Controlled Descent Training

(article starts on next page)

RESEARCH ARTICLE

WILEY

Controlled descent training

Viktor Andersson[1,2] ｜ **Balázs Varga**[2] ｜ **Vincent Szolnoky**[1,2] ｜ **Andreas Syrén**[1] ｜
Rebecka Jörnsten[2] ｜ **Balázs Kulcsár**[2]

[1]R&D, Centiro AB, Borås, Sweden

[2]Chalmers University of Technology, Gothenburg, Sweden

Correspondence
Viktor Andersson, R&D, Centiro AB, Borås, Sweden.
Email: vikta@chalmers.se
Balázs Kulcsár, Chalmers University of Technology, Gothenburg, Sweden.
Email: kulcsar@chalmers.se

Summary

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

convergent learning, gradient decent training, label augmentation, label selection, neural Tangent Kernel, optimal labels

1　｜　INTRODUCTION

Machine learning (ML) and Artificial Intelligence (AI) are able to model complex and highly non-linear input-output relationships. ML is very powerful but often lacks the guarantees and predictability required for system and control theory applications.

Deep Artificial Neural Networks (ANNs) are particularly useful tools in machine learning that are commonly trained with gradient descent methods (GD). ANN architectures have made strides in recent years in solving complicated tasks, like image recognition,[1] natural language processing,[2] artificial image generation,[3] and other engineering tasks[4,5] where classical statistical models might struggle. Problems in the domain of Control Theory have also had a surge in ANN and ML related works.[6–8] The main criticisms of ANNs are their unpredictable training behavior, low output interpretability, and highly hyper-parameter dependent performance.

In this work we aim to improve predictability of ANN learning behavior, with guarantees of convergence and reduced hyper-parameter search space. This would allow 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.

Recent insights into the learning behavior of ANNs come from the study of ANNs with infinite width. Jacot et al.[9] 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[10] demonstrates the empirical region of validity of the Taylor linearization. Moreover, the higher-order terms have been studied.[11] Tangential work studied the infinite NTK for many different architectures like CNNs,[10] RNNs[12] and transformers.[13] Yang et al.[14] 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 Generative Adversarial Networks (GANs) and other encoder-decoder ANN structures are revealed by the NTK framework.[15] Other works study the NTK for collegial ensemble methods.[16] The major criticism of the NTK is its computational complexity. An NTK approximation method for improved computation speed of infinite NTKs has been introduced mitigating this issue.[17] Moreover, both computational times and memory required for finite NTK calculations can be reduced by exploiting the Jacobian symmetric structure.[18] This allows the NTK to be used for real-time training analysis.

The NTK has been used to explain aspects of ANN training dynamics already. For example, convergence properties of the NTK have been used for interpolation of phase transitions.[19] Other work finds bounds on the smallest eigenvalue of the NTK is associated with memorization.[20] The NTK has also been used to compare the informational discrepancy between test and training data sets.[21] As demonstrated, the NTK is highly relevant to assess the training behavior of many ANNs architectures. The NTK description unlocks a more interpretable and dependable perspective on ANN training behavior.

Optimal control has well-established analytic proofs of stability and optimality under conditions of reachability and open-loop stability.[22] Optimal control has made progress in recent years,[23] improving the reliability for high-dimensional and large scale systems.[24,25] Our main contribution is bringing these optimal control methods into the world of ML and ANNs.

In this paper, we define two categories of methods influencing the ANN training dynamic using the NTK. The first category is *indirect* control through NTK reference and model matching. An example of an implicit method is using the eigenvalues of the NTK to improve the convergence rate for physically informed neural networks (PINNs).[26] The NTK has been used to complement existing policy gradient methods[27] and robust Q-learning.[28] The latter papers implement cascade optimal control methods to enhance the convergence and stability of reinforcement learning methods, implicitly.

In this paper, we follow a *direct* approach to influence the ANN training dynamics using optimal control. Firstly, we introduce the concepts of stability and reachability for a given ANN, based purely on the NTK. This allows for directed and reduced hyper-parameter 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 hyper-parameter choices, and guarantees otherwise missing for ANNs without compromising on performance. We evaluate the performance of CDT compared to GD with different fully connected and convolutional ANN architectures. Both training algorithms are benchmarked on datasets selected from classic datasets (regression and classification). We demonstrate and report on the accuracy and numerical results of CDT algorithm with clear indications of its limitation.

Hence, the contribution of the paper is threefold. First, we revisit the ANN training dynamics when using the GD method. We rely on and analyze the Neural Tangent Kernel governed global training dynamics from a novel perspective: systems and control theory. Therefore, we introduce learning equilibria, global and local learning dynamics. Learning goals are assessed by means of stability theory of an equilibrium.

Second, we synthesize optimal controllers to accelerate and optimize the convergence to equilibria by defining a synthetic control input vector called dynamic labels. Unlike existing works, that use kernels to primarily analyze already trained network parameters, we do not consider kernels as a sensitivity metric to asses training quality. Instead of, we use the synthetic labels to optimally reach target equilibria. We achieve it by complementing the GD method with a cascade loop. Controlled Decent Learning guarantees local convergence and optimality via traditional $\mathcal{H}_2$ reference tracking state feedback control law.

Third, we contrast the newly developed CDT method with traditional solutions both in supervised regression and classification environments.

The highlights of the contribution of the paper hence can be seen as it follows.

1. The stability concept is introduced for ANN training dynamics relying on global and local kernel dominated descriptions.
2. The novel method, CDT, hints to inject synthetic labels to change the traditional GD method. We conclude the importance of dynamic labels via reachability analysis.
3. The novel method proposes to inject synthetic-controlled labels to boost the learning with traditional GD method.
4. By optimally choosing the dynamic labels via the proposed Controlled Decent Learning method, we guarantee local $\mathcal{H}_2$ optimal convergence.
5. Numerical comparison between the novel algorithm and traditional gradient descent using a convex loss is demonstrated.
6. The paper lays a ground-work for combined control theory and machine learning research.

The layout of the paper is as follows. In the Preliminaries (Section 2), the NTK and the uncontrolled ANN training dynamics (borrowed from Jacot et al.[9]) are introduced and expanded on to describe the label augmented training dynamics. Section 3 defines stability for the unargumented and reachability for the augmented training dynamics. In Section 4 the CDT algorithm is introduced. Section 5 demonstrates the CDT algorithm performance compared with gradient descent for different architectures and ML problems. We conclude the paper by highlighting future research directions. Additionally, the paper features multiple appendices elaborating on some of the assumptions and findings of the paper.

2 | PRELIMINARIES

In this section, we briefly show how the Neural Tangent Kernel describes the training dynamics of an ANN.

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 parametrization 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.

2.1 | Neural tangent kernel

The NTK is adopted to describe the ANN evolution in function space during gradient descent training.[9] As the ANN width increases, the NTK evolution rate stagnates. For finite-width ANNs the NTK is time-varying resulting in a nonlinear ODE.

This latter NTK is referred to as the *empirical tangent kernel*. The indirectly time-varying NTK ODE can be approximated as a time-invariant system by Taylor linearization around the initial parameters.[10] 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

$$\Theta_{i,j}(k) = \left(\frac{\partial \hat{y}(\theta(k), x_i)}{\partial \theta(k)} \frac{\partial \hat{y}(\theta(k), x_j)}{\partial \theta(k)}^T \right) \in \mathbb{R}^{n_L \times n_L} \tag{1}$$

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

$$\Theta(k) = \begin{bmatrix} \Theta_{0,0}(k) & \Theta_{1,0}(k) & \dots & \Theta_{r,0}(k) \\ \Theta_{0,1}(k) & \Theta_{1,1}(k) & & \vdots \\ \vdots & & \ddots & \vdots \\ \Theta_{0,r}(k) & \dots & \dots & \Theta_{r,r}(k) \end{bmatrix} \in \mathbb{R}^{r n_L \times r n_L} \tag{2}$$

The above mentioned empirical Neural Tangent Kernel $\Theta(k)$ in Equation (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).[9] 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.[29] There exist additional conditions guaranteeing its definiteness.[30]

2.2 | Local and global ANN training dynamics

In this section, with the help of previous works,[9,10] 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

$$\theta(k+1) = \theta(k) - \alpha \frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \theta(k)} = \theta(k) - \alpha \frac{\partial \hat{y}(k)^T}{\partial \theta(k)} \frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} \tag{3}$$

$$\hat{y}(k+1) = \hat{y}(k) - \alpha \Theta(k) \frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} = f_\theta(\hat{y}(k)). \tag{4}$$

Equation (4) captures the evolution of the global training dynamics as a nonlinear time-discrete Ordinary Difference Equation (ODE).

As can be seen in Equation (4) the symmetric empirical kernel $\Theta(k)$ has a central role in describing the training behaviour.

Proposition 1 in Appendix A demonstrates[31] 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 Equation (4) (see Appendix B 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

$$\hat{y}_\vartheta(k+1) = \hat{y}_\vartheta(k) - \alpha \Theta(k_0) \frac{\partial \mathcal{L}(\hat{y}_\vartheta(k), y)}{\partial \hat{y}_\vartheta(k)} = f_\vartheta(\hat{y}_\vartheta(k)). \tag{5}$$

A bound on the error between the local and global dynamics can be found using the Lagrangian error bound (see Appendix C). 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 D).

2.3 | Controlled ANN training dynamics with label augmentation

As mentioned previously, one of the main contributions of the paper is to explicitly control the NTK training dynamics. As such, we introduce a dynamic label augmentation method, that is, inject fictitious, time dependent labels by $y_u(k)$ as

$$\overline{y}(k) = y + y_u(k) \tag{6}$$

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

$$\hat{y}(k+1) = \hat{y}(k) - \alpha \Theta(\theta(k)) \frac{\partial \mathcal{L}(\hat{y}(k), \overline{y}(k))}{\partial \hat{y}(k)}. \tag{7}$$

In Equation (7), the injected labels purports to give a new degree of freedom to influence the local and global training behavior. In Section 4 an optimal way to select the fictitious labels $y_u(k)$ is demonstrated.

