neural networks

ANNs main breakthrough came in 2012 with the convolutional neural network (CNN) architecture outperforming humans on benchmark image recognition tasks [9][10]. Rather than connecting all nodes directly, CNNs use convolution to derive information on nearby inputs (e.g. nearby pixels) for every input (e.g. pixel). Therefore, CNNs capture local relationships in the input explicitly, making them particularly apt at handling image-based tasks. Since their inception CNNs have been applied in many practical settings like radiology [11], face recognition [12] and recommendation systems [13]. However, since CNNs capture local relationships, long-sequenced inputs are challenging. To solve this the recurrent neural network was developed.

Recurrent neural networks

CNNs are good at dealing with local input relationships but struggle with sequential relationships like sentences or video. Recurrent neural network (RNNs) input nodes can create cyclic connections where the output of a node is used as input to subsequent input nodes. The RNN is therefore able to explicitly capture sequential relationships in the input.

RNNs have been used for time series forecasting like end-of-life battery forecasting [14] and mechanical failure diagnosis [15]. RNNs have also been used in natural language processing [16].

RNN architectures do however struggle with long-term memory and very long sequential inputs [17]. Moreover, they are prone to exhibit vanishing or exploding gradients [18] causing stagnant or divergent training respectively. Moreover, the RNN is not parallelizable when training due to the sequential input dependency. Therefore, training requires a lot of time and resources which may be wasted due to vanishing or exploding gradients.

These ANN architectures can not only be used for inference and prediction but for sample generation like the generative adversarial network and the transformer.

Generative adversarial networks

The generative adversarial networks (GAN) architecture is designed to generate novel samples plausibly drawn from a given dataset. The GAN consists of a generative and an adversarial discriminatory model, often neural networks. The two models work against each other. The generative model attempts to generate a novel sample (e.g an image) given some input (e.g text prompt) while the discriminator tries to classify the novel sample as either generated or part of the original dataset.

GANs are used for image segmentation [19], art and music generation [20], and medicinal image augmentation [21].

The adversarial nature of the GAN architecture makes the training unpredictable. The GAN training is prone to divergence and instability due to training rate imbalances between the generator and discriminator [22].

Transformer

The transformer architecture has been the focus of recent ANN research. The transformer is based on the self-attention mechanism which in short encodes the relative importance of each input signal. The transformer can therefore handle both local and sequential input relationships without suffering from the memory decay of RNNs. Furthermore, the transformer architecture training is parallelizable [23] allowing significantly larger models.

The transformer is the basis for well know NLP models like openAIs Generative Pretrained Transformer (GPT)-4 [3] or Googles Bidirectional Encoder Representation from Transformers (BERT) [24]. These models are state-of-the-art at solving complex NLP tasks like text generation, text summarization, and text translation [25] with high accuracy and human-like performance. GPT-4 recently passed advanced exams in multiple fields [3], scoring among the top 10% of students. Moreover, GPT-4 is the NLP model behind GPT-chat, an AI chatbot that has gotten significant media attention for its human-like text generation.

These models are however extremely large and resource intensive to train. GPT-3, the predecessor to GPT-4, and BERT have 175 billion and 360 million parameters respectively. These large-scale models have been criticized for the environmental impact as well as the economic cost of training [25]. As stated in [3] model-specific tuning like hyperparameter optimization is not feasible for such large models. Moreover, the transformer architecture has been observed to be prone to divergent training, yielding unusable models [23]. Guaranteeing convergence of these often large models could result in fewer resources wasted and more efficient training.

ANNs in all their forms have become one of the most widely used tools in ML [10]. They are however unpredictable both during training and in output. Training convergence rate and final network performance are highly dependent on hyperparameters like learning rate for all variations and architectures. Hyperparameter selection is generally only evaluated after training [26]. Hence, ANNs require multiple training repetitions in order to find non-divergent hyperparameters yielding acceptable model performance. The hyperparameter search leads to costly and resource-intensive ANN training and is unfeasible for larger models like GPT. Making ANN training more predictable, guaranteeing convergence, and limiting hyperparameter search spaces would reduce

wasted resources and environmental impact.

1.2 Reinforcement Learning

Another powerful tool of ML is reinforcement learning (RL).

Reinforcement learning (RL) is a method in ML based on Markov decision processes. Decisions are taken in sequence in accordance with an action policy and an environment. The outcome is observed and the policy evaluated. Many policy models exist, but a common modern approach is deep reinforcement learning using ANNs.

There are two approaches to training the policy model, offline and online.

Online

During online RL training, the action policy is evaluated and updated based on observations derived from interacting with the environment.

However, updating the policy model online comes with risks. Interacting with the environment directly may result in unpredictable policy evolution. Moreover, interacting with certain environments can be costly and dangerous (e.g safety-critical systems) [27]. Hence an offline approach to training can be less risky.

Offline

Offline RL relies on training on static datasets similar to traditional ML. Observations are pre-recorded and stored which is used to optimize the action policy. The policy is then deployed in the real environment.

The offline training approach can be safer as the environment is not affected directly. However, without online updates the model becomes less adaptive to changes in the environment. Moreover, the static dataset may not be rich enough to capture all possible interactions or actions [27].

Deep Reinforcement learning

Deep reinforcement learning (DRL) is an RL method where the policy model consists of an ANN. DRL has been used in RL applications requiring com-

plex inputs like computer vision [28] and health care [29]. Deep-Q networks (DQN) are a widespread approach to DRL. DQN has successfully solved complex tasks like playing atari video games [30] and more recently pathfinding for autonomous vehicles [31]. DQN has also been used for process control [32]. However, instability of DQN training has been observed due to the policy changing rapidly, especially in online settings [30].

Reinforcement learning both offline and online has its strengths and drawbacks. Ensuring stable DRL training along with more predictable online training schemes could allow for dynamic and adaptive modeling even in safety-critical applications. Control theory is a field dealing with modeling and control of such safety-critical applications. Drawing inspiration from control theory may assist in the predictability of DRL.

1.3 Control Theory

Control theory is a field of optimization that deals with the control of dynamical processes and systems. In control theory applications, the system dynamics are modeled or approximated, often as an ordinary differential or difference equation (ODE). System inputs are calculated from the model such that the desired system state or output is achieved.

Optimal control

Optimal control is an approach to control theory where the input signals are calculated optimally based on the observed current state of the system with regard to a convex cost function. For linear ODEs optimal control has well-established analytic proofs of stability and optimality under conditions of reachability and open-loop stability[33]. Optimal control and control theory is an old field with well-established applications in multiple industries and fields [34].

Optimal control relies on accurate modeling of the system dynamics. In practice control systems are often nonlinear [35] and require linearization in order to adhere to the optimal control framework. Optimal control gives some overhead in terms of robustness to modeling error. However, even for robust control frameworks operational drift or changing systems can be challenging

to model [36]. Traditionally, high-dimensional systems have been challenging for optimal control due to the curse of high dimensionality. However optimal control has made progress in recent years[37], improving the reliability for high-dimensional and large-scale systems[38][39]. Using ML methods like ANNs or DRL as model estimators could allow optimal control of highly nonlinear or unknown systems.

ML in control theory

Recent works in the fields of control theory and system identification have tried to use machine learning and ANNs in their respective fields. [40] develops a model-free controller using a fully connected ANN with Deep-Q DRL. In this framework, the ANN does both system identification and control design on the fly requiring no modeling. This approach does however not come with the same guarantees as e.g optimal control and [40] states stability analysis as future work. Moreover, the drawbacks of online and deep RL are present in this framework.

[36] uses a simple Gaussian process (GP) to compensate for operational drift in a model predictive controlled (MPC) system. The system is modeled and the controller designed offline while the GP updates the model online based on live observations. The use of a simplistic data-driven model like a GPs mitigates some of the dangers of online RL. The GP does however assume a Gaussian output distribution meaning this approach is limited to certain applications. Moreover, GP scales poorly with the number of data points [41]. An ANN could potentially allow for a less restrictive approach to operational drift estimation at the cost of less predictability and guarantees.

[42] proposes ML to find optimal action policies for constrained or safety-critical systems. ML-safe sub-problems are specified in order to circumvent the unpredictable nature of many complex ML methods. The authors use look-ahead and proactive safety planning along with a control barrier function (CBF) as part of the cost function. If the CBF is violated an optimal controller kicks in as backup ensuring the safety maintained. This approach limits the dangers of online DRL and allows safe environment exploration where the safety constraints are well-defined and modellable. A more predictable policy and ANN evolution would allow less restrictive CBFs.

ML clearly has its applications in control theory but is often restrained in order to guarantee operationally safe applications. The unpredictability of

training restricts online model training to simplistic models or backup safety systems. Making ANN training more predictable, with guarantees of convergence and reduced hyperparameter search space allows these powerful tools to be used in online systems and control applications. Tackling ANNs from a control systems perspective bridges a gap between the two disciplines and gives access to mathematically well-grounded methods and tools such as stability and reachability analysis for ANNs. The theory required to bridge this gap could come from the neural tangent kernel.

1.4 NTK

Recent insights into the learning behavior of ANNs come from the study of ANNs with infinite width. [43] introduces the neural tangent kernel (NTK) for ANNs, describing the gradient descent training behavior as a linear ordinary difference equation (ODE). The authors use a first-order Taylor linearization of the NTK to derive a linear ODE description of the training for finite-width ANNs. Follow-up work by [44] demonstrates the empirical region of validity of this linearization. The higher-order terms of the linearization have been studied in [45]. This novel perspective on ANN evolution allows for more insight into the training dynamics and its analytical properties.

Tangential work studied the infinite NTK for many different architectures like CNNs [44], RNNs [46] and transformers [47]. Moreover, [48] introduces the notion of *architectural universality* for the NTK and demonstrates its existence for any ANN architecture. The NTK has also been used for generative architectures. The internal stability of GANs and other encoder-decoder ANN structures are revealed by the NTK framework [49]. The NTK perspective appears to be applicable to many deep learning methods and ANN architectures.

One criticism of the NTK is its computational complexity. Recent work has limited this drawback. [50] introduce an NTK approximation method for improved computation speed of infinite NTKs. [51] significantly improves both computational times and reduce the memory required for finite NTK calculations by exploiting the Jacobian symmetric structure. This allows the NTK to be used for real-time training analysis. Moreover, [52] reduces the NTK dimensionality decreasing resource cost and requirement in direct NTK applications.

Recent works have applied this NTK perspective in practice. The NTK has been used to prune ANNs, i.e remove uninfluential node connections [53] improving model performance. Moreover, the generalization of ANNs has been analyzed using the NTK [54].

This perspective has had an impact on control theory applications using ANNs as well. In this work, we define two categories of methods influencing the ANN training dynamic using the NTK, indirect and direct.

Indirect

So far in the literature, the NTK and control theory perspective on ANN training have been used *indirectly* through reference and model matching. [55] uses the eigenvalues of the NTK to improve the convergence rate for physically informed neural networks (PINNs). The NTK revealed disparities between different loss components. The authors alter gradient descent using the NTK eigenvalues during training such that the loss converges homogenously over each loss component.

The NTK has been used to complement existing policy gradient methods [56] and robust Q-learning [57]. The latter papers implement cascade optimal control methods to enhance the convergence and stability of reinforcement learning methods, implicitly.

Direct

However, in this research, we follow a *direct* approach to influence the ANN training dynamics using optimal control. Firstly, the concepts of stability and reachability is introduced for a given ANN, based purely on the NTK. This allows for directed and reduced hyperparameter search spaces. Secondly, we develop a novel ANN training algorithm introduced as *Controlled Descent Training* (CDT). CDT is a model-based, $\mathcal{H}_2$ optimal state feedback label augmentation method built on the NTK that provides convergence guarantees (locally) and explicitly minimizes the cumulative training loss. This brings predictability of ANN learning, increased robustness to hyperparameter choices, and guarantees otherwise missing for ANNs without compromising on performance.

1.5 Contributions:

The main contributions of the research can be summarized as:

- The research expands on the NTK description of ANN training introduced by [43]. A dynamic label augment is introduced to the expanded description, allowing more direct control over the training dynamics.

