Unifying Back-Propagation and Forward-Forward Algorithms through Model Predictive Control

Anonymous authors
Paper under double-blind review

Abstract

We propose a Model-Predictive-Control (MPC) view of neural-network training that unifies back-propagation (BP) and the Forward-Forward (FF) algorithm as two extreme instances of a receding-horizon planner family: at each layer the planner solves a length-h terminal-cost subproblem and stitches the partial gradients into a global update. Three planner backends instantiate the family by varying the window stride: chunked and overlap-all both collapse to FF at h=1 and recover full BP at h=T, while deep-supervision also collapses to FF at h=1 but at h=T instead recovers a full deep-supervision schedule, making the horizon h a single dial that gradually interpolates between the FF and BP regimes the title alludes to. Theoretically, on a deep linear network with T blocks (or nonlinear residual networks with bounded block Jacobians) we prove a finite-T gradient-cosine deviation $1 - \cos^2(\theta_h) = O(\delta^p)$ with $\delta = 1 - h/T$, where p=3 for deep-supervision and p=2 for overlap-all and chunked (with a matching $\Omega(\delta^2)$ lower bound); combined with a biased-PL analysis this gives per-iteration suboptimality $1 - r(h) = O(\delta^p)$, so the marginal gain of an extra horizon step diminishes polynomially as $h \to T$ while memory grows linearly, ruling out FF and BP as optimal and forcing an interior $h^\star \in (1, T)$ as the unique minimiser of any reasonable accuracy–memory objective. Empirically, on CIFAR-100 (ViT-b/16+LoRA, ResNet-50 stage- and block-cut) and RoBERTa-base+LoRA on four GLUE tasks, the measured $1-\cos^2\theta_h$ tracks the predicted polynomial decay, and the resulting test-accuracy curve shows the same diminishing returns: an interior horizon $h \approx T/3$ closes 80–98% of the FF–BP accuracy gap across all tasks and, on the majority of them, matches or slightly surpasses full BP ($\leq +1.3$ pp), while peak GPU memory scales only linearly in h. This pins an interior $h^\star$ as the optimal accuracy–memory operating point—ruling out both FF and full BP—and delivers substantially better memory efficiency ($\sim$30–80% lower peak GPU usage on transformer backbones) at no cost to accuracy.

1 Introduction

Neural Networks (NN) are rapidly developing in recent years and have found widespread application across various fields. While Back-Propagation (BP) (Rumelhart et al., 1986) stands as the predominant training method, its high memory requirements limit its application to deep model, large batch sizes, and memory-constrained devices, such as those encountered in large language models (Touvron et al., 2023; OpenAI et al., 2024; Gu & Dao, 2023). To address these limitations, recent research has sought to mitigate the drawbacks associated with BP or explore alternative training methods altogether. Notably, Hinton (2022) proposed a Forward-Forward (FF) algorithm that uses layer-wise local loss to avoid backward propagation through layers. Xiong et al. (2020) also proposed a local loss learning strategy. Local loss methods are theoretically more memory-efficient than BP. However, the performance of FF algorithms is often inferior to that of BP, particularly in deep models. Moreover, the mechanism of FF is unclear, and there generally lacks an understanding of why and when FF can work.

In this paper we study whether there is a principled NN optimization method that can balance memory usage and accuracy. Inspired by the parallels between Back-Propagation and mathematical control theory (Li et al., 2018), we utilize the concept of the Model Predictive Control (MPC), to formulate a general framework for deep network training. In this framework, FF and BP are naturally unified as two extremes of a MPC-type

training scheme. At the same time, a number of previous works on NN training (Xiong et al., 2020; Nøkland & Eidnes, 2019; Belilovsky et al., 2019) (Remark 3.1) can also be incorporated in this framework as special cases. This framework offers not only a spectrum of optimization algorithms with varying horizons to balance performance and memory demand but also a dynamical viewpoint to understand these algorithms, including the FF algorithm. Concretely, a single horizon parameter $h \in \{1, \ldots, T\}$ instantiates three backends that we study throughout the paper: the chunked backend (disjoint h-block windows, one local loss per window), the overlap-all backend (sliding h-block windows that overlap and write gradients back to every covered block), and the deep-supervision backend (a loss at every depth, each backpropagated through the preceding h blocks). All three backends collapse to FF at h=1; at h=T, chunked and overlap-all both recover full BP, while deep-supervision recovers a full deep-supervision schedule. Additionally, our theoretical analysis on a deep linear surrogate shows that the gradient estimation converges polynomially as the horizon approaches full back-propagation (Theorem 3.3), leading to diminishing returns for sufficiently large horizons, with backend-specific rates (δ^3 for deep-supervision vs. δ^2 for overlap-all and chunked, $\delta \triangleq 1 - h/T$). The same rates extend to nonlinear residual networks with bounded block Jacobians (Corollary 3.6). However, the memory demand grows constantly with respect to the horizon, indicating an intermediate region offering a favorable trade-off between memory and accuracy. Our empirical sweeps across ViT, ResNet-50 and RoBERTa confirm that an interior horizon recovers most of the BP accuracy at substantially reduced peak memory (roughly 30–80% of full BP, backbone- and backend-dependent).

The contributions of this paper are summarized as follows:

- We propose a novel MPC framework for deep neural network training, unifying BP and FF algorithms and providing a range of optimization algorithms with different accuracy-memory balances.
- Within the framework we instantiate three concrete backends (chunked, overlap-all and deep-supervision), and prove backend-specific polynomial-in-δ convergence rates of the truncated gradient to its BP (resp. full deep-supervision) reference, together with matching peak-memory bounds.
- We analyze the accuracy-efficiency trade-off within the MPC framework, providing theoretical insights into gradient estimations and memory usage.
- Empirical sweeps across pretrained ViT, ResNet-50 and RoBERTa backbones validate the theory and identify an interior horizon that recovers most of the BP accuracy at substantially reduced peak memory.

The remainder of this paper is structured as follows: Section 2 provides a review of relevant literature. Section 3 briefly reviews BP and FF and presents the MPC framework together with its theoretical analysis. The numerical experiments are reported in Section 4. Finally, we conclude in Section 5.

2 Literature Review

Despite its great success in deep learning, it has been argued that Back-Propagation is memory inefficient and biologically implausible (Hinton, 2022). Recently, numerous studies have been focused on addressing the shortcomings of BP, such as direct feedback alignment algorithm (Nø kland, 2016), synthetic gradient (Jaderberg et al., 2017), and LoCo algorithm (Xiong et al., 2020). Notably, Hinton (2022) proposed a forward-forward algorithm that uses layer-wise local loss to avoid backward propagation through layers. Although these methods mitigate the memory inefficiency of BP, they often suffer from inferior performance or introduce additional structures to the models and the theoretical understanding of when and why these methods work is insufficient.

The connection and comparison between residual neural networks and control systems were observed by E (2017). Subsequently, Li et al. (2018) used relationships between Back-Propagation and the Pontryagin maximum principle in optimal control theory to develop alternative training algorithms. Following this, several works have introduced control methods aimed at enhancing optimization algorithms in neural network training (Li et al., 2018; Weng et al., 2022; Nguyen et al., 2024). These prior works have primarily relied on powerful PID controllers or optimal control theory, which, in practice, are rooted in the concept of backward

propagation and thus inherit the drawbacks associated with BP. Conversely, MPC (Grüne & Pannek, 2017), another well-known controller, has yet to be extensively explored in the realm of deep learning (E et al., 2022).

3 The Model Predictive Control Framework

This section offers a brief overview of the traditional Back-Propagation (BP) algorithm and the more recently proposed Forward-Forward (FF) algorithm. We then establish connections between BP and FF algorithms using the MPC concept, introducing an MPC framework for deep learning models. Finally, we give a theoretical analysis of the MPC framework on linear neural networks, providing insights into the influence of horizon.

We focus on the deep feed-forward neural networks, which have the following form:

$$x(t+1) = f_t(x(t), u(t)), \tag{1}$$

where $t \in \{0, ..., T-1\}$ denotes the block index, $x(t) \in \mathbb{R}^{n_t}$ denotes the input of the t-th block , and $u(t) \in \mathbb{R}^{m_t}$ represents vectorized trainable parameters in the t-th block, n_t, m_t are dimensions of $x(t)$ and $u(t)$, $f_t : \mathbb{R}^{n_t} \times \mathbb{R}^{m_t} \to \mathbb{R}^{n_{t+1}}$ denotes the forward mapping of the t-th block. In deep learning training, the primary objective is to minimize the empirical loss using gradient:

$$J(u) = L(x(T)), \quad u^{\tau+1} = u^{\tau} - \eta g(u^{\tau}), \tag{2}$$

where $u \triangleq (u(0)^\top, \cdots, u(T-1)^\top)^\top \in \mathbb{R}^m$ denotes all trainable parameters in the neural network, $m = \sum_{t=0}^{T-1} m_t$ is the number of trainable parameters in the model and $g(u)$ represents the gradient, η denotes the learning rate, and $L(x(T)) : \mathbb{R}^{n_T} \to \mathbb{R}$ represents the loss on the final output $x(T)$[1], τ denotes the iteration index. In this paper, we further assume the loss to be compatible with the state $x(t)$ for all $t = 1, \cdots, T$, i.e. either $x(t)$ has the same dimension for all $t = 1, \cdots, T$, or there is a small *probe* head (e.g., pooling followed by a learned linear map; see Appendix C.1 for the concrete parametrisations used in our experiments) that maps $x(t)$ to the loss domain. Probe parameters are treated as auxiliary trainable weights: they enter the gradient step of Eq. (2) alongside u, but the theoretical analysis in Section 3.3 takes them to be fixed (and, in the deep-linear surrogate, the identity map) so that the gradient deviation isolates the effect of the planning schedule.

3.1 Back-Propagation and Forward-Forward Algorithm

The Back-Propagation algorithm (Rumelhart et al., 1986) computes the gradient of the loss function for the weights of the neural network:

$$g_{\text{BP}}(u(t)) = \nabla_{u(t)} J(u) = \nabla_{u(t)} L(x(T)). \tag{3}$$

Recently, Hinton (2022) proposed the Forward-Forward (FF) algorithm as an alternative to BP. The FF algorithm computes gradients using the local loss function of the current block:

$$g_{\text{FF}}(u(t)) = \nabla_{u(t)} L(x(t+1)). \tag{4}$$

In the original paper (Hinton, 2022), the author used the Euclidean norm of the layer output as a loss, but here we use the more general form of the loss function.

While the FF algorithm is more memory-efficient than BP, its performance may be worse due to the lack of global information. In contrast, BP is typically more accurate but demands more memory. Moreover, the theoretical understanding of when and why FF might perform well remains limited. Leveraging the concept of MPC in deep learning models, we propose a novel MPC framework that unifies the BP and FF algorithms. This framework offers a spectrum of optimization algorithms with varying horizons, allowing for a balance between performance and memory demand.

[1]Here, the loss is the average of loss in training dataset, we ignore the data index for brevity. We also ignore regularization terms and other possible parameters and labels in the loss function, focusing solely on the loss function that depends on the final output $x(T)$.

3.2 Adopting MPC for Deep Learning

In this section, we introduce the classical Model Predictive Control (MPC) framework (Grüne & Pannek, 2017) and explore its connection with deep learning. Subsequently, we propose an MPC framework tailored to deep learning models and demonstrate how it unifies BP and FF algorithms within this framework.

Before recalling MPC, we point out that the feed-forward network dynamics Eq. (1), $x(t+1) = f_t(x(t), u(t))$, already has the exact form of a discrete-time control system: the block index t plays the role of time, the intermediate activation $x(t)$ is the *state*, the per-block trainable weights $u(t)$ are the *control input*, and the block map f_t is the (time-varying) dynamics; the terminal loss $L(x(T))$ in Eq. (2) then acts as a terminal cost on the final state, which is precisely the setting classical optimal-control algorithms (including MPC) are designed to solve (E, 2017; Li et al., 2018). With this identification in place, we can transcribe the classical MPC machinery to Eq. (1) in a single step below.

MPC is an optimization-based method for the feedback control of dynamic systems. In the infinite horizon control problem, which is commonly considered in the control field, the trajectory loss is used, i.e.:

$$J(u) = \sum_{t=0}^{\infty} l(t, x(t), u(t)), \tag{5}$$

where $l(t, \cdot, \cdot) : \mathbb{R}^{n_t} \times \mathbb{R}^{m_t} \to \mathbb{R}$ is the trajectory loss for state $x(t)$ and control $u(t)$ on time t. Due to the computational complexity of obtaining optimal control for an infinite time horizon, a truncated finite-time control problem is solved at each time step:

$$\begin{aligned} J^h(t, z, u_t^h) = &\sum_{s=t}^{t+h-1} l(s, x(s), u(s)), \\ \text{s.t.} \quad & x(t) = z, \quad x(s+1) = f_s(x(s), u(s)), \quad \forall s = t, \cdots, t+h-1, \end{aligned} \tag{6}$$

where h is the horizon considered by MPC at each time step, $u_t^h \triangleq \{u(t), \cdots, u(t+h-1)\}$. The solution $u_t^{h*} = \arg\min_{u_t^h} J^h(t, x(t), u_t^h)$ is applied to the current state:

$$x(t+1) = f_t(x(t), u_t^{h*}(t)). \tag{7}$$

Specialising Eq. (6) to the network dynamics Eq. (1) is now essentially a change of vocabulary (block $\leftrightarrow$ time, activation $\leftrightarrow$ state, weights $\leftrightarrow$ control input). Since deep learning problems typically involve only a terminal loss applied at the end of the network, we need to define an effective trajectory loss $l(t, x(t), u(t))$ for each block t that add up to the desired terminal loss Eq. (2). One particular choice is as follows:

$$l(t, x(t), u(t)) \triangleq L(f_t(x(t), u(t))) - L(x(t)) = L(x(t+1)) - L(x(t)). \tag{8}$$

We demonstrate that the terminal loss $L(x(T))$ is equivalent to the sum of trajectory loss $\sum_{t=0}^{T-1} l(t, x(t), u(t))$ up to a constant (see Appendix B.1).

Two design choices as variants of classical MPC. The receding-horizon objective J^h is read off exactly as in classical MPC, but the way it is consumed differs in two respects. Each gap exposes a design choice that has no analogue in the control literature, and the three configurations we study fall out as their natural instantiations (Figure 1).

(i) At which positions the window is opened. Classical MPC solves the inner sub-problem for the *optimum* $u_t^{h*} = \arg\min_{u_t^h} J^h(t, x(t), u_t^h)$ and commits only its head $u^*(t)$ to the next state. In deep learning we never solve the inner problem; we merely take one gradient step on u, and a single window already produces a free gradient on *every* block it touches, so we accumulate those gradients on the whole window rather than committing only the head:

$$g_h(u(t')) = \sum_{t \in \mathcal{S}} \mathbf{1}\big[t' \in \{t, \ldots, t+h-1\}\big] \, \nabla_{u(t')} J^h(t, x(t), u_t^h), \tag{9}$$

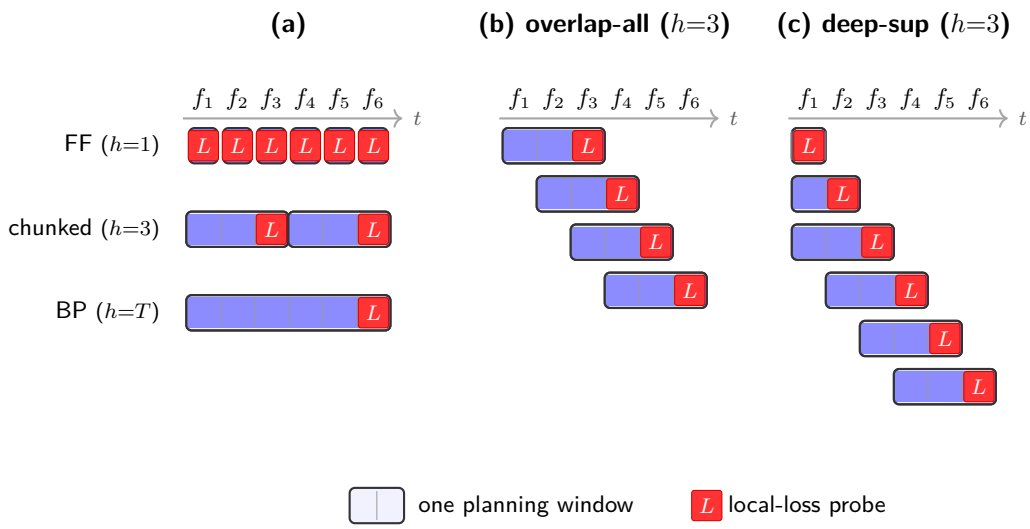

Figure 1: The three MPC backends (chunked, ch; overlap-all, oa; deep-supervision, ds) on a toy network of T=6 blocks at horizon h=3, with h=1 (FF) and h=T (BP) as the two limiting cases. Each column shows the windows that the corresponding algorithm(s) open on the block axis at its top: every bar is one planning window, a red box marked L is a local-loss probe attached at that depth (a small classifier head reading off the intermediate activation, used as a surrogate for the terminal loss when the window does not reach the network's output); every block covered by a window receives a gradient contribution from that window. Column (a) groups the three single-row cases: FF collapses every window to one self-probed block, chunked tiles the axis with $\lceil T/h \rceil$ disjoint windows, and BP is a single full-length window. Column (b) overlap-all uses $T-h+1$ stride-1 windows (staircase), so each block accumulates contributions from up to h shifted windows. Column (c) deep-sup reuses the overlap-all stride-1 schedule (bottom four bars) and adds $h-1$ extra short prefix windows of lengths $1, 2, \ldots, h-1$ (top two bars), so that block 1 already receives a gradient signal from depth 1 instead of waiting until depth h; equivalently, deep-sup attaches a local-loss probe at the output of every block.

where $u_t^h = \{u(t), \ldots, u(\min(t+h-1, T-1))\}$ are the parameters inside the length-h window opened at block t, and $\mathcal{S} \subseteq \{0, 1, \ldots, T-h\}$ is the set of admitted window starts. Two natural instantiations follow: overlap-all keeps stride 1, so each parameter accumulates contributions from up to h shifted windows; chunked advances the window by h at a time, $\mathcal{S} = \{0, h, 2h, \ldots\}$, so the block axis is tiled by $\lceil T/h \rceil$ disjoint windows. Both reduce to a single one-step gradient (FF, $g_{\mathrm{FF}} = g_1$) at h=1 and to a single full BP pass ($g_{\mathrm{BP}} = g_T$) at h=T.