3 | 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.

3.1 | Boundedness of the training dynamics

We relate boundedness of the network output via Equation (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,

$$\hat{y}(k + 1) = \hat{y}(k) = \hat{y}_e. \tag{8}$$

Furthermore, it follows from the dynamics in Equation (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 E. The formal stability definition of Equation (4) can be captured by the following definition.

Definition 2 (Uniform internal stability[32]). 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

$$||\hat{y}(k) - \hat{y}_e||_2 \le \gamma ||\hat{y}(k_0) - \hat{y}_e||_2, \qquad k \ge k_0 \tag{9}$$

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[32]). The discrete-time ANN training dynamics in Equation (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 $0 < \kappa \le 1$ such that for all k_0 and y_0

$$||\hat{y}(k) - \hat{y}_e||_2 \le \gamma \kappa^{k-k_0} ||\hat{y}(k_0) - \hat{y}_e||_2, \quad k \ge k_0. \tag{10}$$

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 Equation (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

$$|\lambda| \le 1, \quad det(\lambda I - (I - \alpha\Theta(k_0))) = 0 \quad \forall \lambda \tag{11}$$

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 Equation (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 F a concise and loss function dependent derivation of stability analysis is provided for certain common loss functions.

Remark 1 (Learning rate adaptation). Finally, some ANN training algorithms[33] 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 C can be propagated through Equation (10) to deduce global stability from the local dynamics. This is further discussed in Appendix G.

3.2 | Reachability

Reachability is a property of the label augmented training dynamics given in Equation (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 Equation (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 Equation (7) can be verified using difference-geometric[34] or set theoretic algorithms.[35] 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.[32] In such cases, we borrow the Popov-Belman-Hautus (PBH) test given by,

$$rank\left[zI - (I - \alpha\Theta(k_0))\alpha\Theta(k_0)\right] = rn_L, \quad \forall z \in \mathbb{C}. \tag{12}$$

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 Equation 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 like Model Gradient Similarity.[36]

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.

4 | CONTROLLED DESCENT TRAINING–LOCALLY OPTIMAL CONTROL OF ANN TRAINING DYNAMICS

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 1, the schematics of the closed-loop and controlled label augmentation for an network trained with MSE is depicted.

In order to find K (in Figure 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 augmented training dynamics is stabilizable (or reachable).
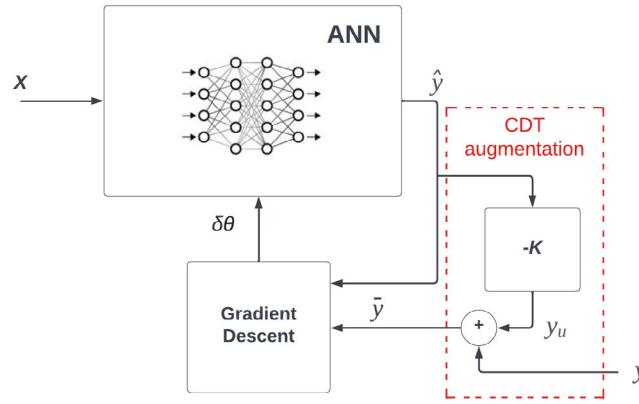
FIGURE 1 Schematic of controlled decent training. The CDT augmentation block calculates the label augment y_u dynamically from the ANN output $\hat{y}$ using the controller feedback matrix K. The new dynamically augmented label $\bar{y} = y + y_u$ is fed to GD rather than the static label y.

The following notation is introduced,

$$\tilde{y}(k) = \begin{bmatrix} \hat{y}_\vartheta(k) \\ 1 \end{bmatrix}, \tag{13}$$

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

$$\min_{y_u} \frac{1}{2} \sum_{i=k_0}^{\infty} \tilde{y}(i)^T \tilde{Q} \tilde{y}(i) + y_u(i)^T R y_u(i) \tag{14}$$

$$s.t. \ \tilde{y}(k+1) = \begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix} \tilde{y}(k) + \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix} y_u(k) \tag{15}$$

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 precisely,

$$\tilde{Q} = \begin{bmatrix} Q & -Qy \\ -y^T Q & y^T Qy \end{bmatrix}. \tag{16}$$

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 Equation (15) 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 Equation (14) describes a generic energy approach to label augment selection where the weighting matrices Q, R shape their relative importance. Finally, the first term in Equation (14) penalizes the deviation from the static targets.

The locally stabilizing and optimal solution to Equations (14)–(16,17) can be found by using the Discrete-time Algebraic Riccati equations (DARE)* (see Appendix H for solution derivation). If the stationary and extremal solution to DARE is P then the optimal feedback gain can be written as

$$K = \left(R + \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix}^T P \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix} \right)^{-1} \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix}^T P \begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix} \tag{17}$$

$$y_u(k) = -K\tilde{y}(k) \tag{18}$$

$$\bar{y}(k) = y_u(k) + y = -K\tilde{y}(k) + y \tag{19}$$

The closed and CDT controlled loop becomes

$$\begin{bmatrix} \hat{y}_\vartheta(k+1) \\ 1 \end{bmatrix} = \left(\begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix} - \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix} K \right) \begin{bmatrix} \hat{y}_\vartheta(k) \\ 1 \end{bmatrix}. \tag{20}$$

The constant feedback gain matrix K maps the ANN output $\tilde{y}$ to *target augments* $y_u(k)$ such that it minimizes the cost in Equation (14,17) 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, the computed controller gain K is static and it can be calculated prior the training sequence. It requires using the initial NTK $\Theta(k_0)$ only. Note, that other control schemes may be used to implement time-varying controller gain $K(k)$ to account for time or parameter dependent NTKs.

In the local dynamical sense, the linear difference equation in Equation (20) 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 guaranteed margins.[22] 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.

Remark 8 (Locally vanishing dynamic labels). The local and asymptotic stability properties of the $\mathcal{H}_2$ state feedback controller guarantees that no bias will be added to the static labels when converged. In this sense, the dynamic label augments locally vanish and act as an error energy optimal booster to the original GD method.[32] Since the state feedback $\mathcal{H}_2$ control method has guaranteed robust stability margins, CDT works well with imperfect linearized NTK cases.

4.1 | The CDT algorithm

In previous sections, the concepts of reachability and stability were introduced for ANNs and their implications on hyper-parameter 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

1:    Calculate $\Theta_0 \leftarrow$ Equation (2)   ▷ Calculate kernel at ANN initialization
2:    Check stability with Equation (11)   ▷ (Stability)
3:    Check reachability with Equation (12)   ▷ (Reachability)
4:    Calculate $K \leftarrow$ Equation (17)   ▷ Calculate control feedback using DARE
ANN Training:
for $k \in 1, 2, \ldots,$ *training steps* **do**
   1:    Calculate $\bar{y}(k) \leftarrow y - K\tilde{y}(k)$ ▷ Calculate label augments based on feedback error
   2:    Update parameters $\theta(k+1) \leftarrow \theta(k) - \delta\mathcal{L}(\hat{y}(k), \bar{y}(k))/\delta\theta(k)$ ▷ Update parameters according to GD
end for

---

5 | EXPERIMENTS

In this section, 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[37] with a single-target fully connected ANN and MSE loss. The second example is a binary image classification problem using ALEXNet[38] 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

$$\alpha_k = \frac{1}{1 + 0.01k}\alpha \tag{21}$$

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 Equation (17) are chosen as scaled diagonal matrices,

$$Q = I_1 \tag{22}$$
$$R = pI_2 \tag{23}$$

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 $p^\dagger$. 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 paper 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.

5.1 | 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 the dataset description.[37] 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.[‡]

5.1.1 | Architecture

For the regression experiment 3 ANN architectures similar to the model description used in previous works[10] is used with initializations given in Appendix I. That is a fully connected feed-forward neural network setup is used according to,

$$\begin{cases} h^{l+1} = z^l W^{l+1} + b^{l+1} \\ z^{l+1} = \phi^l\left(h^{l+1}\right) \end{cases} \qquad \begin{cases} W_{i,j}^l = \frac{\sigma_w}{\sqrt{n_l}} w_{i,j}^l \\ b_j^l = \sigma_b \beta_j^l \end{cases} \tag{24}$$

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 $j = 1, \ldots, n_{l+1}$. 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 $\mathcal{N}\left(0, \frac{\sigma_w^2}{n_l}\right)$ and $\mathcal{N}\left(0, \sigma_b^2\right)$. We define a parameter vector as

$$\theta = \begin{bmatrix} vec\left(W^L\right) & b^L & \ldots & vec(W^1) & b^1 \end{bmatrix}^T \tag{25}$$

where vec means concatenated vector form of the matrix. $\hat{y}(\theta, x) : \mathbb{R}^{n_0 \times r} \rightarrow \mathbb{R}^r$ 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 K1. 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.

5.1.2 | 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 Appendix K Tables K2–K4. In the aforementioned tables[§], α is the learning rate. Final validation loss is the average model performance 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. Figure 2A–C 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 Figure 2A–C are shown in log_{10} scale. Only learning rates where both CDT and GD converge for all initializations are shown in the figures. The MSE losses for each architecture and learning rate can be seen in Figures J1A–J3D in Appendix J.

As can be seen in Tables K2–K4, 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. This is highlighted further by the Figures J4A–J6D in Appendix J. In those Figures, we illustrate the kernel density estimated distributions over the some selected norm of the ANN output error $||\hat{y} - y||_2$ for each and every proposed architecture. 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 K2 that architecture 1 trained with CDT does not diverge for any initialization at the highest learning rate $\alpha = 1.000$ but converges further away from the true labels than at 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.