- A novel training algorithm is introduced as *controlled descent training* (CDT). The novel algorithm uses the expanded NTK description along with optimal control of the label augment to guarantee local training convergence and improve convergence rate.

- Stability is introduced for ANN training. It is demonstrated that local stability is influenced by hyperparameter selection. Hence, it can be demonstrated under which hyperparameters the local training is non-divergent before training the network, greatly limiting the hyperparameter search space.

- Reachability is introduced for ANN training. Examples of when reachability is not achieved is given and the implication on data structure and network architecture discussed.

1.6 Thesis outline

Chapter 2 the NTK and ANN training dynamics given by [43] is recalled and expanded on to include the dynamic training label. Moreover, stability and reachability of the ANN training dynamics is introduced.

Chapter 3 introduces the *controlled descent training* (CDT) algorithm along with the local convergence guarantees.

Chapter 4 demonstrates CDTs performance on benchmarking data sets compared to traditional gradient descent.

Chapter 5 is a summary of the research so far. And finally Chapter 6 includes a conclusion of the thesis as well as future research. Chapter 7 contains additional addendums with supplementary proofs and discussions.

1.7 Ethics

This research follows the ethical guidelines regarding artificial intelligence and machine learning put forth by Centiro. This thesis also follow the general guidelines regarding publication ethics put forth by Chalmers and the Committee On Publication Ethics (COPE). No part of this thesis has been written with the help of GPT or any other large language transformer model.

1.8 Notations

We introduce the notation for an artificial neural network (ANN) as $F(\theta(k), x) : \mathbb{R}^{r \times n_0} \to \mathbb{R}^{r \times n_L}$ where $\theta(k) \in \mathcal{C}$ denotes the parameterization comprising weights and biases at training step k. $\mathcal{C}$ here denotes at least once continuous differentiability. Let the output of the ANN for a fixed n_0-dimensional set of data $x \in \mathbb{R}^{r \times n_0}$ be $\hat{y}(\theta(k), x) = F(\theta(k), x) \in \mathbb{R}^{r \times n_L}$. Assume it is continuously differentiable with respect to $\theta(k)$: $\hat{y}(\theta(k), x) \in \mathcal{C}$. r denotes the number of data points.

CHAPTER 2

Neural Network training dynamics

In the following sections, the NTK and the ANN training dynamics given in [43] is refreshed. Moreover, a dynamic label augment is introduced. Serving as an additional input to the training dynamics the label augment allows greater control of training behavior. Stability and boundedness of unaugmented training are introduced. The effects of hyperparameter selection on stability and local training convergence is discussed. Reachability is then introduced for the augmented training dynamics. The architectural and data structural implications of reachability being fulfilled or unfulfilled by the ANN dynamics is discussed. These two concepts give insight into ANN training behavior while providing the required conditions under which CDT guarantees convergence.

2.1 Neural Tangent Kernel

The NTK is adopted to describe the ANN evolution in function space during gradient descent training[43]. As the ANN width increases, the NTK evolution rate stagnates. For finite-width ANNs the NTK is time-varying resulting in a nonlinear Ordinary Difference Equation (ODE).

This latter NTK is referred to as the *empirical tangent kernel*. The in-

directly time-varying NTK ODE can be approximated as a time-invariant system by Taylor linearization around the initial parameters [44]. Linearization of the finite width NTK ODE description paves the way for our main contribution, analysis, and explicit control of the ANN training dynamics. As such we use the empirical NTK and define it as follows.

Definition 1: ***Neural Tangent Kernel.*** *Given two data points $x_i, x_j \in \mathbb{R}^{n_0} \subset x \in \mathcal{D}$, the NTK for an r -batch size, n_0-input n_L-output ANN $F(\theta(k), x) : \mathbb{R}^{r\times n_0} \to \mathbb{R}^{r\times n_L}$ at time instance $k \in \mathbb{Z}_+$, is*

$$\Theta_{i,j}(k) = \left(\frac{\partial \hat{y}(\theta(k), x_i)}{\partial \theta(k)} \frac{\partial \hat{y}(\theta(k), x_j)}{\partial \theta(k)}^T \right) \in \mathbb{R}^{n_L \times n_L} \tag{2.1}$$

where $\hat{y}(\theta(k), x)$ is the output of the ANN.
We define the full NTK for a subset of data $x \in \mathcal{D} \subseteq \mathbb{R}^{n_0 \times r}$ as

$$\Theta(k) = \begin{bmatrix} \Theta_{0,0}(k) & \Theta_{1,0}(k) & \dots & \Theta_{r,0}(k) \\ \Theta_{0,1}(k) & \Theta_{1,1}(k) & & \vdots \\ \vdots & & \ddots & \vdots \\ \Theta_{0,r}(k) & \dots & \dots & \Theta_{r,r}(k) \end{bmatrix} \in \mathbb{R}^{rn_L \times rn_L} \tag{2.2}$$

The above mentioned empirical Neural Tangent Kernel $\Theta(k)$ in eq. (2.2) is always symmetric and positive semidefinite. Positive-definiteness of the NTK ensures the convergence of the loss to a minimum for a class of loss functions (e.g., quadratic losses) [43]. A weak assumption for positive-definiteness can be made if each pair of training inputs are not parallel and lie within a Euclidean unit ball [58]. Some additional conditions guaranteeing its definiteness are given in[59].

2.2 Local and global ANN training dynamics

In this section, with the help of [43], [44] local and global finite-width ANN training behavior is introduced.

Assuming a constant target vector $y \in \mathcal{Y} \subseteq \mathbb{R}^{n_L \times r}$ (i.e., static labels in supervised learning), the output follows certain dynamics dictated by gradient descent. For the sake of brevity, we denote $\hat{y}(\theta(k), x)$ as $\hat{y}(k)$, bearing in mind that the estimated output $\hat{y}(k)$ still depends on the input data sequence.

Furthermore, we assume the loss function is at least once continuously differentiable $\mathcal{L}(\hat{y}(k), y) \in \mathcal{C}$ with respect to $\theta(k)$ and $\hat{y}(k)$ at any time instance $k \in \mathbb{Z}$. The evolution of the parameter vector $\theta(k)$ and thereof the network output $\hat{y}(k)$ under gradient descent with learning rate α is given by

$$\theta(k+1) = \theta(k) - \alpha \frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \theta(k)} = \theta(k) - \alpha \frac{\partial \hat{y}(k)^T}{\partial \theta(k)} \frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} \quad (2.3)$$

$$\hat{y}(k+1) = \hat{y}(k) - \alpha \Theta(k) \frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} = f_\theta(\hat{y}(k)). \quad (2.4)$$

Eq. (2.4) captures the evolution of the global training dynamics as a nonlinear time-discrete (ODE).

As can be seen in eq. (2.4) the symmetric empirical kernel $\Theta(k)$ has a central role in describing the training behaviour.

Proposition 1 in Appendix 7.1 (taken from [60]) demonstrates that the global training under gradient descent has a unique solution on a discrete time interval $k \in [k_i, k_f]$. A local and linear (in $\hat{y}$) training dynamics can be obtained at any time instance k, by first order Taylor series approximation of eq.(2.4) (see Appendix 7.2 for full derivation). In this case, $\hat{y}(k)$ is approximated at $\theta(k_0)$ when $\vartheta(k) \equiv \theta(k) - \theta(k_0)$, such that

$$\hat{y}_\vartheta(k+1) = \hat{y}_\vartheta(k) - \alpha \Theta(k_0) \frac{\partial \mathcal{L}(\hat{y}_\vartheta(k), y)}{\partial \hat{y}_\vartheta(k)} = f_\vartheta(\hat{y}_\vartheta(k)). \quad (2.5)$$

A bound on the error between the local and global dynamics can be found using the Lagrangian error bound (see Appendix 7.3). This bound allows us to quantify the error introduced via the first order approximation. Additional linearization may be required for certain loss functions to reach input affine form (see Appendix 7.4).

2.3 Controlled ANN training dynamics with label augmentation

As mentioned previously, one of the main contributions of the research is to explicitly control the NTK training dynamics. As such, we introduce a dynamic label augmentation method, i.e. inject fictitious, time dependent

labels by $y_u(k)$ as

$$\bar{y}(k) = y + y_u(k) \tag{2.6}$$

Unlike y, $\bar{y}(k)$ dynamically alters the targets to be estimated by the ANN. The label augmented dynamics (controlled global training dynamics) is then formulated by,

$$\hat{y}(k+1) \quad = \quad \hat{y}(k) - \alpha\Theta(\theta(k))\frac{\partial\mathcal{L}(\hat{y}(k), \bar{y}(k))}{\partial\hat{y}(k)}. \tag{2.7}$$

In eq. (2.7), the injected labels purports to give a new degree of freedom to influence the local and global training behavior. In Section 3.1 an optimal way to select the fictitious labels $y_u(k)$ is demonstrated.

2.4 Analytic properties of discrete-time training dynamics

Before the CDT algorithm is introduced two conditions are established under which CDT guarantees convergence of the local training dynamics; stability and reachability.

Firstly, global and local stability concepts of ANN training dynamics are defined around specific equilibrium values. Stability guarantees boundedness of the unaugmented training dynamics. Secondly, we analyze the local controlled training dynamics from a reachability perspective. If reachability conditions are met, this ensures that the label augments $y_u(k)$ can help us to reach any points in $\mathcal{Y}$ within a finite number of steps. Both properties can be verified before training using the initial kernel Θ_0.

Boundedness of the training dynamics

We relate boundedness of the network output via eq. (2.4) to the context of internal stability. Stability refers to the existence of a finite bound between the ANN output $\hat{y}$ and some equilibrium output $\hat{y}_e$. Here, a training equilibrium point $\hat{y}_e$ is defined as follows,

$$\hat{y}(k+1) = \hat{y}(k) = \hat{y}_e. \tag{2.8}$$

Furthermore, it follows from the dynamics in eq. (2.4) that for most conventional losses equilibrium points may exist at $\hat{y}_e = y$. We discuss the conditions under which an equilibrium point may exist in Appendix 7.5. The formal stability definition of eq. (2.4) can be captured by the following definition.

Definition 2: ***Uniform internal stability [61]*** *The discrete-time ANN training dynamics with network output $\hat{y}(k)$, initial network output $\hat{y}(k_0) = y_0$ and equilibrium point $\hat{y}_e$ is called uniformly bounded if there exists a finite positive constant γ such that for any k_0 and y_0 the corresponding solution satisfies*

$$||\hat{y}(k) - \hat{y}_e||_2 \leq \gamma ||\hat{y}(k_0) - \hat{y}_e||_2, \qquad k \geq k_0 \tag{2.9}$$

In essence, the uniform stability guarantees the ANN output during training $\hat{y}$ does not diverge from the equilibrium point $\hat{y}_e$ to infinity in finite time. The stronger stability condition of exponential stability is defined as,

Definition 3: ***Uniform exponential internal stability [61]*** *The discrete-time ANN training dynamics in eq. (2.4) with network output $\hat{y}(k)$, initial prediction $\hat{y}(k_0) = y_0$, and equilibrium point $\hat{y}_e$ is called uniformly exponentially stable if there exists a finite positive constant γ and a constant $0 < \kappa \leq 1$ such that for all k_0 and y_0*

$$||\hat{y}(k) - \hat{y}_e||_2 \leq \gamma \kappa^{k-k_0} ||\hat{y}(k_0) - \hat{y}_e||_2, \quad k \geq k_0. \tag{2.10}$$

To verify the above mentioned conditions for generic loss functions by using the global training dynamics is an uneasy task. However, the internal stability conditions of the local training dynamics described in eq. (2.5) may result in simplified conditions. As an example, the stability conditions of local training dynamics with quadratic loss reduce to an eigenvalue condition. As such, internal stability reads as

$$|\lambda| \leq 1, \quad det\left(\lambda I - (I - \alpha\Theta(k_0))\right) = 0 \quad \forall\lambda \tag{2.11}$$

where α uniformly scales the eigenvalues of the local-empirical NTK, $\Theta(k_0)$. If $\Theta(k_0)$ is positive semi-definite and none of the scaled eigenvalues of $\alpha\Theta(k_0)$ is larger than 1, the inequality in eq. (2.11) is strict. Hence, α can be choosen such that the local training dynamics is guarantee to be stable stable. Furthermore, this guarantees in the local sense that the equilibrium output is asymptotically reached. In Appendix 7.6 a concise and loss function dependent derivation of stability analysis is provided for certain common loss func-

tions.

