Is Backpropagation Optimal? When Synthetic Gradients Improve Sample Efficiency

Yibo Jacky Zhang
Stanford University
yiboz@stanford.edu

Zeyu Tang
Stanford University
zeyu@cs.stanford.edu

Sanmi Koyejo
Stanford University
sanmi@stanford.edu

Abstract

Backpropagation is the default learning rule for artificial neural networks and is often treated as the settled approach whenever differentiability is available. In this work, we revisit this convention through a theoretical lens of sample efficiency. We introduce a unified vectorized feedback framework for loss-based and reward-based learning on computational graphs, in which synthetic gradients emerge as a natural alternative to backpropagation. We characterize the conditions under which synthetic gradients can achieve a lower gradient-estimation mean squared error than backpropagation. We construct examples illustrating that this sample efficiency advantage can be arbitrarily large. Experiments on contextual bandits and reinforcement learning tasks demonstrate the potential of our theoretical findings.

1 Introduction

Backpropagation [41, 63, 51] has become the default learning rule for artificial neural networks, repeatedly outperforming alternatives (e.g., [40, 37, 24, 52, 21]). As a result, backpropagation (BP) is often treated as the settled answer to neural networks learning whenever differentiability is available. In this paper, we revisit this convention from the perspective of gradient-estimation sample efficiency.

To formalize the sample efficiency analysis, we consider the gradient estimation problem on a computational graph with vectorized terminal feedback. This vectorized feedback framework unifies loss-based learning, where the vectorized feedback is the gradient of a loss, and reward-based learning, where it may be given by policy gradients [64, 60]. In this picture, as detailed in the next section, backpropagation is analogous to a Monte Carlo (MC) estimator, and temporal-difference (TD) learning [59] provides a classical bootstrapped alternative to MC estimation. In the vectorized feedback setting, this bootstrapped alternative turns out to be the synthetic gradients [24, 14].

Synthetic gradients (SG) were originally introduced from the perspectives of decoupled computation and distributed training, and their connection to TD learning was also noted early on [24]. However, prior work largely studied synthetic gradients in essentially chain-structured computational graphs, and established no clear empirical or theoretical advantage over unconstrained backpropagation. As a result, the topic has gradually fallen out of favor, with only limited recent interest [50, 25].

Our point of departure is that this negative picture reflects an incomplete analysis. As with TD versus MC methods, the advantage of synthetic gradient propagation is not universal, but regime-dependent. TD methods can offer practical sample-efficiency advantages over MC methods through bootstrapping, and their theoretical analysis is an active area of research [30, 33, 5, 9, 11]. In contrast, synthetic gradients remain theoretically underexplored, with limited existing work focused mainly on training convergence [14, 23]. However, the analogy between TD and synthetic gradients suggests a possible sample-efficiency advantage over backpropagation, which leads to our central question:

When can synthetic gradients improve gradient-estimation sample efficiency over backpropagation?

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Formally, this paper studies gradient estimation on differentiable computational graphs. We compare the mean squared error of empirical gradient estimators formed from i.i.d. samples under backpropagation and synthetic gradient propagation.

The main contribution of this paper is thus a formal theoretical investigation into this underexplored yet potentially important question. Our results show that *backpropagation can be suboptimal in terms of gradient-estimation quality when gradient feedback contains uncertainty. We identify conditions under which synthetic gradients reduce gradient-estimation MSE.* **Our results can be summarized as follows.**

We introduce a vectorized feedback framework in Section 2 that unifies loss-based and reward-based learning on computational graphs. Motivated by this framework, we formally define the gradient estimation problem in Section 3. We then identify two conditions, defined formally in Section 4, that govern the comparison. Condition (A) says there must be uncertainty in gradient estimations. Specifically, (A) requires *at least one* of the following to hold: $(A.1)$ *randomness in the terminal gradient feedback*; $(A.2)$ *internal stochasticity in the network computation*; $(A.3)$ *sparse connectivity in the network that leads to partial observability at individual nodes*. Condition (B) is a *local sufficiency requirement*: each node's local observation contains sufficient information for the conditional mean of its incoming gradient feedback. Under these conditions, the comparison between backpropagation and synthetic gradients becomes sharp.

Our theory yields the following separation. With condition $\neg(A) \wedge (B)$, backpropagation is optimal in estimating mean squared error (MSE) among all conditionally unbiased gradient estimators. In contrast, with $(A) \wedge (B)$, an oracle synthetic gradient generator achieves a smaller estimation MSE than backpropagation. Thus, for synthetic gradient propagation to improve sample efficiency, some source of randomness or structural sparsity is necessary. This also helps explain why prior work often found synthetic gradients to be worse: synthetic gradients were typically evaluated in deterministic and fully connected networks on relatively clean datasets, essentially where condition (A) is absent. When condition (B) does not hold exactly, or when the synthetic gradient is not an idealized oracle, synthetic gradients may be biased. In this case, we show that a mixture of synthetic and backpropagated gradients can be preferable due to a better bias-variance tradeoff.

Inspired by the conditions (A) and (B), in Section 5 we show examples where synthetic gradient propagation can be arbitrarily more sample efficient than backpropagation with estimated non-oracle gradient predictors. This construction exploits a concrete realization of the local sufficiency condition (B): a network of specialized expert nodes that decomposes a global task into locally sufficient components.

Motivated by our theoretical findings, in Section 6 we apply synthetic gradients to a class of sparsely connected networks. We evaluate this approach on a contextual bandit problem with MNIST contexts [36] and on POPGym, a recent reinforcement learning benchmark for partially observable environments [44]. In both settings, the results reflect our theoretical findings: synthetic gradient propagation improves sample efficiency when non-trivial gradient uncertainty is present. We discuss this connection, along with limitations and future directions, in Section 7. Related work, proofs, experimental details, and ablation studies are provided in the appendix.

2 Vectorized Feedback on Computational Graphs

In this section, we formalize the analogy between TD versus MC estimation and synthetic gradients versus backpropagation on a layered DAG. We begin by fixing some general notation. Vectors and matrices use boldface characters, e.g., $\boldsymbol{g}$ and $\boldsymbol{J}$. Unless otherwise specified, these are random variables. A sample is denoted with a subscript, e.g., $\boldsymbol{g}_i$, while an estimator formed from samples is denoted by a hat, e.g., $\hat{\boldsymbol{g}}$. We write $[n] := \{1, 2, \ldots, n\}$. Additional notation is introduced as needed.

Consider a neural network modeled by a layered DAG $G = (V, E)$. Let $\boldsymbol{x}_v$ denote the activation at a neuron $v \in V$. For each layer $l = 0, \ldots, L$, let V_l denote the set of nodes in that layer. The input-layer (layer $l = 0$) receives external inputs, and then each non-input node $v \in V_l$ computes its activation through a (possibly stochastic) differential function based on the collective activations $\boldsymbol{z}_v$ of its parent neurons $\mathrm{pa}(v) \subseteq V_{l-1}$. Let $v_L \in V_L$ denote a terminal node whose activation $\boldsymbol{y} := \boldsymbol{x}_{v_L}$ serves as system output. Then, the terminal node v_L receives a global vectorized feedback $\boldsymbol{g}_{v_L}$. This vectorized feedback unifies several cases. In loss-based learning, one may take $\boldsymbol{g}_{v_L} = \nabla_{\boldsymbol{y}} \ell$ for a loss

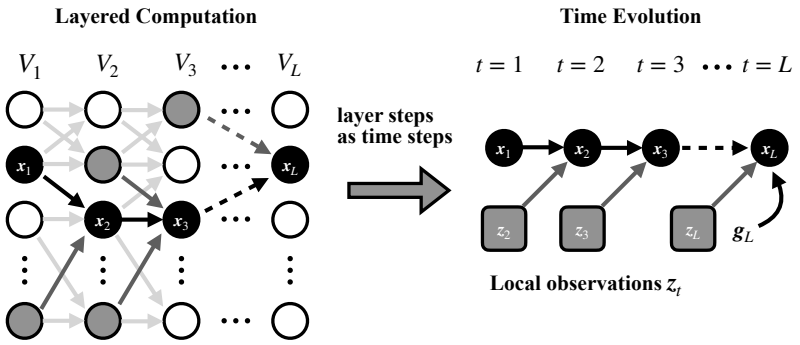


Figure 1: **From a network path to a state evolution trajectory.** Left: a single path through a layered network is highlighted in black. Right: after treating layer steps as time steps, the same path is viewed as the trajectory of a state machine. The local observation $\boldsymbol{z}_t$ includes node variable $\boldsymbol{x}_{t-1}$ by default. Shaded nodes denote local observations available at each step. Dashed arrows indicate information that skips across time. Curved arrow indicates a vectorized feedback $\boldsymbol{g}_L$.

term ℓ that may depend on the input data, target data, and the network output $\boldsymbol{y}$. In reward-based learning, the global feedback $\boldsymbol{g}_{v_L}$ may be obtained from policy gradient.

Backpropagation propagates this terminal feedback backward through the graph by repeated application of the chain rule. This recursion is exact, yet restrictive: each intermediate neuron plays a purely passive role. It receives downstream feedback and multiplies it by a Jacobian. To see the alternative viewpoint, consider a single path through the network, from a neuron $v_1 \in V_1$ in the first layer to $v_L \in V_L$ in the final layer. Since we now focus on a single path, we use $\boldsymbol{x}_l := \boldsymbol{x}_{v_l}$ for notational ease, and similarly drop the node indicator "v" for other relevant variables in the remainder of this section.

Now, let us view its layered computation $v_1 \to v_2 \to \cdots \to v_L$ as the time evolution of a state machine. At step t, the variable $\boldsymbol{x}_t$ is the local state of this machine, while the collection of variables at its parent nodes is viewed as a local observation $\boldsymbol{z}_t$. After L such local transitions, the trajectory receives terminal vectorized feedback $\boldsymbol{g}_L$. This process is illustrated in Figure 1. Under this view, backpropagation resembles Monte Carlo (MC) credit assignment: for any intermediate step t, the signal assigned to x_t is obtained by directly transporting the terminal feedback $\boldsymbol{g}_L$ through the chain rule. Thus, the entire global feedback $\boldsymbol{g}_L$ is effectively given to the whole trajectory.

Reward-based learning did not stop at MC, as TD methods can improve sample efficiency by replacing raw sampled future rewards with predicted values [59, 33]. The analogy between TD and synthetic gradients is particularly clean in a finite-horizon reward process with reward r_L only at the terminal time. For value functions, the task is to estimate the value $V_t(\boldsymbol{z}_t) = \mathbb{E}[r_L \mid \boldsymbol{z}_t]$ from the local information $\boldsymbol{z}_t$. For synthetic gradients, the task is to estimate the gradient feedback $\boldsymbol{h}_t(\boldsymbol{z}_t) = \mathbb{E}[\boldsymbol{g}_t \mid \boldsymbol{z}_t]$ from local information $\boldsymbol{z}_t$. Let $\boldsymbol{J}_t^{t+1}$ denote the transposed local Jacobian that maps feedback from step $t+1$ to step t, and let $\boldsymbol{J}_t^L := \boldsymbol{J}_t^{t+1}\boldsymbol{J}_{t+1}^{t+2}\cdots\boldsymbol{J}_{L-1}^L$ be the full transposed Jacobian from step L back to step t. Then the four estimators can be illustrated as follows.

$$\begin{array}{llllll} \textbf{MC:} & \widehat{V}_t^{\mathrm{MC}}(\boldsymbol{z}_t) & \leftarrow r_L & \textbf{TD:} & \widehat{V}_t^{\mathrm{TD}}(\boldsymbol{z}_t) & \leftarrow \widehat{V}_{t+1}^{\mathrm{TD}}(\boldsymbol{z}_{t+1}), \qquad \widehat{V}_t^{\mathrm{TD}}(\boldsymbol{z}_t) \approx \mathbb{E}[r_L \mid \boldsymbol{z}_t], \\ \textbf{BP:} & \widehat{\boldsymbol{g}}_t & \leftarrow \boldsymbol{J}_t^L\,\boldsymbol{g}_L & \textbf{SG:} & \widehat{\boldsymbol{h}}_t(\boldsymbol{z}_t) & \leftarrow \boldsymbol{J}_t^{t+1}\widehat{\boldsymbol{h}}_{t+1}(\boldsymbol{z}_{t+1}), \qquad \widehat{\boldsymbol{h}}_t(\boldsymbol{z}_t) \approx \mathbb{E}[\boldsymbol{g}_t \mid \boldsymbol{z}_t]. \end{array}$$

This extends the prior TD–SG analogy [24, 50] from a single chain of computations to a generic layered DAG. As we later show in Section 4, this generalization is necessary for discovering conditions under which synthetic gradients can improve sample efficiency. However, let us first formalize the gradient-estimation problem and metrics for sample efficiency.

3 Problem Formulation: Gradient Propagation and Gradient Estimation

In this section, we formalize gradient propagation and define the estimation criterion used throughout the paper. W*e begin by extending the notation from the previous section.* Throughout the paper, we assume that all random variables have finite second moments, so that their mean squared errors are well defined.

Consider a layered DAG $G = (V, E)$ with node-wise random variables. For each layer $l \in \{0, \dots, L\}$, node $(l, i) \in V_l$ denotes the i-th node in layer l, where $i \in [m_l]$, and $V = \cup_{l=0}^{L} V_l$. Let $\boldsymbol{x}_v \in \mathbb{R}^{d_v}$ denote the random variable at node v, for some dimension d_v. The input-layer variables $\{\boldsymbol{x}_{0,i}\}_{i=1}^{m_0}$ are drawn from an external distribution. For each layer $l \in [L]$ and each node $v \in V_l$, the node variable is generated by a differentiable function f_v with a local parameter $\boldsymbol{\theta}_v$:

$$\boldsymbol{x}_v = f_v(\boldsymbol{z}_v; \boldsymbol{\theta}_v), \qquad \text{where } \boldsymbol{z}_v := \big((\boldsymbol{x}_{v^-})_{v^- \in \mathrm{pa}(v)}, \xi_v\big).$$

Here $\mathrm{pa}(v)$ denotes the set of parent nodes of v, and ξ_v is an optional auxiliary random variable, independent of everything else, that governs the stochastic behavior of node v. For input nodes we may define $\boldsymbol{z}_{(0,i)} = \boldsymbol{x}_{(0,i)}$. For each terminal node (L, i), we are given a feedback vector $\boldsymbol{g}_{(L,i)} \in \mathbb{R}^{d_{(L,i)}}$, living in the same Euclidean space as $\boldsymbol{x}_{L,i}$. This vectorized feedback is the starting point for gradient propagation.

We next define the backward propagation of feedback vectors. For each $l \le L-1$ and each node $v \in V_l$, let $\mathrm{ch}(v) \subseteq V_{l+1}$ denote the set of children of v. The backpropagated gradient at node v is defined recursively by

$$\boldsymbol{g}_v = \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} \boldsymbol{g}_{v^+}, \qquad \text{where } \boldsymbol{J}_v^{v^+} := \left(\frac{\partial \boldsymbol{x}_{v^+}}{\partial \boldsymbol{x}_v}\right)^\top.$$

Here $\frac{\partial \boldsymbol{x}_{v^+}}{\partial \boldsymbol{x}_v}$ is the Jacobian of the local computation at node v^+, and it is generally a random matrix because it depends on the f_v's input $\boldsymbol{z}_v$, which is a random variable. This recursion is the usual backpropagation: each node aggregates the feedback from its children under the chain rule.

Synthetic gradients replace part of the exact incoming gradient with a locally generated one. ***We use a tilde to denote quantities associated with synthetic gradient propagation***, e.g., $\tilde{\boldsymbol{g}}$. We formalize this in the following definition.