(ii) How the trajectory loss is defined. Both backends above inherit the telescoping choice Eq. (8), $l(t, x(t), u(t)) = L(x(t+1)) - L(x(t))$, which makes the per-window objective J^h collapse to a single terminal probe $L(x(t+h)) - L(x(t))$ and thereby reproduces standard BP at h=T. Telescoping is a *convenience*, not a requirement: the only constraint imposed by Eq. (2) is that $\sum_t l(t, x(t), u(t))$ recovers a useful global objective. Dropping the difference $-L(x(t))$ and keeping just the look-ahead head gives the alternative design

$$\tilde{l}(t, x(t), u(t)) \;\triangleq\; L\big(f_t(x(t), u(t))\big) \;=\; L(x(t+1)), \tag{10}$$

whose sum $\sum_{t=0}^{T-1} \tilde{l}(t, x(t), u(t)) = \sum_{k=1}^{T} L(x(k))$ is the classical deep-supervision objective. Plugging $\tilde{l}$ into the receding-horizon formulation Eq. (9) with stride 1 yields

$$J^h_{\text{deep-sup}} \;=\; \sum_{k=1}^{T} L\big(x(k)\big), \qquad g^{\text{deep-sup}}(u(t')) \;=\; \sum_{k=t'+1}^{\min(t'+h,T)} \nabla_{u(t')} L\big(x(k)\big),$$

i.e. a terminal-cost head is attached at the output of *every* block and each head back-propagates through at most h blocks; at h=T this recovers standard deep supervision (Lee et al., 2015; Szegedy et al., 2015)

Algorithm 1 MPC training (backend, horizon h).

Require: block chain $\{f_t\}_{t=0}^{T-1}$; terminal loss L with per-cut probe heads (Appendix C.1); horizon $h \in \{1, \ldots, T\}$; backend $\in \{\text{chunked}, \text{overlap-all}, \text{deep-sup}\}$; learning rate η; iterations N.

Backend setup. Fix the window-start set $\mathcal{S}$ and trajectory loss l:
- chunked: $\mathcal{S} \leftarrow \{0, h, 2h, \ldots\}$; $l(t, x, u) \leftarrow L(x(t+1)) - L(x(t))$;
- overlap-all: $\mathcal{S} \leftarrow \{0, 1, \ldots, T-h\}$; same telescoping l;
- deep-sup: $\mathcal{S} \leftarrow \{0, 1, \ldots, T-h\}$ plus short prefix windows of lengths $1, \ldots, h-1$; $l(t, x, u) \leftarrow L(x(t+1))$.

for $\tau = 1$ to N **do**
  Sample a batch $(x(0), y)$; forward pass to obtain $x(1), \ldots, x(T)$.
  Zero the per-block gradient buffer: $g_h(u(t)) \leftarrow 0$ for $t = 0, \ldots, T-1$.
  for $s \in \mathcal{S}$ **do**
    Form the window objective $J^h(s, x(s), u_s^h) = \sum_{k=s}^{s+h-1} l(k, x(k), u(k))$.
    for $t \in \{s, \ldots, s+h-1\}$ **do**
      $g_h(u(t)) \mathrel{+}= \nabla_{u(t)} J^h(s, x(s), u_s^h)$.
    end for
  end for
  Optimiser step: $u \leftarrow u - \eta\, g_h(u)$.
end for

(every block has its own head) rather than full BP. In window-layout terms this is exactly overlap-all stride-1 *plus* $h-1$ extra short prefix windows of lengths $1, 2, \ldots, h-1$, so that block 1 already receives a look-ahead signal from depth 1 instead of waiting until depth h (see the deep-sup column of Figure 1). Because $\tilde{l}$ no longer telescopes, deep-sup falls outside the single-probe receding-horizon family that Theorem 3.3 analyses; we include it as the classical deep-supervision baseline against which the planner backends are compared in Section 4.3. A worked example of all three backends on a small network is given in Appendix A.

Algorithm 1 packages the three instantiations into a single training loop: with the horizon h and the backend fixed, the only per-iteration work beyond a standard optimiser step is to compute the window-averaged gradient $g_h(u)$ from Eq. (9). We refer to any run of Algorithm 1 as *MPC training* in the sequel.

Remark 3.1. *Other algorithms using local loss (e.g. (Nøkland & Eidnes, 2019; Belilovsky et al., 2019)) can also be seen as MPC framework with horizon $h = 1$. The LoCo algorithm (Xiong et al., 2020), which propagates the gradient through two adjacent blocks at every position, is exactly the* overlap-all *backend at $h = 2$ on the same block partition.*

Memory cost. Beyond unifying these existing local-loss algorithms as special cases (Remark 3.1), the MPC framework is also advantageous in terms of memory usage since it considers only part of the model in the computation of J^h. In practice, the memory demand for training a deep learning model largely depend on the need to store intermediate value for Back-Propagation, theoretically proportional to the depth of the model. Therefore, the memory usage will be of linear growth in horizon h, i.e.:

$$M(h) = ah + b, \tag{11}$$

where $M(h)$ is memory usage for horizon h, and a, b are independent of h[2]. This linear scaling is directly verified by the peak-memory measurements in Figure 5 (Section 4.5); Appendix D additionally dissects the implementation-level deviations from Eq. (11) observed on the non-uniform ResNet-50 backbones (residual-block heterogeneity plus CUDA caching-allocator effects flatten the chunked envelope at large h, without changing the $O(h)$ trend).

Performance–memory trade-off. Empirically, the MPC framework realizes the trade-off between performance and memory demand by providing a spectrum of optimization algorithms with varying horizons h. A larger horizon provides an accurate gradient, while a smaller horizon gives memory efficiency. This idea helps to understand the validity of the FF algorithm and its reduced accuracy compared to BP. However, there is still a lack of theoretical analysis on the influence of horizon h on accuracy.

[2]Influence of other factors like batch size are assumed to be constant and absorbed in coefficients a and b.

Remark 3.2. *In the field of control theory, finite time problems are rarely studied in the MPC setting. Existing results on the performance of MPC are primarily either asymptotic or rely on properties that are challenging to verify, or are not suitable for deep learning. For a more detailed introduction to traditional MPC methods, interested readers may refer to Grüne & Pannek (2017).*

In the following sections, we provide theoretical results on the influence of horizon h on deep linear networks (Theorem 3.3; extending to nonlinear residual networks with bounded block Jacobians, Corollary 3.6), and propose a horizon selection algorithm for the given objective.

3.3 Theoretical Analysis on Deep Linear Networks

From the previous discussion, the performance–memory trade-off rests on two established extremes—full BP (h=T) provides the exact gradient at the highest memory cost, and FF (h=1) is the cheapest in memory but produces a heavily biased gradient—together with one implicit claim about their interpolation: that as the discrete horizon h grows from 1 toward T, the MPC gradient g_h becomes progressively closer to its backend-matched full-horizon reference (full BP for chunked, reweighted BP for overlap-all, and full deep-supervision for deep-sup), monotonically closing the accuracy gap between the two extremes. In this section we make that claim precise on a deep linear network by bounding the gradient-cosine deviation between g_h and its reference as a function of the truncation depth $\delta \triangleq 1 - h/T$; the same rates extend to nonlinear residual networks with bounded block Jacobians (Corollary 3.6).

We consider a simple Linear NN problem with linear fully connected layers and quadratic loss, which is widely used in the theoretical analysis of deep learning (Arora et al., 2018; Cohen et al., 2023). The parameter $u(t)$ is the weight $W(t)$ in the t-th layer, omitting bias for brevity:

$$f_t(x(t), u(t)) = W(t)x(t), \;\; L(x(T)) = \frac{1}{2}\|x(T) - y\|_2^2, \tag{12}$$

where $(x(0), y) \in \mathbb{R}^n \times \mathbb{R}^n$ is one sample of a input-label pair in the dataset, $x(t) \in \mathbb{R}^n$ has the same dimension for all $0 \le t \le T$, $W(t) \in \mathbb{R}^{n\times n}$.

We aim to analyze the deviation between the gradient obtained by the MPC framework g_h and the true gradient g_T. Considering that the difference in the scale of the gradient norm can be compensated by adjusting the learning rate, the minimum deviation between the rescaled gradient g_h and true gradient g_T can be determined by the angle between these two vectors.

$$\min_{c_h} \|c_h g_h - g_T\|_2 = \left\| \frac{\|g_T\|_2}{\|g_h\|_2} \cos(\theta_h) g_h - g_T \right\|_2 = \sin(\theta_h)\|g_T\|_2, \tag{13}$$

where $\theta_h = \arccos\big(\frac{\langle g_h, g_T\rangle}{\|g_h\|_2\|g_T\|_2}\big)$ denotes the angle between g_h and g_T.

We further assume the dataset is whitened, i.e. the empirical (uncentered) covariance matrix for input data $\{x(0)\}$ is equal to identity as (Arora et al., 2019). In this case, the problem is equivalent to $L(u) = \frac{1}{2}\|W - \Phi\|_F^2$, where $W \triangleq W(T-1)\cdots W(0)$, Φ is the empirical (uncentered) cross-covariance matrix between inputs $\{x(0)\}$ and labels $\{y\}$, treated as a fixed matrix evaluated on the full whitened batch. We have the following finite-T result for the gradient deviation in the deep linear network in the near-identity regime.

Theorem 3.3 (Gradient-deviation bounds for the three MPC backends)**.** *Let $W(t) = I + \frac{1}{T}\tilde{W}(t)$ for matrices $\{\tilde{W}(t)\}$ with $\|\tilde{W}(t)\|_2 \le c$, let $g_h^\bullet$ denote the per-block gradient produced by backend $\bullet \in \{$oa, ds, ch$\}$ (overlap-all, deep-supervision, chunked, respectively), and set $\delta \triangleq 1 - h/T$. For overlap-all, denote by $c_t^{oa}(h) \triangleq \min(t, T-h) - \max(0, t-h+1) + 1 = \min(t+1, h, T-t)$ the number of stride-1 windows of length h that cover block t. Then there exist constants $K_\bullet(c, \Phi) > 0$, depending only on c and Φ (and not on T or h), such that for every $T \ge 1$ and every $h \in \{1, \dots, T\}$:*

(a) Overlap-all vs. reweighted BP. *Define the per-block reweighted reference $\tilde{g}_T^{oa}(t) \triangleq c_t^{oa}(h)\, g_T(t)$. Then* $1 - \cos^2\big(\angle(g_h^{oa}, \tilde{g}_T^{oa})\big) \;\le\; K_{oa}(c, \Phi)\,\delta^2 + O(\delta^3)$.

(b) Deep-sup vs. full deep supervision. *Let g_T^{ds} denote the deep-supervision gradient at $h{=}T$ (standard full deep supervision; Lee et al., 2015; Szegedy et al., 2015). Then* $1-\cos^2\big(\angle(g_h^{\mathsf{ds}}, g_T^{\mathsf{ds}})\big) \;\le\; K_{\mathsf{ds}}(c,\Phi)\,\delta^3 + O(\delta^4)$.

(c) Chunked vs. BP. $1-\cos^2\big(\angle(g_h^{\mathsf{ch}}, g_T)\big) \;\le\; K_{\mathsf{ch}}(c,\Phi)\,\delta^2 + O(\delta^3)$.

All constants satisfy $K_\bullet(c,\Phi) \le \kappa_0\, c^2 e^{6c}(e^c + \|\Phi\|_F)^2/m(c,\Phi)^2$ for an absolute constant κ_0, where $m(c,\Phi) > 0$ is a uniform lower bound on $\|g_T(t)\|_F$.

Theorem 3.3 confirms the interpolation claim made at the start of §3.3 and gives a quantitative rate for the accuracy leg of the trade-off: the gradient deviation between g_h and its backend-matched reference shrinks as h grows, and does so *polynomially* in $\delta = 1 - h/T$ (cubically for deep-supervision, quadratically for overlap-all and chunked). Hence the marginal gain in gradient accuracy from an extra horizon step diminishes rapidly as $h \to T$, while memory grows linearly in h by Eq. (11), so a non-trivial *interior* horizon should dominate any reasonable accuracy–memory objective—this is the diminishing-returns picture that drives the horizon selection in Section 4.

The bounds are deterministic and finite-T: the only hypothesis is the 2-norm bound on $\{\tilde{W}(t)\}$. The proof strategy (Appendices B.2–B.3) proceeds in three steps. *(i)* A Neumann expansion of the path Jacobian plus Lipschitz estimates yields a single-probe deviation lemma (Lemma B.4) that bounds the Frobenius norm of one length-h window's gradient against its full-BP counterpart. *(ii)* Each backend's per-block gradient deviation is stitched together from single-probe contributions by summing over the backend's window/probe pattern. *(iii)* Lemma B.3 (cosine via residue) converts the Frobenius bound into the cosine bound via the elementary identity $1-\cos^2(g, g+\Delta) \le \|\Delta\|^2/\|g\|^2$, whose denominator is lower-bounded by the reference-gradient estimate $\|g_T\|_F^2 \ge T\, m(c,\Phi)^2$ (Lemma B.2).

Remark 3.4 (Per-block rescaling and Adam-invariance)**.** *The reference in part (a) is BP rescaled by the overlap-all window-multiplicity profile $c_t^{\mathsf{oa}}(h)$. Under per-coordinate adaptive optimisers (Adam, AdamW, RMSProp) any per-block* scalar *rescaling of the gradient is invariant to the update, so g_h^{oa} and $\tilde{g}_T^{\mathsf{oa}}$ induce the same parameter step as g_h^{oa} and g_T; the rescaled cosine in (a) is therefore the operationally relevant alignment for the Adam-trained networks of Section 4. Under plain SGD the multiplicity profile contributes an extra constant gap on top of the δ^2 rate.*

Remark 3.5 (Rate ordering predicted by the bounds)**.** *The rate gap between* deep-sup *and the other two backends comes from the* reference *each backend is measured against.* Chunked *and* overlap-all *are compared to a* single-probe *reference (BP at depth T, possibly reweighted);* deep-sup *is compared to a* multi-probe *reference (full deep supervision, i.e. one probe per depth summed at each block). In the near-identity regime the multi-probe reference has squared norm of order T^3, versus T for the single-probe references, whereas truncating from $h{=}T$ to $h{<}T$ removes only the deepest $T{-}h$ probes per block and contributes squared deviation of order $T^3\delta^3$ (versus $T\delta^2$). Taking the ratio,* deep-sup *attains δ^3 while the two single-probe backends get δ^2.*

Within the quadratic pair, overlap-all *is strictly tighter than* chunked *at any finite T: its stride-1 tiling under-supervises only the $T{-}h$ blocks near the two boundaries, whereas* chunked*'s stride-h tiling under-supervises the entire first chunk of h blocks.*

The proof of Theorem 3.3 uses the matrices $\{\tilde{W}(t)\}$ only through the operator-norm bound $\|\tilde{W}(t)\|_2 \le c$ and the Neumann expansion of the path Jacobian, both of which hold abstractly for any sequence of matrices $A(t)$ with $\|A(t)\|_2 \le L$, regardless of whether $A(t)$ depends on $(x(s), u(s))_{s\le t}$. This immediately yields a unified extension to nonlinear residual networks.