Remark 1. ***Learning rate adaptation****. Finally, some ANN training algorithms[62] suggest altering the learning rate α. Intuitively, the learning rate scales the eigenvalues of the local-empirical NTK and as such impacts stability. Modifying the scalar parameter α may help the convergence of the training dynamics.*

Remark 2. ***Robust stability.*** *The linearization error discussed in Appendix 7.3 can be propagated through eq.*(2.10) *to deduce global stability from the local dynamics. This is further discussed in Appendix 7.7.*

Reachability

Reachability is a property of the label augmented training dynamics given in eq.(2.7). It verifies the existence of a bounded sequence of the label augments $y_u(k)$ such that any targeted ANN output $\hat{y}(k_f)$ can be reached from $\hat{y}_0$ within f finite steps.

Definition 4: ***Reachability.*** *The label augmented training dynamics eq.* (2.7) *is called reachable on $[k_0, k_f]$ if from a given initial state $\hat{y}(k_0)$ there exists at a sequence of $y_u(k)$ such that any $\hat{y}(k_f)$ can be reached $k_0 < k_f < \infty$.*

From Definition (4) the label reachability (with $y(k_f) = y$) can be derived as a specific case of reachability. The reachability condition for the global training dynamics (with input affine label augments) in eq. (2.7) can be verified using difference-geometric [63] or set theoretic algorithms [64]. This also indicates that addressing the reachability question for generic and complex loss functions is hard.

In some specific cases of the loss function (e.g. if the controlled training dynamics is local and the label augments are injected in an input affine way), the reachability analysis is straightforward to perform. Especially, the reachability analysis of quadratic losses and local training dynamics can be concluded by using linear systems and control theory [61]. In such cases, we borrow the Popov-Belman-Hautus (PBH) test given by,

$$rank \begin{bmatrix} zI - (I - \alpha\Theta(k_0)) & \alpha\Theta(k_0) \end{bmatrix} = rn_L, \quad \forall z \in \mathbb{C}. \tag{2.12}$$

Numerically, the PBH condition consists of the finitely many rank tests at the eigenvalues of $(I - \alpha\Theta(k_0))$.

Remark 3. ***Unreachable local training dynamics***. *The importance of reachability in ANN training can easily be captured when it is not full-filled. A specific example is if there exist two identical data points. In such a case, the local empirical NTK has two similar rows or columns (see eq. 2.1) causing rank deficiency in* $\Theta(k_0)$. *The intuitive explanation is that two similar or identical data points will yield the same ANN output. Hence these two points are inseparable in ANN output space. In practice, if these data points have the same label the training will be stabilizable (see remark 5).*

Remark 4. ***Overfitting.*** *Intuitively, if reachability is fulfilled, the augmented training can perfectly fit the training data, causing overfitting. Consequently, if reachability is not fulfilled the ANN cannot perfectly fit the data. Hence reachability can be a good measure of whether or not a network is complex enough to fit the data or if data has conflicting data points. Moreover, the overfitting caused by the augmented learning can be remedied with various regularization techniques (e.g. [65]).*

Remark 5. ***Stabilizability.*** *If the local dynamics is not full state reachable but the non-reachable states partition is locally asymptotically stable, we call the training dynamics locally stabilizable.*

Finally, the above mentioned analytic conditions (stability, reachability, stabilizability) support the deployment of model based and optimal label augmentation solutions.

CHAPTER 3

Locally Optimal Control of ANN training dynamics

In this chapter the novel training algorithm *controlled descent training* is introduced using the label augmented dynamics. Under conditions of reachability and stability, CDT guarantees local convergence and optimally augmented labels.

3.1 Controlled Descent Training

In Section 2.3, label augments, as new fictitious inputs, have been injected into the training dynamics. In the following section, it is demonstrated how to calculate the label augments $y_u(k)$ such that stability and some optimality criteria are (at least locally) satisfied. The main idea is to use $\hat{y}_\vartheta(k)$ and transform it to $y_u(k)$ with a static gain. In Figure 3.1, the schematics of the closed-loop and controlled label augmentation for a network trained with MSE are depicted.

In order to find K (in Figure 3.1), we propose to use an optimal state feedback label augmentation method. More precisely, $y_u(k)$ label injection is aimed at $\mathcal{H}_2$ optimal closed loop training dynamics (CDT). In the following section, we restrict ourselves to quadratic loss functions and assume the

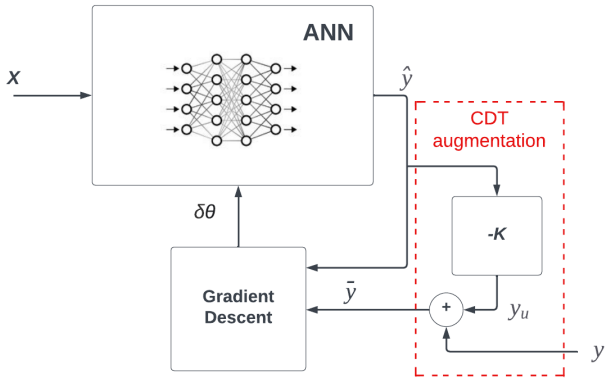


Figure 3.1: Schematic of Controlled Decent Training. The CDT augmentation block calculates the label augment y_u dynamically from the ANN output $\hat{y}$ using the controller feedback matrix K. The new dynamically augmented label $\bar{y} = y + y_u$ is fed to GD rather than the static label y.

augmented training dynamics is stabilizable (or reachable).

The following notation is introduced,

$$\tilde{y}(k) = \begin{bmatrix} \hat{y}_\vartheta(k) \\ 1 \end{bmatrix}, \tag{3.1}$$

$$\tag{3.2}$$

This allows the standard infinite horizon cost to account for the offset introduced by the labels y. More precisely, the following infinite horizon cost is minimized according to

$$\min_{y_u} \tfrac{1}{2} \textstyle\sum_{i=k_0}^{\infty} \tilde{y}(i)^T \tilde{Q} \tilde{y}(i) + y_u(i)^T R y_u(i) \tag{3.3}$$

$$s.t.\ \tilde{y}(k+1) = \begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix} \tilde{y}(k) + \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix} y_u(k) \tag{3.4}$$

$$\tag{3.5}$$

where $\tilde{Q} \in \mathbb{R}^{r(n_L+1)\times r(n_L+1)}$ and $R \in \mathbb{R}_+^{rn_L \times rn_L}$ are real valued positive semi-definite and positive definite weighting matrices, respectively. More pre-

cisely,

$$\tilde{Q} = \begin{bmatrix} Q & -Qy \\ -y^T Q & y^T Q y \end{bmatrix}. \tag{3.6}$$

The cost includes the weighted squared error between the ANN predictions according to the linear dynamics and the targets, as well as the weighted square sum of the label augment y_u. The weighting matrix $Q \in \mathbb{R}^{rn_L \times rn_L}$ can be chosen such that certain data points or ANN outputs are more important than others. Moreover, the local training dynamics in eq. (3.4) captures learning interactions between data points in x which in turn influence the optimal solution. The optimization problem, if solved, delivers an $\mathcal{H}_2$ optimal label augmentation solution. The cost function in eq. (3.3) describes a generic energy approach to label augment selection where the weighting matrices Q, R shape their relative importance. Finally, the first term in eq. (3.3) penalizes the deviation from the static targets.

The locally stabilizing and optimal solution to eq. (3.3)-(3.5) can be found by using the Discrete-time Algebraic Riccati equations (DARE) [1](see Appendix 7.8 for solution derivation). If the stationary and extremal solution to DARE is P then the optimal feedback gain can be written as

$$K = \left(R + \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix}^T P \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix} \right)^{-1} \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix}^T P \begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.7}$$

$$y_u(k) = -K\tilde{y}(k) \tag{3.8}$$

$$\bar{y}(k) = y_u(k) + y = -K\tilde{y}(k) + y \tag{3.9}$$

The closed and CDT controlled loop becomes

$$\begin{bmatrix} \hat{y}_\vartheta(k+1) \\ 1 \end{bmatrix} = \left(\begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix} - \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix} K \right) \begin{bmatrix} \hat{y}_\vartheta(k) \\ 1 \end{bmatrix}. \tag{3.10}$$

The feedback gain matrix K maps the ANN output $\tilde{y}$ to *target augments* $y_u(k)$ such that it minimizes the cost in eq. (3.5) on an infinite horizon. Note, the optimal cost value with the state feedback policy is $\frac{1}{2}\tilde{y}(k_0)^T P \tilde{y}(k_0)$. Finally,

[1]Stabilizability and detectability conditions must hold [33]

the controller gain K can be calculated before training and remains constant during training. In the local dynamical sense, the linear difference equation in eq. (3.10) guarantees asymptotic stability, and therefore convergence.

Remark 6. ***Batch***. *CDT suggests using the local empirical NTK for the whole training dataset (megabatch). In practice, it may be more attractive with a traditional mini-batch approach, recalculating the NTK and feedback controller for each batch. This would call for receding horizon optimal control.*

Remark 7. ***Robustness***. *The proposed $\mathcal{H}_2$ state feedback control policy is robust with a guaranteed magnitude [33]. This makes CDT applicable on the global training dynamics in practice. However, for proper handling of the modeling error between the global and the local training dynamics, robust control methods are proposed.*

The CDT algorithm

In previous sections, the concepts of reachability and stability were introduced for ANNs and their implications on hyperparameter selection examined. The optimal target augment sequence $\bar{y}(k)$ was calculated using LQR such that stability is guaranteed and convergence rate improved. The full CD training algorithm for MSE is summarized in Algorithm 1.

Algorithm 1 CDT summary

Calculate $\Theta_0 \leftarrow$ eq.(2.2) ▷ Calculate kernel at ANN initialization
Check stability with eq.(2.11) ▷ (Stability)
Check reachability with eq.(2.12) ▷ (Reachability)
Calculate $K \leftarrow$ eq.(3.7) ▷ Calculate control feedback using DARE
ANN Training:
for $k \in 1, 2, ...,$ *training steps* **do**
    Calculate $\bar{y}(k) \leftarrow y - K\tilde{y}(k)$ ▷ Label augments based on feedback
    Update parameters $\theta(k+1) \leftarrow \theta(k) - \delta\mathcal{L}(\hat{y}(k), \bar{y}(k))/\delta\theta(k)$

CHAPTER 4

Numerical performance and benchmarking

In this chapter, traditional gradient descent (GD) and CDT are compared numerically using two standard benchmarking datasets. The first example is a regression problem using the Ames Housing dataset [66] with a single-target fully connected ANN and MSE loss. The second example is a binary image classification problem using ALEXNet[9] for the purpose of demonstrating the applicability of CDT on Convolutional Neural Networks (CNN). Both experiments run on a megabatch setup, meaning the data is shuffled and split into validation and train datasets with all train data in a single batch. The loss is averaged over the batch. Each model is trained 10 times with reshuffled data and a new random initialization. For each dataset, the model is trained using both optimization methods for a number of learning rates all with learning rate decay according to

$$\alpha_k = \frac{1}{1 + 0.01k}\alpha \tag{4.1}$$

where k is the training step and α is the initial learning rate. The learning rate decay is not modeled in the training dynamics to ensure the controller does not compensate for the decay by scaling the system. The controller design

cost matrices Q and R in eq.(3.5) are chosen as scaled diagonal matrices,

$$Q = I_1 \tag{4.2}$$
$$R = pI_2 \tag{4.3}$$

with identity matrices $I_1 \in \mathbb{R}^{rn_L \times rn_L}$ and $I_2 \in \mathbb{R}^{r \times r}$ and a pre-selected control input cost p [1]. Smaller values of be p give larger label augment values. In the following experiments, p is a constant ($p = 0.1$) in order to demonstrate that this design parameter is significantly less sensitive than learning rate α. Note however that choosing p will influence performance. There are multiple heuristics involving the choice of p, any of which can be used to yield even lower validation loss. However, this research focuses on demonstrating the applicability of the method and theory, rather than performance improvement. The experiments are written in Python (3.7) using the latest version (1.9.1) of PyTorch released by Facebooks AI Research Lab 2016.