Definition 1 (**Synthetic Gradient Propagation**). *For each edge $v \to v^+$ with $v \in V_l$, $l \le L-1$, and $v^+ \in \mathrm{ch}(v)$, let $\tilde{\boldsymbol{h}}_v^{v^+} \in \mathbb{R}^{d_v}$ denote a synthetic gradient generated locally by node v^+ for node v, and let $\lambda_v^{v^+} \in [0,1]$ be a mixing weight. We define the propagated feedback recursively by*

$$\tilde{\boldsymbol{g}}_v = \sum_{v^+ \in \mathrm{ch}(v)} \Big(\lambda_v^{v^+} \boldsymbol{J}_v^{v^+} \tilde{\boldsymbol{g}}_{v^+} + \big(1 - \lambda_v^{v^+}\big) \tilde{\boldsymbol{h}}_v^{v^+}\Big),$$

with terminal condition $\tilde{\boldsymbol{g}}_v = \boldsymbol{g}_v$ for all $v \in V_L$.

When all mixing weights $\lambda = 1$, it reduces to ordinary backpropagation. When all $\lambda = 0$, backward propagation is fully synthetic and can be decoupled, allowing each node to update without waiting for downstream gradients. Intermediate values allow one to trade off bias and variance.

The propagated feedback at node v, whether exact or synthetic, induces a parameter gradient at the local parameter $\boldsymbol{\theta}_v$:

$$\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} = \boldsymbol{J}_v^{\boldsymbol{\theta}} \tilde{\boldsymbol{g}}_v, \qquad \boldsymbol{g}_v^{\boldsymbol{\theta}} = \boldsymbol{J}_v^{\boldsymbol{\theta}} \boldsymbol{g}_v, \qquad \text{where } \boldsymbol{J}_v^{\boldsymbol{\theta}} := \left(\frac{\partial \boldsymbol{x}_v}{\partial \boldsymbol{\theta}_v}\right)^\top.$$

We now define the criterion of estimator quality used throughout the paper. Let $\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}}$ be an estimator of the parameter gradient at node v, e.g., the empirical mean of i.i.d. samples. We evaluate estimators by mean squared error (MSE) relative to the true target $\mathbb{E}\, \boldsymbol{g}_v^{\boldsymbol{\theta}}$.

Definition 2 (**Gradient Estimator Mean Squared Error**). *Given an estimator $\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}}$ of the parameter gradient at node v, its MSE is*

$$\tilde{\Delta}_v^2 := \mathbb{E} \left\| \hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} - \mathbb{E}\, \boldsymbol{g}_v^{\boldsymbol{\theta}} \right\|_2^2.$$

When the estimator uses only backpropagation, we write $\hat{\boldsymbol{g}}_v^{\boldsymbol{\theta}}$ and Δ_v^2 for the estimator and its MSE.

This is a classic criterion to capture sample complexity. A smaller MSE means fewer samples are needed to estimate the target gradient accurately. One may derive sample complexity bounds from $\tilde{\Delta}_v^2$ using Chebyshev-type inequalities. Stronger assumptions, e.g., sub-Gaussianity, would yield sharper concentration bounds. In this paper, we impose no tail assumptions and work directly with MSE due to its exactness.

To analyze $\tilde{\Delta}_v^2$, we decompose it into the following terms.

Definition 3. *For a fixed node v, define the expected conditional variance ($\tilde{\rho}_v^2$), the variance of the conditional mean ($\tilde{\nu}_v^2$), and the squared bias ($\tilde{b}_v^2$):*

$$\tilde{\rho}_v^2 := \mathbb{E}\left\|\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\right\|_2^2, \quad \tilde{\nu}_v^2 := \mathbb{E}\left\|\mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v] - \mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}\right\|_2^2, \quad \tilde{b}_v^2 := \left\|\mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}\right\|_2^2.$$

Similarly, we define the corresponding backpropagation quantities ρ_v^2 and ν_v^2. By definition, backpropagation is unbiased, so $b_v^2 = 0$.

Intuitively, the quantity $\tilde{\rho}_v^2$ measures the gradient signal that cannot be predicted from the node's local observations. Then, $\tilde{\nu}_v^2$ measures the part of the signal that is predictable from $\boldsymbol{z}_v$. Finally, $\tilde{b}_v^2$ measures the bias introduced by the gradient estimator.

Lemma 1. *Let $\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} = \frac{1}{n}\sum_{i\in[n]} \tilde{\boldsymbol{g}}_{v,i}^{\boldsymbol{\theta}}$ be the empirical mean of n i.i.d. samples. Then*

$$\tilde{\Delta}_v^2 = \frac{1}{n}\left(\tilde{\rho}_v^2 + \tilde{\nu}_v^2\right) + \tilde{b}_v^2.$$

In particular, for a conditionally unbiased estimator, i.e., $\mathbb{E}\left[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right] = \mathbb{E}\left[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right]$, we have

$$\tilde{\Delta}_v^2 = \frac{1}{n}\left(\tilde{\rho}_v^2 + \nu_v^2\right).$$

Lemma 1 makes the bias-variance tradeoffs explicit, which is expected from the MC vs TD analogy. The next section is devoted to characterizing when this tradeoff favors synthetic gradient propagation.

4 Conditions for When Synthetic Gradient Improves Sample Efficiency

We now turn to the central question. *When can synthetic gradients improve sample efficiency?*

To identify the conditions, we need to introduce the following definitions needed for our discovery. Define the sibling set (self-included) of a node $v \in V_l$ by

$$\mathrm{sib}(v) := \{v' \in V_l : \mathrm{pa}(v') \cap \mathrm{pa}(v) \neq \varnothing\}.$$

Definition 4 (Extended Ancestor Set). *For a non-input node $v \in V_l$, we define its sibling-extended ancestor set by the following recursion.*

$$\mathrm{an}_{\mathrm{sib}}(v) := \left(\cup_{v^- \in \mathrm{pa}(\mathrm{sib}(v))} \mathrm{an}_{\mathrm{sib}}(v^-)\right) \cup \mathrm{sib}(v),$$

i.e., $\mathrm{an}_{\mathrm{sib}}(v)$ contains the siblings of v and the extended ancestors of its siblings' parents, including its own ancestors. We define the sibling-extended ancestor set of any input node to be empty.

For any set of nodes $U \subset V$, define $\boldsymbol{Z}_U := \{\boldsymbol{z}_u : u \in U\}$ to be the collection of the corresponding local observables.

Under the MSE criterion, we discover that the sample efficiency is governed by two conditions.

Definition 5. *We introduce conditions (A) and (B).*

***(A): Uncertainty sources in the gradient feedback**: (A) holds if **any** of the following conditions is true.*

- *(A.1)* ***Stochastic terminal gradient feedback.*** *The external gradient feedback $\boldsymbol{g}_{(L,i)}$ to some terminal node $(L,i) \in V_L$ is not almost surely a deterministic function of its output $\boldsymbol{x}_{(L,i)}$.*
- *(A.2)* ***Stochastic network computation.*** *Some node $v \in V$ performs stochastic computation.*
- *(A.3)* ***Sparsely connected network.*** *Some non-input node $v \in V_l$ is sparsely connected to its previous layer, i.e., $\mathrm{pa}(v) \neq V_{l-1}$.*

(B): Conditional mean sufficiency. *Every non-input node's local observation $\boldsymbol{z}_v$ is sufficient for its conditional mean gradient $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v]$ relative to its extended ancestors' information. Formally, for any node $v \in V_l$ and any ancestor subset $U_l \subseteq \mathrm{an}_{\mathrm{sib}}(v)$, we have $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}]$ almost surely.*

Note that both conditions are stated in terms of the raw gradient $\boldsymbol{g}_v$, not in terms of any particular propagation rule. They are properties of the problem, not of the estimator.

Condition (A) ensures that there are uncertainty sources in the incoming gradient feedback at some node, and hence room for variance reduction. The first two cases are direct, while the third case is subtler: even when the terminal feedback and all node computations are deterministic, a sparsely connected node may still see randomness in its incoming gradient due to a partial observation of its upstream states.

Then, condition (B) makes the comparison sharp. First, when (A) fails, it implies that there is no realized uncertainty left in the incoming gradient. Second, when (A) holds, it ensures an oracle synthetic gradient to be unbiased. Condition (B) can also be realized concretely as a form of specialization: an expert node whose functions are largely independent of others. This interpretation will become more transparent in Section 5.

We note that (B) can be viewed as a weaker version of the Markov assumption: standard Markov assumption requires future variables to be conditionally independent of the full history given the current observables, while condition (B) requires only that the conditional mean of the incoming gradient feedback be determined by the node's current observables. Existing sample-complexity analyses comparing TD and MC estimators typically impose Markovian state representations, often in tabular or linear settings [30, 33, 5, 9, 11].

In the next section, we show that these two conditions yield a clean separation between backpropagation and synthetic gradient propagation under an idealized oracle synthetic gradient estimator. As condition (B) and the oracle assumption can be strong, we then relax both of them to develop a more general comparison in Section 4.2.

4.1 Clean Comparisons Under Idealized Conditions

First, we establish the results for the optimality of backpropagation.

Theorem 1. *Under $\neg(A) \wedge (B)$, every gradient estimator $\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}}$ formed by empirical means of i.i.d. conditionally unbiased gradient samples (i.e., $\mathbb{E}\left[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right] = \mathbb{E}\left[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right]$) has its estimation MSE greater or equal to that of the empirical-mean backpropagation estimator: $\tilde{\Delta}_v^2 \geq \Delta_v^2$ for any non-input nodes $v \in V$.*

In other words, under (B), it is necessary for (A) to hold for synthetic gradient propagation to be more sample efficient: one needs a source of uncertainty for any improvement over exact backpropagation to even be possible. We first consider an idealized synthetic gradient oracle, i.e., the prediction of the incoming gradient based on local observables:

$$\tilde{\boldsymbol{h}}_v^{v^+} = \boldsymbol{J}_v^{v^+} \tilde{\boldsymbol{h}}_{v^+}, \quad \text{where } \tilde{\boldsymbol{h}}_{v^+} := \mathbb{E}\left[\tilde{\boldsymbol{g}}_{v^+} \mid \boldsymbol{z}_{v^+}\right]. \tag{1}$$

Note that Conditions $(A.1)$–$(A.3)$ each correspond to a source variable $\boldsymbol{\chi}$ that introduces realized uncertainty at some node $v \in V$:

- under (A.1), $\boldsymbol{\chi}$ is an exogenous source of terminal feedback randomness;
- under (A.2), $\boldsymbol{\chi}$ is an internal source of stochastic computation;
- under (A.3), $\boldsymbol{\chi}$ is a collection of node variables unobserved by v due to sparse connectivity.

We impose the following technical assumption to exclude degenerate cases that would otherwise obscure the main message of our theorem.

Assumption 1 (Non-degenerate uncertainty source). *When condition (A) holds, at least one of its uncertainty sources is non-degenerate: there exist non-terminal $v \in V$ and a source variable $\boldsymbol{\chi}$ corresponding to $(A.1)$–$(A.3)$, such that $\mathbb{E}\left[\boldsymbol{g}_v^{\theta} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}\right] \neq \mathbb{E}\left[\boldsymbol{g}_v^{\theta} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{\chi}\right]$ with positive probability.*

This rules out the degenerate case where the source $\boldsymbol{\chi}$ does not introduce effective uncertainty: it should remain effective to some parameter gradient conditioned on local observables $\{\boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}\}$.

Theorem 2. *Under $(A) \wedge (B)$ and Assumption 1, consider synthetic gradient defined in equation 1 and all mixing weights $\lambda = 0$. Then, for all non-input $v \in V$: $\tilde{\Delta}_v^2 \leq \Delta_v^2$, and there exists some v: $\tilde{\Delta}_v^2 < \Delta_v^2$.*

Therefore, under condition (B), condition (A) cleanly characterizes when the oracle synthetic gradients can achieve better gradient-estimation MSE. In the next section, we relax both the condition (B) and the oracle assumption, moving to a more general regime where there is indeed a bias-variance tradeoff achieved by non-trivial mixing weights λ.

4.2 Comparison Under Generalized Conditions

Definition 6 ((B'): Generalized conditional mean sufficiency). *Every non-input node's local observation $\boldsymbol{z}_v$ is approximately sufficient for its conditional mean gradient $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v]$ relative to its extended ancestors' information. Formally, $\forall v \in V_l$ and any ancestor subset $U_l \subseteq \mathrm{an}_{\mathrm{sib}}(v)$, there is a constant $\epsilon > 0$:*

$$\|\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] - \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}]\|_2 \le \epsilon, \qquad a.s.$$

To relax the synthetic gradient oracle setting, we consider a generic synthetic gradient generator. Specifically, for some constant $\epsilon' > 0$ and for each edge $v \to v^+$, let $\tilde{\boldsymbol{h}}_v^{v^+} = \boldsymbol{J}_v^{v^+} \tilde{\boldsymbol{h}}_{v^+}$ satisfy

$$\left\| \tilde{\boldsymbol{h}}_{v^+} - \mathbb{E}[\tilde{\boldsymbol{g}}_{v^+} \mid \boldsymbol{z}_{v^+}] \right\|_2 \le \epsilon', \qquad a.s. \tag{2}$$

Thus, we allow two sources of approximation: imperfect local sufficiency through (B'), and imperfect synthetic gradient generator through equation 2. We next study how these approximations change the comparison between backpropagation and synthetic gradient propagation, and in particular how they lead to nontrivial optimal choices of the mixing weights λ. In this subsection, for all node v, we consider $\lambda_v = \lambda_v^{v^+}$ for any of its children $v^+ \in \mathrm{ch}(v)$. This means that each node treats the synthetic gradient from their children equally.

If bias is present, the optimal mixing weight must trade bias against variance. Fix a non-terminal node $v \in V_l$. The parameter gradient induced by synthetic gradient propagation can be written as

$$\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} = \lambda_v \underbrace{\boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} \tilde{\boldsymbol{g}}_{v^+}}_{=:\tilde{\boldsymbol{p}}_v} + (1-\lambda_v) \underbrace{\boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \tilde{\boldsymbol{h}}_v^{v^+}}_{=:\tilde{\boldsymbol{s}}_v} = \lambda_v \tilde{\boldsymbol{p}}_v + (1-\lambda_v)\tilde{\boldsymbol{s}}_v.$$

Here $\tilde{\boldsymbol{p}}_v$ is the component obtained by backpropagating gradient feedback through Jacobians, whereas $\tilde{\boldsymbol{s}}_v$ is the component obtained from synthetic gradients.

Let $\hat{\tilde{\boldsymbol{p}}}_v$ and $\hat{\tilde{\boldsymbol{s}}}_v$ denote the empirical means of $\tilde{\boldsymbol{p}}_v$ and $\tilde{\boldsymbol{s}}_v$, respectively, based on n i.i.d. samples. Then the MSE $\tilde{\Delta}_v^2$ of the empirical mean estimator and the optimal mixing weight $\lambda^\star$ is

$$\tilde{\Delta}_v^2 = \mathbb{E}\,\|\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}\|_2^2 = \mathbb{E}\,\left\|\lambda_v \hat{\tilde{\boldsymbol{p}}}_v + (1-\lambda_v)\hat{\tilde{\boldsymbol{s}}}_v - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}\right\|_2^2, \quad \text{and } \lambda_v^\star = \operatorname*{arg\,min}_{\lambda_v \in [0,1]} \tilde{\Delta}_v^2.$$

If $\tilde{\boldsymbol{p}}_v = \tilde{\boldsymbol{s}}_v$ almost surely, then λ_v is irrelevant and we pick $\lambda_v^\star = 0$. Otherwise, standard quadratic optimization shows that there is a unique solution:

$$\lambda_v^\star = \frac{\mathbb{E}\,\|\hat{\tilde{\boldsymbol{s}}}_v - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}\|_2^2 - \mathbb{E}\,\langle \hat{\tilde{\boldsymbol{s}}}_v - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}, \hat{\tilde{\boldsymbol{p}}}_v - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}\rangle}{\mathbb{E}\,\|\hat{\tilde{\boldsymbol{s}}}_v - \hat{\tilde{\boldsymbol{p}}}_v\|_2^2} \quad \text{projection onto } [0,1]. \tag{3}$$

Thus $\lambda_v^\star < 1$ if incorporating a nonzero synthetic component improves over pure backpropagation at node v. The next theorem gives a sufficient condition for this to happen.