Regarding observed global convergence it can be seen in Tables K2–K4 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 2A–C 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 J1B–J3D in Appendix J highlight this further.
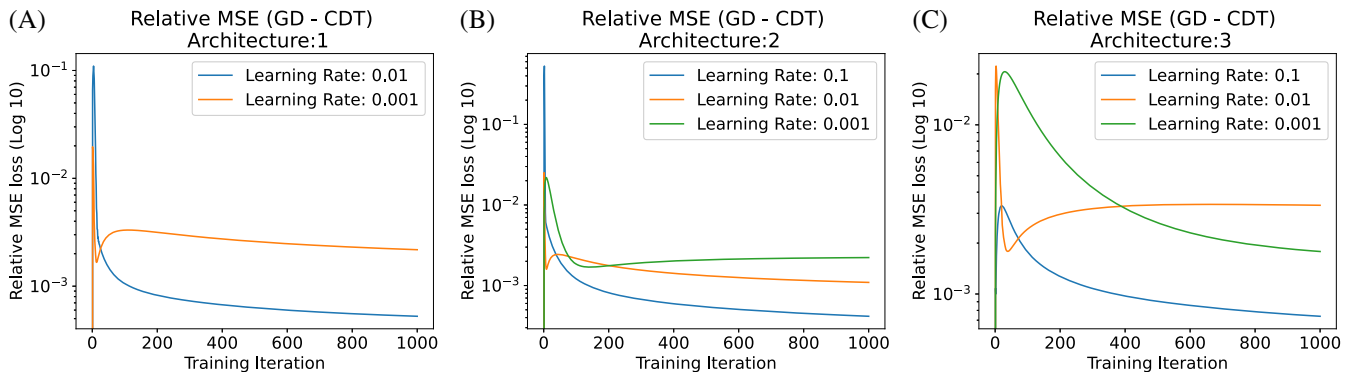

FIGURE 2 Average relative difference between GD and CDTs ($\mathcal{L}_{GD} - \mathcal{L}_{CDT}$) MSE validation loss for all learning-rates where both training methods converge for each of the three regression architectures. CDT has a lower validation loss during the entirety of training for all learning rates and architectures. (A) Architecture 1. (B) Architecture 2. (C) Architecture 3.

Figure 3 shows the ANN output evolution for three randomly selected samples and a single initialization pattern with both of the training methods (static-dynamic labeling). Note that in Figure 3A 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 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.[9] 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. However, for very high learning-rates the linearization validity may decay very quickly resulting in non-vanishing label augments y_u. This can be seen in Figure J7A–C. For such cases increasing the weight penalty ρ allows for increased robustness margins which results in improved training convergence and vanishing label augments. Figure 4 shows the frequency domain interpretation of the CDT versus GD. In this view, the real-value fast Fourier transform (RFFT) of the ANN output evolution relative the true label y in a two-norm sense $|\hat{y}(k) - y|_2$ is plotted for both the CDT and the GD. As can be seen from Figure 4, the label augments appear to enforce a band-pass filtering behaviour aiming at minimizing the magnitude overall the frequency range. At certain frequencies the GD method provides smaller errors, but the CDT method via the dynamic labels and the $\mathcal{H}_2$ feedback policy lowers the error related energy. Figure 4 corroborates the observation that CDT returns smaller errors (convergence to the true labels) at nearly all frequencies. Moreover, for the three randomly selected samples shown in Figure 4 a consistent increase
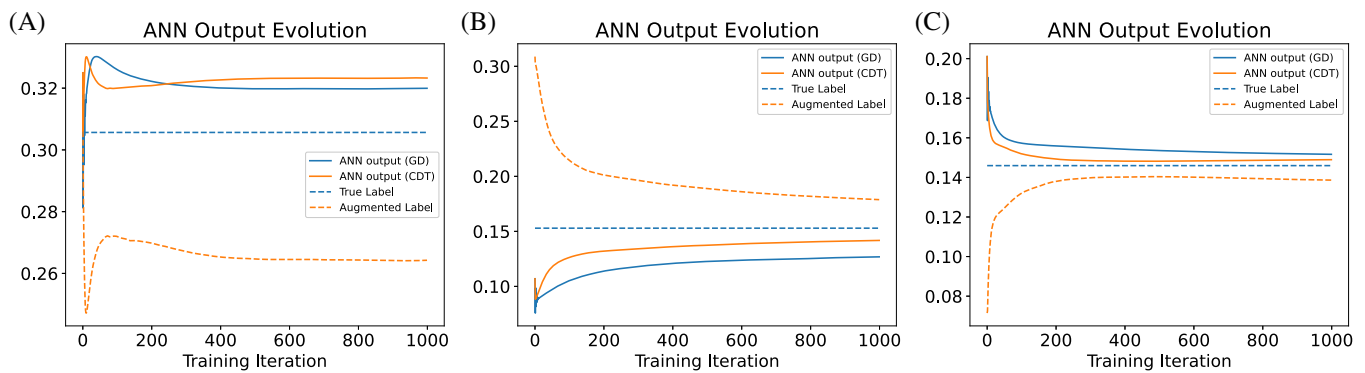

FIGURE 3 ANN outputs $\hat{y}$, true label y and augmented label $\bar{y}$ evolution for random samples when training under CDT and GD. First initialization of architecture 1 with learning rate 0.01. CDT appear to converge closer to true label y for most, but not all data samples in the training batch. (A) Sample 1. (B) Sample 2. (C) Sample 3.
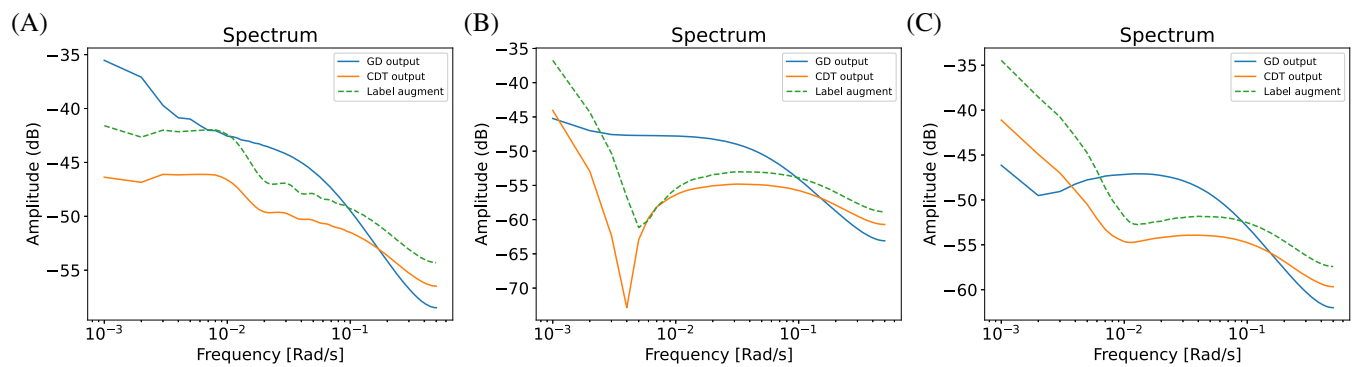

FIGURE 4 The real-valued fast Fourier transform (rfft) of ANN output $\hat{y}$ distance from label y ($|\hat{y} - y|_2$) when training with GD, CDT and the augmented label $\bar{y}$ evolution for random samples. First initialization of architecture 1 with learning rate 0.01. CDT for this architecture appear to band-pass filter out mid-range frequencies. (A) Sample 1. Zero-frequency gain: GD: −30.19 dB CDT: −29.12 dB. (B) Sample 2. Zero-frequency gain: GD: −14.28 dB CDT: −14.72 dB. (C) Sample 3. Zero-frequency gain: GD: −19.18 dB CDT: −22.89 dB.

in higher frequency content is demonstrated. The higher frequency content implies a faster ANN output evolution using CDT.

5.2 | Classification experiments

The Microsoft research Cats versus Dogs dataset[39] 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[38] 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.¶ Figure 5A–D show the MSE validation loss evolution of both CDT and GD for the different learning rates.