4.1 Regression

The Ames Housing Price dataset contains 79 explanatory variables describing houses in Iowa along with their final sale price. The regression target for this dataset is the sale price. A description for each variable can be seen in [66]. The full dataset has 2919 entries. For the experiments, 512 data points are sampled without replacement, normalized around 0 and split into 70% training and 30% validation data. The experiments are run on a mega batch setup meaning all training data is run concurrently in a single batch. [2] I.e. a fully connected feed-forward neural network setup is used according to,

$$\begin{cases} h^{l+1} &= z^l W^{l+1} + b^{l+1} \\ z^{l+1} &= \phi^l(h^{l+1}) \end{cases} \qquad \begin{cases} W^l_{i,j} &= \frac{\sigma_w}{\sqrt{n_l}} w^l_{i,j} \\ b^l_j &= \sigma_b \beta^l_j \end{cases} \tag{4.4}$$

where $l < L \in \mathbb{N}$ is the layer where L is the final layer, n_l is number of input features to the layer l, z^0 is the input data $x \in \mathcal{D} \subseteq \mathbb{R}^{n_0 \times r}$ and r is the batch size. $w_{i,j}$ and β_j is the weight and bias where $i = 1, \ldots, n_l$ and $j = 1, \ldots, n_{l+1}$.

[1] These penalties weights are tuning parameters

[2] For traditional mini-batch gradient descent the control scheme would be recalculated for each batch, analogous to a receding horizon controller or MPC. For the purposes of this research, the mega-batch setup better demonstrates the theory presented.

W^l and b^l are the matrix and vector describing the weights and bias of a layer respectively. ϕ_l is the output activation function for layer l, hence h is the output from layer l and z is the input to the next layer $l+1$. For the regression set up no final activation ϕ^L function is used hence $z^L = h^L$. Finally $\hat{y} = z^L$ is the ANN output. The weights and biases are initialized with a normal distribution respectively $\mathcal{N}(0, \frac{\sigma_w^2}{n_l})$ and $\mathcal{N}(0, \sigma_b^2)$. The parameter vector θ is defined as

$$\theta = \begin{bmatrix} vec(W^L) & b^L & \dots & vec(W^1) & b^1 \end{bmatrix}^T \tag{4.5}$$

where vec means concatenated vector form of the matrix. $\hat{y}(\theta, x) : \mathbb{R}^{n_0 \times r} \to \mathbb{R}^r$ is the output of the ANN using input data x and parameters θ.

The fully connected ANN architectures (of varying widths and depths) used for the Housing price dataset can be seen in Table 4.1. All architectures use ReLU as inter layer activation function with no final activation. This ensures the results are not architecture dependent and demonstrate how CDT is influenced by varying widths and depths.

Architecture

For the regression experiment 3 ANN architectures similar to the model description used in [44] is used with initializations given in Table 4.1.

Architecture	Hidden Layers $(L-1)$	Width $(n_{l+1}\ \forall l \in \{l_0, l_{L-1}\})$
1	1	1500
2	3	500
3	6	250

Table 4.1: Single-target fully connected ANN architectures used for the regression experiments.

Regression experiment

The experiment is run 10 times with different initializations, reshuffled training data, and validation indices for each architecture. Tables comparing the analytical and observed properties of the two training algorithms can be seen in Tables 4.3 to 4.5. In the aforementioned tables, α is the learning rate. The MSE column describle the average model MSE loss over all initializations on

the validation data at the final training iteration. For the purpose of demonstration, if some but not all ANN initializations resulted in divergent training the average loss over all non divergent initializations is indicated. The convergence column describes how many initializations resulted in non divergent training ($\hat{y}$ does not tend toward infinity). $|eig(\Theta(k_0))| < 1$ describes the open-loop local stability of training for all initializations. Reachability describes the reachability of training for all initializations. Figures 4.1a to 4.1c show the average difference ($\mathcal{L}_{GD} - \mathcal{L}_{CDT}$) of MSE validation loss between GD and CDT during training over all initializations for each architecture. The relative difference between the two validation losses is always in favor of CDT (the metric is never < 0) hence Figures 4.1a to 4.1c are shown in log_{10} scale. Only learning rates where both CDT and GD converge for all initializations are shown in the figures. The absolute MSE losses for each architecture and learning rate can be seen in Figures 4.3a to 4.5d. CDT does not affect training time since the NTK and optimization are calculated before training. The NTK and optimization are however computationally heavy. A table with the average total computation time along with the average training time for each architecture can be seen in table 4.2. The training and optimization were done on a single NVIDIA RTX A2000 laptop GPU.

Architecture	CDT Total (s)	CDT Training (s)	GD Training (s)
1	58.5	37.1	37.2
2	18.6	10.3	10.1
3	9.4	5.3	5.3

Table 4.2: CDT Computation time for each architecture on an RTX A2000 GPU. *CDT Total* includes NTK calculation, optimal feedback optimization, and augmented GD training.

α		Reachability	$\lvert eig(\Theta(k_0))\rvert < 1$	Convergence	MSE 10^{-3}
10^0	GD	Yes	No	None	$\infty \pm \infty$
10^0	CDT	Yes	No	All	150.01 ± 350.50
10^{-1}	GD	Yes	No	7/10	3.04 ± 0.59
10^{-1}	CDT	Yes	No	All	1.95 ± 0.24
10^{-2}	GD	Yes	Yes	All	2.70 ± 0.61
10^{-2}	CDT	Yes	Yes	All	2.34 ± 0.49
10^{-3}	GD	Yes	Yes	All	5.90 ± 0.95
10^{-3}	CDT	Yes	Yes	All	3.72 ± 0.76

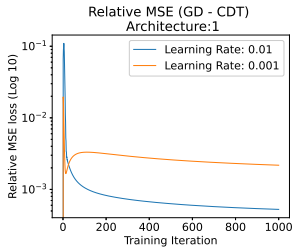
Table 4.3: Analytical and observed properties for fully connected architecture 1 on regression dataset.

α		Reachability	$\lvert eig(\Theta(k_0))\rvert < 1$	Convergence	MSE 10^{-3}
10^0	GD	Yes	No	None	$\infty \pm \infty$
10^0	CDT	Yes	No	All	2.14 ± 0.28
10^{-1}	GD	Yes	No	All	2.31 ± 0.45
10^{-1}	CDT	Yes	No	All	1.90 ± 0.29
10^{-2}	GD	Yes	Yes	All	3.71 ± 0.62
10^{-2}	CDT	Yes	Yes	All	2.62 ± 0.54
10^{-3}	GD	Yes	Yes	All	7.47 ± 1.54
10^{-3}	CDT	Yes	Yes	All	5.24 ± 0.99

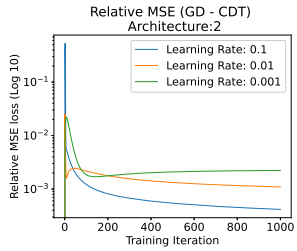
Table 4.4: Analytical and observed properties for fully connected architecture 2 on regression dataset.

α		Reachability	$\lvert eig(\Theta(k_0))\rvert < 1$	Convergence	MSE 10^{-3}
10^0	GD	Yes	No	None	$\infty \pm \infty$
10^0	CDT	Yes	No	All	1.88 ± 0.27
10^{-1}	GD	Yes	No	All	2.63 ± 0.80
10^{-1}	CDT	Yes	No	All	1.90 ± 0.48
10^{-2}	GD	Yes	Yes	All	7.58 ± 1.84
10^{-2}	CDT	Yes	Yes	All	4.23 ± 1.05
10^{-3}	GD	Yes	Yes	All	12.20 ± 2.61
10^{-3}	CDT	Yes	Yes	All	10.42 ± 2.11

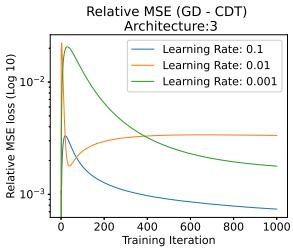
Table 4.5: Analytical and observed properties for fully connected architecture 3 on regression dataset.



(a) Architecture 1



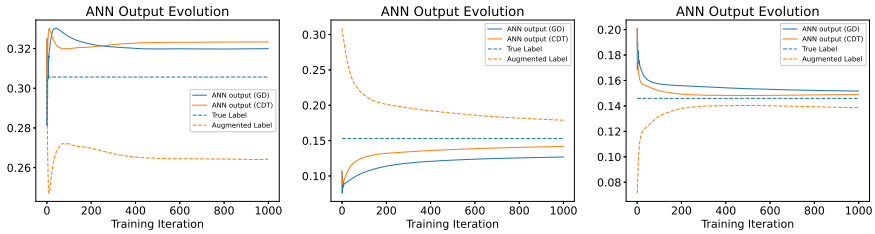
(b) Architecture 2



(c) Architecture 3

Figure 4.1: Average relative difference between GD and CDTs ($\mathcal{L}_{GD} - \mathcal{L}_{CDT}$) MSE validation loss for all learning rates where both training methods converge for each of the three regression architectures. CDT has a lower validation loss during the entirety of training for all learning rates and architectures.

As can be seen in Table 4.3 to 4.5, CDT is more robust to higher learning rates with competitive final MSE validation loss while SGD diverges to infinity. CDT consistently converges to a lower loss for all architectures and learning rates. Moreover, the CDT standard deviation is smaller than for GD hence the augmented training is more consistent between initializations and data shuffles. The difference between the highest and lowest final MSE loss is consistently smaller for CDT and the performance only changes significantly for very low learning rates. Hence, CDT is seemingly less affected by choice of learning rate α than traditional GD. This behavior is expected as the controller may scale the system as required. It can be seen in Table 4.3 that architecture 1 trained with CDT does not diverge for any initialization at the highest learning rate $\alpha = 1$ but converges further away from the true labels than at



(a) Sample 1 **(b)** Sample 2 **(c)** Sample 3

Figure 4.2: ANN outputs $\hat{y}$, true label y and augmented label $\bar{y}$ evolution for random samples when training under CDT and GD. First initialization of architecture 1 with learning rate 0.01. CDT appears to converge closer to true label y for most, but not all data samples in the training batch.

initialization. Due to the high learning rate and relatively few parameters in the single hidden layer architecture, the true kernel $\Theta(k)$ changes rapidly making the global dynamics drift from the local approximation $\Theta(k_0)$. Note however that despite this CDT does not diverge to infinity.

Table 4.2 demonstrates that the training time between GD and CDT is negligible. However, calculating the NTK and solving the CDT optimization does take about as much time as the fitting itself. It is worth noting that both the NTK calculation time and the CDT optimization time scale exponentially with batch size. Hence the mega batch setup exacerbates the computational difference between total computation time and training time.

Regarding observed global convergence it can be seen in Tables 4.3 to 4.5 that for some learning rates, the local stability condition is not fulfilled. Despite this, both GD and CDT are observed to converge to the true labels. This hints at the higher order interactions not modeled by the first order Taylor approximation improves robustness and does not cause divergent training. This requires more analysis to confirm and is left for future work. Furthermore, the reachability condition is always fulfilled for all architectures. The dataset provided is clean and thoroughly examined for duplicates and other issues hence loss of reachability resulting from duplicated data points is not an issue.