Corollary 3.6 (Unified extension to nonlinear residual networks)**.** *Consider any residual network*

$$f_t(x,u) \;=\; x + \tfrac{1}{T}\phi_t(x,u), \qquad A(t) \triangleq \partial_x \phi_t(x(t),u(t)), \qquad \|A(t)\|_2 \le L \text{ along the trajectory.}$$

Then the conclusions of Theorem 3.3 hold with the spectral-radius constant c replaced by L and the same cubic / quadratic exponents in δ. When ϕ_t is only piecewise smooth (e.g. ReLU), $\partial_x\phi_t$ is interpreted as any measurable selection from the Clarke generalised Jacobian at $(x(t),u(t))$; the trajectory-level bound $\|A(t)\|_2 \le L$ then need hold only for the realised activation pattern along the training trajectory (Appendix B.4).

Concrete instances covered are ReLU residual blocks ($\phi_t = \sigma(W(t)x + b(t))$, $\|W(t)\|_2 \le c$), smooth-activation MLPs, Pre-LayerNorm Transformer blocks, and the ResNet-50 bottleneck; each instance is verified in two lines (Appendix B.4) by checking $\|A(t)\|_2 \le L$ for the appropriate L.

The polynomial relationship between $\cos(\theta_h)$ and h is also observed for nonlinear cases (Section 4.2). Combined with the biased-gradient analysis of Bottou et al. (2018); Karimi et al. (2016), this yields, under the Polyak–Łojasiewicz inequality, linear convergence of the loss to a non-vanishing fixed point with bias $O\big(1 - \cos^2\theta_h\big)$ (Appendix B.5). Concretely, defining the loss-progress ratio

$$r(h) = \frac{J_0 - J_h(\tau)}{J_0 - J_T(\tau)}, \tag{14}$$

the biased-PL bound predicts $1 - r(h) = O\big(1 - \cos^2\theta_h\big)$. We use this asymptotic loss–cosine link to motivate interior-horizon choices in the empirical study (Section 4).

Theorem 3.3 indicates that $\cos^2(\theta_h)$ at h=1 is generically bounded away from both 0 and 1 with non-zero derivative (numerical verification in Section 4.2), so FF is typically sub-optimal at the low end just as BP is at the high end. This confirms that the accuracy–memory optimum lies at an interior horizon; a concrete illustration follows.

For instance, suppose the cost is linear in memory, $C(M(h)) = ah + b$, and assume the cubic asymptote of Theorem 3.3 holds globally: $\cos^2(\theta_h) = 1 - c(1 - h/T)^3$ for h in a neighbourhood of T. (This is a heuristic extrapolation of the asymptotic bound to all h.) With objective $-r(h) + \lambda C(M(h))$, $a, b, c > 0$, the first-order condition

$$\frac{3c}{T^3}(T - h^\star)^2 \;=\; a\lambda$$

gives the unique interior optimum

$$h^\star \;=\; T\Big(1 - \sqrt{\tfrac{a\lambda T}{3c}}\Big),$$

which lies strictly between 1 and $T - 1$ when $\frac{3c}{aT^3} < \lambda < \frac{3c(T-2)^2}{aT^3}$. Neither FF ($h = 1$) nor BP ($h = T$) is optimal in this regime.

This example illustrates that the optimal horizon depends on various factors such as the model, dataset, task, and objective. Empirical horizon sweeps under several backbones and objectives are reported in Section 4.

4 Experiments

We evaluate MPC training—i.e. training under one of the three MPC backends of Algorithm 1 at horizon $h \in \{1, \dots, T\}$—on two image-classification backbones (ViT-b/16 fine-tuned with LoRA on CIFAR-100; ResNet-50 trained from scratch on CIFAR-100 in two stage granularities) and on RoBERTa-base fine-tuned with LoRA on four GLUE tasks. Across all settings we sweep the planning horizon h, three planner backends (chunked, overlap-all, deep-supervision), and a backbone-specific terminal probe (headprobe for ViT and RoBERTa, classifier-bank probes shared / stagesh for the two R50 cuts; see Appendix C.1). h=T recovers full back-propagation as a control.

4.1 Setup

For CIFAR-100 (50k/10k) we use AdamW, batch 128, bf16 autocast. ViT-b/16 fine-tuning follows Hu et al. (2021)-style LoRA (r=8, α=16; target modules `qkv` and attention output projection `proj`) and is initialised from `timm` pretrained weights; ResNet-50 is trained from scratch (no LoRA, no pretraining). The MPC chain length T depends on the granularity at which the backbone is cut into blocks. For ResNet-50 (R50 hereafter), we use two granularities: *block-cut* treats the stem and each of the 16 bottleneck blocks of the standard $[3, 4, 6, 3]$ layout as its own MPC block (T=1+16=17), while *stage-cut* groups the bottlenecks into the four ResNet stages and adds the stem (T=1+4=5; the classification head is excluded from the chain, as it is on ViT and RoBERTa below). ViT-b/16 and RoBERTa-base both have 12 transformer blocks, so cutting after every block gives T=12 (the patch embedding on ViT and the classifier heads on both are excluded from the

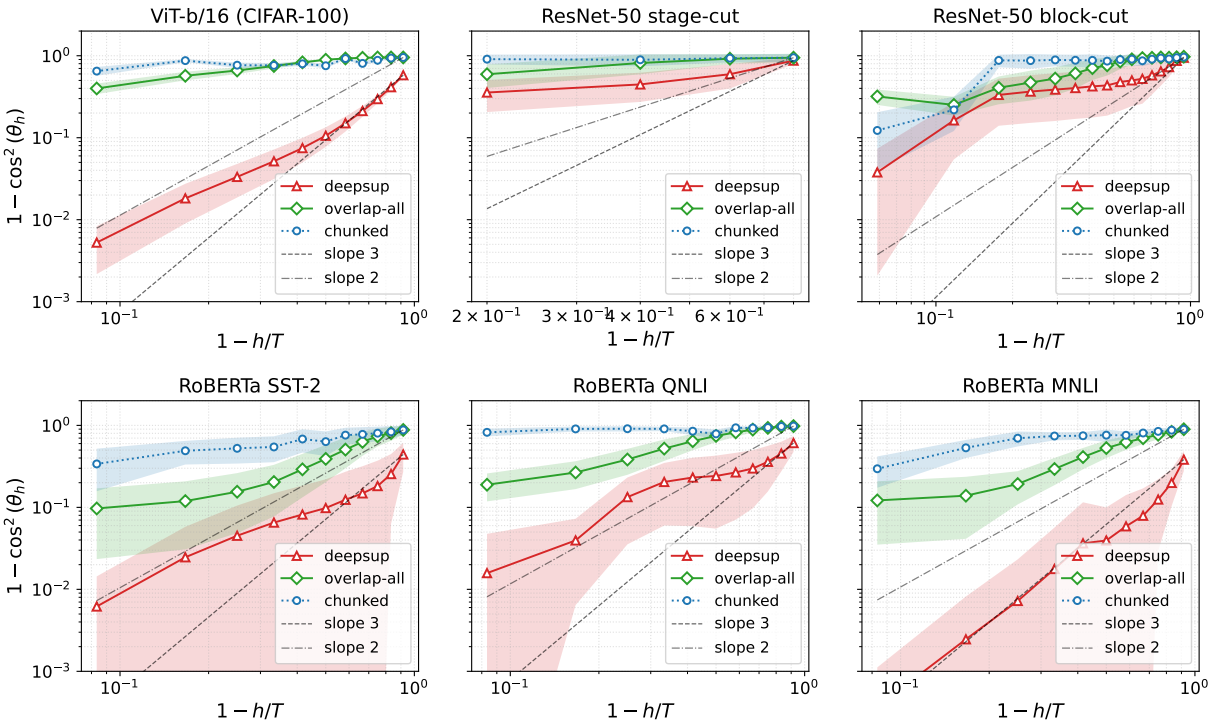

Figure 2: Empirical gradient-deviation cosine $\cos\theta_h = \langle g_h, g_{\text{ref}}\rangle/(\|g_h\|\|g_{\text{ref}}\|)$ on a held-out batch, with backend-matched references: full BP for chunked; multiplicity-rescaled BP (Theorem 3.3(a)) for overlap-all; deep-supervision at $h{=}T$ for deepsup. Probe regime: 200 BP steps of probe warm-up before sampling. Axes are $1-\cos^2(\theta_h)$ against $1-h/T$ on a log–log scale. The dashed reference has slope 3 (cubic bound of Theorem 3.3 for deep-sup); the dash-dotted reference has slope 2 (quadratic bound for overlap-all and chunked).

chain). All MPC backends share an identical training script; the only runtime difference is the planner kernel. As a control, we run full back-propagation ($h{=}T$) for every backbone and report it as *full BP*. For RoBERTa we follow the LoRA paper's GLUE recipe (Table 9 of Hu et al., 2021; LoRA on W_q, W_v) on MNLI, SST-2, QNLI and MRPC, and report best validation accuracy.

Each MPC run also pairs the planner with a probe that maps the intermediate activation at the end of each cut into the loss; we fix one probe per backbone: headprobe (a shared copy of the pretrained classifier head) for ViT and for RoBERTa, and the classifier-bank probes shared/stagesh (a single learned head shared across cuts, with a per-cut or per-stage learned channel projection) for R50 stage-cut and R50 block-cut respectively. See Appendix C.1 for the precise probe structure; the per-backbone choice follows architectural feasibility (headprobe reuses the pretrained classifier head, which is available for the uniform-width backbones ViT and RoBERTa; R50 has varying channel widths across cuts and therefore uses the learned shared/stagesh bank) and is held fixed across the $(h, \text{backend})$ grid.

4.2 Polynomial decay of the gradient-deviation angle

Theorem 3.3, extended to nonlinear residual networks by Corollary 3.6 (which is the form directly applicable to the ViT, ResNet-50 and RoBERTa backbones used here), predicts that the squared sine of the angle between the truncated gradient g_h and a backend-matched reference gradient decays as $1-\cos^2(\theta_h) = O\big((1-h/T)^3\big)$ for the deep-sup backend (part (b), measured against full deep supervision), and as the quadratic bound $1-\cos^2(\theta_h) = O\big((1-h/T)^2\big)$ for overlap-all (part (a), against the multiplicity-rescaled BP reference) and for the chunked backend (part (c), with a matching $\Omega(\delta^2)$ lower bound). We test both predictions on the

same pretrained backbones used in the rest of this section; residual structure holds block-by-block, whereas the block-Jacobian norm bound $\|A(t)\|_2 \leq L$ required by Corollary 3.6 is a working assumption that we do not verify directly. At a fixed sequence of training checkpoints we compute, on a held-out batch and a fixed mini-batch of training examples,

$$\cos\theta_h \;=\; \frac{\langle g_h,\, g_{\mathrm{ref}}\rangle}{\|g_h\|\,\|g_{\mathrm{ref}}\|}\,, \qquad h = 1, 2, \ldots, T,$$

where the inner product is taken over all parameters that the chunked backend would update at horizon h, and the reference g_{ref} is *backend-matched*: full BP for the chunked backend (whose theorem compares against full BP), the multiplicity-rescaled BP $\tilde{g}_T^{\mathsf{oa}}$ of Theorem 3.3(a) for overlap-all, and full deep-supervision at $h{=}T$ for deep-sup (whose theorem compares against full deep supervision; using BP here would conflate the architectural asymmetry of (b) with the truncation effect we want to measure). Figure 2 reports $1-\cos^2(\theta_h)$ versus $1-h/T$ on a log–log scale together with the slope-3 (cubic) and slope-2 (quadratic) reference lines. The left portion of each panel (i.e. $h/T \to 1$, $1-h/T \to 0$) is the asymptotic regime in which the bounds apply. Across every panel the deep-sup curve is visibly steeper than the chunked and overlap-all curves and stays close to the slope-3 reference, while chunked and overlap-all group on the quadratic side of the slope-2/slope-3 dichotomy predicted by Theorem 3.3(a,b,c). The measured log–log slopes are nonetheless well below the asymptotic 2 and 3 in every panel; we discuss the mechanisms below.

A closer look shows that the measured log–log slopes are usually somewhat below the theoretical 3 for deep-sup (and below 2 for overlap-all and chunked), so the empirical exponents should be read as effective rather than asymptotic rates. Two mechanisms contribute. (i) **Pre-asymptotic regime:** the next-order correction is $O(\delta^2)$ for deep-sup, $O(\delta)$ for overlap-all, and $O(\delta)$ for chunked ($\delta \triangleq 1-h/T$), and with only $T \in \{5, 12, 17\}$ stages the smallest sampled δ is bounded away from 0, so the chord we measure bends toward the lower slope. (ii) **Cosine noise and normalisation layers:** finite-batch variance in $\cos\theta_h$ causes the usual errors-in-variables attenuation of log–log slopes, and BN/LN statistics shift between the truncated and full-BP forward passes, adding a horizon- independent offset that flattens the apparent slope. Neither effect breaks the qualitative two-rate prediction: deep-sup remains visibly steeper than overlap-all and chunked in every panel.

4.3 Test accuracy across the horizon sweep

Figure 3 reports best-test accuracy as a function of the planning horizon h for all seven (backbone, task) pairs of the MPC sweep: the three CIFAR-100 image backbones (top row) and RoBERTa-base+LoRA on the four GLUE tasks (bottom row). Each panel overlays the three MPC backends and the full-BP reference (dashed grey line, = chunked at $h{=}T$).

Two findings, both directly mirroring the polynomial gap-closure rate $1-\cos^2\theta_h = O(\delta^p)$ proved in Theorem 3.3. (i) *Test accuracy approaches full BP polynomially fast in $\delta{=}1-h/T$, with strongly diminishing returns.* Looking at the best-of-three-backends cell at each h and tracking the fraction of the FF($h{=}1$)–BP accuracy gap that has been closed, every panel exhibits the same diminishing-returns pattern: the first few horizon steps are paid back richly, but the marginal gain shrinks rapidly as $h \to T$. At the quarter-horizon $\delta \approx 3/4$ ($h{=}2, 3, 4$ depending on T) the gap is already 25–70% closed; by $\delta \approx 2/3$ ($h \approx T/3$) it is 80–98% closed on every task; e.g. QNLI hits 97.8% closure at $h{=}4$, ViT 93.5% at $h{=}4$, R50-block 93.9% at $h{=}8$, and on MNLI the residual gap is already only 2.16 pp at $h{=}4$. The remaining quarter-to-half of the budget (from $h \approx T/2$ to $h{=}T$) is largely wasted, exactly the behaviour predicted by the $O(\delta^p)$ bound: per-step gains behave as $\delta^p - (\delta - 1/T)^p \sim p\delta^{p-1}/T$ and vanish polynomially as $\delta \to 0$. Combined with the linear memory cost of Eq. (11), this is the empirical evidence for the interior optimum $h^\star \in (1, T)$ argued for in Section 3.3. (ii) *The crossing tail: MPC often* beats *BP at large but still sub-T horizons.* On five of the seven tasks (both ResNet-50 cuts, MRPC, SST-2 and MNLI) the best $h{<}T$ cell strictly exceeds full BP by +0.41 to +1.28 pp; on the remaining two (ViT and QNLI) it falls short by at most 0.3 pp. The largest single gain, +1.28 pp on R50-stage at $h{=}4$ (overlap-all), and the runner-up, +0.93 pp on R50-block at $h{=}14$ (overlap-all), are both *below* their respective T, demonstrating that the implicit regularisation provided by truncated planning can be a strict accuracy benefit, not merely a memory trade. Within the deep-supervision backend the $h{=}T$ point

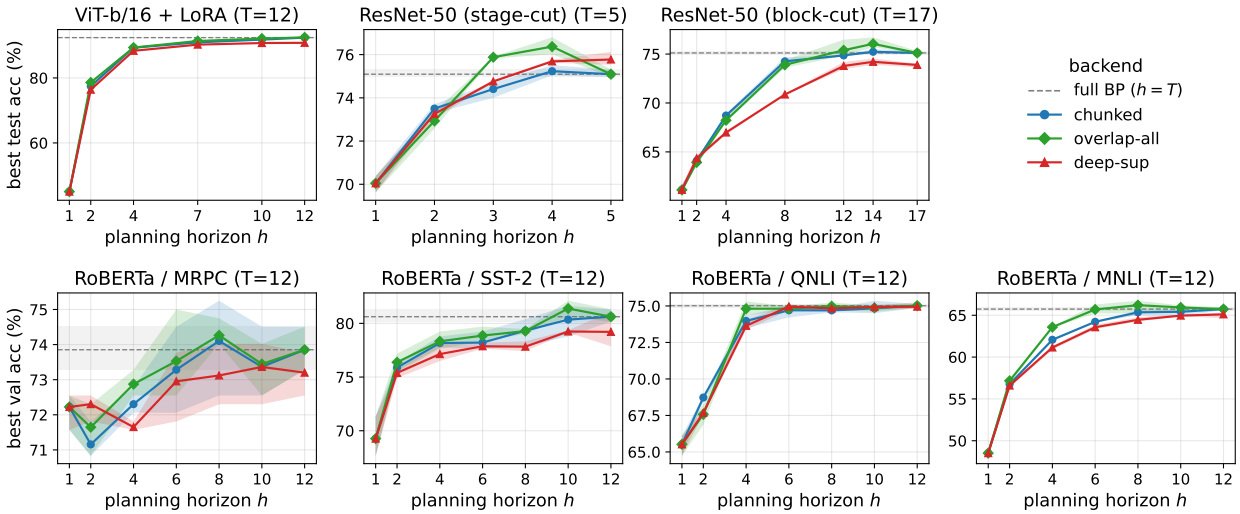

Figure 3: Best-test accuracy (%) versus planning horizon h on the MPC backend sweep. **Top:** CIFAR-100 image backbones (ViT-b/16 + LoRA, ResNet-50 stage-cut, ResNet-50 block-cut). **Bottom:** RoBERTa-base + LoRA on the four GLUE tasks (MRPC, SST-2, QNLI, MNLI). Dashed grey: full BP reference (= chunked at $h{=}T$). Markers: three MPC backends (chunked, overlap-all, deep-supervision). At $h{=}T$ the two non-deep-sup backends are algorithmically identical to plain BP, so only one BP marker (chunked at $h{=}T$) is plotted as the reference and the redundant overlap-all marker is suppressed.

beats full BP only on R50-stage (+0.68 pp) and ties on QNLI; elsewhere it loses 0.6–1.6 pp, consistent with multi-loss regularisation helping only when the per-stage probes have low variance relative to the head loss.