As can be seen in Table K5, 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 5D, CDT accelerates learning for CNNs as well as ANNs for low learning rates. However, Figure 5A 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 5C 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
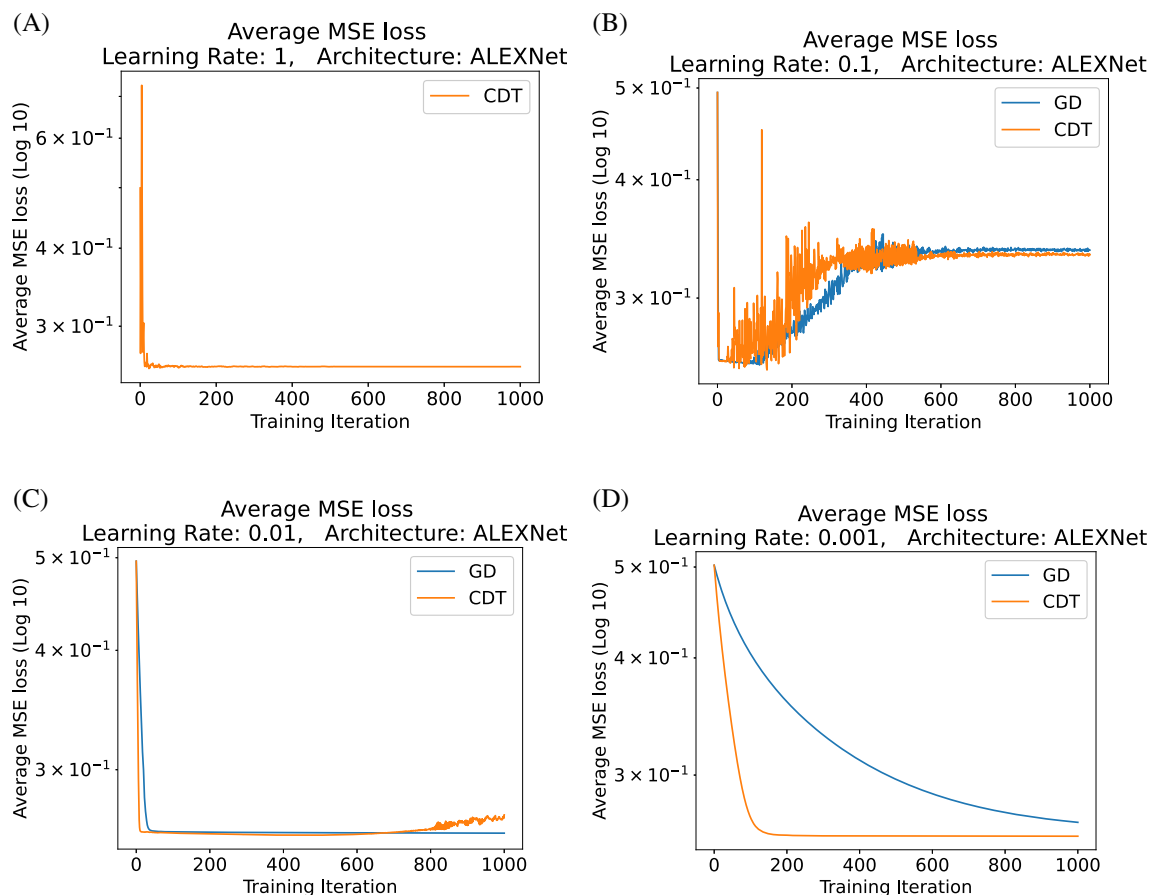

FIGURE 5 MSE loss on validation data for ALEXNet during training with GD and CDT, averaged over all 10 initializations and data shuffles of classification dataset. As can be seen in b, overfitting occurs for high learning rates α. c demonstrate overfitting occurring earlier for CDT due to the acceleration of training.

hence overfit sooner. This hints at using a regularization method together with CDT for optimal performance when using complex network architectures.

6 | CONCLUSION

In this paper, a novel model-based control approach to train ANNs under Gradien Decent is proposed. The method uses the notion of empirical Neural Tangent Kernels (NTK) of ANN training under gradient descent as a model. After analyzing some baseline properties of the model (solvability, stability), a new fictitious label input is created. Label augments equip the training dynamics with dynamically manipulable and artificial labels. These labels give rise to explicit control of the ANNs training behavior.

The newly developed method of Control Decent Learning hence directly manipulates the label augments whilst being (locally) convergent and convergent. In other words, CDT has a locally optimal training behavior via solving a $\mathcal{H}_2$ optimal control problem.

This novel method is demonstrated to improve loss convergence rate for both known CNN architectures and fully connected ANNs with varying widths and depths. Furthermore, CDT gives local convergence guarantees to target labels increasing robustness of ANN training. The stability analysis of ANN training uncovered the effect choice of learning rate has on local ANN training convergence in the upper bound. Reachability is shown to be a good metric for data learnability from the perspective of the chosen ANN architecture. However, CDT and the reachability analysis demonstrated that due to the accelerated training, overfitting is a larger issue for the novel training method. Therefore CDT should be deployed in conjunction with a regularization method to mitigate this effect.

We demonstrated that the theoretical framework of dynamical system theory is directly applicable to ANN training. CDT unlocks the potential to develop additional model-based training solutions. This work is merely the first step in finding a comprehensive description of ANN training suitable for Control Theory applications. We invite the community to further investigate ANN training behavior informed by the NTK from the perspective of dynamical systems and control theory.

CONFLICT OF INTEREST STATEMENT
The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT
The data that support the findings of this study are openly available in the Journal of Statistics Education data archive[37] and Microsoft-Research.[39]

ENDNOTES
*Stabilizability and detectability conditions must hold.[22]
†These penalties weights are tuning parameters.
‡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 paper, the mega-batch setup better demonstrates the theory presented.
§In accordance to the Journal guidelines provided.
¶We do this for two reasons; (1) it is more closely connected to the theory presented which is the main focus of this paper, and (2) it is easier to verify the theory applicability on CNNs with multiple outputs without additional linearizations.

ORCID
Viktor Andersson https://orcid.org/0009-0006-5374-0389

REFERENCES
1. Tripathi M. Analysis of convolutional neural network based image classification techniques. *J Innov Image Process*. 2021;3(2):100-117.

2. Lauriola I, Lavelli A, Aiolli F. An introduction to deep learning in natural language processing: models, techniques, and tools. *Neurocomputing*. 2022;470:443-456.

3. Aggarwal A, Mittal M, Battineni G. Generative adversarial network: an overview of theory and applications. *Int J Inf Manag Data Insights*. 2021;1(1):100004.

4. Pereira M, Lang A, Kulcsár B. Short-term traffic prediction using physics-aware neural networks. *Transp Res Part C-Emerg Technol*. 2022;142:103772.

5. Pereira M, Baykas P, Lang A, Kulcsár B. Parameter and density estimation from real-world traffic data: a kinetic compartmental approach. *Transp Res Part B-Methodol*. 2022;155:210-239.

6. Wan H, Karimi HR, Luan X, Liu F. Model-free self-triggered control based on deep reinforcement learning for unknown nonlinear systems. *Int J Robust Nonlinear Control*. 2023;33(3):2238-2250.

7. Maiworm M, Limon D, Findeisen R. Online learning-based model predictive control with Gaussian process models and stability guarantees. *Int J Robust Nonlinear Control*. 2021;31(18):8785-8812.

8. Marvi Z, Kiumarsi B. Safe reinforcement learning: a control barrier function optimization approach. *Int J Robust Nonlinear Control*. 2023;33(3):2238-2250.

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

10. Lee J, Xiao L, Schoenholz SS, 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.

11. Huang J, Yau H. Dynamics of deep neural networks and neural tangent hierarchy. Proceedings of the 37th International Conference on Machine Learning. 2020 119: 4542–4551.

12. Alemohammad S, Wang Z, Balestriero R, Baraniuk R. The Recurrent Neural Tangent Kernel ArXiv. 2020 2006:10246.

13. Hron J, Bahri Y, Sohl-Dickstein J, Novak R. Infinite attention: NNGP and NTK for deep attention networks. Proceedings of the 37th International Conference on Machine Learning. 2020 119: 4376-4386.

14. Yang G. Tensor programs IIb: architectural universality of neural tangent kernel training dynamics. Proceedings of the 38th International Conference on Machine Learning. 2021 139: 11762-11772.

15. Franceschi J, Bézenac E, Ayed I, Chen M, Lamprier S, Gallinari P. A neural tangent kernel perspective of GANs. Proceedings of the 39th International Conference on Machine Learning. 2021 162: 6660-6704.

16. Littwin E, Myara B, Sabah S, Susskind J, Zhai S, Golan O. Collegial ensembles. Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'33). 2020 18738–18748.

17. Zandieh A, Han I, Avron H, Shoham N, Kim C, Shin J. 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 1062–1073.

18. Novak R, Sohl-Dickstein J, Schoenholz SS. Fast finite width neural tangent kernel. ArXiv. 2022 2206.08720.

19. Montanari A, Zhong Y. The interpolation phase transition in neural networks: memorization and generalization under lazy training. *Ann Stat*. 2022;50(5):2816-2847.

20. Nguyen Q, Mondelli M, Montufar G. Tight bounds on the smallest eigenvalue of the neural tangent kernel for deep ReLU networks. Proceedings of the 38th International Conference on Machine Learning. 2021 139:8119-8129.

21. Jia S, Nezhadarya E, Wu Y, Ba J. Efficient statistical tests: a neural tangent kernel approach. Proceedings of the 38th International Conference on Machine Learning. 2021 139: 4893-4903.

22. Kwakernaak H, Sivan R. *Linear Optimal Control Systems*. John Wiley and Sons; 1972.

23. Dolgui A, Ivanov D, Sethi SP, Sokolov B. Scheduling in production, supply chain and industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications. *Int J Prod Res*. 2019;57(2):432-441.

24. Hagebring F, Lennartson B. Time-optimal control of large-scale systems of systems using compositional optimization. *Discr Event Dyn Syst*. 2019;29:411-443.

25. Zhou Z, Xu H. A novel mean-field-game-type optimal control for very large-scale multiagent systems. *IEEE Trans Cybern*. 2022;52(6):5197-5208.

26. Wanga S, Yua X, Perdikarisb P. When and why PINNs fail to train: a neural tangent kernel perspective. *J Comput Phys*. 2022;449:110768.

27. Varga B, Kulcsár B, Chehreghani MH. Constrained policy gradient method for safe and fast reinforcement learning: a neural tangent kernel based approach. ArXiv. 2021 2006.07678v2.

28. Varga B, Kulcsár B, Chehreghani MH. Deep Q-learning: a robust control approach. *Int J Robust Nonlinear Control*. 2023;33(1):526-544.

29. Chen Z, Cao Y, Gu Q, Zhang T. 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 13363–13373.

30. Ye JC. *Geometry of Deep Learning: A Signal Processing Perspective*. Springer; 2022.

31. Khalil HK. *Nonlinear Systems*. 3rd ed. Patience Hall; 2002.

32. Rugh WJ. *Linear System Theory*. 2nd ed. Patience Hall; 1995.

33. Tong Q, Liang G, Bi J. Calibrating the adaptive learning rate to improve convergence of ADAM. *Neurocomputing*. 2022;481:333-356.