Theorem 3. *Under condition (B') and equation 2, assume bounded jacobians $\forall v, v^+ : \max\{\|\boldsymbol{J}_v^{v^+}\|_2, \|\boldsymbol{J}_v^{\boldsymbol{\theta}}\|_2\} \le J$. A sufficient condition for $\lambda_v^\star < 1$ is:*

$$\frac{\mathrm{tr}\,\mathrm{Cov}(\tilde{\boldsymbol{p}}_v) - \mathrm{tr}\,\mathrm{Cov}(\tilde{\boldsymbol{s}}_v)}{n} > \left(J^2(\epsilon + \epsilon') \sum_{v^+ \in \mathrm{ch}(v)} (1 + c_{v^+}) \right)^2$$

where $\forall u \in V : c_u$ is defined recursively via

$$c_u = J \sum_{u^+ \in \mathrm{ch}(u)} (1 - \lambda_u + c_{u^+}), \quad \textit{and for terminal nodes } \forall u \in V_L : c_u = 0.$$

Here $\mathrm{tr}\,\mathrm{Cov}(\cdot)$ denotes the trace of the covariance matrix, i.e., the total variance of a random vector, i.e., $\mathrm{tr}\,\mathrm{Cov}(\boldsymbol{g}) = \mathbb{E}\,\|\boldsymbol{g} - \mathbb{E}\,\boldsymbol{g}\|_2^2$.

Theorem 3 has a simple interpretation. The left-hand side is the variance reduction obtained by using the synthetic component $\tilde{\boldsymbol{s}}_v$ instead of the backpropagated component $\tilde{\boldsymbol{p}}_v$, after averaging over n samples. The right-hand side is the bias penalty induced by synthetic gradients. Thus, a synthetic mixture can be beneficial whenever the variance reduction dominates the induced bias.

A question remains: how large can this sample efficiency advantage be when the synthetic-gradient generator must itself be estimated from data rather than treated as given? In the next section, we answer this question by constructing examples in which the advantage can be arbitrarily large.

5 An Example of Finite-Partition Expert Network

We consider a two-hidden-layer architecture in which the first hidden layer a single node, and the second hidden layer has nodes $V_2 = \{(2, j)\}_{j\in[m_2]}$. In the rest of the section, we denote $j := (2, j)$. We focus on gradient estimation on the node $(1, 1) \in V_1$ in the first layer, and thus in this section *we omit the subscript when referring to this target node*. For example, $\boldsymbol{g}^{\boldsymbol{\theta}}$ refers to the backpropagated gradient on this target node $(1, 1) \in V_1$.

The first layer node computes a d-dim feature representation $\boldsymbol{x} = f(\boldsymbol{z}; \boldsymbol{\theta}) \in \mathbb{R}^d$, where $\boldsymbol{z}$ is the input sampled from an external source (i.e., from layer $l = 0$). The second layer contains m_2 expert nodes, where each expert $j \in [m_2]$ specializes in a k-dimensional subspace of the feature vector $\boldsymbol{x}$. Formally, each expert j effectively acts on $\boldsymbol{u}_j = P_j\boldsymbol{x}$, where $P_j : \mathbb{R}^d \to \mathbb{R}^k$ is a projection. Each expert outputs $y_j = f_j(\boldsymbol{u}_j) \in \mathbb{R}$. Then, the final node collects the vector of expert outputs $\boldsymbol{y} = (y_1, \ldots, y_{m_2}) \in \mathbb{R}^{m_2}$.

Each expert node j solves an independent sub-task, such that the global loss function decomposes across the experts. The global loss function and its gradient to expert node j are respectively $\mathcal{L} := \sum_{j=1}^{m_2} \ell_j(\boldsymbol{u}_j, y_j)$, and $\boldsymbol{g}_j := \frac{\partial \mathcal{L}}{\partial y_j} = \frac{\partial \ell_j(\boldsymbol{u}_j, y_j)}{\partial y_j}$. We compare ordinary backpropagation and synthetic-gradient propagation under three sources of stochastic expert feedback, each inspired by a sub-condition in condition (A).

- **Case 1: noisy terminal feedback.** The external feedback carries i.i.d. noise: $\hat{\boldsymbol{g}}_j = \boldsymbol{g}_j + \xi_j, \qquad \mathbb{E}[\xi_j] = 0, \qquad \mathbb{E}[\xi_j^2] = \sigma^2$.
- **Case 2: stochastic expert activation.** Each expert is activated with probability p independently every time. The gradient feedback is thus $\hat{\boldsymbol{g}}_j = \frac{\eta_j}{p}\boldsymbol{g}_j, \qquad \eta_j \sim \text{Bernoulli}(p)$.
- **Case 3: partial observation.** In this case, the terminal node sees some other variables $\boldsymbol{z}'_j$, independent across j, such that the global loss term $\mathcal{L}' = \sum_{j=1}^{m_2} \ell_j(\boldsymbol{u}_j, y_j, \boldsymbol{z}'_j)$, and its gradient to expert j becomes $\hat{\boldsymbol{g}}_j = \frac{\partial \mathcal{L}'}{\partial y_j} = \frac{\partial \ell_j(\boldsymbol{u}_j, y_j, \boldsymbol{z}'_j)}{\partial y_j}$.

The synthetic gradient from expert j to the target node $(1, 1)$ needs to be estimated: $\hat{\tilde{\boldsymbol{h}}}^j(\boldsymbol{u}_j) \approx \boldsymbol{J}^j\mathbb{E}[\boldsymbol{g}_j \mid \boldsymbol{z}_j]$. To characterize sample efficiency when the synthetic gradient estimator is itself learned from data, we restrict attention to discrete inputs. This yields a tabular setting, analogous to those commonly used in sample-complexity analyses of TD and MC estimators [9, 11].

Let $\boldsymbol{z} \in \mathcal{X}$ denote the input variable to our target node $(1, 1) \in V_1$. Let the input space $\mathcal{X}$ be partitioned into $R = 2^d$ sets $\mathcal{X} = \bigsqcup_{r=1}^{R} \mathcal{X}_r$. We write $\boldsymbol{z}_r$ for the representative input in partition r, and suppose the input variable $\boldsymbol{z}$ is sampled from a uniform distribution over these representative inputs. The target node's function is piecewise parameterized:

$$f(\boldsymbol{z}; \boldsymbol{\theta}) = \sum_{r=1}^{R} \mathbf{1}\{\boldsymbol{z} \in \mathcal{X}_r\} f_{\boldsymbol{\theta}_r}(\boldsymbol{z}), \quad \text{where } \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R). \tag{4}$$

Without loss of generality, let $\boldsymbol{x}_r = f_{\boldsymbol{\theta}_r}(\boldsymbol{z}_r) \in \{\pm 1\}^d$. We assume that the map $r \mapsto \boldsymbol{x}_r$ is one-to-one and onto $\{\pm 1\}^d$. Therefore, drawing r uniformly from $[R]$ is equivalent to drawing $\boldsymbol{x}_r$ uniformly from $\{\pm 1\}^d$. Let each expert projection P_j selects exactly k coordinates of $\boldsymbol{x}_r$. Thus $\boldsymbol{u}_{j,r} := P_j\boldsymbol{x}_r \in \{\pm 1\}^k$. For any $\boldsymbol{u} \in \{\pm 1\}^k$, let the equivalent class under this projection be

$$\mathcal{F}_j(r) := \{s \in [R] : P_j\boldsymbol{x}_s = P_j\boldsymbol{x}_r\}, \qquad \text{and thus} \quad \forall r \in [R] : \ |\mathcal{F}_j(r)| = 2^{d-k}.$$

In this example, this state-pooling effect is the source of the sample-efficiency advantage of each expert nodes' synthetic gradient $\tilde{\boldsymbol{h}}^j \approx \boldsymbol{J}^j \mathbb{E}[\hat{\boldsymbol{g}}_j \mid \boldsymbol{u}_j]$. We defer the full construction of the example and the theorem statement to Appendix G, and give an informal version below.

Theorem 4 (Informal version of Theorem 5)**.** *Under the settings described in this section, for all the three cases we have*

$$\Delta^2 = 2^{d-k}\,\tilde{\Delta}^2.$$

Thus, in this example, the estimation MSE of backpropagation can be arbitrarily larger than that of synthetic gradient propagation. In particular, the gap scales exponentially with the dimension reduction induced by the expert nodes' subspace specialization.

Our theoretical analysis focuses on gradient-estimation MSE. We discuss its connection to optimization convergence in Appendix J.1. In this example of the expert network, Condition (B) (Definition 5) holds by construction. We discuss when this condition can be expected to hold more generally in Section J.2. We next turn to numerical experiments to investigate the benefits of synthetic gradients.

6 Experiments

The experiments serve a single purpose: *to investigate whether synthetic gradients can still provide sample-efficiency benefits in more applied settings where the conditions in Section 4 may not hold and the synthetic-gradient predictors must be learned from data.* We investigate synthetic gradients in three experimental settings[1]: from the finite-partition expert network as seen in the previous section, to learning contextual bandits and partially observable RL.

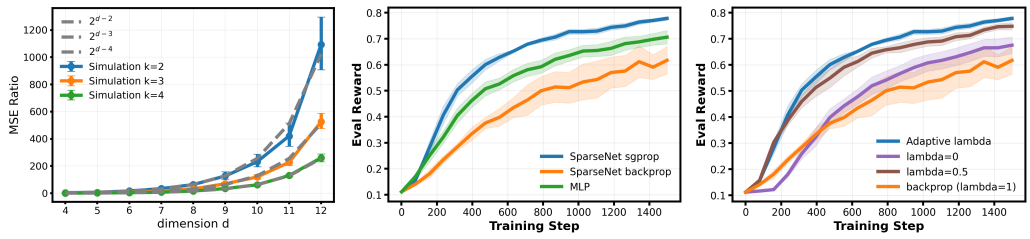
Simulation of the finite-partition expert network. We first simulate the expert network from Section 5. Figure 2 (a) shows the estimated MSE ratio for the setting of noisy terminal feedback (Case 1). The simulated MSE ratio grows exponentially with $d-k$, matching the predicted results from Theorem 5. implementation details and simulation results for other cases are shown in Appendix I.1.

SparseNet. Following our theoretical findings, we investigate a sparsely connected neural network architecture, where every neuron can propagate synthetic gradients. For a neuron to perform both non-linear computations and synthetic gradient prediction, a natural idea is to use a soft-gating mechanism where it has shared keys for input matching and separate heads for non-linear transformation and synthetic gradient prediction. Each neuron also estimates their optimal mixing weight according to equation 3. This architecture is named `SparseNet`, as detailed in Appendix I.2.

MNIST contextual bandit. We next test a contextual bandit task learned with policy gradient. The context is a sampled MNIST image [36], the action is a sampled digit label, and the reward is one for a correct label and zero otherwise; during training the reward is flipped with probability $p_{\text{flip}} = 0.4$ to simulate a noisy environment. The SparseNet is a five-layer sparse feedforward network with width 200 and 5 parents per neuron. The first layer receives input images, and final layer's output is converted into probability logits for sampling actions. We compare SparseNet with synthetic gradient propagation and backpropagation, and a dense MLP. Figures 2 (b)&(c) show that synthetic gradient with adaptive mixing weights improves convergence speed relative to the number of samples seen. All curves report means over 6 seeds with standard error of the mean (SEM). Implementation details and ablation on the number of parents, p_{flip}, learning rates, and λ dynamics are shown in Appendix I.3.

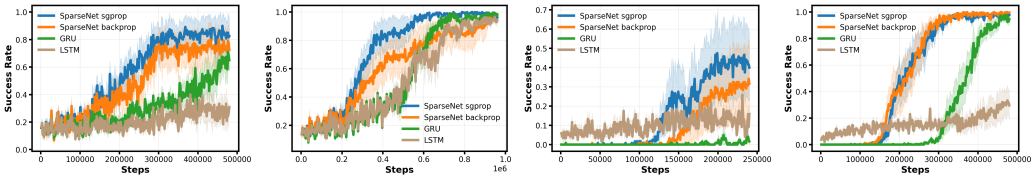
POPGym reinforcement learning. We investigate whether synthetic gradient propagation can help recurrent neural network learning in RL. We use two maze navigation environments from POPGym [44]: `labyrinth escape` and `labyrinth explore`, where only adjacent tiles can be observed by the agent. SparseNet is made recurrent by applying the same sparse neuron transition through time: 128 neurons are each connected to 6 other neurons and optionally the external input, where the connectivity is uniformly randomly picked and fixed at initialization. During each step, the collection of all neuron outputs is converted into probability logits for sampling actions. Synthetic gradient propagation is applied during backpropagation through time (BPTT) to mix backpropagated gradients with synthetic gradients. We compare SparseNet (sgprop), SparseNet (backprop), GRU [12] and LSTM [53]. Figure 3 shows the four main training curves. All curves report means over 6 seeds

[1] Code is available at `https://github.com/jackyzyb/sgprop`



(a) Expert network (b) bandit training dynamics (c) bandit mixing weight

Figure 2: **Finite-partition and MNIST bandit results.** (a) In the expert simulation, synthetic gradients reduces gradient estimation MSE relative to backpropagation, where the reduction ratio fits the theoretical prediction. (b) In the MNIST contextual bandit, synthetic gradients requires fewer steps (numer of samples seen) to converge. (c) Mixing weights sensitivity in the MNIST contextual bandit experiment.



(a) `escape`, 20 trials. (b) `escape`, 40 trials. (c) `explore`, 20 trials. (d) `explore`, 40 trials.

Figure 3: **POPGym `labyrinth escape` and `labyrinth explore`.** Training dynamics for all methods with 20 sampled trials per gradient update and 40 sampled trials per gradient update. Across these settings, synthetic gradients improve over backpropagation in few-sample regimes.

with standard error of the mean (SEM). Implementation details and ablations on learning rates and adaptive mixing weights are presented in Appendix I.4.

Ablation studies. We provide three additional ablation studies. In Appendix K.1, by varying the training effort devoted to synthetic-gradient predictors, we show that lower gradient-estimation MSE is associated with higher evaluation reward on the MNIST bandit task. In Appendix K.2, we examine how closely Condition (B) holds in SparseNet by comparing locally predicted gradients with their ground truth. In Appendix K.3, we explore a preliminary extension to GPT-style transformers with MoE layers, observing lower validation loss than backpropagation when training with noisy next-token targets.

7 Discussion, Limitation, and Future Work

Our analysis suggests that synthetic gradients have a sample-efficiency role that is previously under-explored. The conditions under which such benefits may arise can be interpreted as: $(A.1)$ stochastic external feedback; $(A.2)$ stochastic neuron computation; $(A.3)$ neuron sparsely connected to others; and (B) neuron specialization and local sufficiency. We discuss this perspective in relation to prior work on synthetic gradients, TD versus MC value estimation, and biological learning in Appendix A. Several limitations remain. First, Condition (B) can be strong, and it is not comprehensively explored how well it is satisfied in real network systems. Analysis under further relaxations of this condition is an important direction for future theoretical work. Second, our theory focuses primarily on estimation MSE. A more complete sample-complexity analysis could impose explicit distributional assumptions and derive concentration bounds. Moreover, the adaptive mixing weight used in our experiment is rather rudimentary, and better algorithms for estimating these weights may have important practical benefits. Finally, our experiments are intended as proof-of-concept demonstrations rather than a comprehensive empirical evaluation. Future work should test synthetic gradient propagation across broader architectures, tasks, and scales.

Acknowledgments

This work is partially supported by NSF 2046795, 2205329, and 2504264, NIH, ARPA-H, the MacArthur Foundation, Good Ventures, Schmidt Sciences, the Hasso Plattner Foundation, and Stanford HAI.

References

[1] J Yu Angela and Peter Dayan. Uncertainty, neuromodulation, and attention. *Neuron*, 46(4):681–692, 2005.

[2] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pages 736–745. PMLR, 2020.

[3] Guillaume Bellec, Franz Scherr, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049*, 2019.