4.4 Loss-progress ratio tracks the PL bound

The biased-PL analysis (Appendix B.5, summarised in Section 3.2) predicts that the per-iteration loss-progress ratio $r_{\text{loss}}(h) \triangleq \big(J_0 - J_h(\tau)\big)/\big(J_0 - J_T(\tau)\big)$ satisfies $1 - r_{\text{loss}}(h) = O\big(1 - \cos^2\theta_h\big)$ as $\theta_h \to 0$, where $J_h(\tau)$ is the final training loss of the h-horizon run and $J_T(\tau)$ is the same-backend reference at the deepest available horizon. Combined with the cubic / quadratic decay of $1 - \cos^2\theta_h$ in $1 - h/T$ (Theorem 3.3), the PL bound therefore predicts that the loss-progress deficit itself decays polynomially in $1 - h/T$, with rate at least cubic for deep-sup and at least quadratic for overlap-all and chunked. Figure 4 verifies this scaling on the three image backbones used in Section 4.2: it plots the loss-progress deficit $1 - r_{\text{loss}}(h)$ computed from the end-to-end training logs against the horizon-truncation depth $1 - h/T$, with $J_T(\tau)$ taken as the same-backend training loss at $h{=}T$ on each backbone. The bottom row repeats the test with the validation-accuracy deficit $1 - r_{\text{acc}}(h) \triangleq (A_T - A_h)/(A_T - A_0)$ in place of the loss; this quantity has *no* clean PL link (the PL inequality bounds training-loss progress, not generalisation) and is included only for empirical comparison.

The figure delivers two messages. First, in every panel of the loss row the deficit decays as $1 - h/T$ shrinks, with empirical log–log slopes that are broadly consistent with the predicted rate ordering: the chunked backend is the flattest at $\hat{s}{\approx}2.7$ (matching its theoretical quadratic floor), deep-sup ($\hat{s}{\approx}3.09$) sits above its predicted δ^3 floor, and overlap-all ($\hat{s}{\approx}2.96$) lies between its theoretical δ^2 floor and the slope-3 curve. All three slopes sit slightly above their theoretical δ^2/δ^3 floors for the reasons already discussed in Section 4.2 (pre-asymptotic $O(\delta)$–$O(\delta^2)$ corrections plus finite-batch cosine noise); with only a handful of horizons per backbone the fitted chord is dominated by the next-order term. Second, the validation-accuracy row is included only for empirical comparison; the PL analysis bounds training-loss progress, not test accuracy, so no theoretical curve applies here. The observed decay is noisier and shallower than for the training loss because validation accuracy saturates near 100% at moderate h, which shrinks the deficit denominator $A_T{-}A_0$ and compresses the dynamic range of $1 - r_{\text{acc}}(h)$; the downward trend in $1 - h/T$ is nonetheless clear on every panel.

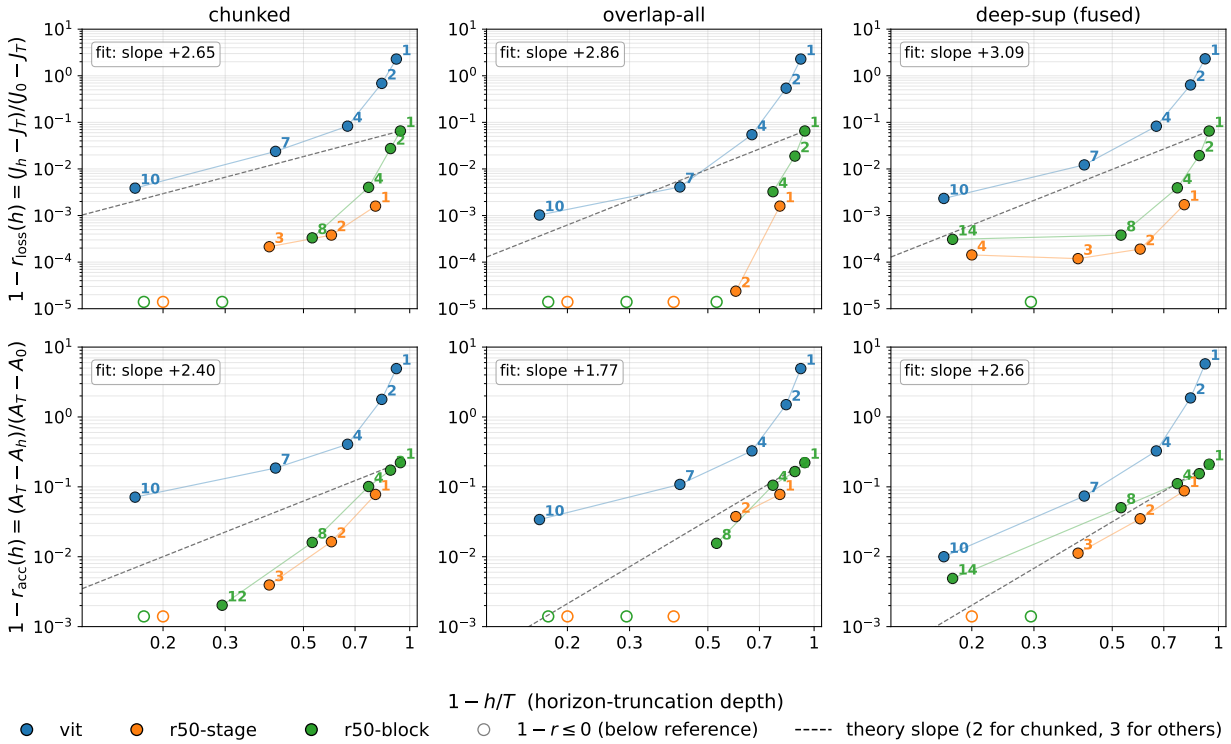

Figure 4: Progress deficit vs horizon-truncation depth $1 - h/T$, per backend. **Top row:** training-loss deficit $1 - r_{\text{loss}}(h) = (J_h - J_T)/(J_0 - J_T)$, the quantity bounded by the biased-PL analysis as $O(1 - \cos^2 \theta_h)$. **Bottom row:** validation-accuracy deficit $1 - r_{\text{acc}}(h) = (A_T - A_h)/(A_T - A_0)$, plotted for empirical comparison only; no PL bound applies. Each filled marker is one (model, horizon) cell with the horizon annotated; open markers (rare) flag runs whose final loss or accuracy *exceeded* the reference by an LR/seed quirk. Both deficits trend downward with $1 - h/T$ across every (backend, backbone) cell. The accuracy row is noisier and decays more gradually, consistent with the fact that validation accuracy saturates near the BP reference well before the training loss does.

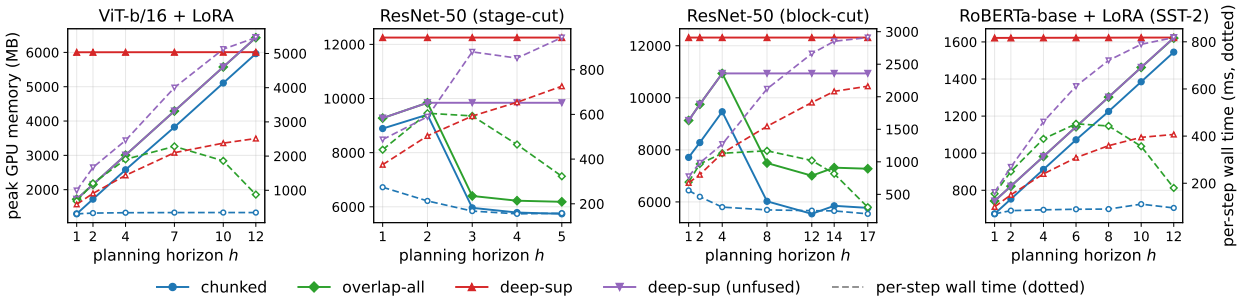

Figure 5: Peak GPU memory (solid, filled markers, left axis) and per-step wall time (dashed, hollow markers, right axis) versus planning horizon h on the three CIFAR-100 image backbones, batch 128, bf16, RTX 3090. Same markers and colours as Figure 3.

4.5 Compute and memory cost of MPC

Figure 5 plots peak GPU memory and per-step wall time against h for the three image backbones. On the two transformer backbones (ViT and RoBERTa), chunked memory scales linearly in h as Eq. (11) predicts, saving 30–80% against full BP; overlap-all tracks chunked's memory closely but pays a 1.5–2× step-time

premium for the extra forward passes; deep-supervision keeps the full activation graph alive and matches BP memory throughout. On the two R50 cuts chunked's savings largely disappear (within $\pm 5\%$ of BP once $h \geq T/3$), an artefact of ResNet block-width heterogeneity and the CUDA caching allocator. Appendix D explains this together with the other implementation-level quirks visible in the figure (chunked<overlap-all even at h=1; non-monotonic R50 memory; quadratic-in-T deep-supervision wall time).

5 Discussion

This study presents a systematic framework that integrates the Forward-Forward algorithm with Back-Propagation, offering a family of algorithms to train deep neural networks. Theoretical analysis on deep linear networks demonstrates that the gradient deviation decays as δ^3 for deep-supervision and as δ^2 for overlap-all and chunked (with a matching $\Omega(\delta^2)$ lower bound for the latter family) as the horizon approaches full BP, leading to diminishing returns for large horizons. Empirical sweeps across pretrained ViT, ResNet-50 and RoBERTa backbones confirm the picture and identify an interior horizon that recovers most of the BP accuracy at substantially reduced peak memory (roughly 30–80% of full BP, backbone- and backend-dependent) while keeping per-step wall time within a small constant factor of BP. Taken together, the theoretical rates and the empirical sweeps point toward a concrete deployment recipe.

Practical recommendation. The accuracy-vs.-cost picture in Section 4 suggests a simple decision rule for practitioners. When only *peak memory* is scarce (e.g. training a larger backbone or a larger batch on a fixed GPU), an interior horizon $h \approx T/2$ with the chunked backend recovers nearly all of the full-BP accuracy at 30–80% of its activation memory, and is the configuration we would default to. When *both* memory and wall-clock time are scarce, however, this same operating point is no longer obviously preferred: per-step wall time is roughly constant in h for chunked (so an interior h buys memory but not time), grows linearly in h for overlap-all (peaking near $h \approx T/2$), and is dominated by deep-supervision at $h = T$. In that regime a small horizon (h close to 1 with chunked, trading some accuracy for both memory and time) or full BP itself ($h = T$, paying the full memory cost but matching BP accuracy at BP wall time) are the more attractive operating points than a mid-horizon MPC step. In short: the interior horizon is the right pick whenever memory is the only bottleneck, and the endpoints of the sweep are the right pick when wall time also matters.

Limitations and outlook. The results above come with several caveats that also indicate natural directions for further study.

Firstly, the cubic / quadratic decay rates are proved in a near-identity regime: deep linear networks with block-Jacobian spectra clustered around the identity, and the nonlinear extensions of Appendix B.4–B.4 additionally require the per-block Jacobians (of ϕ_t w.r.t. its input, evaluated along the training trajectory) to stay close to a common reference. Removing this near-identity assumption, and in particular handling non-residual architectures whose block-Jacobians depart far from the identity, would require a different proof technique that controls the bias of the truncated gradient outside this regime.

Secondly, this paper does not propose an automatic objective-based horizon-selection routine; our experiments instead sweep h explicitly and report the empirical accuracy / memory frontier.

Thirdly, our study focuses solely on the horizon's influence, and does not examine other MPC hyperparameters such as block selection or the loss-splitting scheme.

We leave these investigations to future work.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv:1610.01644*, 2016.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization, June 2018. URL http://arxiv.org/abs/1802.06509. arXiv:1802.06509 [cs].

Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks, October 2019. URL http://arxiv.org/abs/1810.02281. arXiv:1810.02281 [cs, stat].

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy Layerwise Learning Can Scale to ImageNet, April 2019. URL http://arxiv.org/abs/1812.11446. arXiv:1812.11446 [cs, stat].

Dimitri P. Bertsekas. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena Scientific, 2020.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization Methods for Large-Scale Machine Learning, February 2018. URL http://arxiv.org/abs/1606.04838. arXiv:1606.04838 [cs, math, stat].

Nadav Cohen, Govind Menon, and Zsolt Veraszto. Deep Linear Networks for Matrix Completion – An Infinite Depth Limit, May 2023. URL http://arxiv.org/abs/2210.12497. arXiv:2210.12497 [cs, math].

Weinan E. A Proposal on Machine Learning via Dynamical Systems. *Communications in Mathematics and Statistics*, 5(1):1–11, March 2017. ISSN 2194-6701, 2194-671X. doi: 10/ggrtrp. URL http://link.springer.com/10.1007/s40304-017-0103-z.

Weinan E, Jiequn Han, and Jihao Long. Empowering Optimal Control with Machine Learning: A Perspective from Model Predictive Control. *IFAC-PapersOnLine*, 55(30):121–126, January 2022. ISSN 2405-8963. doi: 10/gsmnbx. URL https://www.sciencedirect.com/science/article/pii/S2405896322026672.

Klaus Greff, Rupesh K. Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. In *International Conference on Learning Representations (ICLR)*, 2017.

Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control*. Communications and Control Engineering. Springer International Publishing, Cham, 2017. ISBN 978-3-319-46023-9 978-3-319-46024-6. doi: 10.1007/978-3-319-46024-6. URL http://link.springer.com/10.1007/978-3-319-46024-6.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, December 2023. URL http://arxiv.org/abs/2312.00752. arXiv:2312.00752 [cs].

Geoffrey Hinton. The Forward-Forward Algorithm: Some Preliminary Investigations, December 2022. URL http://arxiv.org/abs/2212.13345. arXiv:2212.13345 [cs].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, June 2021. URL https://arxiv.org/abs/2106.09685v2.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled Neural Interfaces using Synthetic Gradients, July 2017. URL http://arxiv.org/abs/1608.05343. arXiv:1608.05343 [cs].

Stanisław Jastrzębski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. In *International Conference on Learning Representations (ICLR)*, 2018.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pp. 795–811, 2016.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 562–570, 2015.

Qianxiao Li, Long Chen, Cheng Tai, and Weinan E. Maximum Principle Based Algorithms for Deep Learning, June 2018. URL http://arxiv.org/abs/1710.09513. arXiv:1710.09513 [cs, stat].

D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000. ISSN 0005-1098. doi: 10.1016/s0005-1098(99)00214-9. URL https://www.sciencedirect.com/science/article/pii/S0005109899002149.

Hanna Michalska and David Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, 1993.

Tam Nguyen, César A. Uribe, Tan M. Nguyen, and Richard G. Baraniuk. PIDformer: Transformer Meets Control Theory, February 2024. URL http://arxiv.org/abs/2402.15989. arXiv:2402.15989 [cs, eess].

Arild Nø kland. Direct Feedback Alignment Provides Learning in Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/d490d7b4576290fa60eb31b5fc917ad1-Abstract.html.

Arild Nøkland and Lars Hiller Eidnes. Training Neural Networks with Local Error Signals, May 2019. URL http://arxiv.org/abs/1901.06656. arXiv:1901.06656 [cs, stat].

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, et al. GPT-4 Technical Report, March 2024. URL http://arxiv.org/abs/2303.08774. arXiv:2303.08774 [cs].

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0. Publisher: Nature Publishing Group.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, February 2023. URL http://arxiv.org/abs/2302.13971. arXiv:2302.13971 [cs].

Andreas Veit, Michael Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Boxi Weng, Jian Sun, Alireza Sadeghi, and Gang Wang. AdaPID: An Adaptive PID Optimizer for Training Deep Neural Networks. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3943–3947, May 2022. doi: 10/gsmnbq. ISSN: 2379-190X.