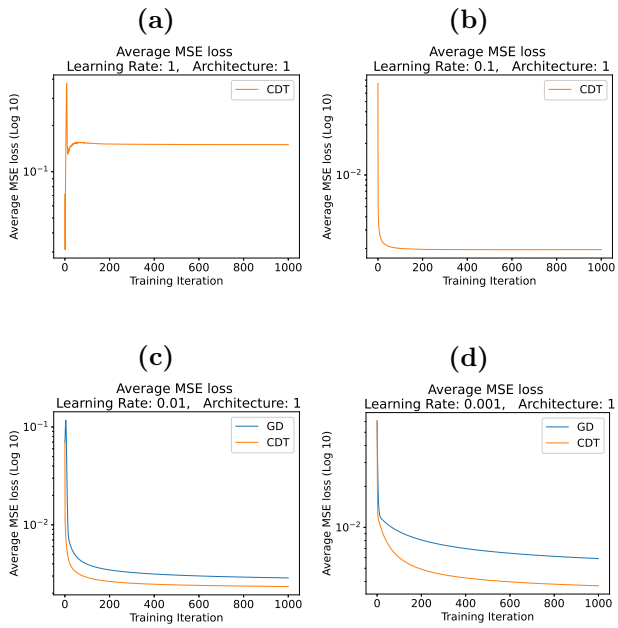


Figure 4.3: Absolute MSE loss on validation data for fully connected ANN architecture 1 averaged over all 10 initializations and data shuffles during training with GD and CDT. CDT converges with fewer iterations than GD for all learning rates. Moreover, CDT is stable at higher learning rates ($\alpha \geq 0.1$), while GD diverges to ∞_+. While reaching a finite steady-state behavior for learning rate $\alpha = 1$ CDT is not observed to converge to the true labels.

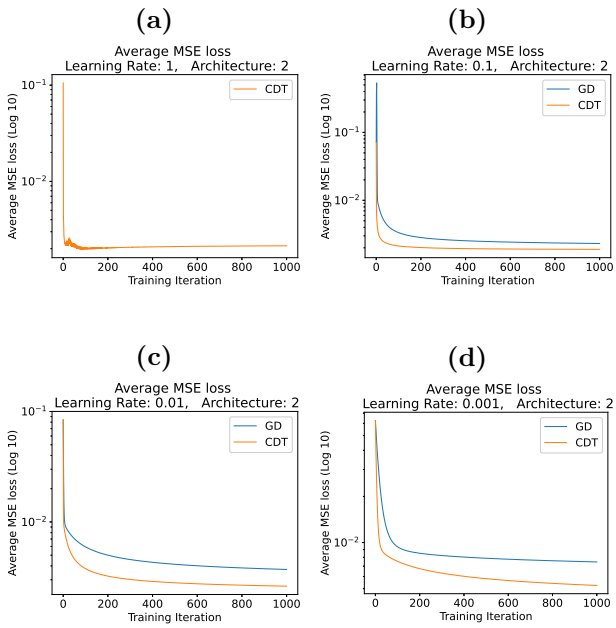


Figure 4.4: Absolute MSE loss on validation data for fully connected ANN architecture 2 averaged over all 10 initializations and data shuffles during training with GD and CDT. CDT appears to converge with fewer iterations than GD for all learning rates. Moreover, CDT is stable at higher learning rates ($\alpha \geq 1$), while GD diverges to ∞_+.

Figure 4.1a to 4.1c demonstrate that CDT converges in fewer iterations compared to GD. The difference in performance is largest at the start of training, meaning CDT has already converged to a lower loss than GD during early iterations. Figures 4.3a to 4.5d highlight this further.

Figure 4.2 show the ANN output evolution for 3 randomly selected samples and a single initialization under both training methods along with the augmented $\bar{y}(k)$ and static y labels. Not that in Figure 4.2a CDT converges further away from the true label than GD. For this initialization, the ANN trained with CDT is closer to the true labels for 253 out of 357 samples in the training batch at the final iteration. Since CDT gives a lower average loss and outputs closer to the true targets on most samples but not all, it can be

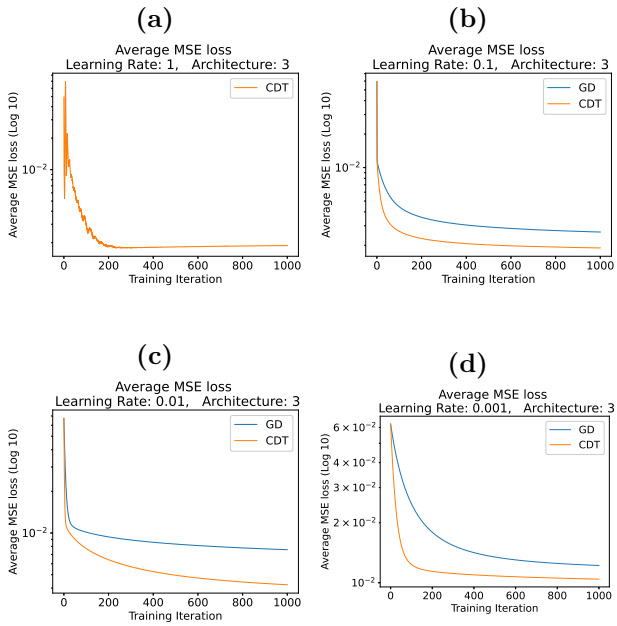


Figure 4.5: Absolute MSE loss on validation data for fully connected ANN architecture 3 averaged over all 10 initializations and data shuffles during training with GD and CDT. CDT converges with fewer iterations than GD for all learning rates. Moreover, CDT is stable at higher learning rates ($\alpha \geq 1$), while GD diverges to ∞_+.

concluded that some data samples are prioritized by the CDT method while others are not. As stated previously the empirical kernel $\Theta(k_0)$ is a matrix describing the effect of each sample on all other samples during training[43]. Hence the CDT algorithm will prioritize samples with a large effect on others such that minimal loss is achieved. Since K is a static linear transform of the ANN output as $\hat{y}$ approaches the true static labels y the augmented label $\bar{y}$ converges to the true static label y.

MSE local stability criteria

An experiment is made to compare the local stability criteria on learning rate for MSE loss with different learning rates. Only gradient descent is used for the following experiment. An adaptive[3] learning rate is calculated using the NTK according to the stability criterion given in eq. 2.11 and trained using gradient descent. The stable learning rate is calculated offline before training. The comparison for each architecture can be seen in Figures 4.6a to 4.6c.

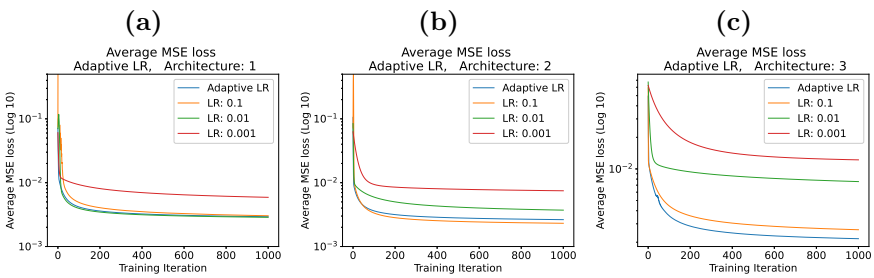


Figure 4.6: MSE loss on validation data for fully connected ANN architectures averaged over all 10 initializations and data shuffles during training with GD. Adaptive learning rate calculated using the NTK and local stability criteria compared to different learning rates.

It can be seen in Figures 4.6a to 4.6c that the calculated learning rate is competitive with the learning rate yielding the lowest validation loss for all dense architectures. However, the calculated adaptive learning rate only outperforms the lowest loss learning rate for architecture 3. The adaptive learning

[3] Adaptive in the sense that it adapts to ensure local training stability based on the data and architecture. The learning rate is still static during training.

rate guarantees local stability but higher learning rates may still yield globally stable training and lower loss. The comparison between the calculated adaptive learning rate and hyperparameter optimization needs to be investigated further. This experiment hints at the adaptive learning rate being close to optimal without any additional training reruns. If optimal hyperparameters are required then the adaptive learning rate can serve as a starting point for other hyperparameter optimization schemes, reducing hyperparameter search space.

4.2 Classification

The Microsoft research Cats vs. Dogs dataset [67] contains 25k images depicting cats and dogs equally distributed. However, for the demonstrative purposes of this work, a subset of 256 images are sampled without replacement. In order to verify the generalization properties of CDT 70% of the data is placed in the validation set. The random sampling makes no distinction between the classes, therefore the sampled dataset is not balanced between the classes. Each image is resized to 96×96 pixels with all color channels retained. The ALEXNet [9] CNN architecture is used for this dataset. This architecture is very complex compared to the previous regression example hence overfitting is expected. For the purposes of demonstration, the CNN is trained with multi-target MSE rather than the standard cross-entropy loss.[4] Figures 4.7a to 4.7d show the MSE validation loss evolution of both CDT and GD for the different learning rates.

As can be seen in Table 4.6, CDT improves training robustness for CNNs with multiple outputs at higher learning rates. Due to the small training batch size, the performance is poor for both models. ALEXNet is a complex network and will easily overfit the training data. As can be seen in Figure 4.7d, CDT accelerates learning for CNNs as well as ANNs for low learning rates. However, Figure 4.7b demonstrates that both training algorithms overfit quickly for high learning rates. CDT however stabilizes at a lower loss, indicating higher generalizability after many iterations. More robust experimentation is required to verify this observation. As can be seen in Figure 4.7c

[4] We do this for two reasons; (1) it is more closely connected to the theory presented which is the main focus of this research, and (2) it is easier to verify the theory applicability on CNNs with multiple outputs without additional linearizations.

α		Reachability	$\lvert eig(\Theta(k_0))\rvert < 1$	Convergence	MSE 10^{-3}
10^0	GD	Yes	No	None	$\infty \pm \infty$
10^0	CDT	Yes	No	All	2.59 ± 0.15
10^{-1}	GD	Yes	Yes	All	3.37 ± 0.38
10^{-1}	CDT	Yes	Yes	All	3.33 ± 0.37
10^{-2}	GD	Yes	Yes	All	2.58 ± 0.11
10^{-2}	CDT	Yes	Yes	All	2.69 ± 0.29
10^{-3}	GD	Yes	Yes	All	2.67 ± 0.10
10^{-3}	CDT	Yes	Yes	All	2.58 ± 0.11

Table 4.6: Analytical and observed properties of ALEXNet on classification dataset.

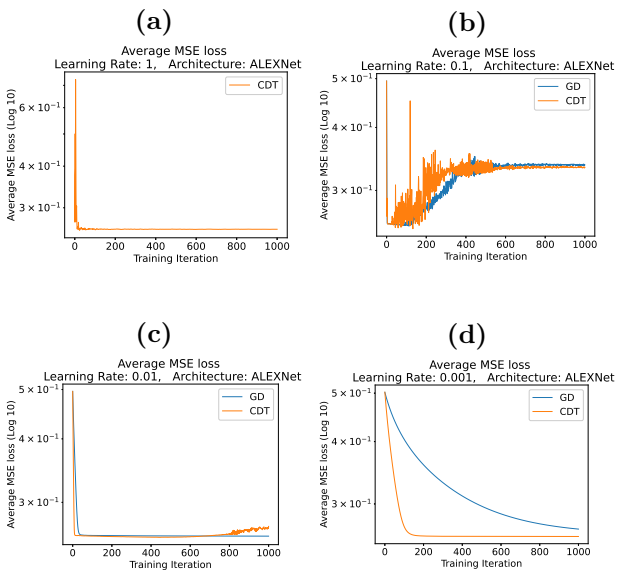


Figure 4.7: Absolute MSE loss on validation data for ALEXNet during training with GD and CDT, averaged over all 10 initializations and data shuffles of classification dataset. As can be seen in b, overfitting occurs for high learning rates α. c demonstrate overfitting occuring earlier for CDT due to the acceleration of training.

CDT converges at a few iterations and reach a lower loss than GD but does however overfit earlier than GD. The observed behavior is expected as CDT accelerates training hence overfit sooner. This hints at using a regularization method together with CDT for optimal performance when using complex network architectures. invite the community to further investigate the learning behaviour of controlled gradient descent.