34. Isidori A. *Nonlinear Control Systems*. 3rd ed. Springer; 1995.

35. Maidens J, Arcak M. Reachability analysis of nonlinear systems using matrix measures. *IEEE Trans Automat Contr*. 2015;60(1): 265-270.

36. Szolnoky V, Andersson V, Kulcsar B, Jörnsten R. 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.

37. Cock DD. Dataset. 2011 https://jse.amstat.org/v19n3/decock/AmesHousing.txt

38. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Advances in Neural Information Processing Systems (NeurIPS'25). 2012.

39. Microsoft-Research. Dataset. 2022 https://www.microsoft.com/en-us/download/details.aspx?id=54765

40. Sohl-Dickstein J, Novak R, Schoenholz SS, Lee J. On the infinite width limit of neural networks with a standard parameterization. ArXiv. 2020 abs/2001.07301.

41. Park D, Sohl-Dickstein J, Le Q, Smith S. The effect of network width on stochastic gradient descent and generalization: an empirical study. Proceedings of the 36th International Conference on Machine Learning. 2019 97: 5042–5051.

APPENDIX A. 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 Khalil, Nonlinear Systems.[31]

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

$$||f_\theta(\hat{y}_1) - f_\theta(\hat{y}_2)||_2 \leq L||\hat{y}_1 - \hat{y}_2||_2 \tag{A1}$$

$\forall \hat{y}_1, \hat{y}_2 \in \mathbb{R}^o, \forall k \in [k_i, k_f]$ 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 Khalil, Nonlinear Systems[31] (Ch. 2.2, p. 67, theorem 2.4). ∎

It follows that when training an ANN under gradient descent a solution to Equation (4) always exists and that the solution is unique on the time interval $[k_i, k_f]$.

APPENDIX B. LOCAL TRAINING DYNAMICS

The following appendix details the first order Taylor linearization used to derive Equation (5). The following derivation closly follow the linearization used in previous NTK papers.[10] 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,

$$\hat{y}(k) = \hat{y}(k_0) + \left(\frac{\partial \hat{y}(k)^T}{\partial \theta(k)}\right)^T_{\theta(k_0)} \vartheta(k) + \sum_{i=2}^{\infty} \frac{\partial^{(i)} \hat{y}(k)^T}{\partial^{(i)} \theta(k)}^T_{\theta(k_0)} \frac{\vartheta^{(i)}(k)}{i!}. \tag{B1}$$

Using only the first term from the Taylor expansion (Equation B1) results in

$$\hat{y}_\vartheta(k) \equiv \hat{y}(k_0) + \left(\frac{\partial \hat{y}(k)^T}{\partial \theta(k)}\right)^T_{\theta(k_0)} \vartheta(k). \tag{B2}$$

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

$$\hat{y}_\vartheta(k + 1) = \hat{y}_\vartheta(k) - \alpha\Theta(k_0)\frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} = f_\vartheta(\hat{y}(k)). \tag{B3}$$

APPENDIX C. 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[31] is used,

$$||\hat{y}(k) - \hat{y}_\vartheta(k)||_2 \leq \max_{\theta(\kappa)} \frac{1}{2} \left\| \left(\frac{\partial^2 \hat{y}(k)^T}{\partial^2 \theta(k)} \right)^T_{\theta(\kappa)} \right\|_2 \cdot ||\vartheta(k)||_2^2 \qquad (C1)$$

where for any κ on the discrete interval $[k_0, k]$ $\theta(\kappa)$ is evaluated. Note that Equation (C1) 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.

APPENDIX D. LOSS LINEARIZATION

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:

$$\left(\frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} \right)_{\theta(k_0)} = \left(\frac{\partial \mathcal{L}(\hat{y}(k_0), y)}{\partial \hat{y}(k_0)} \right)_{\theta(k_0)} + \left(\frac{\partial^2 \mathcal{L}(\hat{y}(k_0), y)}{\partial^2 \hat{y}(k_0)} \right)_{\theta(k_0)} (\hat{y}(k) - \hat{y}(k_0))$$
$$+ \sum_{i=2}^{\infty} \left(\frac{\partial^{(i+1)} \mathcal{L}(\hat{y}(k_0), y)}{\partial^{(i+1)} \hat{y}(k_0)} \right)_{\theta(k_0)} \frac{(\hat{y}(k) - \hat{y}(k_0))^{(i)}}{i!} \qquad (D1)$$

and

$$\left(\frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} \right)_L = \left(\frac{\partial \mathcal{L}(\hat{y}(k_0), y)}{\partial \hat{y}(k_0)} \right)_{\theta(k_0)} + \left(\frac{\partial^2 \mathcal{L}(\hat{y}(k_0), y)}{\partial^2 \hat{y}(k_0)} \right)_{\theta(k_0)} (\hat{y}(k) - \hat{y}(k_0)) \qquad (D2)$$

with Lagrange error bound

$$\left\| \left(\frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} \right)_{\theta(k_0)} - \left(\frac{\partial \mathcal{L}(\hat{y}(k), y)}{\partial \hat{y}(k)} \right)_L \right\|_2$$
$$\leq \frac{1}{2} \left\| \left(\frac{\partial^3 \mathcal{L}(\hat{y}(k_0), y)}{\partial^3 \hat{y}(k_0)} \right)_{\theta(k_0)} \right\|_2 \cdot ||(\hat{y}(k) - \hat{y}(k_0))||_2^2. \qquad (D3)$$

Next, insert the linearized loss into Equation (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

$$\hat{y}_\vartheta(k+1) = \hat{y}_\vartheta(k) - \alpha\Theta(k_0)\left(\left(\frac{\partial \mathcal{L}(\hat{y}(k_0), y)}{\partial \hat{y}(k_0)} \right)_{\theta(k_0)} + \left(\frac{\partial^2 \mathcal{L}(\hat{y}(k_0), y)}{\partial^2 \hat{y}(k_0)} \right)_{\theta(k_0)} (\hat{y}_\vartheta(k) - \hat{y}(k_0)) \right)$$
$$= \left(I - \alpha\Theta(k_0) \left(\frac{\partial^2 \mathcal{L}(\hat{y}(k_0), y)}{\partial^2 \hat{y}(k_0)} \right)_{\theta(k_0)} \right) \hat{y}_\vartheta(k)$$
$$+ \alpha\Theta(k_0) \left(\frac{\partial^2 \mathcal{L}(\hat{y}(k_0), y)}{\partial^2 \hat{y}(k_0)} \right)_{\theta(k_0)} \hat{y}(k_0) - \alpha\Theta(k_0) \left(\frac{\partial \mathcal{L}(\hat{y}(k_0), y)}{\partial \hat{y}(k_0)} \right)_{\theta(k_0)}. \qquad (D4)$$

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 Equations (C1) and (D3) can be given as follows. Denote the left hand side of Equation (C1) with E_y and Equation (D3) with E_L. Consider Equation (D3) and inject the linearization error of $\hat{y}(k)$ as

$$E_L(E_y) \leq \frac{1}{2} \left\| \left(\frac{\partial^3 \mathcal{L}(\hat{y}(k_0), y)}{\partial^3 \hat{y}(k_0)} \right)_{\theta(k_0)} \right\|_2 \cdot ||(\hat{y}_\vartheta(k) + E_y - \hat{y}(k_0))||_2^2. \qquad (D5)$$

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

$$E \leq E_y - \alpha\Theta(k_0)E_L(E_y). \tag{D6}$$

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.

APPENDIX E. 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, that is, $\hat{y}(k+1) = \hat{y}(k)$. 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 $\Theta(k) = \underline{0}$. However, it can only occur in some very specific cases, for example, 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 $\Theta(k) \neq \underline{0}$, and $\left(\frac{\partial\mathcal{L}(\hat{y}(k),y)}{\partial\hat{y}(k)}\right)_{\theta(k)} \neq 0$. That is, 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.

APPENDIX F. BOUNDEDNESS FOR COMMON LOSSES

The boundedness of some common loss functions is analyzed, assuming static target y.

1. **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 Equation (5) one gets

$$\hat{y}_\vartheta(k+1) = \hat{y}_\vartheta(k) - \frac{\alpha\Theta(k_0)}{rn_L}(\hat{y}_\vartheta(k) - y). \tag{F1}$$

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

$$\hat{y}_\vartheta(k+1) = \left(I - \frac{\alpha\Theta(k_0)}{rn_L}\right)\hat{y}_\vartheta(k) + \frac{\alpha\Theta(k_0)}{rn_L}y. \tag{F2}$$

In Equation (F2), 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 if

$$|\lambda| < 1 \quad \forall\lambda \in eig\left(I - \frac{\alpha\Theta(k_0)}{rn_L}\right). \tag{F3}$$

The proof for this can be found in Rugh, Linear System Theory.[32]

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

$$|\lambda| < 1 \quad \forall\lambda \in eig(I - \alpha\Theta(k_0)), \tag{F4}$$

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.

3. **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

$$\frac{\partial\mathcal{L}(\hat{y}_\vartheta(k), y_i)}{\partial\hat{y}_\vartheta(k)} = \frac{1}{rn_L}\sum_{i=1}^{rn_L}\frac{\hat{y}_{\vartheta,i}(k) - y}{|\hat{y}_{\vartheta,i}(k) - y_i|} \tag{F5}$$

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

$$y_\vartheta(k+1) = y_\vartheta(k) - \frac{\alpha\Theta(k_0)}{rn_L}sgn(\hat{y}_\vartheta(k) - y). \tag{F6}$$

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.