Yuwen Xiong, Mengye Ren, and Raquel Urtasun. Loco: Local contrastive representation learning. *Advances in neural information processing systems*, 33:11142–11153, 2020.

A An example of the MPC framework

We illustrate the three backends introduced in Section 3 on a concrete 5-layer residual MLP (T=5) with horizon h=3. At h=1 all three backends collapse to FF; at h=T the two single-probe backends collapse to BP while deep-sup recovers standard deep supervision (Lee et al., 2015), so the backend distinction is visible in the interior and (for deep-sup) at h=T as well.

Architecture and trajectory loss. Each layer is a fully-connected residual block,

$$f_t(x(t), u(t)) = x(t) + \sigma(\mathrm{Linear}(x(t); u(t))), \quad t = 0, \ldots, 4,$$

and the terminal cost is $L(x) = \mathrm{MSE}(x, y)$. The trajectory loss $l(t, x(t), u(t)) = L(x(t+1)) - L(x(t))$ telescopes, so for any window $[a, b]$ we have $\sum_{t=a}^{b-1} l(t, x(t), u(t)) = L(x(b)) - L(x(a))$. In what follows we write $g_h^{\mathrm{back}}(u(t))$ for the gradient produced at block t by backend back $\in \{\mathrm{cf}, \mathrm{oa}, \mathrm{ds}, \mathrm{BP}\}$ at horizon h.

Chunked (h=3). Disjoint windows of length h with stride h: $[0, 3]$ and $[3, 5]$ (the tail is truncated to length 2). Every block inside a chunk is updated against that chunk's terminal:

$$g_3^{\mathrm{cf}}(u(0)) = \nabla_{u(0)} L(x(3)), \quad g_3^{\mathrm{cf}}(u(1)) = \nabla_{u(1)} L(x(3)), \quad g_3^{\mathrm{cf}}(u(2)) = \nabla_{u(2)} L(x(3)),$$
$$g_3^{\mathrm{cf}}(u(3)) = \nabla_{u(3)} L(x(5)), \quad g_3^{\mathrm{cf}}(u(4)) = \nabla_{u(4)} L(x(5)).$$

Note that intermediate blocks "see" a target only $h - (t \bmod h)$ steps ahead, never the full horizon T.

Overlap-all (h=3). Sliding windows of length h with stride 1, attributing each window's gradient to *every* block it covers:

$$\begin{aligned}
g_3^{\mathrm{oa}}(u(0)) &= \nabla_{u(0)} L(x(3)), \\
g_3^{\mathrm{oa}}(u(1)) &= \nabla_{u(1)} L(x(3)) + \nabla_{u(1)} L(x(4)), \\
g_3^{\mathrm{oa}}(u(2)) &= \nabla_{u(2)} L(x(3)) + \nabla_{u(2)} L(x(4)) + \nabla_{u(2)} L(x(5)), \\
g_3^{\mathrm{oa}}(u(3)) &= \nabla_{u(3)} L(x(4)) + \nabla_{u(3)} L(x(5)), \\
g_3^{\mathrm{oa}}(u(4)) &= \nabla_{u(4)} L(x(5)).
\end{aligned}$$

A mid-network block now accumulates contributions from up to h look-aheads. This is the backend that arises from the receding-horizon MPC formulation; its quadratic decay (and the $\Omega(\delta^2)$ matching lower bound) is established in Section B.2. The LoCo algorithm of Xiong et al. (2020) is the special case h=2.

Deep-supervision variant (h=3). A terminal-cost head is attached at the output of *every* block and each head back-propagates through at most h preceding blocks, so block t accumulates the gradients of all heads $L(x(k))$ with $k \in (t, \min(t + h, T)]$:

$$\begin{aligned}
g_3^{\mathrm{ds}}(u(0)) &= \nabla_{u(0)} L(x(1)) + \nabla_{u(0)} L(x(2)) + \nabla_{u(0)} L(x(3)), \\
g_3^{\mathrm{ds}}(u(1)) &= \nabla_{u(1)} L(x(2)) + \nabla_{u(1)} L(x(3)) + \nabla_{u(1)} L(x(4)), \\
g_3^{\mathrm{ds}}(u(2)) &= \nabla_{u(2)} L(x(3)) + \nabla_{u(2)} L(x(4)) + \nabla_{u(2)} L(x(5)), \\
g_3^{\mathrm{ds}}(u(3)) &= \nabla_{u(3)} L(x(4)) + \nabla_{u(3)} L(x(5)), \\
g_3^{\mathrm{ds}}(u(4)) &= \nabla_{u(4)} L(x(5)).
\end{aligned}$$

This is exactly overlap-all *plus* the two extra short prefix windows of lengths 1 and 2 that supply block 0 with the look-ahead signals $L(x(1))$ and $L(x(2))$ and block 1 with $L(x(2))$; overlap-all delays these signals until depth h. Unlike the other three backends, deep-sup does not collapse to BP at h=T: instead each block keeps its own head and the scheme reduces to standard deep supervision (Lee et al., 2015).

Two limiting horizons. At h=1 every backend (including deep-sup) reduces to the Forward-Forward style local update $g_1(u(t)) = \nabla_{u(t)} L(f_t(x(t), u(t)))$, $t = 0, \ldots, 4$, because each window/head covers a single block. At h=T=5 the two single-probe backends chunked and overlap-all both reduce to standard Back-Propagation $g_T(u(t)) = \nabla_{u(t)} L(x(T))$, $t = 0, \ldots, 4$, because there is a unique window $[0, T]$ that covers every block and attributes its full terminal gradient; deep-sup instead sums one head per block, $g_T^{\mathrm{ds}}(u(t)) = \sum_{k=t+1}^{T} \nabla_{u(t)} L(x(k))$. The interesting backend-discrimination among chunked and overlap-all thus lives entirely in $1 < h < T$, exactly the regime probed by the experiments in Section 4 and by the gradient-cosine measurements of Figure 2; deep-sup additionally remains distinct from BP at the right endpoint.

B Proof of Theorem

B.1 Equivalence of Terminal Loss and Trajectory Loss

In this section, we prove the equivalence of terminal loss and trajectory loss. We use $J_{tra}(u)$ to denote the trajectory loss and $J_{ter}(u)$ to denote the terminal loss in the section to distinguish.

Trajectory Loss to Terminal Loss To transform trajectory loss to terminal loss, we need to add an auxiliary state x_0 and let the auxiliary state x_0 record the accumulation of trajectory loss and $x_0(T)$ be considered as the terminal loss:

$$x_0(t+1) = x_0(t) + l(t, x(t), u(t)), x_0(0) = 0 \tag{15}$$

$$J_{ter}(u) = L(x(T), u) = x_0(T). \tag{16}$$

It is easy to prove using induction on $x_0(t)$ that

$$J_{ter}(u) = x_0(T) = \sum_{t=0}^{T-1} l(t, x(t), u(t)) = J_{tra}(u). \tag{17}$$

Terminal Loss to Trajectory Loss On the other hand, to turn terminal loss to trajectory loss, we use split loss Eq. (8) $l(t, x(t), u(t)) \triangleq L(x(t+1)) - L(x(t))$. The terminal loss will become the accumulation of loss decreases for each block, and the equivalence can be proven by simple calculation:

$$\begin{aligned} J_{tra}(u) &= \sum_{t=0}^{T-1} l(t, x(t), u(t)) \\ &= \sum_{t=1}^{T} (L(x(t)) - L(x(t-1))) \\ &= L(x(T)) - L(x(0)) = J_{ter}(u) - L(x(0)). \end{aligned} \tag{18}$$

Since $L(x(0))$ only depends on input and is unrelated to the model, it is a constant for the model.

B.2 Single-probe deviation lemma

This appendix establishes the single-probe deviation estimate Eq. (25) that the per-backend proofs in Appendix B.3 all rely on. The lemma bounds, in Frobenius norm, the difference between the BP per-block gradient $g_T(t)$ and the gradient $g(t;k)$ obtained by truncating the chain at any depth $k \in \{t+1, \dots, T\}$. The derivation is a direct Neumann-series + Lipschitz argument that uses only the 2-norm bound $\|\tilde{W}(t)\|_2 \le c$ and the $1/T$ near-identity scaling $W(t) = I + \tilde{W}(t)/T$.

Notation and the per-block gradient. For $0 \le a \le b \le T$ denote by $W_a^b \triangleq W(b-1)W(b-2)\cdots W(a)$ the chain product over blocks $a, \dots, b-1$ (with $W_a^a = I$); in particular W_0^T is the full network W of §3, and the whitened loss reads $L = \frac{1}{2}\|W_0^T - \Phi\|_F^2$. More generally, evaluating the same quadratic at the intermediate feature $x(k) = W_0^k\, x(0)$ defines the depth-k probe $L\big(x(k)\big) = \frac{1}{2}\|W_0^k - \Phi\|_F^2$. Applying the chain rule to $W_0^k = W_{t+1}^k\, W(t)\, W_0^t$ (for $t < k$) and averaging over the whitened batch ($\mathbb{E}[x(0)x(0)^\top] = I$), the depth-$k$ probe gradient with respect to block t is the matrix

$$g(t;k) \;\triangleq\; \frac{\partial L(x(k))}{\partial W(t)} \;=\; \big(W_{t+1}^k\big)^\top \big(W_0^k - \Phi\big) \big(W_0^t\big)^\top. \tag{19}$$

In particular $g_T(t) = g(t;T)$ is the BP gradient. The single-probe window at block t has terminal target $L(x(\min(t+h, T)))$ and is attributed only to block t, so by Eq. (19),

$$g_h(t) \;\triangleq\; \frac{\partial L(x(\min(t+h,T)))}{\partial W(t)} \;=\; \begin{cases} \big(W_{t+1}^{t+h}\big)^\top \big(W_0^{t+h} - \Phi\big) \big(W_0^t\big)^\top & 0 \le t < T-h, \\ \big(W_{t+1}^{T}\big)^\top \big(W_0^{T} - \Phi\big) \big(W_0^t\big)^\top & T-h \le t \le T-1, \end{cases} \tag{20}$$

and $g_T(t)$ is the second branch above for every t (the BP gradient is the special case $h = T$, with the window saturating $\min(t+T, T) = T$ for every t). For $t \ge T-h$ the window already saturates the chain ($\min(t+h, T) = T$) and hence $g_h(t) = g_T(t)$ identically, so the only non-zero contribution to $g_h - g_T$ comes from the $T-h$ *transient* blocks $0 \le t < T-h$. Throughout we write $\delta_t \triangleq (T-t-h)/T \in (0, \delta]$ for the tail

length of transient block t and use $\|\cdot\| = \|\cdot\|_F$, $\|\cdot\|_2$ for Frobenius and spectral norm respectively. The bookkeeping below is stated for $g_h(t)$ because the algebra specialises most cleanly to a single probe at depth $k = t+h$; the generalisation to an arbitrary probe depth $k \in \{t+1, \dots, T\}$, which is what the per-backend proofs of Appendix B.3 actually invoke, is verbatim ($t+h \to k$, $\delta_t \to (T-k)/T$).

The proof of Lemma B.4 below uses three elementary lemmas.

Lemma B.1 (Chain bounds)**.** *For every $0 \le a \le b \le T$,*

$$\|W_a^b - I\|_2 \;\le\; \exp\bigl(c(b-a)/T\bigr) - 1 \;\le\; \frac{c(b-a)}{T}\, e^c, \qquad e^{-c} \;\le\; \|W_a^b\|_2 \;\le\; e^c,$$

provided $c \le T$.

Proof. By sub-multiplicativity, $\|W_a^b\|_2 \le \prod_{t=a}^{b-1}(1 + c/T) \le e^{c(b-a)/T} \le e^c$. For the lower bound, $\sigma_{\min}(W_a^b) \ge \prod_{t=a}^{b-1}(1 - c/T) \ge e^{-c}$ when $c \le T$. For the difference, telescope $W_a^b - I = \sum_{k=a}^{b-1} W_{k+1}^b\,(\tilde{W}(k)/T)\,W_a^k$ and apply the spectral bounds to each factor; the geometric series sums to $e^{c(b-a)/T} - 1$. □

Lemma B.2 (Per-block gradient norm)**.** *For every $T \ge 1$ and every $t \in \{0, \dots, T-1\}$,*

$$m(c, \Phi) \;\le\; \|g_T(t)\|_F \;\le\; M(c, \Phi), \qquad m \triangleq e^{-2c}\bigl(e^{-c} - \|\Phi\|_2\bigr)_+, \qquad M \triangleq e^{2c}\bigl(e^c + \|\Phi\|_F\bigr).$$

We assume c is small enough that $m(c, \Phi) > 0$, i.e. that the target Φ is not too far from the trajectory of the near-identity chain; this is the regime in which the deviation bound is informative.

Proof. Apply Lemma B.1 to each of the three factors of $g_T(t)$ in Eq. (20); bound the middle factor $W_0^T - \Phi$ above by $\|W_0^T\|_F + \|\Phi\|_F \le e^c + \|\Phi\|_F$ and below by $\sigma_{\min}(W_0^T) - \sigma_{\max}(\Phi) \ge e^{-c} - \|\Phi\|_2$. □

Lemma B.3 (Cosine via residue)**.** *For any two non-zero vectors g, g' in a Hilbert space, write $g' = g + \Delta$. Then*

$$1 - \cos^2(\angle(g', g)) \;\le\; \frac{\|\Delta\|^2}{\|g'\|^2}.$$

Proof. Decompose $g' = \alpha g + r$ with $r \perp g$. Then $\|\Delta\|^2 = (1-\alpha)^2\|g\|^2 + \|r\|^2 \ge \|r\|^2$, and $\cos^2(\angle(g', g)) = \alpha^2\|g\|^2/\|g'\|^2$, so $1 - \cos^2 = \|r\|^2/\|g'\|^2 \le \|\Delta\|^2/\|g'\|^2$. □

Lemma B.4 (Single-probe deviation; master estimate)**.** *Under the assumptions $W(t) = I + \tilde{W}(t)/T$ with $\|\tilde{W}(t)\|_2 \le c$, for every $T \ge 1$, every $t \in \{0, \dots, T-1\}$, and every probe depth $k \in \{t+1, \dots, T\}$, the depth-k probe gradient $g(t; k)$ of Eq. (19) satisfies*

$$\|g_T(t) - g(t; k)\|_F \;\le\; A_1\,\frac{T-k}{T} \;+\; A_2\,\frac{(T-k)^2}{T^2},$$

where $A_1 \triangleq 3\,c\,e^{3c}(e^c + \|\Phi\|_F)$ and $A_2 \triangleq 2\,c^2\,e^{4c}(e^c + \|\Phi\|_F)$. Taking $k = t+h$ recovers the bound Eq. (22) below for the single-probe window of Eq. (20).