CHAPTER 5

Summary of included papers

5.1 Paper A

Viktor Andersson, Balázs Varga, Vincent Szolnoky, Andreas Syrén, Rebecka Jörnsten, Balázs Kulcsár
Controlled descent training
Submitted to the *International Journal of Robust and Nonlinear Control*, Mar. 2023 **https://arxiv.org/abs/2303.09216** .

In this article a novel optimal control-based ANN training algorithm *controlled gradient descent* (CDT) is introduced. CDT is demonstrated to guarantee local training convergence and improve convergence rate. CDT is based on the neural tangent kernel framework, allowing ANN training to be analyzed from the perspective of control theory. Moreover, reachability and stability are introduced for ANN training. These two concepts are used to derive insights regarding hyper-parameter impact on training and data trainability. CDT performance compared to traditional gradient descent is demonstrated numerically and shows faster convergence and more robust training dynamics.

CHAPTER 6

Concluding Remarks and Future Work

6.1 Concluding remarks

So far the research has demonstrated that the NTK ODE description is apt for traditional control theory methods and analysis. For convex losses like MSE stability analysis demonstrates an upper limit on learning rate guaranteeing local training convergence, greatly reducing hyperparameter search space. Furthermore, the implications on stability for other losses were discussed. Moreover, reachability is demonstrated theoretically to impact overfitting and data suitability from the ANN architectural perspective.

The novel ANN training algorithm *controlled gradient descent* (CDT) was introduced, under conditions of reachability and stability. The novel algorithm uses an optimal feedback controller based on a quadratic optimization of the NTK ODE training description to alter the training labels such that minimal cumulative training loss is achieved. Moreover, CDT guarantees local training convergence allowing more predictable training behavior. CDT is demonstrated to converge to a lower validation loss than traditional gradient descent for benchmark datasets in both regression and classification.

Future work

The focus has been to demonstrate the validity of control theoretical concepts on ANN architecture. The next steps would include expanding these concepts to mini-batch gradient descent making a more robust comparative performance analysis. This research can however be expanded in multiple directions.

Expand CDT applications

The research so far shows good performance for fully-connected and convolutional ANNs. This could be expanded to include the large-scale transformer architecture, ensuring training convergence while allowing limited hyperparameter optimization or selection even for large-scale models.

More exotic losses can be explored and tackled using the same approach, like cross-entropy common for classification problems. The effect of additional linearization of for example softmax or other nonlinear final activations can also be studied. Other controller frameworks like robust or nonlinear control can be applied to deal with these additional uncertainties.

Deep reinforcement learning could also benefit from more predictable training behavior, especially in the online setting. Hence, CDT performance in online DRL applications can be investigated.

While less intuitive, the control input could augment the ANN prediction rather than the label. This could mitigate some of the issues regarding input affinity and nonlinearity of certain losses.

Deeper ANN analysis

The stability analysis can be much expanded. Assumptions on the initial training step yielding the largest parameter change could be used to more robustly prove stability during the entire training. Moreover, the upper bound on learning rate found by the stability analysis can be further investigated and implemented in common hyperparameter optimization schemes like grid-search.

The insights generated by the reachability analysis can also be expanded upon. The examples given like data duplicates can be more robustly demonstrated with numerical examples. Moreover, a deep dive into the mechanics behind could yield insights into both generalization and data trainability.

This work is a first step in bridging the gap between control theory and machine learning. Further work in this field could yield robust deep-learning based adaptive estimators and controllers fulfilling the hard safety and predictability constraints put on control theory applications. Moreover, control theory analytical tools would lend themselves well to the field of ML, reducing time and resources required for training high-performing large-scale deep-learning models.

CHAPTER 7

Appendecies

7.1 Existence and uniqueness of solution

The analysis in Section 3 and onward require the ANN training dynamics to have a unique solution on the interval $[k_i, k_f]$. The following proposition is a variation of a proposition on Liptschitzness given in [60].

Proposition 1: *Suppose that $f_\theta(\hat{y}(k))$ is bounded on the discrete interval $k \in [k_i, k_f]$ and satisfies*

$$||f_\theta(\hat{y}_1) - f_\theta(\hat{y}_2)||_2 \leq L||\hat{y}_1 - \hat{y}_2||_2 \tag{7.1}$$

$\forall \hat{y}_1, \hat{y}_2 \in \mathbb{R}^o, \forall k \in [k_i, k_f]$ with L being the Lipschitz constant. Then, for all initial conditions $||f_\theta(\hat{y}_i)||_2 \leq \phi$ with a bounded real scalar ϕ. The discrete difference equation $\hat{y}(k+1) = f_\theta(\hat{y}(k))$, with $\hat{y}_i = \hat{y}(k_i)$ has a unique solution over the time interval $[k_i, k_f]$.

Proof. By means of the continuity assumption of $\hat{y}(k)$ in $\theta(k)$, the proof is a direct consequence of [60] (Ch. 2.2, pp 67, Theorem 2.4). □

It follows that when training an ANN under gradient descent a solution to

eq. (2.4) always exists and that the solution is unique on the time interval $[k_i, k_f]$.

7.2 Local training dynamics

The following appendix details the first order Taylor linearization used to derive eq. 2.5. The following is a variation of the linearization used in [44]. We define the time instance of linearization as k_0. In this local aspect, $\hat{y}(k)$ is described with $\theta(k_0)$ and $\vartheta(k) \equiv \theta(k) - \theta(k_0)$. That is,

$$\hat{y}(k) = \hat{y}(k_0) + \left(\frac{\partial \hat{y}(k)^T}{\partial \theta(k)}\right)^T_{\theta(k_0)} \vartheta(k) + \sum_{i=2}^{\infty} \frac{\partial^{(i)} \hat{y}(k)^T}{\partial^{(i)} \theta(k)}^T_{\theta(k_0)} \frac{\vartheta^{(i)}(k)}{i!}. \tag{7.2}$$

Using only the first term from the Taylor expansion (eq. (7.2)) results in

$$\hat{y}_\vartheta(k) \equiv \hat{y}(k_0) + \left(\frac{\partial \hat{y}(k)^T}{\partial \theta(k)}\right)^T_{\theta(k_0)} \vartheta(k). \tag{7.3}$$

The smoothness (once continuously differentiable) of the loss function enables the definition of the *local training dynamics* by,

$$\hat{y}_\vartheta(k+1) = \hat{y}_\vartheta(k) - \alpha \Theta(k_0) \frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} = f_\vartheta(\hat{y}(k)). \tag{7.4}$$

7.3 Lagrange error bounds for local training dynamics

In order to quantify the error between the local and the global training dynamics the Lagrange error bound [60] is used,

$$||\hat{y}(k) - \hat{y}_\vartheta(k)||_2 \leq \max_{\theta(\kappa)} \frac{1}{2} \left|\left| \left(\frac{\partial^2 \hat{y}(k)^T}{\partial^2 \theta(k)}\right)^T_{\theta(\kappa)} \right|\right|_2 \cdot ||\vartheta(k)||_2^2 \tag{7.5}$$

where for any κ on the discrete interval $[k_0, k]$ $\theta(\kappa)$ is evaluated. Note that eq. (7.5) expresses the overbound of the deviation between the outputs obtained from the local training dynamics $\Theta(k_0)$ and the global training dynamics $\Theta(k)$.

Meaning, while the linear dynamics are not replicating the learning behavior exactly we can still quantify the goodness of the approximation.

7.4 Loss Linearization

[1] Although, the local training dynamics are linearized w.r.t. $\theta(k_0)$, the derivative of the loss $\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{\theta(k_0)}$ can still be a nonlinear function of the output $\hat{y}(k)$ (e.g., for cross entropy loss). When this is the case we propose a further linearization step and apply a first-order Taylor series approximation on the loss derivative:

$$\begin{aligned}\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{\theta(k_0)} = &\left(\frac{\partial \mathcal{L}(\hat{y}(k_0),y)}{\partial \hat{y}(k_0)}\right)_{\theta(k_0)} + \left(\frac{\partial^2 \mathcal{L}(\hat{y}(k_0),y)}{\partial^2 \hat{y}(k_0)}\right)_{\theta(k_0)} (\hat{y}(k) - \hat{y}(k_0)) \\ &+ \sum_{i=2}^{\infty} \left(\frac{\partial^{(i+1)} \mathcal{L}(\hat{y}(k_0),y)}{\partial^{(i+1)} \hat{y}(k_0)}\right)_{\theta(k_0)} \frac{(\hat{y}(k)-\hat{y}(k_0))^{(i)}}{i!} \end{aligned} \tag{7.6}$$

and

$$\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{L} = \left(\frac{\partial \mathcal{L}(\hat{y}(k_0),y)}{\partial \hat{y}(k_0)}\right)_{\theta(k_0)} + \left(\frac{\partial^2 \mathcal{L}(\hat{y}(k_0),y)}{\partial^2 \hat{y}(k_0)}\right)_{\theta(k_0)} (\hat{y}(k) - \hat{y}(k_0)) \tag{7.7}$$

with Lagrange error bound

$$\begin{aligned}&\left|\left|\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{\theta(k_0)} - \left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{L}\right|\right|_2 \\ &\leq \frac{1}{2}\left|\left|\left(\frac{\partial^3 \mathcal{L}(\hat{y}(k_0),y)}{\partial^3 \hat{y}(k_0)}\right)_{\theta(k_0)}\right|\right|_2 \cdot ||(\hat{y}(k) - \hat{y}(k_0))||_2^2. \end{aligned} \tag{7.8}$$

[1] The following work was done together with Balázs Varga

Next, insert the linearized loss into eq. (2.5) and assume $\hat{y}(k) \approx \hat{y}_\vartheta(k)$ and $\hat{y}(k_0) = \hat{y}_\vartheta(k_0)$. We define the *control oriented training dynamics* as

$$\begin{aligned}
&\hat{y}_\vartheta(k+1) = \\
&\hat{y}_\vartheta(k) - \alpha\Theta(k_0)\left(\left(\frac{\partial\mathcal{L}(\hat{y}(k_0),y)}{\partial\hat{y}(k_0)}\right)_{\theta(k_0)} + \left(\frac{\partial^2\mathcal{L}(\hat{y}(k_0),y)}{\partial^2\hat{y}(k_0)}\right)_{\theta(k_0)}(\hat{y}_\vartheta(k) - \hat{y}(k_0))\right) = \\
&\left(I - \alpha\Theta(k_0)\left(\frac{\partial^2\mathcal{L}(\hat{y}(k_0),y)}{\partial^2\hat{y}(k_0)}\right)_{\theta(k_0)}\right)\hat{y}_\vartheta(k) \\
&+\alpha\Theta(k_0)\left(\frac{\partial^2\mathcal{L}(\hat{y}(k_0),y)}{\partial^2\hat{y}(k_0)}\right)_{\theta(k_0)}\hat{y}(k_0) - \alpha\Theta(k_0)\left(\frac{\partial\mathcal{L}(\hat{y}(k_0),y)}{\partial\hat{y}(k_0)}\right)_{\theta(k_0)}. \qquad (7.9)
\end{aligned}$$

Note that there is a bias term $\alpha\Theta(k_0)\left(\frac{\partial^2\mathcal{L}(\hat{y}(k_0),y)}{\partial^2\hat{y}(k_0)}\right)_{\theta(k_0)}\hat{y}(k_0) - \alpha\Theta(k_0)\left(\frac{\partial\mathcal{L}(\hat{y}(k_0),y)}{\partial\hat{y}(k_0)}\right)_{\theta(k_0)}$ that only offsets the dynamics. The cumulative error bound (based on eq. (7.5) and eq. (7.8) can be given as follows. Denote the left hand side of eq. (7.5) with E_y and eq. (7.8) with E_L. Consider eq. (7.8) and inject the linearization error of $\hat{y}(k)$ as

$$E_L(E_y) \leq \frac{1}{2}\left|\left|\left(\frac{\partial^3\mathcal{L}(\hat{y}(k_0),y)}{\partial^3\hat{y}(k_0)}\right)_{\theta(k_0)}\right|\right|_2 \cdot ||(\hat{y}_\vartheta(k) + E_y - \hat{y}(k_0))||_2^2. \qquad (7.10)$$

Then, the error bound for the control-oriented training dynamics can be given by

$$E \leq E_y - \alpha\Theta(k_0)E_L(E_y). \qquad (7.11)$$

In certain cases, when the loss is a quadratic function of the output (e.g., SSE, or MSE losses), the linearization error of the loss disappears.