4. **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

$$y_\vartheta(k+1) = y_\vartheta(k) + \alpha\Theta(k_0)\breve{y}_\vartheta(k)y, \tag{F7}$$

where $\breve{y}_\vartheta(k)$ 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 $\alpha\Theta(k_0)log(\breve{y}_\vartheta(k)y)$ is a null vector. That is, y is an equilibrium point if it is in the nullspace of the matrix $\alpha\Theta(k_0)\breve{y}_\vartheta(k)$. A trivial solution to this if $y = 0$, and this is the only solution if the columns in $\alpha\Theta(k_0)\breve{y}_\vartheta(k)$ 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.[31] Let $V(x) = x^T x$ be a Lyapunov function. Then for Equation (F7) the Lyapunov boundedness criteria is

$$\left(y_\vartheta(k) + \alpha\Theta(k_0)\breve{y}_\vartheta(k)y\right)^T\left(y_\vartheta(k) + \alpha\Theta(k_0)\breve{y}_\vartheta(k)y\right) - y_\vartheta(k)^T y_\vartheta(k) < 0 \tag{F8}$$

which can be simplified to

$$\alpha^2 y^T \breve{y}_\vartheta^T(k)\Theta^T(k_0)\Theta(k_0)\breve{y}_\vartheta(k)y - 2\alpha y_\vartheta^T(k)\Theta(k_0)\breve{y}_\vartheta(k)y < 0. \tag{F9}$$

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, that is, $y, y_\vartheta(k) \in (0, 1) \subset \mathbb{R}^{m_L}$. In such a case (in a local sense) α is always positive, y, and $y_\vartheta(k)$ are positive vectors, $\breve{y}_\vartheta(k)$ is a diagonal matrix with positive elements. $\Theta(k_0)$ is symmetric and if the input is normalized, it is positive-definite too.[9] 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.

APPENDIX G. BOUNDEDNESS OF THE GLOBAL DYNAMICS

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 (Equation C1) from Equation (10) giving a less conservative bound for the global training

dynamics:

$$||\hat{y}(k) - \hat{y}_e||_2 \leq \gamma e^{-\lambda(k-k_0)}||\hat{y}_\vartheta(k_0) - \hat{y}_e||_2 - \max_{\theta(r)} \frac{1}{2} \left|\left|\left(\frac{\partial^2 \hat{y}(k)^T}{\partial^2 \theta(k)}\right)^T_{\theta(r)}\right|\right|_2 \cdot ||\vartheta^2(k)||_2. \tag{G1}$$

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. That is, if

$$||\hat{y}_\vartheta(k_c) - \hat{y}_e||_2 > \gamma e^{-\lambda(k_c-k_0)}||\hat{y}_\vartheta(k_0) - \hat{y}_e||_2$$
$$- \max_{\theta(r)} \frac{1}{2} \left|\left|\left(\frac{\partial^2 \hat{y}(k_c)^T}{\partial^2 \theta(k_c)}\right)^T_{\theta(r)}\right|\right|_2 \cdot ||\vartheta^2(k_c)||_2 \tag{G2}$$

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

APPENDIX H. THE DARE EQUATION

The following appendix describes the standard Discrete-time Algebraic Riccati Equation (DARE)[22] which is deployed in order to find the solution P to the quadratic infinite optimization problem in Equation (17). The proof that Equation (H1) solves the cost given in Equation (17) is given in Kwakernaak, Linear Optimal Control Systems.[22]

$$A = \begin{bmatrix} I - \alpha\Theta(k_0) & \alpha\Theta(k_0)y \\ \mathbf{0} & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \alpha\Theta(k_0) \\ \mathbf{0} \end{bmatrix}, \quad P = A^T P A + Q - A^T P B (R + B^T P B)^{-1} B^T P A \tag{H1}$$

APPENDIX I. INITIALIZATION OF THE ANN

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

1. Standard initialization. The weight for each neuron are given as $\mathcal{N}\left(0, \frac{\sigma_w^2}{sn_l}\right)$ $\left(\mathcal{N}\left(0, \frac{\sigma_w^2}{sn_l n_m}\right)\right.$ for convolutional layers$\left.\right)$, and biases are $\mathcal{N}\left(0, \sigma_b^2\right)$ 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 infinite width-limit the entries of the NTK diverge.
2. NTK initialization, proposed by Jacot et al.[9] In this case, weights and biases are initialized with normalized gaussian distributions $\mathcal{N}(0, 1)$. The weights are multiplied with $\frac{\sigma_w}{\sqrt{sn_l}}$, $\left(\frac{\sigma_w}{\sqrt{sn_l n_m}}\right.$ for convolutional layers$\left.\right)$, and the biases are scaled with σ_b. That is to make the NTK values converge.
3. Improved standard initialization.[40] The difference between the standard and the improved version is that the width-scaling factor is pulled out from the normal distribution, that is, $\frac{1}{\sqrt{s}}\mathcal{N}\left(0, \frac{\sigma_w^2}{n_l}\right)$ and $\frac{1}{\sqrt{s}}\mathcal{N}\left(0, \frac{\sigma_w^2}{n_l n_m}\right)$.

The initializations are summarized in Table K6.

According to Park et al.,[40,41] infinite width networks with various architectures achieve similar error regardless of initialization. That is, 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 two as recommended by Jacot et al.[9]

APPENDIX J. SUPPLEMENTARY FIGURES

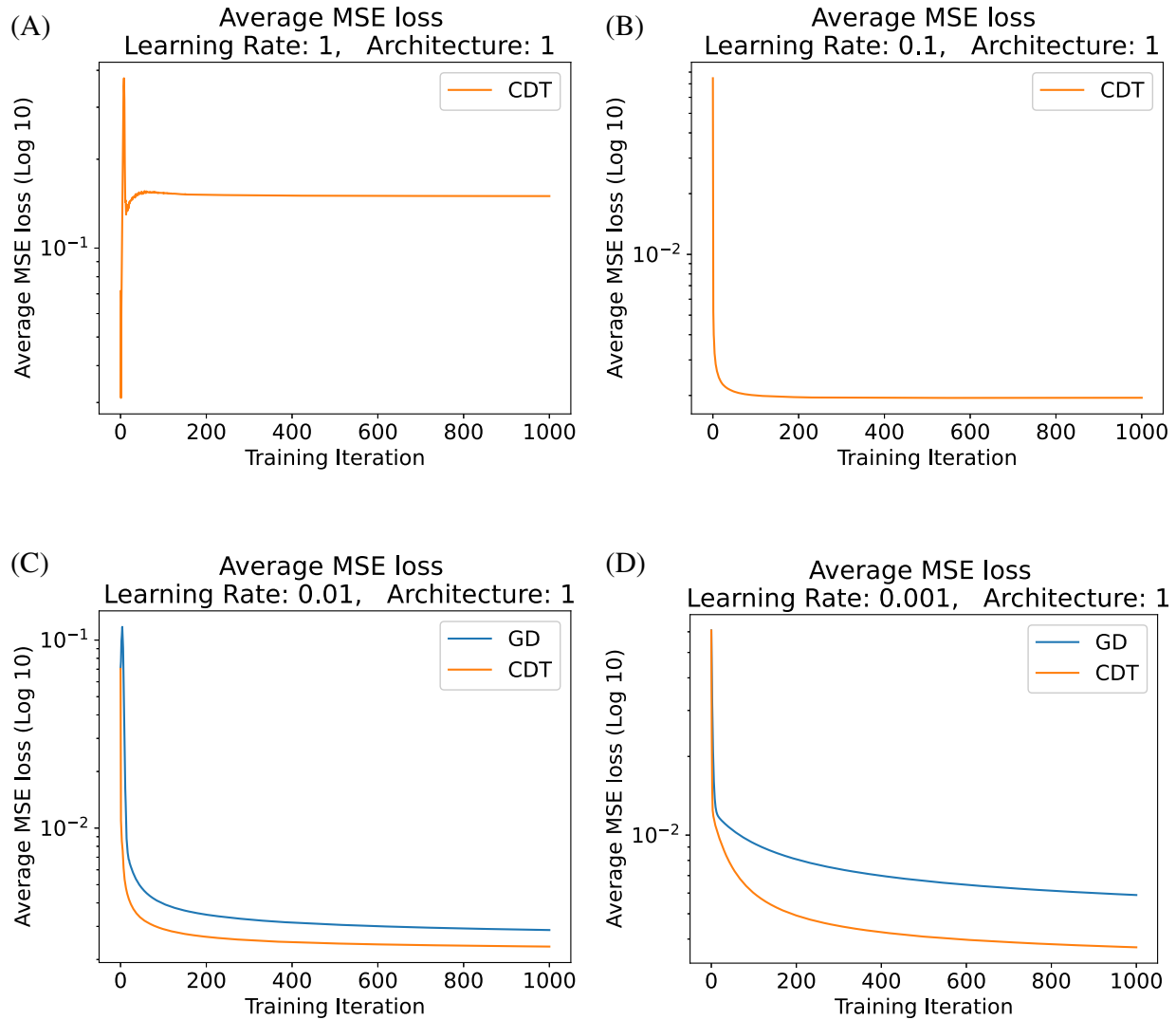### J.1 Regression experiment supplements