Proof of Lemma B.4. Fix a transient block $0 \le t < T - h$ (the saturated case $t \ge T - h$ gives $g_h(t) = g_T(t)$ identically, so the bound is trivial). Set $E_t \triangleq W_{t+h}^T - I$. Lemma B.1 applied to the chain of length $T - (t+h) = \delta_t T$ gives

$$\|E_t\|_2 \;\le\; e^{c\delta_t} - 1 \;\le\; c\,\delta_t\,e^c. \tag{21}$$

Step 1: pointwise gradient deviation. Using $W_{t+1}^T = W_{t+h}^T W_{t+1}^{t+h} = (I + E_t)W_{t+1}^{t+h}$ and $W_0^T = W_{t+h}^T W_0^{t+h} = (I + E_t)W_0^{t+h}$, we get the two left-action decompositions $W_{t+1}^T = W_{t+1}^{t+h} + E_t\, W_{t+1}^{t+h}$ and $W_0^T - \Phi = (W_0^{t+h} - \Phi) + E_t\, W_0^{t+h}$. Substituting in Eq. (20), the difference $g_T(t) - g_h(t)$ splits into three terms whose middle factor is, respectively, $E_t^\top(W_0^{t+h} - \Phi)$, $E_t\, W_0^{t+h}$, and the cross term $E_t^\top E_t\, W_0^{t+h}$, sandwiched between $(W_{t+1}^{t+h})^\top$ on the left and $(W_0^t)^\top$ on the right. Bounding each by Lemmas B.1–B.2 gives Frobenius norms at most $e^{2c}(e^c + \|\Phi\|_F)\|E_t\|_2$ for the first, $2e^{2c}(e^c + \|\Phi\|_F)\|E_t\|_2$ for the second (after splitting $W_0^{t+h} = (W_0^{t+h} - \Phi) + \Phi$ to expose a $\|\Phi\|_F$ factor), and $2e^{2c}(e^c + \|\Phi\|_F)\|E_t\|_2^2$ for the cross term. Adding all three contributions and using $\|E_t\|_2 \le c\delta_t e^c$ from Eq. (21),

$$\|g_T(t) - g_h(t)\|_F \;\le\; 3\, e^{2c}(e^c + \|\Phi\|_F)\|E_t\|_2 \;+\; 2\, e^{2c}(e^c + \|\Phi\|_F)\|E_t\|_2^2 \;\le\; A_1\, \delta_t \;+\; A_2\, \delta_t^2, \tag{22}$$

with $A_1 \triangleq 3\, c\, e^{3c}(e^c + \|\Phi\|_F)$ and $A_2 \triangleq 2\, c^2\, e^{4c}(e^c + \|\Phi\|_F)$. Since $\delta_t = (T - (t+h))/T$, this is the $k = t + h$ specialisation of the lemma; the general k case is obtained by relabelling $t + h \to k$ throughout. The remaining $t \ge T - h$ blocks of the single-probe window are saturated, so Eq. (22) also holds trivially there with both sides equal to zero. □

B.3 Proof of Theorem 3.3: per-backend bounds

We retain the notation of Appendix B.2: $W(t) = I + \frac{1}{T}\tilde{W}(t)$ with $\|\tilde{W}(t)\|_2 \le c$; chain product $W_a^b = W(b-1)\cdots W(a)$ with $W_a^a = I$; whitened loss $L = \frac{1}{2}\|W_0^T - \Phi\|_F^2$; depth-$k$ probe $L(x(k)) = \frac{1}{2}\|W_0^k - \Phi\|_F^2$. Throughout this section we write $a \wedge b \triangleq \min(a, b)$ for the truncation of indices to the chain length T. By Eq. (19), the gradient of the depth-k probe with respect to block $t < k$ is the matrix

$$g(t; k) \;\triangleq\; \frac{\partial L(x(k))}{\partial W(t)} \;=\; \big(W_{t+1}^k\big)^\top \big(W_0^k - \Phi\big)\big(W_0^t\big)^\top, \tag{23}$$

identical in form to the single-probe gradient Eq. (20) with $t + h$ replaced by k. In particular $g_T(t) = g(t; T)$ is the BP gradient. Reading off each backend's per-block formula from the code's window/probe pattern (cf. §A and fig. 1): for overlap-all the stride-1 schedule places $T-h+1$ *full-length* windows $[w, w+h]$ for $w = 0, \ldots, T-h$, each carrying the probe $L(x(w+h))$ and attributing its gradient to every block in $[w, w+h)$; deep-sup attaches a probe at every depth $k \in \{1, \ldots, T\}$ and back-propagates each into the h preceding blocks; chunked tiles the block axis with $Q = \lceil T/h \rceil$ disjoint windows of length h (with the tail truncated to T) and assigns every block of chunk q to the chunk's single terminal probe $L(x(k_q))$ with $k_q = (q+1)h \wedge T$. This gives

$$\begin{aligned}
g_h^{\mathsf{ch}}(t) \;&=\; g\big(t;\, k_q\big), \qquad k_q \;\triangleq\; (q+1)h \wedge T, \qquad q \;\triangleq\; \lfloor t/h \rfloor,\\
g_h^{\mathsf{oa}}(t) \;&=\; \sum_{w=(t-h+1)\vee 0}^{t\wedge(T-h)} g\big(t;\, w+h\big),\\
g_h^{\mathsf{ds}}(t) \;&=\; \sum_{k=t+1}^{(t+h)\wedge T} g(t; k).
\end{aligned} \tag{24}$$

The overlap-all window range $w \in [(t-h+1) \vee 0,\, t \wedge (T-h)]$ counts the windows whose full-length span $[w, w+h)$ contains block t. Write $c_t^{\mathsf{oa}}(h) \triangleq t \wedge (T-h) - (t-h+1) \vee 0 + 1$ for this multiplicity (the symmetric triangle of fig. 1(c)); in the regime $2h > T$ (i.e. $\delta < 1/2$) of interest,

$$c_t^{\mathsf{oa}}(h) \;=\; \begin{cases} t+1 & 0 \le t \le T-h-1,\\ T-h+1 & T-h \le t \le h-1,\\ T-t & h \le t \le T-1, \end{cases}$$

with $\sum_t c_t^{\mathsf{oa}}(h) = (T-h+1)h$ (total window-block incidences) and, in particular, $c_t^{\mathsf{oa}}(T) = 1$ for all t, so $g_T^{\mathsf{oa}} = g_T$ exactly (the single window $[0, T]$ broadcasts to every block). Overlap-all and deep-sup agree on the central plateau $t \in [T-h,\, h-1]$, where both equal $\sum_{k=h}^T g(t; k)$ for overlap-all (probes from depth h to

T) or $\sum_{k=t+1}^{T} g(t;k)$ for deep-sup (probes from depth $t{+}1$ to T), a difference of $t{-}h{+}1$ shallow probes that vanishes only in the symmetric centre $t = (T{-}1)/2$.

For the residual / nonlinear variants of §B.4 the matrices $\tilde{W}$ are replaced by the actual block Jacobians $A(t)$ throughout; the calculation below uses only the bound $\|\tilde{W}(t)\|_2 \le c$, so the substitution is verbatim.

Single-probe deviation (master estimate). Lemma B.4 of Appendix B.2 bounds $\|g_T(t) - g(t;k)\|_F$ for any probe depth $k \in \{t{+}1, \dots, T\}$ (and any block $t < k$):

$$\|g_T(t) - g(t;k)\|_F \;\le\; A_1 \frac{T-k}{T} \;+\; A_2 \frac{(T-k)^2}{T^2}, \tag{25}$$

with constants $A_1 = 3c\, e^{3c}(e^c + \|\Phi\|_F)$ and $A_2 = 2c^2 e^{4c}(e^c + \|\Phi\|_F)$ as in Appendix B.2. By Lemma B.2, $\|g_T(t)\|_F \ge m(c,\Phi)$ uniformly in t, so $\|g_T\|^2 = \sum_t \|g_T(t)\|_F^2 \ge T\, m(c,\Phi)^2$. Applying Lemma B.3 to the gradient stacks gives, for any choice of *per-block* rescaling profile $\{\beta_t^\bullet\}_t > 0$,

$$1 - \cos^2(\theta_h^\bullet) \;\le\; \frac{\sum_t \|g_h^\bullet(t) - \beta_t^\bullet\, r^\bullet(t)\|_F^2}{\sum_t \|g_h^\bullet(t)\|_F^2}, \tag{26}$$

where $r^\bullet(t)$ is the per-block reference vector ($r^\bullet(t) = g_T(t)$ for $\bullet = \mathsf{ch}$; $r^{\mathsf{ds}}(t) = g_T^{\mathsf{ds}}(t)$ for deep-sup; the same identification with $\beta_t^{\mathsf{oa}} = c_t^{\mathsf{oa}}(h)$ for overlap-all). Per-coordinate adaptive optimisers (Adam, AdamW) absorb any $\{\beta_t^\bullet\}$ in their second-moment normalisation, so allowing $\beta_t^\bullet$ to depend on t is operationally free for the Adam-trained networks of Section 4; we keep $\beta_t^\bullet$ constant in t for backend `chunked` (where the natural reference $g_T(t)$ has uniform multiplicity 1) and depth-varying for `overlap-all` (to match its multiplicity profile $c_t^{\mathsf{oa}}(h)$).

Part (a): `overlap-all` vs reweighted BP. Pick the per-block rescaling $\beta_t^{\mathsf{oa}} \triangleq c_t^{\mathsf{oa}}(h)$ so that the reference becomes $\tilde{g}_T^{\mathsf{oa}}(t) = c_t^{\mathsf{oa}}(h)\, g_T(t)$. This is the natural target because at $h{=}T$ the single window $[0,T]$ broadcasts $g_T(t)$ to every block with multiplicity $c_t^{\mathsf{oa}}(T) = 1$, so $g_T^{\mathsf{oa}} = g_T$ identically; for any $h < T$ the per-block sum $g_h^{\mathsf{oa}}(t) = \sum_w g(t; w{+}h)$ is the natural $c_t^{\mathsf{oa}}(h)$-fold near-BP target (under Adam this rescaling is absorbed as discussed above; the grad-sim plots in Section 4.2 measure the unrescaled SGD-relevant cosine).

We split the per-block deviation into early-boundary, central-plateau, and late-boundary contributions (using the three-piece formula for $c_t^{\mathsf{oa}}(h)$ above; in the regime $\delta < 1/2$ all three regions are non-empty and the central plateau dominates).

Early boundary ($0 \le t \le T{-}h{-}1$, $c_t^{\mathsf{oa}}(h) = t{+}1$). The probes are $k = h, h{+}1, \dots, t{+}h$, each with missing depth $T{-}k \le T{-}h = \delta T$:

$$\begin{aligned} g_h^{\mathsf{oa}}(t) - (t{+}1)\, g_T(t) \;&=\; \sum_{k=h}^{t+h} \big[g(t;k) - g_T(t)\big], \\ \big\|g_h^{\mathsf{oa}}(t) - (t{+}1)\, g_T(t)\big\|_F \;&\le\; \frac{A_1}{T} \sum_{k=h}^{t+h} (T-k) \;\le\; (t{+}1)\, A_1\, \delta. \end{aligned} \tag{27}$$

Squaring and summing over $t = 0, \dots, T{-}h{-}1$,

$$\sum_{t=0}^{T-h-1} \big((t{+}1)\, A_1\, \delta\big)^2 \;\le\; \frac{A_1^2 (T{-}h)^3}{3}\, \delta^2 \;=\; \frac{A_1^2}{3}\, T^3\, \delta^5. \tag{28}$$

Central plateau ($T{-}h \le t \le h{-}1$, $c_t^{\mathsf{oa}}(h) = T{-}h{+}1 = \delta T{+}1$). Every block sees the *same* probes $k = h, h{+}1, \dots, T$, so

$$\begin{aligned} g_h^{\mathsf{oa}}(t) - (\delta T{+}1)\, g_T(t) \;&=\; \sum_{k=h}^{T} \big[g(t;k) - g_T(t)\big], \\ \big\|g_h^{\mathsf{oa}}(t) - (\delta T{+}1)\, g_T(t)\big\|_F \;&\le\; \frac{A_1}{T} \sum_{k=h}^{T} (T-k) \;=\; \frac{A_1\, \delta T(\delta T{+}1)}{2T} \;\le\; A_1\, \delta^2\, T, \end{aligned} \tag{29}$$

where the last inequality uses $\delta T{+}1 \le 2\,\delta T$, valid whenever the plateau is non-empty (i.e. $\delta T \ge 1$). The plateau has $(2h{-}T) = (1{-}2\delta)T$ blocks, so

$$\sum_{t=T-h}^{h-1} \big(A_1\,\delta^2 T\big)^2 \;\le\; A_1^2\,(1{-}2\delta)\,T^3\,\delta^4. \tag{30}$$

Late boundary ($h \le t \le T{-}1$, $c_t^{\mathsf{oa}}(h) = T{-}t$). The probes are $k = t{+}1, \dots, T$, each with missing depth $T{-}k \le T{-}t{-}1$:

$$\big\|g_h^{\mathsf{oa}}(t) - (T{-}t)\,g_T(t)\big\|_F \;\le\; \frac{A_1}{T}\sum_{k=t+1}^{T}(T{-}k) \;=\; \frac{A_1(T{-}t)(T{-}t{-}1)}{2T}. \tag{31}$$

Squaring and summing over $t = h, \dots, T{-}1$ (substitute $i = T{-}t$, $i = 1, \dots, T{-}h = \delta T$),

$$\sum_{t=h}^{T-1}\Big(\tfrac{A_1\,i(i-1)}{2T}\Big)^2 \;\le\; \frac{A_1^2}{4T^2}\sum_{i=1}^{\delta T} i^4 \;\le\; \frac{A_1^2\,(\delta T)^5}{20\,T^2} \;=\; \frac{A_1^2}{20}\,T^3\,\delta^5. \tag{32}$$

Assemble. Combining Eq. (28), Eq. (30), Eq. (32), the central-plateau term Eq. (30) of order $T^3\delta^4$ dominates the two $T^3\delta^5$ boundary terms. The denominator of Eq. (26) satisfies

$$\sum_t \|g_h^{\mathsf{oa}}(t)\|_F^2 \;\ge\; m^2\sum_t (c_t^{\mathsf{oa}}(h))^2(1-O(\delta)) \;\ge\; m^2\,(1{-}2\delta)\,T\,(\delta T{+}1)^2(1-O(\delta)) \;\ge\; \tfrac{1}{2}\,m^2\,T^3\,\delta^2, \tag{33}$$

(taking the plateau contribution alone), so

$$1-\cos^2\!\big(\angle(g_h^{\mathsf{oa}},\,\tilde g_T^{\mathsf{oa}})\big) \;\le\; \frac{A_1^2\,T^3\,\delta^4}{\frac{1}{2}\,m^2\,T^3\,\delta^2} \;=\; \frac{2\,A_1^2}{m^2}\,\delta^2 \;+\; O(\delta^3). \qquad \square \tag{34}$$

Where the quadratic rate comes from. Although each probe-deviation $\|g(t;k){-}g_T(t)\|$ is only $O(\delta)$ and the central-plateau block sees a sum of δT such deviations (giving a per-block deviation $O(\delta\cdot\delta T) = O(\delta^2 T)$, hence per-block squared $O(\delta^4 T^2)$ and stack sum $O(\delta^4 T^3)$), the denominator scales only as $O(\delta^2 T^3)$ because the central multiplicity $c_t^{\mathsf{oa}}(h) = \delta T{+}1$ vanishes with δ. Their ratio is δ^2, irrespective of how the per-block reference scalars are chosen, so no global rescaling can lift overlap-all above the quadratic rate.

Part (b): deep-sup vs full deep-sup. The reference $g_T^{\mathsf{ds}}(t) = \sum_{k=t+1}^{T} g(t;k)$ is the full deep-sup gradient at $h{=}T$ and is the same vector that the deep-sup loop computes for $h \ge T-t$; in particular $g_h^{\mathsf{ds}}(t) = g_T^{\mathsf{ds}}(t)$ identically for $t \ge T-h$. For transient blocks $t < T{-}h$,

$$g_T^{\mathsf{ds}}(t) - g_h^{\mathsf{ds}}(t) \;=\; \sum_{k=t+h+1}^{T} g(t;k), \tag{35}$$

a sum of $T{-}t{-}h$ deep-probe gradients each close to $g_T(t)$. Decompose each summand by Eq. (25): $g(t;k) = g_T(t) + r(t;k)$ with $\|r(t;k)\|_F \le A_1(T-k)/T + A_2(T-k)^2/T^2$. Setting $\beta_t^{\mathsf{ds}} \equiv 1$, the per-block deviation splits cleanly into a piece parallel to $g_T(t)$ and a small residual:

$$d_t \;\triangleq\; g_T^{\mathsf{ds}}(t) - g_h^{\mathsf{ds}}(t) \;=\; \underbrace{(T{-}t{-}h)\,g_T(t)}_{\text{parallel piece}} \;+\; \underbrace{\sum_{k=t+h+1}^{T} r(t;k)}_{\text{residual }\rho_t} \qquad (t < T{-}h), \tag{36}$$

and $d_t \equiv 0$ for $t \ge T{-}h$.

Stack-cosine identity. Write the full deviation stack $d = g_T^{\mathsf{ds}} - g_h^{\mathsf{ds}}$ and the three scalars

$$A = \|g_T^{\mathsf{ds}}\|_F^2, \qquad B = \langle d, g_T^{\mathsf{ds}}\rangle_F, \qquad C = \|d\|_F^2.$$

Then $g_h^{\mathsf{ds}} = g_T^{\mathsf{ds}} - d$ has squared norm $A - 2B + C$ and inner product $A - B$ with g_T^{ds}, so direct expansion of $1 - (A{-}B)^2/[A(A{-}2B{+}C)]$ collects to

$$1 - \cos^2\!\big(\angle(g_h^{\mathsf{ds}}, g_T^{\mathsf{ds}})\big) \;=\; \frac{A\,C - B^2}{A\,(A - 2B + C)}. \tag{37}$$

Equivalently $\sin^2\theta = \min_{\lambda\in\mathbb{R}} \|g_T^{\mathsf{ds}} - \lambda g_h^{\mathsf{ds}}\|_F^2/A$, so a single global rescaling of g_h^{ds} cannot improve the cosine. The rate is now determined by the orders of A, B, C, for which we use the parallel-residual split $d_t = d_t^{\parallel} + \rho_t$ with $d_t^{\parallel} = (T{-}t{-}h)g_T(t)$ from Eq. (36).

Order of A. Using $g_T^{\mathsf{ds}}(t) = (T{-}t)g_T(t) + O((T-t)\delta\,\|g_T(t)\|)$ from the same probe-deviation bound,

$$A \;=\; \sum_{t=0}^{T-1} \|g_T^{\mathsf{ds}}(t)\|_F^2 \;\geq\; m^2 \sum_{t=0}^{T-1} (T{-}t)^2\,(1 - O(\delta)) \;\geq\; \tfrac{1}{3}m^2T^3\,(1 - O(\delta)),$$

and analogously $A \leq \frac{1}{3}M^2T^3(1 + O(\delta))$; in particular $A = \Theta(T^3)$.