[4] Eloïse Berthier, Ziad Kobeissi, and Francis Bach. A non-asymptotic analysis of non-parametric temporal-difference learning. *Advances in Neural Information Processing Systems*, 35:7599–7613, 2022.

[5] Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1691–1692. PMLR, 06–09 Jul 2018.

[6] Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on learning theory*, pages 1691–1692. PMLR, 2018.

[7] Qi Cai, Zhuoran Yang, Jason D Lee, and Zhaoran Wang. Neural temporal-difference learning converges to global optima. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] Semih Cayci, Niao He, and R Srikant. Finite-time analysis of natural actor-critic for pomdps. *SIAM Journal on Mathematics of Data Science*, 6(4):869–896, 2024.

[9] David Cheikhi and Daniel Russo. On the statistical benefits of temporal difference learning. In *International Conference on Machine Learning*, pages 4269–4293. PMLR, 2023.

[10] Yitao Chen, Kaiqi Zhao, Baoxin Li, and Ming Zhao. Exploring the use of synthetic gradients for distributed deep learning across cloud and edge resources. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[11] Xiaoou Cheng and Jonathan Weare. The surprising efficiency of temporal difference learning for rare event prediction. *Advances in Neural Information Processing Systems*, 37:81257–81286, 2024.

[12] Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1724–1734, 2014.

[13] Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.

[14] Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *International Conference on Machine Learning*, pages 904–912. PMLR, 2017.

[15] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

[16] Yury Demidovich, Grigory Malinovsky, Igor Sokolov, and Peter Richtárik. A guide through the zoo of biased sgd. *Advances in Neural Information Processing Systems*, 36:23158–23171, 2023.

[17] Derek Driggs, Jingwei Liang, and Carola-Bibiane Schönlieb. On biased stochastic gradient estimation. *Journal of Machine Learning Research*, 23(24):1–43, 2022.

[18] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[19] Steffen Grunewalder, Sepp Hochreiter, and Klaus Obermayer. Optimality of lstd and its relation to mc. In *2007 International Joint Conference on Neural Networks*, pages 338–343. IEEE, 2007.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2(3):5, 2022.

[22] Shell Xu Hu, Pablo Garcia Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil Lawrence, and Andreas Damianou. Empirical bayes transductive meta-learning with synthetic gradients. In *International Conference on Learning Representations*, 2020.

[23] Zhouyuan Huo, Bin Gu, Heng Huang, et al. Decoupled parallel backpropagation with convergence guarantee. In *International Conference on Machine Learning*, pages 2098–2106. PMLR, 2018.

[24] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International conference on machine learning*, pages 1627–1635. PMLR, 2017.

[25] Vahid Janfaza, Shantanu Mandal, Farabi Mahmud, and Abdullah Muzahid. Ada-gp: Accelerating dnn training by adaptive gradient prediction. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 1092–1105, 2023.

[26] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

[27] Nancy Kanwisher, Josh McDermott, and Marvin M Chun. The fusiform face area: a module in human extrastriate cortex specialized for face perception. *Journal of neuroscience*, 17(11):4302–4311, 1997.

[28] Belhal Karimi, Blazej Miasojedow, Eric Moulines, and Hoi-To Wai. Non-asymptotic analysis of biased stochastic approximation scheme. In *Conference on Learning Theory*, pages 1944–1974. PMLR, 2019.

[29] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[30] Michael J Kearns and Satinder Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pages 142–147, 2000.

[31] Koulik Khamaru, Ashwin Pananjady, Feng Ruan, Martin J Wainwright, and Michael I Jordan. Is temporal difference learning optimal? an instance-dependent analysis. *SIAM Journal on Mathematics of Data Science*, 3(4):1013–1040, 2021.

[32] Alan Larkman, Timo Hannay, Ken Stratford, and Julian Jack. Presynaptic release probability influences the locus of long-term potentiation. *Nature*, 360(6399):70–73, 1992.

[33] Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Finite-sample analysis of lstd. In *ICML-27th International Conference on Machine Learning*, pages 615–622, 2010.

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

[35] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[37] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

[38] Gen Li, Weichen Wu, Yuejie Chi, Cong Ma, Alessandro Rinaldo, and Yuting Wei. High-probability sample complexities for policy evaluation with linear function approximation. *IEEE transactions on information theory*, 70(8):5969–5999, 2024.

[39] Tianjiao Li, Guanghui Lan, and Ashwin Pananjady. Accelerated and instance-optimal policy evaluation with linear function approximation. *SIAM Journal on Mathematics of Data Science*, 5(1):174–200, 2023.

[40] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276, 2016.

[41] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT*, 16(2):146–160, June 1976.

[42] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

[43] Takeru Miyato, Daisuke Okanohara, Shin-ichi Maeda, and Masanori Koyama. Synthetic gradient methods with virtual forward-backward networks. 2017.

[44] Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. Popgym: Benchmarking partially observable reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[45] Robby Neven, Marian Verhelst, Tinne Tuytelaars, and Toon Goedemé. Feed-forward on-edge fine-tuning using static synthetic gradient modules. In *European Conference on Computer Vision*, pages 131–146. Springer, 2020.

[46] Seung Wook Oh, Julie A Harris, Lydia Ng, Brent Winslow, Nicholas Cain, Stefan Mihalas, Quanxin Wang, Chris Lau, Leonard Kuan, Alex M Henry, et al. A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214, 2014.

[47] Shogo Ohmae and Javier F Medina. Climbing fibers encode a temporal-difference prediction error during cerebellar learning in mice. *Nature neuroscience*, 18(12):1798–1803, 2015.

[48] Gergő Orbán, Pietro Berkes, József Fiser, and Máté Lengyel. Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, 92(2):530–543, 2016.

[49] Joseph Pemberton, Ellen Boven, Richard Apps, and Rui Ponte Costa. Cortico-cerebellar networks as decoupling neural interfaces. *Advances in neural information processing systems*, 34:7745–7759, 2021.

[50] Joseph Oliver Pemberton and Rui Ponte Costa. BP($\mathbf{\lambda}$): Online learning via synthetic gradients. *Transactions on Machine Learning Research*, 2024.

[51] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[52] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[53] Jürgen Schmidhuber, Sepp Hochreiter, et al. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.

[54] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[55] David Silver, Anirudh Goyal, Ivo Danihelka, Matteo Hessel, and Hado van Hasselt. Learning by directional gradient descent. In *International Conference on Learning Representations*, 2021.

[56] William R Softky and Christof Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsps. *Journal of neuroscience*, 13(1):334–350, 1993.

[57] Sen Song, Per Jesper Sjöström, Markus Reigl, Sacha Nelson, and Dmitri B Chklovskii. Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS biology*, 3(3):e68, 2005.

[58] Tao Sun, Dongsheng Li, and Bao Wang. Finite-time analysis of adaptive temporal difference learning with deep neural networks. *Advances in Neural Information Processing Systems*, 35:19592–19604, 2022.

[59] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[60] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[61] Bosiljka Tasic, Zizhen Yao, Lucas T Graybuck, Kimberly A Smith, Thuc Nghi Nguyen, Darren Bertagnolli, Jeff Goldy, Emma Garren, Michael N Economo, Sarada Viswanathan, et al. Shared and distinct transcriptomic cell types across neocortical areas. *Nature*, 563(7729):72–78, 2018.

[62] Xiao-Jing Wang. *Theoretical neuroscience: Understanding cognition*. CRC press, 2025.

[63] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. Springer, 1981.

[64] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Appendix

A Related Work

Synthetic Gradients. Synthetic gradients were introduced as decoupled neural interfaces to remove the locking constraints imposed by backpropagation, allowing layers to update using locally predicted gradients rather than waiting for a full backward pass [24]. Their convergence was later analyzed to show that synthetic-gradient training can lead to different layer-wise representations [14]. Subsequent variants improved the design and training of synthetic-gradient models [13, 43, 2], including evaluations on larger-scale architectures such as convolutional neural networks [35, 34] and ResNets [20]. Synthetic gradients, defined as a local learning rule, has also been studied in settings where globally backpropagated gradients are not fully available, including online recurrent learning [3, 55, 50], transductive meta-learning [22], distributed or edge training [10, 45], and biologically learning [3, 26, 49]. These works suggest application-specific benefits of synthetic gradients, as discussed below.

Benefits of Synthetic Gradients versus Backpropagation. One of the clearest benefits of synthetic gradients discovered by prior work is a reduced horizon dependence in recurrent learning [24, 55, 50]. In this setting, backpropagation through time (BPTT) is often applied only over a truncated window, while synthetic gradients can estimate feedback gradients beyond the truncation boundary. Another benefit arises when training is bottlenecked by synchronization or communication: synthetic gradients allow modules to update asynchronously and can improve throughput in distributed settings [10, 45]. However, when exact backpropagation is available, prior work generally finds that synthetic gradients, without additional mechanisms, do not match the performance of backpropagation. In this work, we investigate synthetic gradients from a fresh perspective: sample efficiency, and we compare against unconstrained backpropagation. As discussed in Section 2, the sample efficiency benefit is analogous to the advantage of temporal-difference methods over Monte Carlo methods in value function estimation.

TD versus MC. Temporal-difference (TD) methods [59] have long been studied as alternatives to Monte Carlo (MC) value function estimation, replacing unbiased full-return targets with bootstrapped recursive targets. Despite empirical success, understanding when this bootstrapping improves sample efficiency remains a longstanding question and an active research area. Early work on phased TD established an explicit bias-variance tradeoff [30]. Related oracle-style analyses further showed that TD methods can exploit Markov structure more efficiently than MC in Markov reward processes [19]. Subsequent work developed nonasymptotic guarantees for TD and its variants in tabular and linear function approximation settings [33, 6, 9, 38, 11], while lower bounds showed that plain TD is not always statistically optimal [31, 39]. More recent analyses extend finite-sample TD theory to nonlinear function approximation [7, 58, 4] and partially observable environments [8]. Overall, the literature suggests that TD is not uniformly superior to MC, but can be substantially more sample-efficient when it successfully exploits the Bellman structure.

Discussion of Biological Learning. As detailed in Section 4 and illustrated in Section 5, we identify four conditions under which synthetic gradients may offer sample-efficiency benefits: $(A.1)$ stochastic external feedback, $(A.2)$ stochastic neuron computation, $(A.3)$ sparse connectivity, and (B) neuron specialization and local sufficiency. Some of these conditions have analogies in biological learning, which we briefly discuss here as broader context. Biological learning in real world is intrinsically stochastic, which may be modeled by noisy and delayed external reward signals [54, 1, 47]. Internally, stochasticity is also a well-established feature of neural computation: cortical neurons exhibit trial-to-trial spiking and synaptic transmission is probabilistic [32, 56, 15, 48, 62]. Moreover, biological networks are sparse and structured rather than densely connected [57, 46, 62]. Finally, evidence suggests that cortical areas or neuronal populations can exhibit a certain level of specialization and task decomposition [27, 61, 42, 62]. These observations offer a tentative biological perspective on our assumptions; whether they support a more concrete relationship with synthetic-gradient propagation remains an open question for future investigation.

B Proof of Lemma 1

Lemma 1. Let $\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} = \frac{1}{n}\sum_{i\in[n]} \tilde{\boldsymbol{g}}_{v,i}^{\boldsymbol{\theta}}$ be the empirical mean of n i.i.d. samples. Then

$$\tilde{\Delta}_v^2 = \frac{1}{n}\left(\tilde{\rho}_v^2 + \tilde{\nu}_v^2\right) + \tilde{b}_v^2.$$

In particular, for an conditionally unbiased estimator, i.e., $\mathbb{E}\left[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right] = \mathbb{E}\left[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right]$, we have

$$\tilde{\Delta}_v^2 = \frac{1}{n}\left(\tilde{\rho}_v^2 + \nu_v^2\right).$$

Proof. We start from the standard bias-variance decomposition:

$$\begin{aligned}
\tilde{\Delta}_v^2 &= \mathbb{E}\left\|\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}\right\|_2^2 \\
&= \mathbb{E}\left\|\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}\right\|_2^2 + \left\|\mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\boldsymbol{g}_v^{\boldsymbol{\theta}}\right\|_2^2 \\
&= \mathbb{E}\left\|\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}\right\|_2^2 + \tilde{b}_v^2.
\end{aligned}$$

Since $\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}}$ is the empirical mean of n i.i.d. samples,

$$\mathbb{E}\left\|\hat{\tilde{\boldsymbol{g}}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}\right\|_2^2 = \frac{1}{n}\mathbb{E}\left\|\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}\right\|_2^2.$$

Now add and subtract $\mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]$, and apply the law of total variance:

$$\begin{aligned}
\mathbb{E}\left\|\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}\right\|_2^2 &= \mathbb{E}\left\|\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\right\|_2^2 + \mathbb{E}\left\|\mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v] - \mathbb{E}\,\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}\right\|_2^2 \\
&= \tilde{\rho}_v^2 + \tilde{\nu}_v^2.
\end{aligned}$$

Combining the two results shows

$$\tilde{\Delta}_v^2 = \frac{1}{n}\left(\tilde{\rho}_v^2 + \tilde{\nu}_v^2\right) + \tilde{b}_v^2.$$

For an unbiased estimator, i.e., $\mathbb{E}\left[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right] = \mathbb{E}\left[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right]$, we have $\tilde{b}_v^2 = 0$ and $\tilde{\nu}_v^2 = \nu_v^2$. Thus,

$$\tilde{\Delta}_v^2 = \frac{1}{n}\left(\tilde{\rho}_v^2 + \nu_v^2\right).$$

□

C Proof of Theorem 1

Theorem 1. Under $\neg(A) \wedge (B)$, every gradient estimator $\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}}$ formed by empirical means of i.i.d. conditionally unbiased gradient samples (i.e., $\mathbb{E}\left[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right] = \mathbb{E}\left[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v\right]$) has its estimation MSE greater or equal to that of the empirical-mean backpropagation estimator: $\tilde{\Delta}_v^2 \geq \Delta_v^2$ for any non-input nodes $v \in V$..

Proof. First, we may examine the $\neg(A)$ in more detail. $\neg(A)$ needs **all** the following conditions to be true.

- ¬(A.1) **Deterministic terminal gradient feedback.** The gradient feedback $\boldsymbol{g}_{(L,i)}$ to any terminal nodes $(L,i) \in V_L$ is almost surely a deterministic function of the terminal output $\boldsymbol{x}_{(L,i)}$.
- ¬(A.2) **Deterministic network computation.** All nodes $v \in V$ perform deterministic computations.
- ¬(A.3) **Fully connected network.** All non-input nodes are fully connected to their previous layer.

Next, we prove the following statement for all non-input nodes v.

$$\neg(A.1) \wedge \neg(A.2) \wedge \neg(A.3) \wedge (B) \implies \rho_v^2 = 0.$$

We prove this theorem by induction.

Define $S(l)$ to be the following statement: *the gradient feedback $\boldsymbol{g}_{(l,i)}$ to any node $(l,i) \in V_l$ is almost surely a deterministic function of the node's local observation $\boldsymbol{z}_{(l,i)}$.*

Note that $S(L)$ automatically holds under $\neg(A.1)$. Now suppose $S(l)$ holds for a layer $l > 1$, i.e., any node $v \in V_l$ satisfies

$$\boldsymbol{g}_v = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v], \qquad a.s.. \tag{5}$$

Since the jacobian $\boldsymbol{J}_v^{\boldsymbol{\theta}}$ is a function of $\boldsymbol{z}_v$, by equation 5 we have

$$\boldsymbol{g}_v^{\boldsymbol{\theta}} = \boldsymbol{J}_v^{\boldsymbol{\theta}} \boldsymbol{g}_v = \boldsymbol{J}_v^{\boldsymbol{\theta}} \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{J}_v^{\boldsymbol{\theta}} \boldsymbol{g}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v], \qquad a.s..$$

This implies

$$\rho_v^2 = \mathbb{E}\Big[\big\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\big\|_2^2\Big] = 0.$$

This proves the zero conditional variance for nodes $v \in V_l$ in this layer. Next, we prove that $S(l-1)$ holds.