7.5 Examples of equilibrium points

The following appendix discusses the conditions under which an equilibrium point may exist. The definition of an equilibrium point is a point where no change to the ANN output occurs, i.e $\hat{y}(k+1) = \hat{y}(k)$. In case of the global training dynamics $\hat{y}(k+1) = \hat{y}(k)$ can only occur if $\alpha\Theta(k)\left(\frac{\partial\mathcal{L}(\hat{y}(k),y)}{\partial\hat{y}(k)}\right)_{\theta(k)} = 0$. More precisely, there is an equilibrium point if any of the following conditions are fulfilled.

1. The most important case is when the loss is at a (local) minimum,

$\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{\theta(k)} = 0.$

2. The learning is frozen $\alpha = 0$.

3. The kernel is a null matrix $\Theta(k) = \underline{\underline{0}}$. However, it can only occur in some very specific cases, e.g., if $\frac{\partial \hat{y}(k,x_i)^T}{\partial \theta(k)}$ and $\frac{\partial \hat{y}(k,x_j)^T}{\partial \theta(k)}$ for all data combinations x_i, x_j.

4. A less trivial case is when $\Theta(k)\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{\theta(k)}$ is a zero vector while $\Theta(k) \neq \underline{\underline{0}}$, and $\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{\theta(k)} \neq 0$. I.e., the derivative of the loss $\left(\frac{\partial \mathcal{L}(\hat{y}(k),y)}{\partial \hat{y}(k)}\right)_{\theta(k)}$ is in the null space of the kernel.

7.6 Boundedness for common losses

The boundedness of some common loss functions is analyzed, assuming static target y. This work was mainly conducted by Balázs Varga, post-doc at Chalmers University.

- **Mean squared error (MSE) loss**: The MSE loss is given as $\mathcal{L}(\hat{y}(k), y) = \frac{1}{2rn_L}(\hat{y}_\vartheta(k) - y)^2$. Substituting the MSE loss in eq. (2.5) one gets

$$\hat{y}_\vartheta(k+1) = \hat{y}_\vartheta(k) - \frac{\alpha\Theta(k_0)}{rn_L}(\hat{y}_\vartheta(k) - y). \tag{7.12}$$

  This difference equation has an equilibrium point at a bounded y, which is proven in later sections. For the linear time-discrete ANN training dynamics under MSE loss

$$\hat{y}_\vartheta(k+1) = \left(I - \frac{\alpha\Theta(k_0)}{rn_L}\right)\hat{y}_\vartheta(k) + \frac{\alpha\Theta(k_0)}{rn_L}y. \tag{7.13}$$

  In eq. (7.13), trajectories of $\hat{y}_\vartheta(k)$ can be checked for boundedness by looking at the eigenvalues of the system matrix $\left(I - \frac{\alpha\Theta(k_0)}{rn_L}\right)$. The local training dynamics are internally exponentially bounded iff

$$|\lambda| < 1 \quad \forall \lambda \in eig\left(I - \frac{\alpha\Theta(k_0)}{rn_L}\right). \tag{7.14}$$

The proof for this can be found in [61].

- **Sum of Squared Error (SSE) loss**: The SSE loss is similar to the MSE loss without the normalization with rn_L, i.e., $\mathcal{L}(\hat{y}_\vartheta(k), y) = \frac{1}{2}(\hat{y}_\vartheta(k) - y)^2$. Therefore, following the same line of thought as for the MSE, if

$$|\lambda| < 1 \quad \forall \lambda \in eig(I - \alpha\Theta(k_0)), \tag{7.15}$$

then $\hat{y}_\vartheta(k)$ does not diverge from y. Since rn_L is a positive integer, the overbound for a non-divergent α with SSE loss is smaller than with MSE loss.

- **Mean absolute error (MAE) loss**: The mean absolute error loss is given as $\mathcal{L}(\hat{y}_\vartheta(k), y) = \frac{1}{rn_L}||\hat{y}_\vartheta(k) - y||_1$ and its derivative w.r.t. $\hat{y}_\vartheta(k)$ is

$$\frac{\partial \mathcal{L}(\hat{y}_\vartheta(k), y_i)}{\partial \hat{y}_\vartheta(k)} = \frac{1}{rn_L} \sum_{i=1}^{rn_L} \frac{\hat{y}_{\vartheta,i}(k) - y}{|\hat{y}_{\vartheta,i}(k) - y_i|} \tag{7.16}$$

for $\hat{y}_{\vartheta,i}(k) \neq y_i \forall i$. Index i denotes one element of the vector-valued outputs and labels. Outside of $\hat{y}_{\vartheta,i}(k) = y_i$, the derivative is -1 if $\hat{y}_{\vartheta,i}(k) < y_i$ and 1 if $\hat{y}_{\vartheta,i}(k) > y_i$. Therefore, the discrete learning dynamics with MAE loss can be written as

$$y_\vartheta(k+1) = y_\vartheta(k) - \frac{\alpha\Theta(k_0)}{rn_L} sgn(\hat{y}_\vartheta(k) - y). \tag{7.17}$$

Intuitively, this means the loss will uniformly converge to the $\frac{\alpha\Theta(k_0)}{rn_L}$ radius of y. The conditions for exponential internal boundedness are not fulfilled.

- **Cross entropy loss**: The cross entropy loss or log loss is used for classification, rather than regression tasks. It can be computed as $\mathcal{L}(\hat{y}_\vartheta(k), y) = -y^T log(\hat{y}_\vartheta(k))$. Then, the nonlinear difference-equation for the learning dynamics is

$$y_\vartheta(k+1) = y_\vartheta(k) + \alpha\Theta(k_0)\check{y}_\vartheta(k)y, \tag{7.18}$$

where $\check{y}_\vartheta(k)$ is a diagonal matrix $\in \mathbb{R}^{rn_L \times rn_L}$ of the element-wise inverses of $\hat{y}_\vartheta(k)$, assuming $\hat{y}_\vartheta(k)$ has no zero elements. Then, y is an equilibrium point for the difference equation if $\alpha\Theta(k_0)log(\check{y}_\vartheta(k)y$ is a

null vector. I.e., y is an equilibrium point if it is in the nullspace of the matrix $\alpha\Theta(k_0)\check{y}_\vartheta(k)$. A trivial solution to this if $y = 0$, and this is the only solution if the columns in $\alpha\Theta(k_0)\check{y}_\vartheta(k)$ are linearly independent. If they are linearly dependent, there are infinitely many equilibrium points. For more in-depth analysis a Lyapunov function is sought to give boundedness conditions for the cross entropy loss. In discrete-time, Lyapunov boundedness is fulfilled if $V(f(x)) - V(x) < 0$, where $V(x)$ is a Lyapunov function [60]. Let $V(x) = x^T x$ be a Lyapunov function. Then for eq. (7.18) the Lyapunov boundedness criteria is

$$\left(y_\vartheta(k) + \alpha\Theta(k_0)\check{y}_\vartheta(k)y\right)^T \left(y_\vartheta(k) + \alpha\Theta(k_0)\check{y}_\vartheta(k)y\right) - y_\vartheta(k)^T y_\vartheta(k) < 0 \tag{7.19}$$

which can be simplified to

$$\alpha^2 y^T \check{y}_\vartheta^T(k)\Theta^T(k_0)\Theta(k_0)\check{y}_\vartheta(k)y - 2\alpha y_\vartheta^T(k)\Theta(k_0)\check{y}_\vartheta(k)y < 0. \tag{7.20}$$

Although, this equation is easy to check whether it is fulfilled or not, a universal conclusion cannot be drawn for the global boundedness. On the other hand, the cross entropy loss is mainly used for classification tasks rather than regression where the target y and the output $y_\vartheta(k)$ are normalized, i.e., $y,\ y_\vartheta(k) \in (0,1) \subset \mathbb{R}^{rn_L}$. In such a case (in a local sense) α is always positive, y, and $y_\vartheta(k)$ are positive vectors, $\check{y}_\vartheta(k)$ is a diagonal matrix with positive elements. $\Theta(k_0)$ is symmetric and if the input is normalized, it is positive-definite too [43]. Then, for a sufficiently small α, Lyapunov boundedness is fulfilled.

From the above list it is obvious, that from a control-oriented perspective, the SSE and MSE losses are the most appropriate.

7.7 Boundedness of the global dynamics

[2] A criteria for the boundedness of the global training dynamics can be given based on the linearized dynamics. To this end, we subtract the Lagrange error (in appendix eq. (7.5)) from eq. (2.10) giving a less conservative bound for

[2] The following work was done together with Balázs Varga

the global training dynamics:

$$||\hat{y}(k)-\hat{y}_e||_2 \leq \gamma e^{-\lambda(k-k_0)}||\hat{y}_\vartheta(k_0)-\hat{y}_e||_2 - \max_{\theta(r)} \frac{1}{2}\left|\left|\left(\frac{\partial^2 \hat{y}(k)^T}{\partial^2 \theta(k)}\right)^T_{\theta(r)}\right|\right|_2 \cdot ||\vartheta^2(k)||_2. \tag{7.21}$$

The above expression suggests that the global training dynamics is not exponentially bounded given the Lagrange error is nonzero. On the other hand, it has important implications on the validity of the linearized training dynamics. Since the linearization error grows over time, a time instant $k_c > k_0$ can be found where exponential boundedness for the local training dynamics gets violated. I.e., if

$$\begin{aligned} ||\hat{y}_\vartheta(k_c) - \hat{y}_e||_2 > \quad & \gamma e^{-\lambda(k_c-k_0)}||\hat{y}_\vartheta(k_0) - \hat{y}_e||_2 \\ - \quad & \max_{\theta(r)} \frac{1}{2}\left|\left|\left(\frac{\partial^2 \hat{y}(k_c)^T}{\partial^2 \theta(k_c)}\right)^T_{\theta(r)}\right|\right|_2 \cdot ||\vartheta^2(k_c)||_2 \end{aligned} \tag{7.22}$$

we can explicitly say that the linear model is poor and must be recalculated.

7.8 The DARE equation

The following appendix describes the standard Discrete-time Algebraic Riccati Equation (DARE) [33] which is deployed in order to find the solution P to the quadratic infinite optimization problem in eq. (3.5). The proof that eq. (7.23) solves the cost given in eq. (3.5) is given in [33].

$$A = \begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix}, \; B = \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix}, \; P = A^T PA + Q - A^T PB\left(R + B^T PB\right)^{-1}$$

7.9 Initialization of the ANN

There are three common ways to initialize neural networks of infinite width to derive fixed kernels.

- Standard initialization. The weight for each neuron are given as $\mathcal{N}(0, \frac{\sigma_w^2}{sn_l})$ ($\mathcal{N}(0, \frac{\sigma_w^2}{sn_l n_m})$ for convolutional layers), and biases are $\mathcal{N}(0, \sigma_b^2)$ with σ_w, and σ_b being initialization variances, n_l is the width of each layer, n_m

is the number of spatial positions in the convolution kernel, and s is a width-scaling factor that goes to ∞ for infinite width networks. The main issue with this initialization is that in the infinie width-limit the entries of the NTK diverge.

- NTK initialization, proposed by [43]. In this case, weights and biases are initialized with normalized gaussian distributions $\mathcal{N}(0, 1)$. The weights are multiplied with $\frac{\sigma_w}{\sqrt{sn_l}}$, ($\frac{\sigma_w}{\sqrt{sn_l n_m}}$ for convolutional layers), and the biases are scaled with σ_b. That is to make the NTK values converge.

- Improved standard initialization [68]. The difference between the standard and the improved version is that the width-scaling factor is pulled out from the normal distribution, i.e. $\frac{1}{\sqrt{s}}\mathcal{N}(0, \frac{\sigma_w^2}{n_l})$ and $\frac{1}{\sqrt{s}}\mathcal{N}(0, \frac{\sigma_w^2}{n_l n_m})$.