Order of C. $C = \|d^{\parallel}\|_F^2 + 2\langle d^{\parallel}, \rho\rangle_F + \|\rho\|_F^2$. The parallel piece sums to

$$\|d^{\parallel}\|_F^2 \;=\; \sum_{t=0}^{T-h-1} (T{-}t{-}h)^2\,\|g_T(t)\|_F^2 \;\leq\; \tfrac{1}{3}M^2\,T^3\,\delta^3.$$

The residual bound $\|\rho_t\|_F^2 \leq A_1^2(T{-}t{-}h)^4/(3T^2)$ (Cauchy–Schwarz on $\sum_k r(t;k)$, same as part (a)) sums to $\|\rho\|_F^2 \leq (A_1^2/(3T^2))\sum_{j=1}^{T-h} j^4 = O(T^3\delta^5)$, and the cross-term is bounded by $\sqrt{\|d^{\parallel}\|_F^2 \cdot \|\rho\|_F^2} = O(T^3\delta^4)$. Therefore

$$C \;=\; \tfrac{1}{3}M^2T^3\,\delta^3\,(1 + O(\delta)).$$

Order of B. Decompose $B = \langle d^{\parallel}, g_T^{\mathsf{ds}}\rangle_F + \langle \rho, g_T^{\mathsf{ds}}\rangle_F$. The first piece is

$$\langle d^{\parallel}, g_T^{\mathsf{ds}}\rangle_F \;=\; \sum_{t=0}^{T-h-1} (T{-}t{-}h)(T{-}t)\,\|g_T(t)\|_F^2\,(1 + O(\delta)) \;=\; \Theta(T^3\delta^2),$$

and the second is bounded by $\sqrt{\|\rho\|_F^2 \cdot A} = O(T^3\delta^{5/2})$, strictly smaller. Hence $B = \Theta(T^3\delta^2)$ and $B^2/A = \Theta(T^3\delta^4)$.

Conclude. Plugging the orders into Eq. (37): the numerator is $AC - B^2 = AC(1 - O(\delta))$ since $B^2/A = O(C\delta)$, and the denominator is $A(A - 2B + C) = A^2(1 - O(\delta^2))$ since $B, C = O(A\delta^2)$. Thus

$$1 - \cos^2\!\big(\angle(g_h^{\mathsf{ds}}, g_T^{\mathsf{ds}})\big) \;=\; \frac{C}{A}\,(1 + O(\delta)) \;\leq\; \frac{M^2}{m^2}\,\delta^3 \;+\; O(\delta^4). \tag{38}$$

This is the δ^3 rate quoted for part (b); the prefactor satisfies $K_{\mathsf{ds}}(c, \Phi) = \Theta(M^2/m^2)$. □

Part (c): chunked. With $\beta_{\mathsf{ch}} = 1$, block $t = qh + r$ ($r \in \{0, \ldots, h-1\}$, $q \in \{0, \ldots, Q-1\}$, $Q = \lceil T/h \rceil$) receives the single probe at $k_q = (q{+}1)h \wedge T$, and Eq. (25) gives $\|g_T(t) - g_h^{\mathsf{ch}}(t)\|_F \leq A_1(T - k_q)/T$. Crucially $T - k_q$ is *constant within each chunk*: all h blocks of chunk q contribute the same squared deviation. In particular the *last* chunk ($q = Q{-}1$) has $k_{Q-1} = T$, so $g_h^{\mathsf{ch}}(t) = g(t;T) = g_T(t)$ for $t \geq (Q{-}1)h$ and these blocks reproduce the BP gradient exactly — they drop out of the sum. The first chunk ($q = 0$, missing depth $T - h = \delta T$) alone contributes $h \cdot (A_1\delta)^2 = A_1^2 h\delta^2$, and summing across the remaining chunks,

$$\sum_t \|g_T(t) - g_h^{\mathsf{ch}}(t)\|_F^2 \;\leq\; A_1^2 h \sum_{q=0}^{Q-2} \big((T - k_q)/T\big)^2 \;=\; O\big(A_1^2\,T\,\delta^2\big),$$

where the $q = 0$ term dominates as $h \to T$ (in this regime $Q = 2$ and only chunk 0 plus the trivial last one are present). With $\|g_T\|^2 \geq Tm^2$,

$$1 - \cos^2(\theta_h^{\mathsf{ch}}) \;\leq\; \frac{A_1^2}{m^2}\,\delta^2 + O(\delta^3). \tag{39}$$

Lower bound (genuine quadraticity). The h blocks of chunk 0 all share missing depth δT, so the second-order Neumann term in Eq. (25) contributes $\geq c'\delta$ to $\|g_T(t) - g_h^{\mathsf{ch}}(t)\|_F$ for each, and no global rescaling can absorb their joint $\Theta(\delta^2)$ contribution to $1 - \cos^2$. □

Constants. Modulo absolute numerical factors, all three constants satisfy

$$K_{\mathsf{oa}},\; K_{\mathsf{ds}},\; K_{\mathsf{ch}} \;=\; O\Big(\frac{c^2 e^{6c}(e^c + \|\Phi\|_F)^2}{m(c,\Phi)^2}\Big),$$

uniformly in T, h and depending on the data only through $\|\Phi\|_F$.

Remark on per-sample vs. whitened-batch. The derivation above uses the whitened-batch convention of Appendix B.2 ($\mathbb{E}[x(0)x(0)^\top] = I$, $L = \frac{1}{2}\|W_0^T - \Phi\|_F^2$, matrix gradients). The per-sample form is recovered by writing $g(t;k) = \rho(t;k)\,x(t)^\top$ with $\rho(t;k) = (W_{t+1}^k)^\top(x(k) - y)$ and $x(t) = W_0^t\,x(0)$ for a single sample; the rank-one factorisation makes the right factor $x(t)^\top$ cancel in the cosine ratio, which is the more economical form for direct numerical evaluation but is bookkeeping-equivalent to the matrix derivation above.

B.4 Unified nonlinear extension

The deterministic structure of the proof in Sections B.2 and B.3 makes a clean transfer to nonlinear residual networks possible without re-doing the bookkeeping. The sequence $\{\tilde{W}(t)\}$ enters the proofs only through two structural facts: (i) the deterministic operator-norm bound $\|\tilde{W}(t)\|_2 \leq c$, and (ii) the Neumann expansion of the path Jacobian $W_a^b = \prod_{s=a}^{b-1}\big(I + \frac{1}{T}\tilde{W}(s)\big)$ used in Lemmas B.1–B.3 and in the proof of Lemma B.4. Both facts continue to hold for *any* sequence of matrices $A(t)$ with $\|A(t)\|_2 \leq L$, irrespective of whether $A(t)$ depends on the trajectory $(x(s), u(s))_{s \leq t}$. This gives the following unified statement.

Theorem B.5 (Unified nonlinear extension)**.** *Let $f_t(x,u) = x + \frac{1}{T}\phi_t(x,u)$ with ϕ_t locally Lipschitz in x (in particular differentiable, piecewise differentiable, or piecewise-linear in x). Let $A(t)$ denote any measurable selection from the Clarke generalised Jacobian $\partial_x\phi_t$ at $(x(t), u(t))$ (for smooth ϕ_t this is the usual derivative, for piecewise-smooth/-linear ϕ_t it is the one-sided directional derivative along the realised activation pattern), and assume the trajectory-level bound*

$$\|A(t)\|_2 \;\leq\; L \qquad \textit{for all } t = 0, \ldots, T-1. \tag{40}$$

Then the conclusions of Theorem 3.3 hold for ϕ_t with the spectral-radius constant c replaced by L in every constant $K_\bullet(c,\Phi)$. In particular, the cubic exponent δ^3 for deep-sup *and the quadratic exponent δ^2 for* overlap-all *and* chunked *are preserved.*

Proof. Substitute $A(t)$ for $\tilde{W}(t)$ throughout Lemmas B.1–B.3 and in the Neumann expansion of W_a^b. Properties (i)–(ii) above hold by assumption Eq. (40), so each lemma yields the same bound with c replaced by L. The per-backend bookkeeping of Appendix B.3 (single-probe deviation Eq. (25), cosine master estimate Eq. (26)) is combinatorial in the window pattern and does not touch the Jacobians, so it applies as written. The chain-rule structure of the gradient w.r.t. the parameter $u(t)$ factors out a common $(\partial_u\phi_t)^\top$ matrix on both sides of the cosine and therefore drops out (Lemma B.1 applied to u in place of W). □

Instance (a): ReLU residual blocks. For $f_t(x,u) = x + \frac{1}{T}\sigma(W(t)x + b(t))$ with $\sigma(z) = \max(0,z)$ and $\|W(t)\|_2 \leq c$, the realised Jacobian is $A(t) = D(t,x(t))\,W(t)$ where $D(t,x) = \mathrm{diag}\big(\mathbb{1}[W(t)x + b(t) > 0]\big)$ is a 0/1 projector. Then $\|A(t)\|_2 \leq \|D(t)\|_2\,\|W(t)\|_2 \leq c$, so Eq. (40) holds with $L = c$. The cosine inherits the same projector $D(t)$ on both sides of the inner product (the gradient w.r.t. $W(t)$ picks up $D(t)$ from the outer σ', identically for g_h and g_T), so inactive coordinates contribute zero to both and the rates of Theorem 3.3 are attained in the active subspace.

Instance (b): smooth residual blocks. For ϕ_t differentiable in x with $\|\partial_x \phi_t(x,u)\|_2 \le L$, condition Eq. (40) holds directly. Examples: smooth activations (tanh, GELU, SiLU); the Pre-LayerNorm Transformer block, where $\partial_x \phi_t$ is the Jacobian of LN $\to$ MHA or LN $\to$ MLP composed with a residual sum, with softmax bounded by 1 and value / FFN gain bounded by weight-norm products; and the ResNet-50 bottleneck, a composition of bounded convolutional Jacobians with ReLU projectors (which adds only D-factors of operator norm ≤ 1). All have finite L.

Why a naive expectation argument fails (ReLU case). A tempting shortcut is to replace $D(t)$ by $\mathbb{E}[D(t)] = \frac{1}{2}I$ and appeal to concentration. Three issues block this: (i) the iterates $x(t) = x(t-1) + \frac{1}{T}\sigma(\cdot) \succeq x(t-1)$ are non-decreasing component-wise, so $\mathbb{E}[D(t,x(t))] \to I$ rather than $\frac{1}{2}I$ as t/T grows; (ii) the $\{D(t,x(t))\}$ share the underlying $\{W(s)\}_{s<t}$ and $x(0)$, hence are far from independent and Hoeffding does not apply; (iii) the cosine is non-linear in the Jacobians, so $\mathbb{E}[\cos\theta_h] \ne \cos\theta_{\mathbb{E}[\mathrm{Jac}]}$. The trajectory-level bound Eq. (40) side-steps all three because it is asserted at the realised $A(t)$ and never invokes linearity of expectation across blocks.

B.5 Loss Decrease under the Polyak–Łojasiewicz Inequality

The convergence rate established below requires strong convexity. Most overparameterised deep networks do not satisfy strong convexity but do satisfy the Polyak–Łojasiewicz (PL) inequality along the training trajectory for wide overparameterised networks.

Theorem B.6 (Loss Decrease under PL with biased gradient)**.** *Suppose L is β-smooth and satisfies the PL inequality $\frac{1}{2}\|\nabla L\|^2 \ge \mu(L - L^*)$. Suppose at every iterate $\cos\theta_h^k \ge \cos\bar{\theta}_h > 0$. With the projection-rescaled MPC update*

$$u^{k+1} \;=\; u^k - \tfrac{1}{\beta}\, c^k\, g_h^k, \qquad c^k \;=\; \tfrac{\|g_T^k\|\cos\theta_h^k}{\|g_h^k\|}, \tag{41}$$

we have

$$L(u^{k+1}) - L^* \;\le\; \big(1 - \tfrac{\mu\cos^2\bar{\theta}_h}{\beta}\big)\big(L(u^k) - L^*\big), \tag{42}$$

so

$$L(u^k) - L^* \le (1 - \mu\cos^2\bar{\theta}_h/\beta)^k (L^0 - L^*). \tag{43}$$

Proof. We expand smoothness, plug in the projection-rescaled MPC update, and close with PL.

Smoothness. β-smoothness gives, for any direction v and step $\eta > 0$,

$$L(u^k - \eta v) \;\le\; L(u^k) - \eta\,\langle \nabla L(u^k), v\rangle + \tfrac{\eta^2\beta}{2}\,\|v\|^2. \tag{44}$$

The MPC update sets $\eta = 1/\beta$ and $v = c^k g_h^k$. Using $g_T^k = \nabla L(u^k)$ and $\cos\theta_h^k = \langle g_T^k, g_h^k\rangle/(\|g_T^k\|\,\|g_h^k\|)$, both relevant quantities reduce to the same scalar:

$$\langle \nabla L,\, c^k g_h^k\rangle \;=\; \frac{\|g_T^k\|\cos\theta_h^k}{\|g_h^k\|}\cdot\|g_T^k\|\,\|g_h^k\|\cos\theta_h^k \;=\; \|\nabla L\|^2\,\cos^2\theta_h^k \;=\; \|c^k g_h^k\|^2, \tag{45}$$

so Eq. (44) collapses to the descent inequality

$$L(u^{k+1}) - L(u^k) \;\le\; -\tfrac{1}{2\beta}\,\|\nabla L\|^2\,\cos^2\theta_h^k. \tag{46}$$

The role of the projection rescaling c^k is exactly to make the inner-product and squared-norm terms agree on this common scalar and thus eliminate the $1/2$ overhead one would otherwise carry. Applying the PL inequality $\frac{1}{2}\|\nabla L\|^2 \ge \mu(L - L^*)$ and the floor $\cos^2\theta_h^k \ge \cos^2\bar{\theta}_h$ to Eq. (46),

$$L(u^{k+1}) - L^* \;\le\; \Big(1 - \tfrac{\mu\cos^2\bar{\theta}_h}{\beta}\Big)\big(L(u^k) - L^*\big), \tag{47}$$

and iterating gives the stated linear rate. □

Stochastic version. Replacing g_h^k by an unbiased mini-batch with variance $\leq \sigma^2$, the standard biased-SGD argument gives, for fixed $\eta \leq 1/\beta$,

$$\mathbb{E}[L(u^k) - L^*] \;\leq\; \big(1 - \eta\mu\cos^2\bar{\theta}_h\big)^k (L^0 - L^*) \;+\; \tfrac{\eta\sigma^2\beta}{2\mu\cos^2\bar{\theta}_h}. \tag{48}$$

A small horizon hurts both the rate *and* the asymptotic noise floor, while a horizon close to T pays linear memory for a polynomially vanishing return on $\cos^2\bar{\theta}_h$ (Theorems 3.3 and 3.3). The optimum is therefore interior in h and depends on (μ, β, σ^2), which are model- and dataset-specific; this is exactly why the empirical study (Section 4) sweeps h on a held-out criterion rather than fixing it a priori.

B.6 Probe as Terminal Cost: a Receding-Horizon View

The MPC framework introduced in Section 3 treats the probe loss $L_{\text{probe}}(x(t+h))$ as an estimator of the true cost-to-go

$$V^\star\big(x(t+h)\big) \;\triangleq\; \min_{u(t+h),\ldots,u(T-1)} L\big(x(T)\big), \tag{49}$$

i.e. the optimal loss attainable from intermediate state $x(t+h)$ by optimising the remaining blocks. We connect this choice to classical receding-horizon control and show that the MPC objective is a valid surrogate for the global one whenever the probe *upper-bounds* $V^\star$.

Proposition B.7 (Per-window descent under probe terminal cost)**.** *Let $\hat{L}_t(x)$ be a probe loss attached at position t satisfying the* terminal-cost condition

$$\hat{L}_{t+h}\big(x(t+h)\big) \;\geq\; V^\star\big(x(t+h)\big), \qquad t = 0, \ldots, T-h-1. \tag{50}$$

Apply one step of MPC to window $[t, t+h]$ with the projection-rescaled update of Theorem B.6 starting from parameters u^k, producing u^{k+1} that differs from u^k only on block index t. Then

$$L\big(x^{k+1}(T)\big) \;\leq\; \hat{L}_{t+h}\big(x^{k+1}(t+h)\big) \;\leq\; \hat{L}_{t+h}\big(x^k(t+h)\big), \tag{51}$$

so $L(x(T))$ is non-increasing under the receding-horizon scheme that sweeps $t = 0, \ldots, T-h-1$ once per epoch.

Proof. The first inequality is Eq. (50) applied at $x^{k+1}(t+h)$ together with the definition of $V^\star$ as a *minimum* over $u(t+h:T-1)$ which the current u^k achieves at most. The second inequality is the per-window PL-descent of Theorem B.6 applied to $\hat{L}_{t+h} \circ f_{t+h-1} \circ \cdots \circ f_t$ (a β_h-smooth, μ_h-PL function in the active block-t parameters whenever $\hat{L}_{t+h}$ is). Composing the two yields $L(x^{k+1}(T)) \leq L(x^k(T))$. □

This is the classical Mayne–Rawlings–Scokaert stability argument (Mayne et al. (2000)): receding-horizon control with a terminal cost dominating the value function is stable for *any* horizon $h \geq 1$. In our setting the conclusion is global descent of $L(x(T))$, sidestepping the cosine-bias rate in Section B.5 for the qualitative direction question.