FIGURE J1   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 significantly fewer iterations than traditional GD for all learning rates. Moreover, CDT is stable at higher learning rates ($\alpha \geq 0.1$), while GD diverges. We believe that the robustness margins of the state feedback $\mathcal{H}_2$ control algorithm makes the CDT powerful.
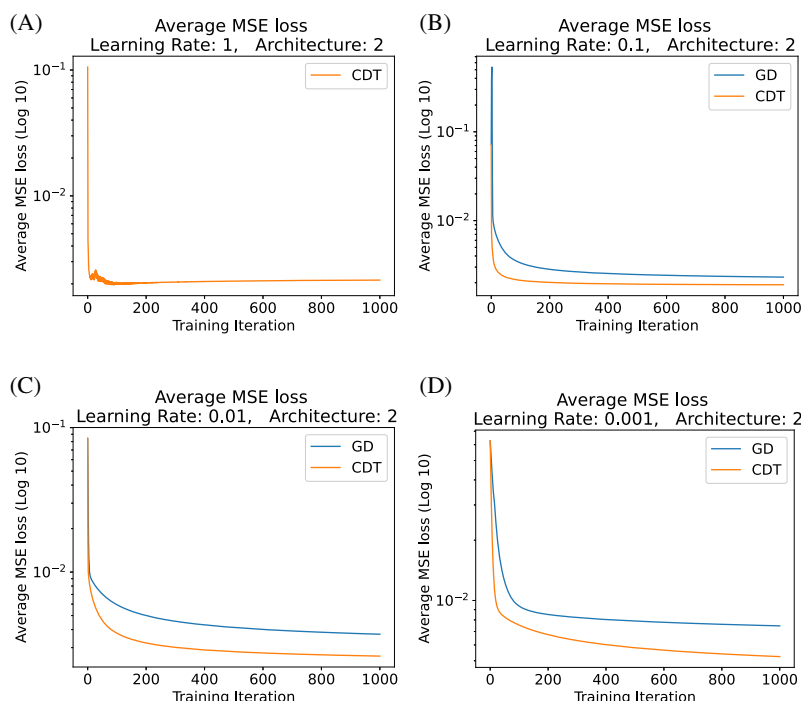
FIGURE J2   MSE loss on validation data for fully connected ANN architecture 2 averaged over all 10 initializations and data shuffles during training with GD and CDT. CDT appears to converge with fewer iterations than GD for all learning rates. Moreover, CDT is stable at higher learning rates ($\alpha \geq 1$), while GD diverges. We believe that the robustness margins of the state feedback $\mathcal{H}_2$ control algorithm makes the CDT powerful. Note, the speed increase in the convergence via CDT.
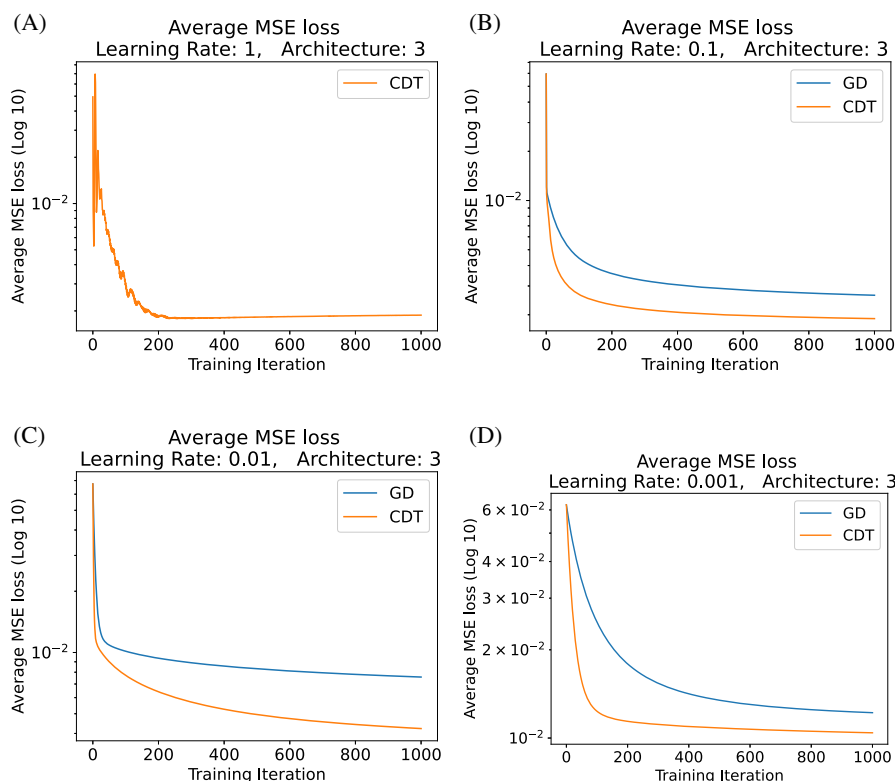

FIGURE J3   MSE loss on validation data for fully connected ANN architecture three averaged over all 10 initializations and data shuffles during training with GD and CDT. CDT converges with fewer iterations than GD for all learning rates. Moreover, CDT is stable at higher learning rates ($\alpha \geq 1$), while GD diverges.
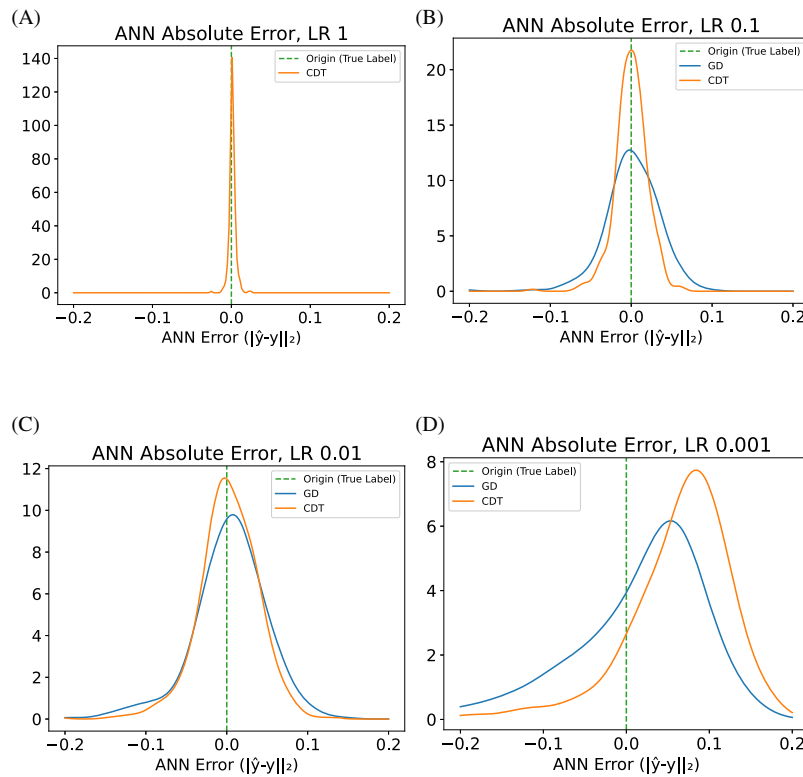
FIGURE J4 Kernel density estimated distribution of ANN output two norm related distance from label y at final final training step for GD and CDT using architecture 1. It can be seen that error variance is smaller for CDT (stochastic interpretation of $\mathcal{H}_2$ norm). GD did not converge for all initializations using $\alpha = 1$.