For any node $v \in V_l$ in the layer l, by $\neg(A.3)$ and (B) we have

$$\forall v^- \in V_{l-1}: \;\; \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{z}_{v^-}]. \tag{6}$$

For $\forall v^- \in V_{l-1}$, we start by definition and then apply those conditions:

$$\begin{aligned}
\boldsymbol{g}_{v^-} &= \sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \boldsymbol{g}_v \\
&= \sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{z}_{v^-}] \qquad a.s. && \text{(by (5) and (6))} \\
&= \sum_{v \in \mathrm{ch}(v^-)} \mathbb{E}[\boldsymbol{J}_{v^-}^{v} \boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{z}_{v^-}] \qquad a.s. && (\boldsymbol{J}_{v^-}^{v} \text{ is a function of } \boldsymbol{z}_v) \\
&= \sum_{v \in \mathrm{ch}(v^-)} \mathbb{E}[\boldsymbol{J}_{v^-}^{v} \boldsymbol{g}_v \mid \boldsymbol{z}_{v^-}] \qquad a.s. \qquad && \text{(by } \neg(A.2) \text{ and } \neg(A.3), \boldsymbol{z}_v \text{ is a function of } \boldsymbol{z}_{v^-}) \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \boldsymbol{g}_v \,\middle|\, \boldsymbol{z}_{v^-}\right] \qquad a.s. \\
&= \mathbb{E}\left[\boldsymbol{g}_{v^-} \mid \boldsymbol{z}_{v^-}\right] \qquad a.s.
\end{aligned}$$

This proves $S(l-1)$. Therefore, by induction, backpropagation achieves $\rho_v^2 = 0$ for all non-input nodes v. By Lemma 1, the estimator MSE from backpropagation is.

$$\Delta_v^2 = \frac{1}{n}\left(\rho_v^2 + \nu_v^2\right) = \frac{1}{n}\nu_v^2.$$

Therefore, by Lemma 1, any unbiased estimator cannot have smaller estimation MSE:

$$\tilde{\Delta}_v^2 = \frac{1}{n}\left(\tilde{\rho}_v^2 + \nu_v^2\right) \geq \frac{1}{n}\nu_v^2 = \Delta_v^2.$$

□

D Proof of Theorem 2

Theorem 2 Under $(A) \wedge (B)$ and Assumption 1, consider synthetic gradient defined in equation 1 and all mixing weights $\lambda = 0$. Then, for all non-input $v \in V$: $\tilde{\Delta}_v^2 \leq \Delta_v^2$, and there exists some v: $\tilde{\Delta}_v^2 < \Delta_v^2$.

Proof. We begin by proving the following lemma.

Lemma 2. *Under (B), synthetic gradient propagation with equation 1 and all $\lambda = 0$ leads to unbiased $\tilde{\boldsymbol{g}}_v$. Formally, for all non-input nodes v we have*

$$\mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v].$$

Proof. We prove by induction. First, for terminal nodes $\forall v \in V_L$, $\tilde{\boldsymbol{g}}_v = \boldsymbol{g}_v$ by definition, and thus $\mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v]$.

Now, suppose for a layer $l > 1$ we have $\forall v \in V_l : \mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v]$. Then, for any $v^- \in V_{l-1}$:

$$\begin{aligned}
\mathbb{E}[\tilde{\boldsymbol{g}}_{v^-} \mid \boldsymbol{z}_{v^-}] &= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \tilde{\boldsymbol{h}}_{v^-}^{v} \;\middle|\; \boldsymbol{z}_{v^-}\right] \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] \;\middle|\; \boldsymbol{z}_{v^-}\right] && \text{(by (1))} \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] \;\middle|\; \boldsymbol{z}_{v^-}\right] && \text{(by induction)} \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{z}_{v^-}] \;\middle|\; \boldsymbol{z}_{v^-}\right] && \text{(by (B))} \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \mathbb{E}[\boldsymbol{J}_{v^-}^{v} \boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{z}_{v^-}] \;\middle|\; \boldsymbol{z}_{v^-}\right] && (\boldsymbol{J}_{v^-}^{v} \text{ is a function of } \boldsymbol{z}_v) \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \boldsymbol{g}_v \;\middle|\; \boldsymbol{z}_{v^-}\right] && \text{(double expectation)} \\
&= \mathbb{E}\left[\boldsymbol{g}_{v^-} \mid \boldsymbol{z}_{v^-}\right].
\end{aligned}$$

Therefore, by induction, $\mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v]$ holds for all non-input nodes v. □

To proceed, we first prove the following inequality that explicitly shows how uncertainty appears under Condition (A) and Assumption 1:

$$\begin{aligned}
&\mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}]\|_2^2 \\
&= \mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{\chi}] + \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{\chi}] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}]\|_2^2 \\
&= \mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{\chi}]\|_2^2 + \mathbb{E}\|\mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{\chi}] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}]\|_2^2 \\
&\geq \mathbb{E}\|\mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{\chi}] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}]\|_2^2 \\
&> 0. && \text{(by Assumption 1)}
\end{aligned}$$

Therefore, under Condition (A) and Assumption 1, there is some node $v \in V_l$ having

$$\mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}]\|^2 > 0, \tag{7}$$

where

$$\boldsymbol{Z}_{\mathrm{ch}(v)} := \{\boldsymbol{z}_{v^+} : v^+ \in \mathrm{ch}(v)\}.$$

With Lemma 2 and equation 7, we are ready to prove the theorem. Note that Lemma 2 implies

$$\begin{aligned}
\mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v] &= \mathbb{E}[\boldsymbol{J}_v^{\boldsymbol{\theta}} \tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] = \boldsymbol{J}_v^{\boldsymbol{\theta}} \mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] && (\boldsymbol{J}_v^{\boldsymbol{\theta}} \text{ is a function of } \boldsymbol{z}_v) \\
&= \boldsymbol{J}_v^{\boldsymbol{\theta}} \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] && \text{(by Lemma 2)} \\
&= \mathbb{E}[\boldsymbol{J}_v^{\boldsymbol{\theta}} \boldsymbol{g}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]. && (8)
\end{aligned}$$

We start from the conditional MSE. For any non-input node v:

$$\tilde{\rho}_v^2 = \mathbb{E}\left[\left\|\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\right\|_2^2\right] = \mathbb{E}\left[\left\|\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\right\|_2^2\right]. \qquad \text{(by (8))}$$

Recall synthetic gradient propagation with equation 1 and that all $\lambda_v = 0$, we have

$$\begin{aligned}
\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} &= \boldsymbol{J}_v^{\boldsymbol{\theta}} \tilde{\boldsymbol{g}}_v = \boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \tilde{\boldsymbol{h}}_v^{v^+} = \boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} \mathbb{E}[\tilde{\boldsymbol{g}}_{v^+} \mid \boldsymbol{z}_{v^+}] \\
&= \boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} \mathbb{E}\left[\boldsymbol{g}_{v^+} \mid \boldsymbol{z}_{v^+}\right] && \text{(Lemma 2)} \\
&= \boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} \mathbb{E}\left[\boldsymbol{g}_{v^+} \mid \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{z}_v\right] && \text{(by (B))} \\
&= \mathbb{E}\left[\sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{\boldsymbol{\theta}} \boldsymbol{J}_v^{v^+} \boldsymbol{g}_{v^+} \mid \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{z}_v\right] && (\boldsymbol{J}_v^{\boldsymbol{\theta}}, \boldsymbol{J}_v^{v^+} \text{ are functions of } \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{z}_v) \\
&= \mathbb{E}\left[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{Z}_{\mathrm{ch}(v)}, \boldsymbol{z}_v\right] && (9)
\end{aligned}$$

Because conditioning does not increase variance, we conclude that

$$\tilde{\rho}_v^2 \leq \rho_v^2.$$

By Lemma 2, $\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}}$ is unbiased, and then by Lemma 1:

$$\tilde{\Delta}_v^2 = \frac{1}{n}(\tilde{\rho}_v^2 + \nu_v^2) \leq \frac{1}{n}(\rho_v^2 + \nu_v^2) = \Delta_v^2. \qquad (10)$$

It remains to show that some node v achieves strict inequality. It follows from equation 10 that

$$\Delta_v^2 - \tilde{\Delta}_v^2 = \frac{1}{n}\left(\rho_v^2 - \tilde{\rho}_v^2\right).$$

By equation 7, there is some non-input node v such that

$$\mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}]\|_2^2 > 0,$$

and thus

$$\begin{aligned}
\rho_v^2 &= \mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\|^2 = \mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}] + \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\|_2^2 \\
&= \mathbb{E}\|\boldsymbol{g}_v^{\boldsymbol{\theta}} - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}]\|_2^2 + \mathbb{E}\|\mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\|_2^2 \\
&> \mathbb{E}\|\mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v, \boldsymbol{Z}_{\mathrm{ch}(v)}] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\|_2^2 && \text{(by (7))} \\
&= \mathbb{E}\|\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} - \mathbb{E}[\tilde{\boldsymbol{g}}_v^{\boldsymbol{\theta}} \mid \boldsymbol{z}_v]\|_2^2 && \text{(by (9) and Lemma 2)} \\
&= \tilde{\rho}_v^2.
\end{aligned}$$

Therefore,

$$\Delta_v^2 - \tilde{\Delta}_v^2 = \frac{1}{n}\left(\rho_v^2 - \tilde{\rho}_v^2\right) > 0.$$

□

E Proof of Lemma 3

We first quantify how the two approximation errors induce bias in synthetic gradient propagation.

Lemma 3. *Under (B'), synthetic gradient propagation with equation 2. Then, for any non-input nodes $v \in V_l$ and collection $\boldsymbol{Z}_{U_l} \subseteq \mathrm{an}_{\mathrm{sib}}(v)$ of extended ancestors, assuming bounded jacobians $\|\boldsymbol{J}_v^{v^+}\|_2 \leq J$, we have*

$$\|\mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}] - \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}]\|_2 \leq c_v(\epsilon + \epsilon'), \qquad a.s. \tag{11}$$

where $\forall v \in V_L : c_v = 0$ for terminal nodes, and a node $v \in V_l$ in layer $l < L$:

$$c_v = J \sum_{v^+ \in \mathrm{ch}(v)} (1 - \lambda_v + c_{v^+})$$

Proof. We begin by introducing a convenient notation.

Definition 7. *We use $\mathcal{O}(\epsilon)$ to denote a vector that satisfies $\|\mathcal{O}(\epsilon)\|_2 \leq \epsilon$ almost surely.*

With this notation, we can easily keep track of the scale of errors, e.g., $\mathcal{O}(\epsilon) + \mathcal{O}(\epsilon) = \mathcal{O}(2\epsilon)$; $\boldsymbol{J}\mathcal{O}(\epsilon) = \mathcal{O}(\|\boldsymbol{J}\|_2\epsilon)$.

Therefore, condition (B') can be expressed as

$$\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}] + \mathcal{O}(\epsilon).$$

Similarly, equation 2 can be expressed as

$$\tilde{\boldsymbol{h}}_{v^+} = \mathbb{E}[\tilde{\boldsymbol{g}}_{v^+} \mid \boldsymbol{z}_{v^+}] + \mathcal{O}(\epsilon').$$

Next, we prove the lemma by induction. First, for terminal nodes $\forall v \in V_L$, $\tilde{\boldsymbol{g}}_v = \boldsymbol{g}_v$ by definition, and thus $\mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{Z}_{U_L}] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{Z}_{U_L}]$ for any subset $U_L \subseteq \mathrm{an}_{\mathrm{sib}}(v)$.

Now, suppose equation 11 holds for a layer $l > 1$. Then, consider any $v^- \in V_{l-1}$ and a subset $U_{l-1} \subseteq \mathrm{an}_{\mathrm{sib}}(v^-)$. Let $U_l := \{v^-\} \cup U_{l-1}$. Note that for any $v \in \mathrm{ch}(v^-)$, we have $U_l \subseteq \mathrm{an}_{\mathrm{sib}}(v)$ by Definition 4. Therefore,

$$\begin{aligned}
\mathbb{E}[\tilde{\boldsymbol{g}}_{v^-} \mid \boldsymbol{z}_{v^-}, \boldsymbol{Z}_{U_{l-1}}] &= \mathbb{E}[\tilde{\boldsymbol{g}}_{v^-} \mid \boldsymbol{Z}_{U_l}] \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \lambda_{v^-} \boldsymbol{J}_{v^-}^{v} \tilde{\boldsymbol{g}}_v + (1 - \lambda_{v^-})(\boldsymbol{J}_{v^-}^{v}(\mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] + \mathcal{O}(\epsilon'))) \;\middle|\; \boldsymbol{Z}_{U_l}\right] \\
&= \lambda_{v^-} \underbrace{\mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \tilde{\boldsymbol{g}}_v \;\middle|\; \boldsymbol{Z}_{U_l}\right]}_{A} + (1 - \lambda_{v^-}) \underbrace{\mathbb{E}\left[\sum_{v \in \mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v} \mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] \;\middle|\; \boldsymbol{Z}_{U_l}\right]}_{B} \\
&\quad + (1 - \lambda_{v^-}) \cdot \mathcal{O}\left(|\mathrm{ch}(v^-)| \cdot J\epsilon'\right)
\end{aligned}$$

Applying double expectation, let $\epsilon'' := \epsilon + \epsilon'$, we have

$$
\begin{aligned}
A &= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \mathbb{E}\left[\boldsymbol{J}_{v^-}^{v}\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}\right] \,\middle|\, \boldsymbol{Z}_{U_l}\right] \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\mathbb{E}\left[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}\right] \,\middle|\, \boldsymbol{Z}_{U_l}\right] \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\left(\mathbb{E}\left[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}(c_v\epsilon'')\right) \,\middle|\, \boldsymbol{Z}_{U_l}\right] && \text{(by induction)} \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\mathbb{E}\left[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}\right] \,\middle|\, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right) \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \mathbb{E}\left[\boldsymbol{J}_{v^-}^{v}\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}\right] \,\middle|\, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right) \\
&= \mathbb{E}\left[\boldsymbol{g}_{v^-} \mid \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right). && \text{(double expectation)}
\end{aligned}
$$

Similarly, for the other term:

$$
\begin{aligned}
B &= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\mathbb{E}[\tilde{\boldsymbol{g}}_v \mid \boldsymbol{z}_v] \,\middle|\, \boldsymbol{Z}_{U_l}\right] \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\left(\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] + \mathcal{O}(c_v\epsilon'')\right) \,\middle|\, \boldsymbol{Z}_{U_l}\right] && \text{(by induction)} \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] \,\middle|\, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right) \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\left(\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}] + \mathcal{O}(\epsilon)\right) \,\middle|\, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right) && \text{(by (B'))} \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \boldsymbol{J}_{v^-}^{v}\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}] \,\middle|\, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon\cdot|\mathrm{ch}(v^-)| + J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right) \\
&= \mathbb{E}\left[\sum_{v\in\mathrm{ch}(v^-)} \mathbb{E}[\boldsymbol{J}_{v^-}^{v}\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{U_l}] \,\middle|\, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon\cdot|\mathrm{ch}(v^-)| + J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right) \\
&= \sum_{v\in\mathrm{ch}(v^-)} \mathbb{E}[\boldsymbol{J}_{v^-}^{v}\boldsymbol{g}_v \mid \boldsymbol{Z}_{U_l}] + \mathcal{O}\left(J\epsilon\cdot|\mathrm{ch}(v^-)| + J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right) && \text{(double expectation)} \\
&= \mathbb{E}\left[\boldsymbol{g}_{v^-} \,\middle|\, \boldsymbol{Z}_{U_l}\right] + \mathcal{O}\left(J\epsilon\cdot|\mathrm{ch}(v^-)| + J\epsilon''\sum_{v\in\mathrm{ch}(v^-)} c_v\right).
\end{aligned}
$$