When does Eq. (50) hold? Empirically, Alain & Bengio (2016) establish that the linear-probe cross-entropy of a trained ResNet decreases monotonically with depth, $\hat{L}_t(x(t)) \geq \hat{L}_{t+1}(x(t+1)) \geq \cdots \geq \hat{L}_T(x(T)) = L$, so the probe loss is a discrete Lyapunov function on the activation trajectory of any sufficiently trained model. Jastrzębski et al. (2018) strengthen this from a theoretical angle by showing that residual blocks implement gradient descent on an implicit loss, so each block strictly decreases that loss under mild smoothness assumptions; see also Greff et al. (2017) (iterative refinement view) and Veit et al. (2016) (block-removal lesion experiments). Belilovsky et al. (2019) show the property is achievable *by construction* via greedy layerwise probe minimisation. Together these results identify a regime — BP-trained or greedy-probe-trained residual networks near a minimiser of L — where condition Eq. (50) holds and Proposition B.7 applies.

When it fails: at random initialisation. Early in training the probe is itself untrained and Eq. (50) cannot be expected. The shared/stage-shared probe variants of Section 4 address this by training the probe jointly with the network, so $\hat{L}_t$ tracks $V^\star$ as both co-evolve. The cosine-bias analysis of Section B.5 provides the rate guarantee in this joint-training regime, while Proposition B.7 explains the late-stage stability we observe empirically.

Remark (rollout / policy-improvement view, Bertsekas (2020)). A complementary monotonicity statement is the Bertsekas rollout inequality: h-step MPC with terminal probe $\hat{L}$ improves on the zero-step greedy policy $u_t \leftarrow \arg\min L_{\text{probe},t}(x(t))$ whenever the probe is consistent with the chain $\hat{L}_{t+1} \circ f_t \leq \hat{L}_t$. In our framework this gives "MPC-with-probe $\succeq$ greedy layerwise training" (Belilovsky et al. (2019)) for every $h \geq 1$. We do not pursue the quantitative version (the Grüne suboptimality bound Grüne & Pannek (2017) requires positive stage cost $\ell_t(x_t, u_t) > 0$ which our formulation lacks: stage cost is zero except at the terminal block) and leave it as an interesting direction.

Open: time-varying horizon. Combining Proposition B.7 with a training-step-varying horizon h^k (small h^k during the transient when probe quality is low, larger h^k near convergence when the probe is informative) yields a dual-mode MPC scheme analogous to classical dual-mode receding horizon Michalska & Mayne (1993). We defer empirical study to the companion experiments on per-block horizon scheduling (Section B.7); the temporal axis h^k is orthogonal to the spatial axis $\{h_t\}$ developed below.

B.7 Per-block Optimal Horizon via Lagrangian

The horizon-selection algorithm of Section 3.3 uses a single global horizon h. When the gradient-norm profile $G_t \triangleq \|g_T(u(t))\|$ varies substantially across blocks, a per-block schedule $\{h_t\}_{t=0}^{T-1}$ improves the accuracy/memory trade-off.

Apply Theorem B.6 block-by-block: the per-block contribution to the loss-decrease bound is $\propto (1 - \delta_t^2(h_t))G_t^2$ where $\delta_t(h) \triangleq \sin\theta_h(u(t))$. Maximising over the budget B:

$$\min_{\{h_t\}} \sum_{t=0}^{T-1} \delta_t^2(h_t)\, G_t^2 \quad \text{s.t.} \quad \sum_{t=0}^{T-1} a_t h_t \;\leq\; B - \sum_t b_t, \;\; 1 \leq h_t \leq T, \tag{52}$$

where $a_t, b_t \geq 0$ are the per-block memory cost coefficients (window-linear $a_t h_t$ + fixed b_t).

Closed form (continuous relaxation). Using the parametric form $\delta_t^2(h) = c_t(1 - h/T)^p$ from Theorem 3.3 ($p = 3$, linear) or Theorem B.5 ($p = 3$, nonlinear), the KKT conditions $p\, c_t(1 - h_t/T)^{p-1} G_t^2/T = \lambda a_t$ give

$$\boxed{h_t^* \;=\; T\Big(1 - \Big(\tfrac{\lambda a_t T}{p c_t G_t^2}\Big)^{1/(p-1)}\Big)_{[1,T]}\,.} \tag{53}$$

λ is set by the budget constraint.

Validity caveats. The decomposition treats per-block updates as orthogonal. For residual networks without weight sharing this is clean: each $u(t)$ lives in a different parameter subspace, so the cross-block inner products $\langle v_s, v_t\rangle$, $s \neq t$, vanish in the parameter space; they re-appear only as higher-order corrections $\mathcal{O}((\eta/T)^2)$ from the coupled forward through J. The estimates G_t also drift during training; recomputing the schedule every K epochs (cost: one BP probe + $T \cdot |H|$ overlap-MPC probes) is cheap and removes the drift error.

C Training Details

All cells of the MPC sweep cited in the main text use the fixed per-backbone configuration listed in Table C1: a single optimiser (AdamW), bf16 autocast, batch size 128 on CIFAR-100 and per-task batches on GLUE, and a single fixed terminal probe per backbone (ViT: `headprobe`; ResNet-50/stage-cut: `shared`; ResNet-50/block-cut: `stagesh`; RoBERTa: `headprobe`). The MPC chain length T excludes the patch-embedding / classifier head on ViT and the stem on ResNet-50, so that every block on the chain participates in the planning horizon h.

Table C1: Training settings used in the current end-to-end MPC sweep. Image backbones are fine-tuned on CIFAR-100 (50k/10k); RoBERTa-base is fine-tuned on the four GLUE tasks with LoRA (r=8, α=16, W_q, W_v) following the per-task schedule of Hu et al. (2021, Table 9). All cells use AdamW, bf16 autocast, and a fixed per-backbone probe.

Setting	ViT-b/16+LoRA	R50-stage	R50-block	RoBERTa-base+LoRA on GLUE			
				MRPC	SST-2	QNLI	MNLI
Optimiser	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	4×10^{-4}	5×10^{-4}	4×10^{-4}	5×10^{-4}
Weight decay	1×10^{-4}	0	0	0	0	0	0
Batch size	128	128	128	16	16	32	16
Epochs	10	70	70	30	60	25	30
Precision	bf16	bf16	bf16	bf16	bf16	bf16	bf16
Probe	headprobe	shared	stagesh	headprobe	headprobe	headprobe	headprobe
T (MPC chain)	12	5	17	12	12	12	12
Dataset	CIFAR-100	CIFAR-100	CIFAR-100	MRPC	SST-2	QNLI	MNLI

C.1 Probe parametrisations

A probe is the concrete realisation of the surrogate loss $L\big(x(k)\big)$ that the theory treats abstractly: it takes the intermediate activation $x(k) \in \mathbb{R}^{B\times C_k\times\cdots}$ at the output of block k, projects it to class logits, and applies the same task loss as the terminal head. We design all probes under two principles: (a) reuse the backbone's pretrained classifier head whenever the cuts have a uniform feature width, and (b) add as few extra parameters and activation memory as possible; in particular all variants pool the spatial / sequence dimensions *before* any channel projection, so the projection is a small Linear($C_k \to d$) acting on a (B, C_k) tensor rather than a 1×1 convolution that would materialise a (B, d, H, W) feature map per cut. The three variants used in the sweep are:

- shared: a learned channel projection pool+Linear($C_k \to d$) *per cut position* (with d=256) and a single learned head Linear($d \to C$) shared across all cuts. Used for R50 stage-cut.
- stagesh: identical to shared except the channel projections are pooled *per stage* rather than per cut (e.g. R50-block uses 4 stage projections for its 17 cuts instead of 17 per-position projections); the head is still a single shared Linear($d \to C$). Used for R50 block-cut.
- headprobe: the backbone's own *pretrained* classifier head, shared across cuts; for uniform-width architectures (ViT, RoBERTa) no projection is needed, and the head consumes the intermediate activation directly after pooling. Used for ViT and RoBERTa.

The depth-k probe of the theory ($L(x(k)) = \frac{1}{2}\|W_0^k - \Phi\|_F^2$ in Appendix B.3) is the limit in which the learned head is the identity map: it isolates the gradient deviation induced by the planning schedule itself rather than by the choice of probe parametrisation.

D Memory and per-step wall time (image backbones)

This appendix complements the main-text §4.5 and Figure 5 with a discussion of three implementation-level artefacts visible in that figure. Each backbone is measured with the single fixed probe used in the training sweep (ViT: headprobe; ResNet-50 stage-cut: shared; ResNet-50 block-cut: stagesh; see Appendix C.1 for the probe parametrisations).

Three implementation-level artefacts in Figure 5. On ViT, where every transformer block produces an activation of the same shape, peak memory grows essentially linearly in h for both chunked and overlap-all, which is the expected behaviour: each extra horizon step keeps one more block of stored activations alive. deep-supervision keeps the full activation graph alive at every h, so its memory matches full BP across the whole sweep; h only affects its step time.

Three implementation-level effects are worth flagging because they are visible in the figure but do not reflect the asymptotic algorithm.

(i) Chunked is uniformly the lower envelope, even at h=1. The ∼5–30% gap between chunked and overlap-all persists even at h=1 where all three backends compute the identical per-block gradient. The reason is that our overlap-all implementation pre-stores the full detached activation cascade acts$[0..T]$ once and then re-forwards each window with grad, whereas chunked streams the cascade and drops every activation via `z.detach()` between chunks. At h=1 the gap therefore equals T detached activation tensors of the full network, which matches the observed offsets (≈400 MiB on ViT, ≈1.3 GiB on R50-block). A fused overlap implementation that shares the streamed cascade would close the gap, but at the cost of the numerical compatibility with chunked that the current code path guarantees.

(ii) ResNet-50 memory is non-monotonic in h. On both ResNet-50 cuts chunked and overlap-all have peak memory that rises until $h \approx 2$ (stage cut) or $h \approx 4$ (block cut) and then drops by 30–40% as h grows further. This is a ResNet-specific artefact of two combined factors: (a) the per-block activation footprint is extremely non-uniform across the four stages (layer-1 activations 56×56×256 are roughly 8× larger than layer-4's 7×7×2048), and (b) at small h the implementation issues many short re-forward/backward windows, which the CUDA caching allocator services with separate transient buffers (peak inflated by fragmentation), whereas at large h a single wide window is serviced from one reused reserved segment. ViT and RoBERTa show no such non-monotonicity precisely because their blocks are size-uniform.

(iii) Deep-supervision wall time is quadratic in T at large h. The fused deep-supervision implementation does one forward pass (O(T) block-ops, the same as BP) but then issues a separate `autograd.grad` call for each loss depth $k \in \{1, \ldots, T\}$, each traversing $\min(k, h)$ blocks backward (with `retain_graph=True` for all but the last). The total backward block-op count is therefore $\sum_{k=1}^{T} \min(k, h) = T(T{+}1)/2$ at $h{=}T$, quadratic in T, versus T for full BP. The observed slowdowns at $h{=}T$ match this prediction: ViT (T=12) shows 78/12≈6.5× theory vs 7.4× measured, R50-stage (T=5) shows 15/5=3.0× vs 4.4×, R50-block (T=17) shows 153/17=9.0× vs 12.9×; the measured overhead exceeds the kernel count by a small constant due to `retain_graph` keeping activations alive and the per-call CUDA-launch overhead of many small `autograd.grad` invocations. The unfused deep-supervision backend adds the same quadratic to the forward pass, doubling the slowdown (e.g. 16× on ViT at $h{=}T$). This wall-time premium is a property of the multi-loss schedule, not of the MPC asymptotic per-step gradient cost.

Across backends chunked remains the memory-efficient lower envelope (savings range from 30% at $h \approx T/2$ to 80% at h=1 relative to full BP); overlap-all pays a moderate 1.5×–2× step-time premium for the additional forward passes. The per-step wall-time traces are noisier than the memory traces and show backend-dependent spikes at a few horizons (e.g. the overlap-all corners on R50-block); these are engineering artefacts of the current implementation (non-fused probe kernels, an unbatched overlap-all dispatch loop, cuts that happen to fall on residual-block boundaries) and are reported as observed rather than as evidence against the $O(h)$ behaviour predicted by the kernel count.

E Probe warm-up ablation for the gradient-deviation angle

The gradient-deviation diagnostic in Section 4.2 samples $\cos\theta_h$ with the terminal probe at its random initialisation (no warm-up). A natural concern is that the probe noise inflates the small-h end of every curve and hides the true asymptotic rate. We therefore repeat the measurement after 200 BP steps of probe warm-up, using the same per-(architecture, task) layout as the main-text Figure 2. Figure E1 shows the result. Compared with the no-warm-up figure in the main text, the small-h end of the overlap-all and deep-sup curves drops by roughly an order of magnitude, while the chunked curve is essentially unchanged, consistent with the view that the probe noise lowers the *baseline* of the identity-aligned curves. The log–log *slopes*, however, are not visibly improved by warm-up: deep-sup still sits above the slope-3 reference at small $1 - h/T$, overlap-all and chunked still track the slope-2 envelope, and the relative ordering of the three backends is preserved. We therefore report the no-warm-up regime in the main text as the more honest depiction of the truncation effect, and include the warm-up version here for completeness.

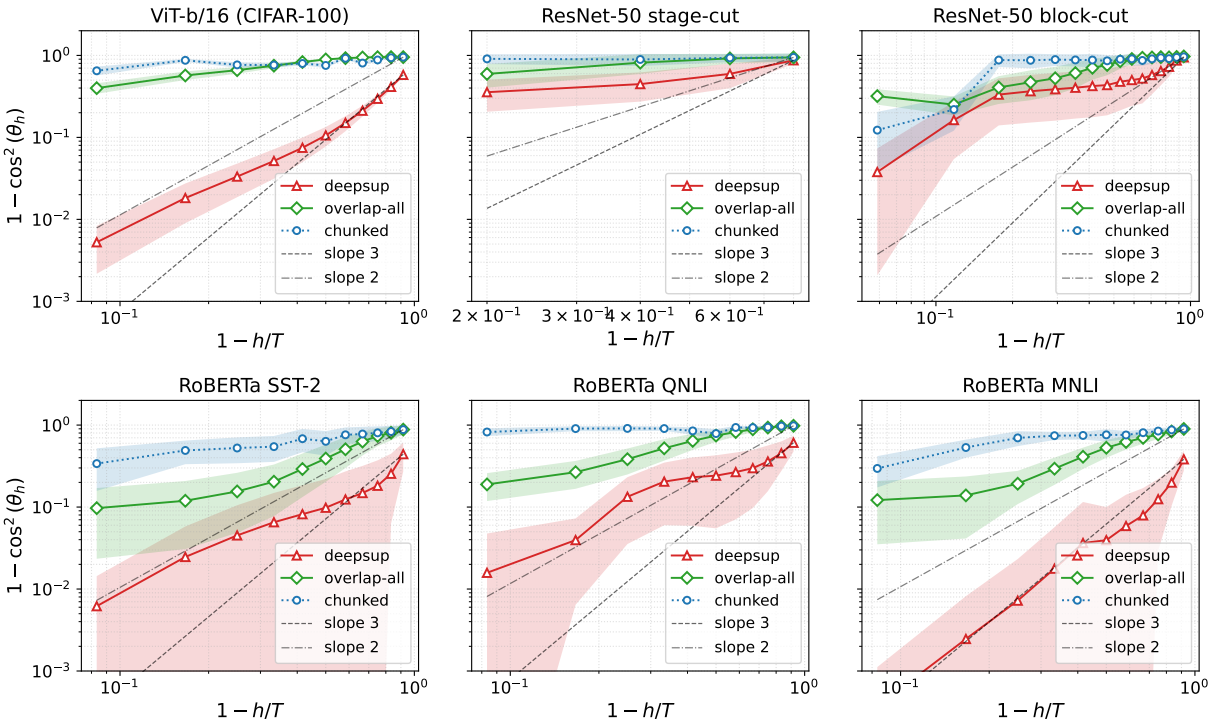

Figure E1: Empirical gradient-deviation cosine $\cos\theta_h = \langle g_h, g_{\mathrm{ref}}\rangle/(\|g_h\|\|g_{\mathrm{ref}}\|)$ on a held-out batch, with backend-matched references: full BP for chunked; multiplicity-rescaled BP (Theorem 3.3(a)) for overlap-all; deep-supervision at $h{=}T$ for deepsup. Probe regime: 200 BP steps of probe warm-up before sampling. Axes are $1-\cos^2(\theta_h)$ against $1-h/T$ on a log–log scale. The dashed reference has slope 3 (cubic bound of Theorem 3.3 for deep-sup); the dash-dotted reference has slope 2 (quadratic bound for overlap-all and chunked).