The initializations are summarized in Table **??**.

According to [68], [69], infinite width networks with various architectures achieve similar error regardless of initialization. I.e., if they converge, the final value will be similar in output space, regardless of initialization. On the other hand, it is not the case in parameter space; the NTK will take different final numerical values depending on initialization. This means it will traverse a different trajectory during learning since the eigenvalues of the NTK will influence the learning dynamics.

All experiments, both regression and classification, implement initialization 2 as recommended by [43].

References

[1] M. Tripathi, "Analysis of convolutional neural network based image classification techniques," *Journal of Innovative Image Processing*, vol. 3, no. 2, pp. 100–117, 2021.

[2] R. Olu-Ajayi, H. Alaka, I. Sulaimon, F. Sunmola, and S. Ajayi, "Building energy consumption prediction for residential buildings using deep learning and other machine learning techniques," *Journal of Building Engineering*, vol. 45, no. 103406, 2022.

[3] OpenAI, "Gpt-4 technical report," *ArXiv*, vol. 2303.08774, 2020.

[4] D. Baptista and F. Morgado-Dias, "A survey of artificial neural network training tools," *Neural Computing and Applications*, vol. 23, pp. 609–615, 2013.

[5] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, 2012.

[6] F. Martínez, F. Charte, M. P. Frías, and A. M. Martínez-Rodríguez, "Strategies for time series forecasting with generalized regression neural networks," *Neurocomputing*, vol. 491, pp. 509–521, 2022.

[7] H. Salem, A. Kabeel, E. M. El-Said, and O. M. Elzeki, "Predictive modelling for solar power-driven hybrid desalination system using artificial neural network regression with adam optimization," *Desalination*, vol. 522, no. 115411, 2022.

[8] V. W. Tam, A. Butera, K. N. Le, L. C. D. Silva, and A. C. Evangelista, "A prediction model for compressive strength of co2 concrete using regression analysis and artificial neural networks," *Construction and Building Materials*, vol. 324, no. 126689, 2022.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proceedings of the 25th International Conference on Advances in Neural Information Processing Systems (NeurIPS'25 2012)*,

[10] L. Alzubaidi, J. Zhang, A. Humaidi, and et al., "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, p. 53, 2021.

[11] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611–629, 2018.

[12] M. P. Aneesa, N. Saabina, and K. Meera, "Face recognition using cnn: A systematic review," *International Journal of Engineering Research & Technology*, vol. 11, 2022.

[13] A. Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," *Proceedings of the 26th International Conference of Advances in Neural Information Processing Systems (NeurIPS'26 2013)*,

[14] J. Lu, R. Xiong, J. Tian, *et al.*, "Battery degradation prediction against uncertain future conditions with recurrent neural network enabled deep learning," *Energy Storage Materials*, vol. 50, pp. 139–151, 2022.

[15] J. Zhu, Q. Jiang, Y. Shen, C. Qian, F. Xu, and Q. Zhu, "Application of recurrent neural network to mechanical fault diagnosis: A review," *Journal of Mechanical Science and Technology*, vol. 36, pp. 527–542, 2022.

[16] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," 2011, pp. 5528–5531.

[17] Y. Su and C.-C. J. Kuo, "On extended long short-term memory and dependent bidirectional recurrent neural network," *Neurocomputing*, vol. 356, pp. 151–161, 2019.

[18] A. Ribeiro, K. Tiels, L. A. Aguirre, and T. Schön, "Beyond exploding and vanishing gradients: Analysing rnn training using attractors and smoothness," ser. Proceedings of Machine Learning Research, vol. 108, Aug. 2020, pp. 2370–2380.

[19] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 100004, 2021.

[20] S. Shahriar, "Gan computers generate arts? a survey on visual arts, music, and literary text generation using generative adversarial network," *Displays*, vol. 73, no. 102237, 2022.

[21] Y. Chen, X. Yang, Z. Wei, *et al.*, "Generative adversarial networks in medical image augmentation: A review," *Computers in Biology and Medicine*, vol. 144, no. 105382, 2022.

[22] D. Saxena and J. Cao, "Generative adversarial networks (gans): Challenges, solutions, and future directions," vol. 54, no. 3, 2021.

[23] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Proceedings of the 30nd International Conference on Advances in Neural Information Processing Systems (NeurIPS'30 2017)*,

[24] J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," vol. 1810.04805, 2018.

[25] C. Xu and J. McAuley, "A survey on model compression and acceleration for pretrained language models," 2022.

[26] Z. S. Kadhim, H. S. Abdullah., and K. I. Ghathwan, "Artificial neural network hyperparameters optimization: A survey," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 18, no. 15, pp. 59–87, 2022.

[27] R. F. Prudencio, M. R. O. A. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–, 2023.

[28] N. Le, V. S. Rathour, K. Yamazaki, K. Luu, and M. Savvides, "Deep reinforcement learning in computer vision: A comprehensive survey," *Artificial Intelligence Review*, vol. 55, pp. 2733–2819, 2022.

[29] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," vol. 55, no. 1, 2023.

[30] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: An overview," *Proceedings of SAI Intelligent Systems Conference (IntelliSys)*, pp. 426–440, 2016.

[31] L. Luo, N. Zhao, Y. Zhu, and Y. Sun, "A guiding dqn algorithm for automated guided vehicle pathfinding problem of robotic mobile fulfillment systems," *Computers & Industrial Engineering*, vol. 178, no. 109112, 2023.

[32] R. R. Faria, B. D. O. Capron, A. R. Secchi, and M. B. de Souza, "Where reinforcement learning meets process control: Review and guidelines," *Processes*, vol. 10, no. 11, 2022.

[33] H. Kwakernaak and R. Sivan, "Linear optimal control systems," *John Wiley and Sons, New York*, 1972.

[34] A. E. Bryson, "Applied optimal control," *Hemisphere Publishing Corporation, California*, 1975.

[35] Y. Zhang, S. Li, and X. Zhou, "A survey of near-optimal control of nonlinear systems," *Deep Reinforcement Learning with Guaranteed Performance: A Lyapunov-Based Approach*, pp. 1–20, 2020.

[36] M. Maiworm, D. Limon, and R. Findeisen, "Online learning-based model predictive control with gaussian process models and stability guarantees," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8785–8812, 2021.

[37] A. Dolgui, D. Ivanov, S. P. Sethi, and B. Sokolov, "Scheduling in production, supply chain and industry 4.0 systems by optimal control: Fundamentals, state-of-the-art and applications," *International Journal of Production Research*, vol. 57, no. 2, pp. 441–432, 2019.

[38] F. Hagebring and B. Lennartson, "Time-optimal control of large-scale systems of systems using compositional optimization," *Discrete Event Dynamic Systems*, vol. 29, pp. 411–443, 2019.

[39] Z. Zhou and H. Xu, "A novel mean-field-game-type optimal control for very large-scale multiagent systems," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 5197–5208, 2022.

[40] H. Wan, H. R. Karimi, X. Luan, and F. Liu, "Model-free self-triggered control based on deep reinforcement learning for unknown nonlinear systems," *International Journal of Robust and Nonlinear Control*, vol. 33, no. 3, pp. 2238–2250, 2023.

[41] M. Seeger, "Guassian processes for machine learning," *International Journal of Neural Systems*, vol. 14, pp. 69–106, 2004.

[42] Z. Marvi and B. Kiumarsi, "Safe reinforcement learning: A control barrier function optimization approach," *International Journal of Robust and Nonlinear Control*, vol. 33, no. 3, pp. 2238–2250, 2023.

[43] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Proceedings of the 31st Internation Conference on Advances Neural Information Processing Systems (NeurIPS'31 2018)*,

[44] J. Lee, L. Xiao, S. S. Schoenholz, *et al.*, "Wide neural networks of any depth evolve as linear models under gradient descent," *Proceedings of the 32nd International Conference on Advances Neural Information Processing Systems (NeurIPS'32 2019)*,

[45] J. Huang and H. Yau, "Dynamics of deep neural networks and neural tangent hierarchy," *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, pp. 4542–4551, 2020.

[46] S. Alemohammad, Z. Wang, R. Balestriero, and R. Baraniuk, "The recurrent neural tangent kernel," *ArXiv*, vol. 2006.10246, 2020.

[47] J. Hron, Y. Bahri, J. Sohl-Dickstein, and R. Novak, "Infinite attention: Nngp and ntk for deep attention networks," *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, pp. 4376–4386, 2020.

[48] G. Yang, "Tensor programs iib: Architectural universality of neural tangent kernel training dynamics," *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, pp. 11 762–11 772, 2021.

[49] J. Franceschi, E. Bézenac, I. Ayed, M. Chen, S. Lamprier, and P. Gallinari, "A neural tangent kernel perspective of gans," *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, pp. 6660–6704, 2021.

[50] A. Zandieh, I. Han, H. Avron, N. Shoham, C. Kim, and J. Shin, "Scaling neural tangent kernels via sketching and random features," *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'34 2021)*, pp. 1062–1073,

[51] R. Novak, J. Sohl-Dickstein, and S. S. Schoenholz, "Fast finite width neural tangent kernel," *ArXiv*, vol. 2206.08720, 2022.

[52] N. Ailon and S. Shit, "Efficient ntk using dimensionality reduction," 2022.

[53] Y. Wang, D. Li, and R. Sun, "Ntk-sap: Improving neural network pruning by aligning training dynamics," 2023.

[54] P. Ju, X. Lin, and N. Shroff, "On the generalization power of the overfitted three-layer neural tangent kernel model," pp. 26 135–26 146.

[55] S. Wanga, X. Yua, and P. Perdikarisb, "When and why pinns fail to train: A neural tangent kernel perspective," *Journal of Computational Physics*, vol. 449, no. 110768, 2022.

[56] B. Varga, B. Kulcsár, and M. H. Chehreghani, "Constrained policy gradient method for safe and fast reinforcement learning: A neural tangent kernel based approach," *ArXiv*, vol. 2006.07678v2, 2021.

[57] ——, "Deep q-learning: A robust control approach," *International Journal of Robust and Nonlinear Control*, vol. 33, no. 1, pp. 526–544, 2023.

[58] Z. Chen, Y. Cao, Q. Gu, and T. Zhang, "A generalized neural tangent kernel analysis for two-layer neural networks," *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'33 2020)*, pp. 13 363–13 373,

[59] J. C. Ye, "Geometry of deep learning: A signal processing perspective," *Springer, Singapore*, 2022.

[60] H. K. Khalil, "Nonlinear systems (3rd edition)," *Patience Hall, Michigan*, 2002.

[61] W. J. Rugh, "Linear system theory (2nd edition)," *Patience Hall, Michigan*, 1995.

[62] Q. Tong, G. Liang, and J. Bi, "Calibrating the adaptive learning rate to improve convergence of adam," *Neurocomputing*, vol. 481, pp. 333–356, 2022.

[63] A. Isidori, "Nonlinear control systems (3rd edition)," *Springer, New York*, 1995.

[64] J. Maidens and M. Arcak, "Reachability analysis of nonlinear systems using matrix measures," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 265–270, 2015.

[65] V. Szolnoky, V. Andersson, B. Kulcsar, and R. Jörnsten, "On the interpretability of regularisation for neural networks through model gradient similarity," *Proceedings of the 36th International Conference of Advances in Neural Information Processing Systems (NeurIPS'36 2022)*,

[66] D. D. Cock, "Ames house pricing dataset," *https://www.kaggle.com/c/house-prices-advanced-regression-techniques*, 2011.

[67] Microsoft-Research, "Cats vs. dogs," $https://www.microsoft.com/en-us/download/details.aspx?id = 54765$, 2022.

[68] J. Sohl-Dickstein, R. Novak, S. S. Schoenholz, and J. Lee, "On the infinite width limit of neural networks with a standard parameterization," *ArXiv*, vol. abs/2001.07301, 2020.

[69] D. Park, J. Sohl-Dickstein, Q. Le, and S. Smith, "The effect of network width on stochastic gradient descent and generalization: An empirical study," *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 5042–5051, 2019.