Therefore, combining the above, we have

$$\begin{aligned}\mathbb{E}[\tilde{\boldsymbol{g}}_{v^-} \mid \boldsymbol{z}_{v^-}, \boldsymbol{Z}_{U_{l-1}}] &= \lambda_{v^-} A + (1-\lambda_{v^-})B + (1-\lambda_{v^-}) \cdot \mathcal{O}\left(|\mathrm{ch}(v^-)| \cdot J\epsilon'\right) \\ &= \mathbb{E}[\boldsymbol{g}_{v^-} \mid \boldsymbol{z}_{v^-}, \boldsymbol{Z}_{U_{l-1}}] + \mathcal{O}\left((1-\lambda_{v^-})J(\epsilon+\epsilon') \cdot |\mathrm{ch}(v^-)| + J\epsilon'' \sum_{v \in \mathrm{ch}(v^-)} c_v\right) \\ &= \mathbb{E}[\boldsymbol{g}_{v^-} \mid \boldsymbol{z}_{v^-}, \boldsymbol{Z}_{U_{l-1}}] + \mathcal{O}\left(c_{v^-}(\epsilon+\epsilon')\right),\end{aligned}$$

where

$$c_{v^-} = J \sum_{v \in \mathrm{ch}(v^-)} (1 - \lambda_{v^-} + c_v).$$

By induction, equation 11 holds for every node. □

F Proof of Theorem 3

Theorem 3 Under condition (B') and equation 2, assume bounded jacobians $\forall v, v^+ : \max\{\|\boldsymbol{J}_v^{v^+}\|_2, \|\boldsymbol{J}_v^{\boldsymbol{\theta}}\|_2\} \leq J$. A sufficient condition for $\lambda_v^\star < 1$ is:

$$\frac{\mathrm{tr}\,\mathrm{Cov}(\tilde{\boldsymbol{p}}_v) - \mathrm{tr}\,\mathrm{Cov}(\tilde{\boldsymbol{s}}_v)}{n} > \left(J^2(\epsilon+\epsilon') \sum_{v^+ \in \mathrm{ch}(v)} (1 + c_{v^+})\right)^2$$

where $\forall u \in V : c_u$ is defined recursively via

$$c_u = J \sum_{u^+ \in \mathrm{ch}(u)} (1 - \lambda_u + c_{u^+}), \quad \text{and for terminal nodes } \forall u \in V_L : c_u = 0.$$

Proof. Let $\epsilon'' := \epsilon + \epsilon'$ and we use Definition 7 for a bounded vector such as $\mathcal{O}(\epsilon'')$. We start by applying Lemma 3 to the following quantity.

$$\begin{aligned}\mathbb{E}[\tilde{\boldsymbol{s}}_v] &= \mathbb{E}\left[\boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} (\mathbb{E}[\tilde{\boldsymbol{g}}_{v^+} \mid \boldsymbol{z}_{v^+}] + \mathcal{O}(\epsilon'))\right] \\ &= \mathbb{E}\left[\boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} (\mathbb{E}[\boldsymbol{g}_{v^+} \mid \boldsymbol{z}_{v^+}] + \mathcal{O}(\epsilon' + c_{v^+}\epsilon''))\right] && \text{(by Lemma 3)} \\ &= \mathbb{E}\left[\boldsymbol{J}_v^{\boldsymbol{\theta}} \sum_{v^+ \in \mathrm{ch}(v)} \boldsymbol{J}_v^{v^+} (\mathbb{E}[\boldsymbol{g}_{v^+} \mid \boldsymbol{z}_{v^+}, \boldsymbol{z}_v] + \mathcal{O}(\epsilon' + \epsilon + c_{v^+}\epsilon''))\right] && \text{(by (B'))} \\ &= \mathbb{E}\left[\sum_{v^+ \in \mathrm{ch}(v)} \mathbb{E}[\boldsymbol{J}_v^{\boldsymbol{\theta}} \boldsymbol{J}_v^{v^+} \boldsymbol{g}_{v^+} \mid \boldsymbol{z}_{v^+}, \boldsymbol{z}_v]\right] + \mathcal{O}\left(J^2 \sum_{v^+ \in \mathrm{ch}(v)} ((c_{v^+}+1)\epsilon'')\right) \\ &= \mathbb{E}\left[\boldsymbol{g}_v^{\boldsymbol{\theta}}\right] + \mathcal{O}\left(J^2\epsilon'' \sum_{v^+ \in \mathrm{ch}(v)} (1 + c_{v^+})\right).\end{aligned}$$

It is standard to show from equation 3 that $\lambda^\star < 1$ if

$$\mathbb{E}\left[\left\|\hat{\tilde{\boldsymbol{s}}}_v - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2\right] < \mathbb{E}\left[\left\|\hat{\tilde{\boldsymbol{p}}}_v - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2\right]. \tag{12}$$

The LHS can be decomposed as follows.

$$\begin{aligned}
\mathbb{E}\left[\left\|\hat{\tilde{\boldsymbol{s}}}_v - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2\right] &= \mathbb{E}\left[\left\|\hat{\tilde{\boldsymbol{s}}}_v - \mathbb{E}[\tilde{\boldsymbol{s}}_v] + \mathbb{E}[\tilde{\boldsymbol{s}}_v] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2\right] \\
&= \frac{1}{n}\mathbb{E}\left[\|\tilde{\boldsymbol{s}}_v - \mathbb{E}[\tilde{\boldsymbol{s}}_v]\|_2^2\right] + \left\|\mathbb{E}[\tilde{\boldsymbol{s}}_v] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2 \\
&\leq \frac{1}{n}\mathbb{E}\left[\|\tilde{\boldsymbol{s}}_v - \mathbb{E}[\tilde{\boldsymbol{s}}_v]\|_2^2\right] + \left(J^2\epsilon'' \sum_{v^+ \in \mathrm{ch}(v)} (1 + c_{v^+})\right)^2 \\
&= \frac{1}{n}\operatorname{tr}\operatorname{Cov}(\tilde{\boldsymbol{s}}_v) + \left(J^2\epsilon'' \sum_{v^+ \in \mathrm{ch}(v)} (1 + c_{v^+})\right)^2.
\end{aligned}$$

Similarly, the RHS shows

$$\begin{aligned}
\mathbb{E}\left[\left\|\hat{\tilde{\boldsymbol{p}}}_v - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2\right] &= \mathbb{E}\left[\left\|\hat{\tilde{\boldsymbol{p}}}_v - \mathbb{E}[\tilde{\boldsymbol{p}}_v] + \mathbb{E}[\tilde{\boldsymbol{p}}_v] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2\right] \\
&= \frac{1}{n}\mathbb{E}\left[\|\tilde{\boldsymbol{p}}_v - \mathbb{E}[\tilde{\boldsymbol{p}}_v]\|_2^2\right] + \left\|\mathbb{E}[\tilde{\boldsymbol{p}}_v] - \mathbb{E}[\boldsymbol{g}_v^{\boldsymbol{\theta}}]\right\|_2^2 \\
&\geq \frac{1}{n}\mathbb{E}\left[\|\tilde{\boldsymbol{p}}_v - \mathbb{E}[\tilde{\boldsymbol{p}}_v]\|_2^2\right] \\
&= \frac{1}{n}\operatorname{tr}\operatorname{Cov}(\tilde{\boldsymbol{p}}_v).
\end{aligned}$$

Therefore, a sufficient condition for equation 12 is

$$\frac{1}{n}\operatorname{tr}\operatorname{Cov}(\tilde{\boldsymbol{s}}_v) + \left(J^2\epsilon'' \sum_{v^+ \in \mathrm{ch}(v)} (1 + c_{v^+})\right)^2 < \frac{1}{n}\operatorname{tr}\operatorname{Cov}(\tilde{\boldsymbol{p}}_v).$$

□

G Detailed Setup of the Example of Finite-Partition Expert Network

We consider a two-hidden-layer architecture in which the first hidden layer has a single node, and the second hidden layer has nodes $V_2 = \{(2, j)\}_{j \in [m_2]}$. In the rest of the section, we denote $j := (2, j)$. We focus on gradient estimation on the node $(1, 1) \in V_1$ in the first layer, and thus in this section *we omit the subscript when referring to this target node*. For example, $\boldsymbol{g}^{\boldsymbol{\theta}}$ refers to the backpropagated gradient on this target node $(1, 1) \in V_1$.

The first layer node computes a d-dim feature representation $\boldsymbol{x} = f(\boldsymbol{z}; \boldsymbol{\theta}) \in \mathbb{R}^d$, where $\boldsymbol{z}$ is the input sampled from an external source (i.e., from layer $l = 0$). The second layer contains m_2 expert nodes, where each expert $j \in [m_2]$ specializes in a k-dimensional subspace of the feature vector $\boldsymbol{x}$. Formally, each expert j effectively acts on $\boldsymbol{u}_j = P_j\boldsymbol{x}$, where $P_j : \mathbb{R}^d \to \mathbb{R}^k$ is a projection. Each expert outputs $y_j = f_j(\boldsymbol{u}_j) \in \mathbb{R}$. Then, the final node collects the vector of expert outputs $\boldsymbol{y} = (y_1, \ldots, y_{m_2}) \in \mathbb{R}^{m_2}$.

Each expert node j solves an independent sub-task, such that the global loss function decomposes across the experts. The global loss function and its gradient to expert node j are respectively

$$\mathcal{L} := \sum_{j=1}^{m_2} \ell_j(\boldsymbol{u}_j, y_j), \qquad \text{and } \boldsymbol{g}_j := \frac{\partial \mathcal{L}}{\partial y_j} = \frac{\partial \ell_j(\boldsymbol{u}_j, y_j)}{\partial y_j}.$$

We compare ordinary backpropagation and synthetic-gradient propagation under three sources of stochastic expert feedback, each inspired by a sub-condition in condition (A).

- **Case 1: noisy terminal feedback.** The external feedback carries i.i.d. noise: $\hat{\boldsymbol{g}}_j = \boldsymbol{g}_j + \xi_j, \qquad \mathbb{E}[\xi_j] = 0, \qquad \mathbb{E}[\xi_j^2] = \sigma^2$.

- **Case 2: stochastic expert activation.** Each expert is activated with probability p independently every time. The gradient feedback is thus $\hat{\boldsymbol{g}}_j = \frac{\eta_j}{p}\boldsymbol{g}_j, \qquad \eta_j \sim \text{Bernoulli}(p)$.
- **Case 3: partial observation.** In this case, the terminal node sees some other variables $\boldsymbol{z}'_j$, independent across j, such that the global loss term $\mathcal{L}' = \sum_{j=1}^{m_2} \ell_j(\boldsymbol{u}_j, y_j, \boldsymbol{z}'_j)$, and its gradient to expert j becomes $\hat{\boldsymbol{g}}_j = \frac{\partial \mathcal{L}'}{\partial y_j} = \frac{\partial \ell_j(\boldsymbol{u}_j, y_j, \boldsymbol{z}'_j)}{\partial y_j}$.

The synthetic gradient from expert j to the target node $(1,1)$ needs to be estimated: $\hat{\tilde{\boldsymbol{h}}}^j(\boldsymbol{u}_j) \approx \boldsymbol{J}^j \mathbb{E}[\boldsymbol{g}_j \mid \boldsymbol{z}_j]$. To characterize sample efficiency when the synthetic gradient estimator is itself estimated from data, we restrict attention to discrete inputs. This yields a tabular setting, analogous to those commonly used in sample-complexity analyses of TD and MC estimators [9, 11]. We analyze this setting next.

Let $\boldsymbol{z} \in \mathcal{X}$ denote the input variable to our target node $(1,1) \in V_1$. Let the input space $\mathcal{X}$ be partitioned into $R = 2^d$ sets $\mathcal{X} = \bigsqcup_{r=1}^{R} \mathcal{X}_r$. We write $\boldsymbol{z}_r$ for the representative input in partition r, and suppose the input variable $\boldsymbol{z}$ is sampled from a uniform distribution over these representative inputs. The target node's function is piecewise parameterized:

$$f(\boldsymbol{z}; \boldsymbol{\theta}) = \sum_{r=1}^{R} \mathbf{1}\{\boldsymbol{z} \in \mathcal{X}_r\} f_{\boldsymbol{\theta}_r}(\boldsymbol{z}), \quad \text{where } \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R).$$

Let $\boldsymbol{x}_r = f_{\boldsymbol{\theta}_r}(\boldsymbol{z}_r) \in \{\pm 1\}^d$. We assume that the map $r \mapsto \boldsymbol{x}_r$ is one-to-one and onto $\{\pm 1\}^d$. Therefore, drawing r uniformly from $[R]$ is equivalent to drawing $\boldsymbol{x}_r$ uniformly from $\{\pm 1\}^d$. Let each expert projection P_j selects exactly k coordinates of $\boldsymbol{x}_r$. Thus $\boldsymbol{u}_{j,r} := P_j \boldsymbol{x}_r \in \{\pm 1\}^k$. For any $\boldsymbol{u} \in \{\pm 1\}^k$, let the equivalent class under this projection be

$$\mathcal{F}_j(r) := \{s \in [R] : P_j \boldsymbol{x}_s = P_j \boldsymbol{x}_r\}, \qquad \text{and thus} \quad \forall r \in [R] : \ |\mathcal{F}_j(r)| = 2^{d-k}.$$

For each expert j and partition r, its has a unique output $y_{j,r} = f_j(\boldsymbol{u}_j)$. let

$$\boldsymbol{g}_{j,r} := \frac{\partial \ell_j(\boldsymbol{u}_{j,r}, y_{j,r})}{\partial y_j}$$

denote the exact scalar gradient received by expert j. Here, we have fixed all the random variables and thus $\boldsymbol{g}_{j,r}$ becomes a constant.

Since the parameter $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R)$ is also partitioned, by definition of the piecewise function (equation 4), when $\boldsymbol{z}_r$ is given, only the parameter $\boldsymbol{\theta}^r$ receives gradient feedback:

$$\boldsymbol{g}^{\boldsymbol{\theta}_r} = \sum_{j=1}^{m_2} \boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j \boldsymbol{g}_{j,r}.$$

Similarly, here $\boldsymbol{g}^{\boldsymbol{\theta}_r}$ becomes a constant vector.