FIGURE J5 Kernel density estimated distribution of ANN output two-norm related distance from label y at final final training step for GD and CDT using architecture 2. GD did not converge for all initializations using learning rate $\alpha = 0.1$ or 1. CDT results in smaller error variance.
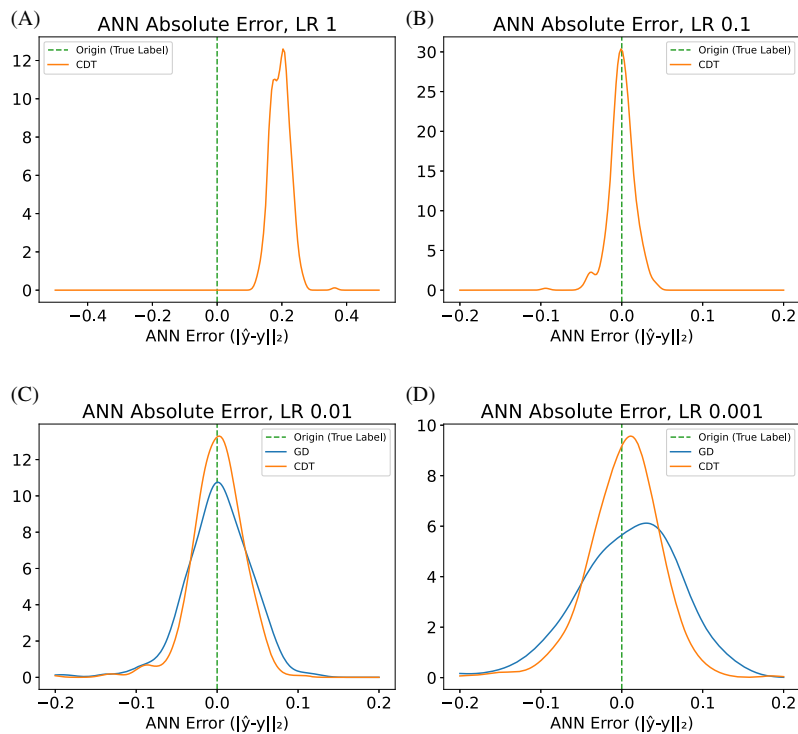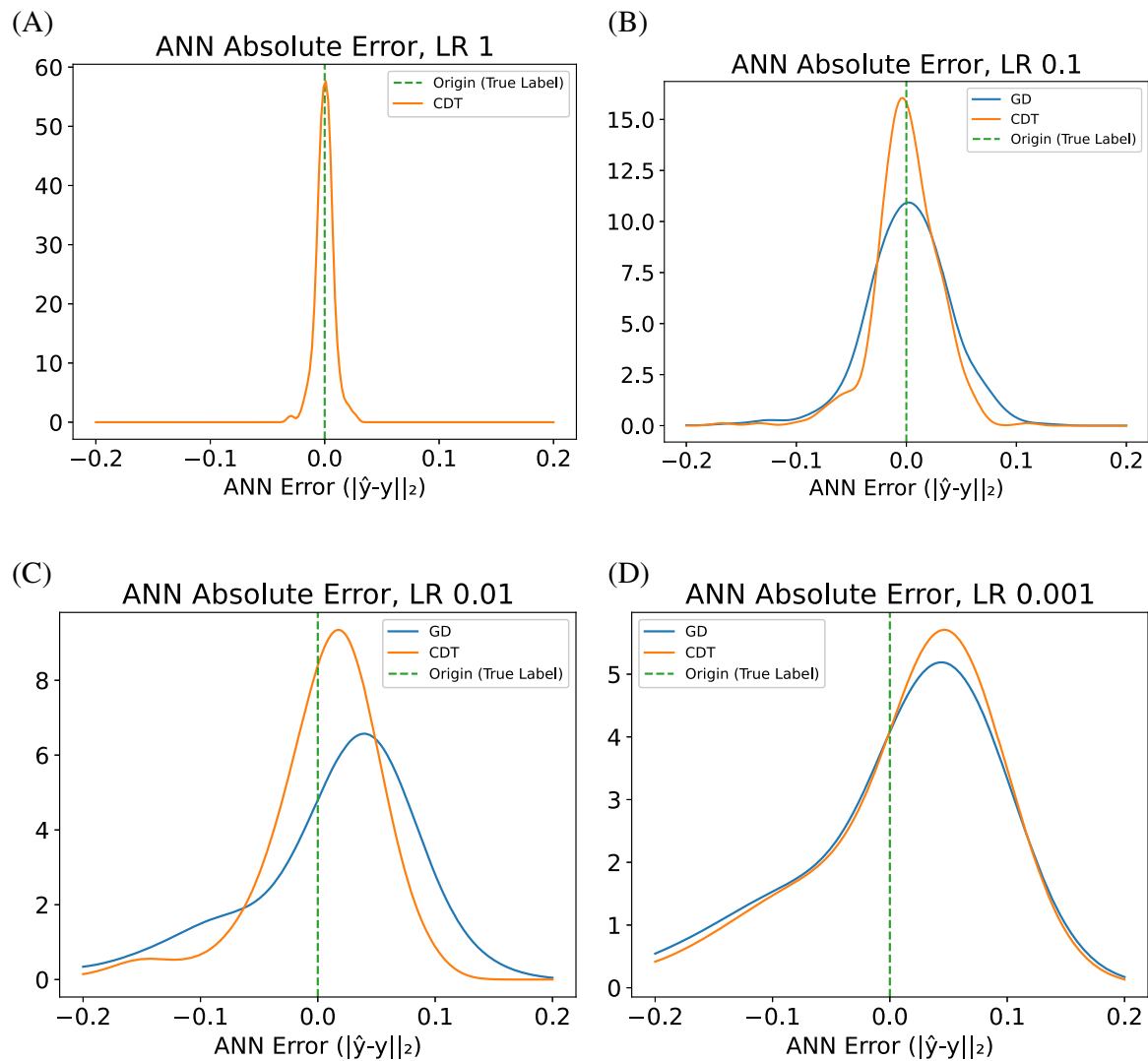
FIGURE J6 Kernel density estimated distribution of ANN output two-norm related distance from label y at final final training step for GD and CDT using architecture 3. GD did not converge for all initializations using learning rate $\alpha = 1$. CDT results in smaller error variance.
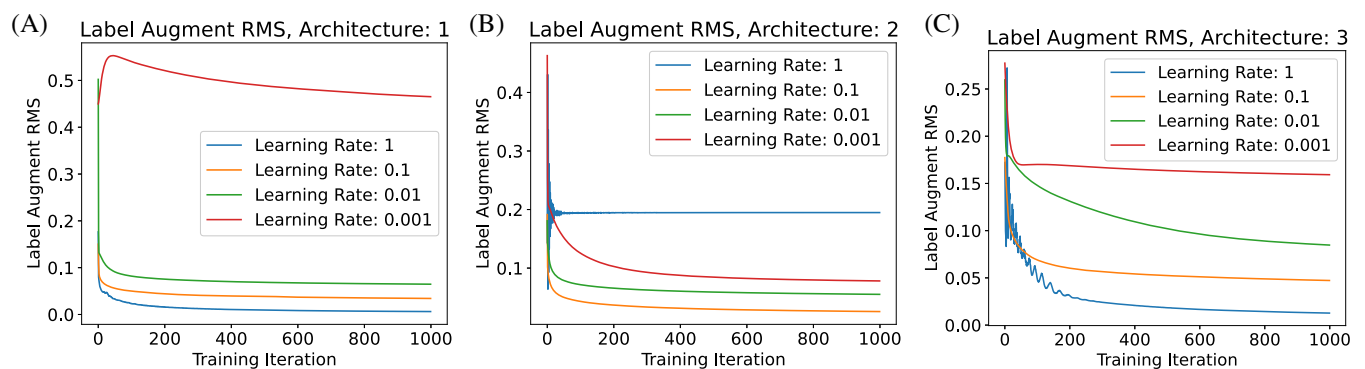
FIGURE J7 The root-mean-square (RMS) of the label augment y_u for all learning-rates and architectures. The label augments tend toward zero as the training progress. There is one slowly converging learning-rate for architecture 3 where $\alpha = 0.001$. This can be explained by observing from plot Figure J1D that the training converges slowly. Moreover, the label augments do not vanish for architecture 2 with large learning rate $\alpha = 1$. The high learning rate tightens the domain of validity of the linearized NTK and causes instability of the training as can be seen in Table K3. The poor linear NTK based CDT results in an offset. Note, that changing the $\mathcal{H}_2$ weighting penalty via ρ expands the robustness margins and hence improves the convergence.

APPENDIX K. TABLES

T A B L E K1 Single-target fully connected ANN architectures used for the regression experiments.

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

T A B L E K2 Analytical and observed properties for fully connected architecture 1 on regression dataset.

α		Reachability	$\lvert eig(\Theta(k_0)) \rvert < 1$	Convergence	Final validation loss (MSE 10^{-3})
1.000	GD	Yes	No	No initialization	$\infty \pm \infty$
1.000	CDT	Yes	No	All initializations	150.01 ± 350.50
0.100	GD	Yes	No	Some initializations (7/10)	3.04 ± 0.59
0.100	CDT	Yes	No	All initializations	1.95 ± 0.24
0.010	GD	Yes	Yes	All initializations	2.70 ± 0.61
0.010	CDT	Yes	Yes	All initializations	2.34 ± 0.49
0.001	GD	Yes	Yes	All initializations	5.90 ± 0.95
0.001	CDT	Yes	Yes	All initializations	3.72 ± 0.76

T A B L E K3 Analytical and observed properties for fully connected architecture 2 on regression dataset.

α		Reachability	$\lvert eig(\Theta(k_0)) \rvert < 1$	Convergence	Final validation loss (MSE 10^{-3})
1.000	GD	Yes	No	No initializations	$\infty \pm \infty$
1.000	CDT	Yes	No	All initializations	2.14 ± 0.28
0.100	GD	Yes	No	All initializations	2.31 ± 0.45
0.100	CDT	Yes	No	All initializations	1.90 ± 0.29
0.010	GD	Yes	Yes	All initializations	3.71 ± 0.62
0.010	CDT	Yes	Yes	All initializations	2.62 ± 0.54
0.001	GD	Yes	Yes	All initializations	7.47 ± 1.54
0.001	CDT	Yes	Yes	All initializations	5.24 ± 0.99

TABLE K4 Analytical and observed properties for fully connected architecture 3 on regression dataset.

α		Reachability	$\|eig(\Theta(k_0))\| < 1$	Convergence	Final validation loss (MSE 10^{-1})
1.000	GD	Yes	No	No initializations	$\infty \pm \infty$
1.000	CDT	Yes	No	All initializations	1.88 ± 0.27
0.100	GD	Yes	No	All initializations	2.63 ± 0.80
0.100	CDT	Yes	No	All initializations	1.90 ± 0.48
0.010	GD	Yes	Yes	All initializations	7.58 ± 1.84
0.010	CDT	Yes	Yes	All initializations	4.23 ± 1.05
0.001	GD	Yes	Yes	All initializations	12.20 ± 2.61
0.001	CDT	Yes	Yes	All initializations	10.42 ± 2.11

TABLE K5 Analytical and observed properties of ALEXNet on classification dataset.

α		Reachability	$\|eig(\Theta(k_0))\| < 1$	Convergence	Final validation loss (MSE 10^{-1})
1.000	GD	Yes	No	No initializations	$\infty \pm \infty$
1.000	CDT	Yes	No	All initializations	2.59 ± 0.15
0.100	GD	Yes	Yes	All initializations	3.37 ± 0.38
0.100	CDT	Yes	Yes	All initializations	3.33 ± 0.37
0.010	GD	Yes	Yes	All initializations	2.58 ± 0.11
0.010	CDT	Yes	Yes	All initializations	2.69 ± 0.29
0.001	GD	Yes	Yes	All initializations	2.67 ± 0.10
0.001	CDT	Yes	Yes	All initializations	2.58 ± 0.11

TABLE K6 Different ANN weight and bias initializations.

	Standard init.	NTK init.	Improved standard
Weight initialization	$\mathcal{N}\left(0, \frac{\sigma_w^2}{sn_l}\right)$	$\frac{\sigma_w}{\sqrt{sn_l}}\mathcal{N}(0,1)$	$\frac{1}{\sqrt{s}}\mathcal{N}\left(0, \frac{\sigma_w^2}{n_l}\right)$
Weight initialization (conv.)	$\mathcal{N}\left(0, \frac{\sigma_w^2}{sn_l n_m}\right)$	$\frac{\sigma_w}{\sqrt{sn_l n_m}}\mathcal{N}(0,1)$	$\frac{1}{\sqrt{s}}\mathcal{N}\left(0, \frac{\sigma_w^2}{n_l n_m}\right)$
Bias initialization	$\mathcal{N}(0, \sigma_b^2)$	$\mathcal{N}(0, \sigma_b^2)$	$\mathcal{N}(0, \sigma_b^2)$