Because the parameter blocks $\boldsymbol{\theta}_r$ are specific to the input $\boldsymbol{z}_r$, and each $\boldsymbol{z}_r$ is sampled with equal probability, the population mean gradient can be written as

$$\mathbb{E}\, \boldsymbol{g}^{\boldsymbol{\theta}} = \frac{1}{R}(\boldsymbol{g}^{\boldsymbol{\theta}_1}, \ldots, \boldsymbol{g}^{\boldsymbol{\theta}_R}).$$

Then, for any gradient estimator $\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}} = \frac{1}{R}(\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_1}, \ldots, \hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_R})$, the estimation MSE decomposes into the sum of per-partition MSE:

$$\tilde{\Delta}^2 = \mathbb{E} \left\| \hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}} - \mathbb{E}\, \boldsymbol{g}^{\boldsymbol{\theta}} \right\|_2^2 = \frac{1}{R^2} \sum_{r=1}^{R} \mathbb{E} \left\| \hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_r} - \boldsymbol{g}^{\boldsymbol{\theta}_r} \right\|_2^2 = \frac{1}{R^2} \sum_{r=1}^{R} \tilde{\Delta}_r^2, \tag{13}$$

where we denote the per-partition MSE by

$$\tilde{\Delta}_r^2 := \mathbb{E} \left\| \hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_r} - \boldsymbol{g}^{\boldsymbol{\theta}_r} \right\|_2^2.$$

For every partition $r \in [R]$, we draw n i.i.d. samples. This is analogue to a phased sampling model used in TD learning analyses (e.g., see [30]) to isolate the statistical mechanism estimators in a

single frozen-parameter phase. Thus, the samples can be labeled by $(r, i) \in [R] \times [n]$. Define the backpropagation estimator to a parameter block $\boldsymbol{\theta}^r$ as

$$\hat{\boldsymbol{g}}^{\boldsymbol{\theta}_r} := \sum_{j=1}^{m_2} \boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j \left(\frac{1}{n} \sum_{i=1}^{n} \hat{\boldsymbol{g}}_{j,r,i} \right).$$

Note that expert j's local observation is $\boldsymbol{z}_j = P_j \boldsymbol{x} = \boldsymbol{u}_j$. As the synthetic-gradient generator approximates $\boldsymbol{J}^j \mathbb{E}[\hat{\boldsymbol{g}}_j \mid \boldsymbol{u}_j]$, let the conditional empirical mean as the synthetic gradient generator:

$$\hat{\tilde{\boldsymbol{h}}}^j(\boldsymbol{u}_j) := \boldsymbol{J}^j \left(\frac{1}{n \sum_r \mathbf{1}(P_j \boldsymbol{x}_r = \boldsymbol{u}_j)} \sum_{r,i} \mathbf{1}(P_j \boldsymbol{x}_r = \boldsymbol{u}_j)\, \hat{\boldsymbol{g}}_{j,r,i} \right) = \boldsymbol{J}^j \frac{1}{n|\mathcal{F}_j(r)|} \sum_{s \in \mathcal{F}_j(r)} \sum_{i=1}^{n} \hat{\boldsymbol{g}}_{j,s,i}.$$

Thus, the synthetic-gradient estimator to a parameter block $\boldsymbol{\theta}^r$ is

$$\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_r} := \sum_{j=1}^{m_2} \boldsymbol{J}_r^{\boldsymbol{\theta}^r} \hat{\tilde{\boldsymbol{h}}}_j(\boldsymbol{u}_{j,r}).$$

Theorem 5. *Consider the settings described in this section. Assume that the norm of the gradient feedback to each expert node has a constant norm across experts, i.e., $\forall j, j' : |\boldsymbol{g}_{j,r}| = |\boldsymbol{g}_{j',r}|$, and for Case 3: $\forall j, j' : \mathrm{Var}(\hat{\boldsymbol{g}}_{j,r} \mid r) = \mathrm{Var}(\hat{\boldsymbol{g}}_{j',r} \mid r)$ has a constant variance across experts. Then, for all three cases, we have*

$$\Delta^2 = 2^{d-k}\, \tilde{\Delta}^2.$$

H Proof of Theorem 5

We first prove the following lemma.

Lemma 4. *Fix a state r. Consider the settings and objects described in Theorem 5. Suppose*

$$\hat{\boldsymbol{g}}_{j,s,i} = \boldsymbol{g}_{j,s} + \varepsilon_{j,s,i},$$

where the variables $\varepsilon_{j,s,i}$ are independent across j, s, i, have mean zero, and have variance

$$v_{j,s} := \mathbb{E}[\varepsilon_{j,s,i}^2].$$

We have

$$\Delta_r^2 = \frac{1}{n} \sum_{j=1}^{m_2} \|\boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j\|^2 v_{j,r}, \qquad \text{and} \quad \tilde{\Delta}_r^2 = \frac{1}{n 2^{d-k}} \sum_{j=1}^{m_2} \|\boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j\|^2 v_{j,r}.$$

In particular, when $v_{j,r}$ is a constant across expert j, we have

$$\Delta_r^2 = 2^{d-k} \tilde{\Delta}_r^2.$$

Proof. For ordinary backpropagation, define

$$\overline{\varepsilon}_{j,r} := \frac{1}{n} \sum_{i=1}^{n} \varepsilon_{j,r,i}.$$

Then

$$\frac{1}{n} \sum_{i=1}^{n} \hat{\boldsymbol{g}}_{j,r,i} = \boldsymbol{g}_{j,r} + \overline{\varepsilon}_{j,r},$$

and therefore

$$\hat{\boldsymbol{g}}^{\boldsymbol{\theta}_r} - \boldsymbol{g}^{\boldsymbol{\theta}_r} = \sum_{j=1}^{m_2} \boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j \overline{\varepsilon}_{j,r}.$$

Since $\mathbb{E}[\overline{\varepsilon}_{j,r}] = 0$, the estimator is unbiased. Also,

$$\mathbb{E}[\overline{\varepsilon}_{j,r}^2] = \frac{v_{j,r}}{n}.$$

The variables $\overline{\varepsilon}_{j,r}$ are independent across j. Hence the cross terms vanish, and

$$\begin{aligned}\Delta_r^2 = \mathbb{E}\left[\left\|\hat{\boldsymbol{g}}^{\boldsymbol{\theta}_r} - \boldsymbol{g}^{\boldsymbol{\theta}_r}\right\|^2\right] &= \mathbb{E}\left[\left\|\sum_{j=1}^{m_2} \boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j \overline{\varepsilon}_{j,r}\right\|^2\right] \\ &= \sum_{j=1}^{m_2} \|\boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j\|^2 \mathbb{E}[\overline{\varepsilon}_{j,r}^2] = \frac{1}{n}\sum_{j=1}^{m_2} \|\boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j\|^2 v_{j,r}.\end{aligned}$$

We next analyze the synthetic-gradient estimator. Since $\boldsymbol{g}_{j,s} = \boldsymbol{g}_{j,r}$ for every $s \in \mathcal{F}_j(r)$,

$$\begin{aligned}\hat{\tilde{\boldsymbol{h}}}_j(\boldsymbol{x}_r) &= \frac{1}{n|\mathcal{F}_j(r)|}\sum_{s\in\mathcal{F}_j(r)}\sum_{i=1}^{n}(\boldsymbol{g}_{j,s} + \varepsilon_{j,s,i}) \\ &= \boldsymbol{g}_{j,r} + \frac{1}{n|\mathcal{F}_j(r)|}\sum_{s\in\mathcal{F}_j(r)}\sum_{i=1}^{n}\varepsilon_{j,s,i}.\end{aligned}$$

Define

$$\widetilde{\varepsilon}_{j,r} := \frac{1}{n|\mathcal{F}_j(r)|}\sum_{s\in\mathcal{F}_j(r)}\sum_{i=1}^{n}\varepsilon_{j,s,i}.$$

Then

$$\hat{\tilde{\boldsymbol{h}}}_j(\boldsymbol{x}_r) = \boldsymbol{g}_{j,r} + \widetilde{\varepsilon}_{j,r}.$$

Since the feedback errors are mean zero,

$$\mathbb{E}[\tilde{\boldsymbol{h}}_j(\boldsymbol{x}_r)] = \boldsymbol{g}_{j,r},$$

and hence

$$\mathbb{E}[\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_r}] = \boldsymbol{g}^{\boldsymbol{\theta}_r}.$$

Because P_j selects exactly k coordinates and $\boldsymbol{x}_r$ ranges uniformly over $\{\pm 1\}^d$,

$$|\mathcal{F}_j(r)| = 2^{d-k}.$$

Using $v_{j,s} = v_{j,r}$ for $s \in \mathcal{F}_j(r)$ and independence,

$$\mathbb{E}[\widetilde{\varepsilon}_{j,r}^2] = \frac{v_{j,r}}{n|\mathcal{F}_j(r)|} = \frac{v_{j,r}}{n2^{d-k}}.$$

Finally,

$$\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_r} - \boldsymbol{g}^{\boldsymbol{\theta}_r} = \sum_{j=1}^{m_2} \boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j \widetilde{\varepsilon}_{j,r}.$$

The feedback-error variables are independent across experts j, so the cross terms again vanish. Hence

$$\begin{aligned}\tilde{\Delta}_r^2 = \mathbb{E}\left[\left\|\hat{\tilde{\boldsymbol{g}}}^{\boldsymbol{\theta}_r} - \boldsymbol{g}^{\boldsymbol{\theta}_r}\right\|^2\right] &= \sum_{j=1}^{m_2} \|\boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j\|^2 \mathbb{E}[\widetilde{\varepsilon}_{j,r}^2] \\ &= \frac{1}{n2^{d-k}}\sum_{j=1}^{m_2} \|\boldsymbol{J}_r^{\boldsymbol{\theta}^r} \boldsymbol{J}_r^j\|^2 v_{j,r}.\end{aligned}$$

Therefore, when $v_{j,r}$ is a constant across expert j, we have

$$\Delta_r^2 = 2^{d-k}\tilde{\Delta}_r^2.$$

□

Theorem 5. Consider the settings described in this section. Assume that the norm of the gradient feedback to each expert node has a constant norm across experts, i.e., $\forall j, j' : |\boldsymbol{g}_{j,r}| = |\boldsymbol{g}_{j',r}|$, and for Case 3: $\forall j, j' : \mathrm{Var}(\hat{\boldsymbol{g}}_{j,r} \mid r) = \mathrm{Var}(\hat{\boldsymbol{g}}_{j',r} \mid r)$ has a constant variance across experts. Then, for all three cases, we have

$$\Delta^2 = 2^{d-k}\,\tilde{\Delta}^2.$$

Proof. The proof for all three cases are applications of Lemma 4.

For Case 1, set $\varepsilon_{j,s,i} = \xi_{j,s,i}$. Then, $\varepsilon_{j,s,i}$ is i.i.d. across j, s, i with zero mean and constant variance $v_s = \sigma^2$. Invoking Lemma 4 gives the result for Case 1.

For Case 2, set

$$\varepsilon_{j,s,i} = \left(\frac{\eta_{j,s,i}}{p} - 1\right) \boldsymbol{g}_{j,s}.$$

Since $\mathbb{E}[\eta_{j,s,i}/p] = 1$, the error has mean zero. Also,

$$\mathbb{E}\left[\left(\frac{\eta_{j,s,i}}{p} - 1\right)^2\right] = \frac{1-p}{p},$$

and hence

$$v_{j,s} = \frac{1-p}{p}\boldsymbol{g}_{j,s}^2.$$

As each $\boldsymbol{g}_{j,s}^2$ is constant across expert j, we know $v_{j,s}$ is constant across experts. Then, invoking Lemma 4 proves the result for Case 2.

It remains to prove Case 3. Recall

$$\hat{\boldsymbol{g}}_j = \frac{\partial \mathcal{L}'}{\partial y_j} = \frac{\partial \ell_j(\boldsymbol{u}_j, y_j, \boldsymbol{z}'_j)}{\partial y_j}.$$

The mean gradient for parameter block $\boldsymbol{\theta}^s$ is

$$\mathbb{E}\,\boldsymbol{g}_s^{\boldsymbol{\theta}_s} = \mathbb{E}\left[\boldsymbol{J}_s^{\boldsymbol{\theta}_s} \sum_{j\in[m_2]} \boldsymbol{J}_s^j \hat{\boldsymbol{g}}_j \mid \boldsymbol{z}_s\right] = \boldsymbol{J}_s^{\boldsymbol{\theta}_s} \sum_{j\in[m_2]} \boldsymbol{J}_s^j\, \mathbb{E}\left[\hat{\boldsymbol{g}}_j \mid \boldsymbol{z}_s\right].$$

Define $\bar{\boldsymbol{g}}_{j,s} := E\left[\hat{\boldsymbol{g}}_j \mid \boldsymbol{z}_s\right]$ and set

$$\epsilon_{j,s,i} = \hat{\boldsymbol{g}}_{j,s,i} - \bar{\boldsymbol{g}}_{j,s}.$$

By definition, $\epsilon_{j,s,i}$ has zero mean, independent across j, s, i. Moreover, by assumption in the theorem statement, its variance $v_{j,s} = \mathbb{E}[\epsilon_{j,s,i}^2]$ is constant across experts. Therefore, invoking Lemma 4 proves:

$$\Delta_r^2 = 2^{d-k}\,\tilde{\Delta}_r^2, \quad \forall r \in [R].$$

Then, applying equation 13 proves the theorem.

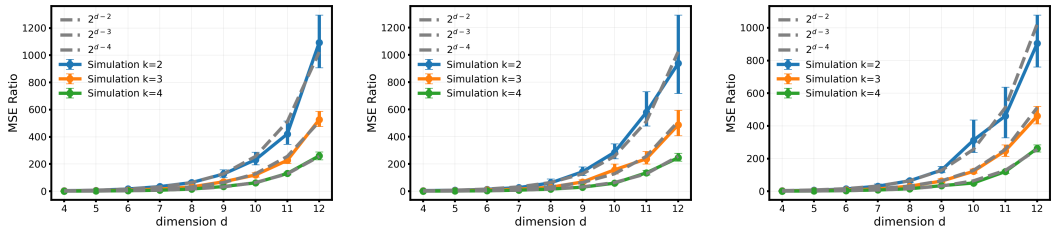
□

I Experiment Details

This section provides the implementation details and full results for the experiments in Section 6.

I.1 Simulation of the Expert Network

We first simulate the expert network from Section 5. For each dimension d, the input space is represented by $R = 2^d$ partition representatives. The target node output is a bijection onto Boolean states $\boldsymbol{x}_r \in \{-1, +1\}^d$, and the target node is piecewise parameterized as $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R)$, so only block $\boldsymbol{\theta}_r$ receives gradient on state r. In the simulation, the local map for each block is



(a) Noisy terminal feedback. (b) Stochastic expert activation. (c) Partial observation.

Figure 4: **Finite-partition expert simulation.** MSE ratio $\Delta^2/\tilde{\Delta}^2$ between the squared error of the backpropagation estimator and the squared error of the synthetic-gradient estimator for the full population gradient. In all three cases, the ratio grows with dimension reduction $d - k$, matching the predicted advantage from theory.

linear, $f_{\boldsymbol{\theta}_r}(x_r) = \boldsymbol{A}_r\boldsymbol{\theta}_r$, with $\boldsymbol{A}_r \sim \mathcal{N}(0, \frac{1}{d}\boldsymbol{I})$; the Boolean state $\boldsymbol{x}_r$ is used as the expert-visible representation, while $\boldsymbol{A}_r$ supplies the local Jacobian for pulling gradients back to $\boldsymbol{\theta}_r$.

The second layer contains m expert nodes. Expert j observes a k-coordinate projection $\boldsymbol{u}_{j,r} = P_j\boldsymbol{x}_r \in \{-1, +1\}^k$, computes $y_{j,r} = \boldsymbol{a}_j^\top \boldsymbol{u}_{j,r} + b_j$, and is trained against a linear target $t_j(\boldsymbol{u}_{j,r}) = \boldsymbol{\beta}_j^\top \boldsymbol{u}_{j,r} + \gamma_j$ with squared loss. The clean scalar expert feedback is therefore $g_{j,r} = y_{j,r} - t_j(\boldsymbol{u}_{j,r})$, which depends only on $P_j\boldsymbol{x}_r$ and is constant on the preimage $\mathcal{F}(r) := \{s : P_j\boldsymbol{x}_s = P_j\boldsymbol{x}_r\}$ of size 2^{d-k}.

We compare ordinary backpropagation, which averages stochastic feedback separately at each state, with synthetic-gradient propagation, which estimates the conditional expert feedback table by pooling all states sharing the same projected expert input. The full population gradient is computed by exact enumeration over all R representatives; estimator error is the squared Euclidean distance to this exact gradient.

We evaluate the three stochasticity sources from Section 5: noisy terminal feedback, stochastic expert activation, and partial observation. For noisy terminal feedback, each expert receives its clean feedback corrupted by independent additive Gaussian noise, $\widehat{g}_{j,r} = g_{j,r} + \xi_j$, where $\xi_j \sim \mathcal{N}(0, \sigma^2)$. For stochastic expert activation, expert j is active with probability p, and its feedback is reweighted as $\widehat{g}_{j,r} = (\eta_j/p)g_{j,r}$ with $\eta_j \sim \text{Bernoulli}(p)$. For partial observation, the expert target depends on an unobserved zero-mean Gaussian variable $q_j \sim \mathcal{N}(0, \tau^2)$, producing feedback $\widehat{g}_{j,r} = g_{j,r} + \alpha_j q_j$, where each α_j is a scale constant sampled once from the standard Gaussian for each expert. Each construction is unbiased, so $\mathbb{E}[\widehat{g}_{j,r} \mid \mathbf{z}_r] = g_{j,r}$.

The sweep uses $d \in \{4, \ldots, 12\}$, $k \in \{2, 3, 4\}$, $m = 5$ experts and 20 Monte Carlo trials. Every possible input variable is sampled once. The reported metric is the ratio $\Delta^2/\tilde{\Delta}^2$ between the squared error of the backpropagation estimator and the squared error of the synthetic-gradient estimator for the full population gradient.

I.2 SparseNet and Adaptive Gradient Mixing

The MNIST bandit and POPGym experiments use the same SparseNet neuron template. A SparseNet hidden state is a collection of W neurons,

$$H = \{h_i\}_{i=1}^W, \qquad h_i \in \mathbb{R}^q,$$

where q is called the neuron dimension. Each neuron connects to n_{parents} other neurons, where the connections are uniformly randomly picked and fixed at initialization. Each neuron observes the concatenated outputs $\boldsymbol{u} \in \mathbb{R}^d$ from its parent neurons and optionally from external inputs.

Each neuron i contains $M = n_{\text{slots}}$ slot maps. For slot s, the policy parameters are a key $k_{i,s}$, a matrix $A_{i,s}$, and a bias $b_{i,s}$. The neuron first routes its local input over slots,

$$\alpha_{i,s} = \frac{\exp\left(\langle \boldsymbol{u}_i, k_{i,s}\rangle/\sqrt{d}\right)}{\sum_{r=1}^M \exp\left(\langle \boldsymbol{u}_i, k_{i,r}\rangle/\sqrt{d}\right)},$$

then computes slot outputs

$$y_{i,s} = \tanh(A_{i,s}\boldsymbol{u}_i + b_{i,s}),$$

and returns the mixture

$$\bar{h}_i = \sum_{s=1}^{M} \alpha_{i,s} y_{i,s}.$$

For MNIST, hidden layers after the first use the residual update implemented in the sweep:

$$h_{\ell,i} = \tanh(\bar{h}_{\ell,i}) + h_{\ell-1,i}, \qquad \ell > 0,$$

while POPGym uses the sparse transition recurrently through time. The MNIST action layer flattens the final hidden state and forms scaled dot-product logits. The POPGym actor uses an analogous slot mechanism at the action layer: for each action, actor keys route over action slots and the slot logits are mixed to produce the final action logit.

This neuron design makes the synthetic gradient predictor convenient in a sparse network: it uses the same local input $\boldsymbol{u}_i$ and the same routing weights $\alpha_{i,s}$, but has predictor-specific slot parameters $\tilde{A}_{i,s}, \tilde{b}_{i,s}$:

$$\hat{\tilde{\boldsymbol{h}}}_i(\boldsymbol{u}_i) = \sum_{s=1}^{M} \alpha_{i,s} \left(\tilde{A}_{i,s}\boldsymbol{u}_i + \tilde{b}_{i,s} \right).$$

Thus, the predictor estimates an incoming hidden-state gradient in $\mathbb{R}^q$ using only the information available at that neuron.

The adaptive λ rule estimates the per-neuron bias-variance tradeoff from the current mini-batch. Let $\tilde{\boldsymbol{p}}_b$ be the received gradient on this neuron's parameter on sample b. Let $\tilde{\boldsymbol{s}}_b$ be the gradient on this neuron's parameter induced by its synthetic gradient prediction. We use the batch empirical mean of $\tilde{\boldsymbol{p}}_b$ to approximate the true expected gradient, and use equation 3 to calculate a mixing weight λ_{batch} for this batch. In practice, we found it to be more stable to normalize the gradients to unit ℓ_2 norm before estimating the mixing weight. Then, the mixing weight is updated with momentum,

$$\lambda \leftarrow \mu\lambda + (1-\mu)\lambda_{\text{batch}}.$$

I.3 MNIST Contextual Bandit

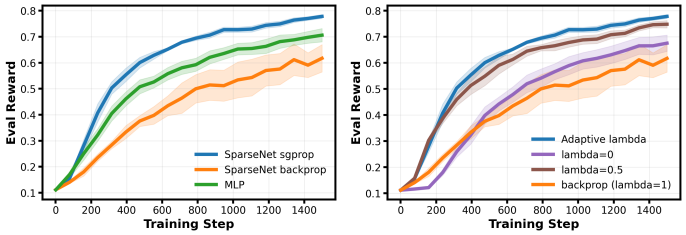
The experiment is a contextual bandit simulation. The context is an MNIST image, and the action set consists of the class predictions $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The policy samples an action from its softmax distribution. The true reward is one for a correct sampled label and zero otherwise; during training this reward is flipped with probability p_{flip}, while evaluation uses the noiseless reward.

The SparseNet policy is multi-layer sparse feedforward network (SparseNet neuron operation and adaptive gradient-mixing rule detailed in Section I.2). The first layer samples $d_{\text{in}} = qn_{\text{parents}}$ random raw pixel coordinates per neuron, where q is the neuron dimension and n_{parents} is the number of parents. In later layers, each neuron receives input from the concatenation of the states of its n_{parents} parents. Sparse connections are uniformly randomly picked, fixed at initialization. In this experiment, it has five sparse layers of width 200, neuron output dimension 10, neuron slots number 5, and number of parents 5 in the base setting. The final action layer flattens the last sparse hidden state and uses learned action keys to produce logits. We compare SparseNet trained by synthetic gradient propagation (sgprop), backpropagation (backprop), and a dense MLP backpropagation baseline. The MLP uses five layers, width 745, matching the SparseNet parameter count. For every batch, once the gradient is estimated, we use Adam to optimize every network. We plot the evaluation rewards along training steps in Figure 5.

We observe that each method has a different optimal learning rate. For example, as synthetic gradient reduces variance, it may benefit from a bigger learning rate. we use their corresponding optimal learning rate according to Figure 6 (a). More ablations (reward noise scale, number of parents, adaptive mixing weight dynamics) are shown in Figure 6; detailed hyperparameters are detailed in Table 1.

Table 1: **MNIST contextual bandit settings.** Learning rate are chosen to be their optimal ones respectively according to Figure 6 (a). MLP layers and width are chosen to match the parameter count of SparseNet.

Component	Setting	Value
Task	Base reward flip probability	$p_{\text{flip}} = 0.4$
Training	Steps / seeds	1500 / $\{0, 1, 2, 3, 4, 5\}$
Training	Batch size / eval batch size	64 / 4000
SparseNet	synthetic grad learning rate	0.005
SparseNet	Layers / width / neuron dim	5 / 200 / 10
SparseNet	Num. of parents	5
SparseNet SGProp	Base learning rate	0.005
SparseNet backprop	Base learning rate	0.001
MLP backprop	Layers / width / learning rate	5 / 745 / $2 \cdot 10^{-5}$



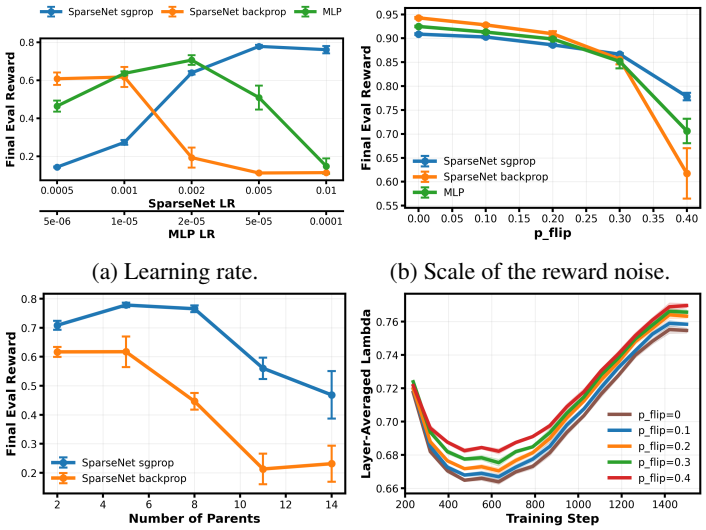
(a) Training dynamics (b) Compare adaptive and fixed-λ

Figure 5: **MNIST contextual bandit.** Mean evaluation rewards along training steps. Each curve aggregates six seeds, with the standard error of the mean (SEM) reported. For SparseNet, synthetic gradient improves sample efficiency relative to backpropagation, out performing the dense MLP baseline. The fixed-λ comparison shows that the benefit depends on balancing synthetic and back-propagated gradients.

I.4 POPGym Reinforcement Learning

This experiment tests synthetic gradients in a recurrent SparseNet for partially observable RL. We use two maze navigation environments from POPGym: `labyrinth_escape` (the agent escapes the maze) and `labyrinth_explore` (the agent explores the entire maze). If the agent successfully escapes the maze in `labyrinth_escape` or successfully visits all maze squares in `labyrinth_explore`, a terminal reward $r = 1$ is given. For both environments, there is a small penalty $r = -0.01$ for each step taken. In the exploration task, a reward $r = 1/(\text{total maze squre count})$ is given with probability 0.75 for each unseen square the agent visits. For both environments we use maze size $(6, 6)$. Both POPGym labyrinth environments are partially observable. The raw maze observation (adjacent tiles) is wrapped with the previous action and then flattened as the resulting observation. A learnable linear mapping is used to map the observation from the environment to an input embedding. For SparseNet, we use 128 neurons, 16 of which are used to receive the input embedding (each receives the same input embedding), while the other neurons only receive inputs from their connected neurons.

Training uses a full-episode REINFORCE policy gradient update through backpropagation through time (BPTT): episodes are collected from parallel environments, shorter episodes are padded and masked, discounted Monte Carlo returns are computed, and one policy gradient update is applied. For `labyrinth_escape` we use an episode length of 80; for `labyrinth_escape` we use an episode length of 40. Rollout length is the same as the episode length in both environments. As mentioned in Appendix I.2, to estimate the mixing weight adaptively, since the true gradient in equation 3 is not available, we simply use the downstream propagated gradient to replace the true gradient. This, however, could accumulate excessive bias in the gradient estimation, especially for earlier time steps. To make sure that the portion of synthetic gradients is consistent across time steps, we employ the following scheduling heuristic. Each neuron estimates its mixing weight θ normally, but a schedule is applied during gradient propagation through time. For node i in the BPTT graph, let $\mathcal{C}(i)$ be its children and define the mean child deficit as $\delta_{t+1,i} = |\mathcal{C}(i)|^{-1} \sum_{j \in \mathcal{C}(i)} (1 - \theta_{t+1,j})$. The update is



(a) Learning rate. (b) Scale of the reward noise.

(c) Num. of parents. (d) Adaptive λ along training steps.

Figure 6: **MNIST contextual bandit ablations.** (a) Each method has different optimal learning rate due. (b) Synthetic gradient propagation shows advantages in high variance regime. (c) Synthetic gradient propagation favors sparse connectivity. (d) Dynamics of the adaptive mixing weight along training steps.

$\theta_{t,i} = \min\{1, \theta_{t+1,i} + \rho\,\delta_{t+1,i}\}$, so earlier values increase in proportion to the downstream deficit and are maxed out at 1. In this experiment, we use $\rho = 0.1$.

SparseNet is recurrent in this experiment; the common SparseNet functions and adaptive mixing rule are described in Section I.2. Here, the synthetic gradient mixing is applied during BPTT on the time unrolled computational graph. We compare SparseNet sgprop, SparseNet backprop, GRU [12], and LSTM [53]. The GRU and LSTM baselines use backpropagation and use the same observation preprocessing and actor head layout as SparseNet. The recurrent hidden states of all models are reset to zero at the episode starting step. For the baseline choices: LSTM and GRU are both popular RNN architectures, and in particular GRU is shown to be the best in POPGym environments, outperforming other baselines including MLP, LSTM, and Fast Autoregressive Transformers (FART) [29].

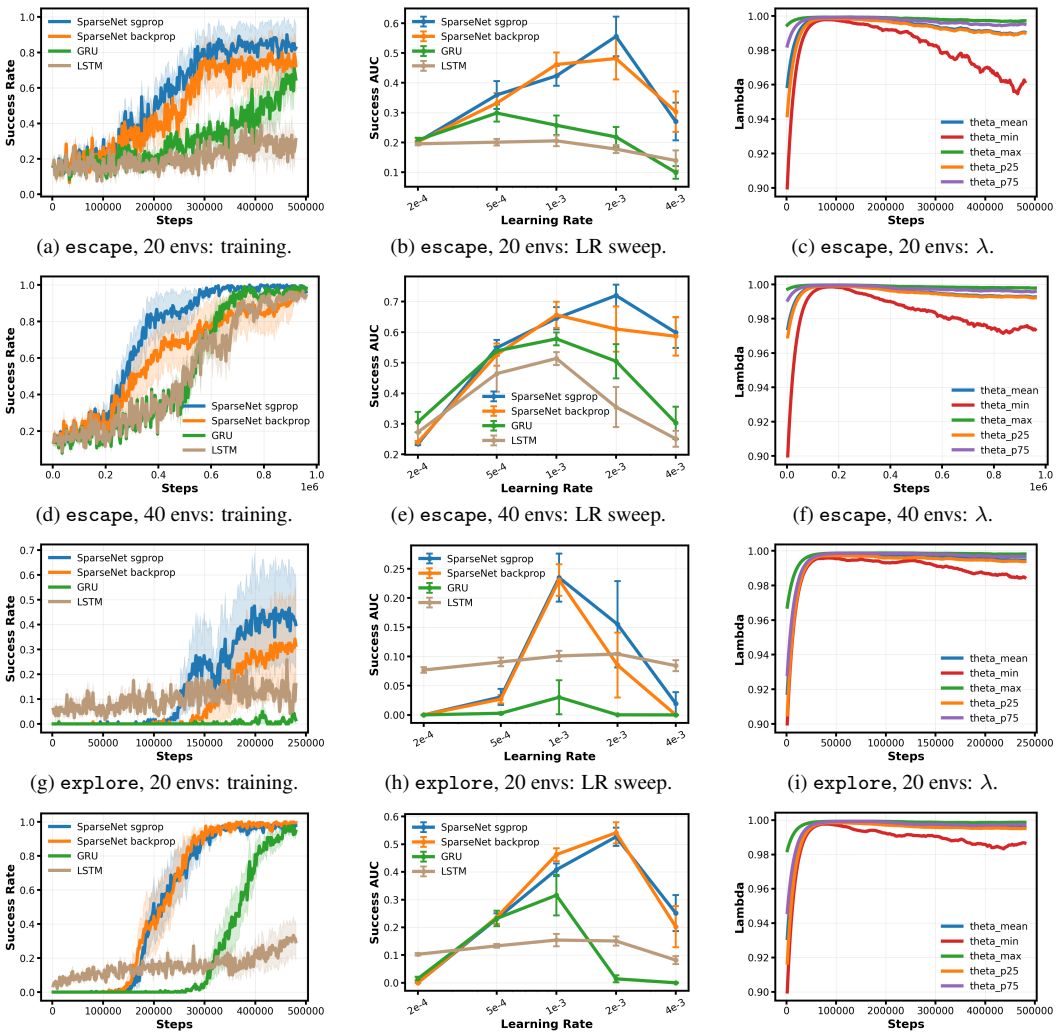
The results are plotted and discussed in Figure 7. Other detailed experimental settings are shown in Table 2 and Table 3.

Table 2: **POPGym model dimensions.**

Model	Dimensions
GRU	input size 128, hidden size 256, actor hidden size 128, one recurrent layer
LSTM	input size 128, hidden size 256, actor hidden size 128, one recurrent layer
SparseNet	input size 128, num. of neuron 128, neuron dim 6, num. of slots 6, input neurons 16, each neuron is sparsely connected with 6 parents

Table 3: **POPGym settings for `labyrinth_escape` and `labyrinth_explore`.**

Parallel environments	$\{20, 40\}$
Learning rates	$\{2 \cdot 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 2 \cdot 10^{-3}, 4 \cdot 10^{-3}\}$
Synhtetic grad predictor learning rate	10^{-5}
Seeds	$\{0, 1, 2, 3, 4, 5\}$
Total batches	300
Discount factor	$\gamma = 0.95$
Entropy coefficient	0.01



(a) `escape`, 20 envs: training. (b) `escape`, 20 envs: LR sweep. (c) `escape`, 20 envs: λ.

(d) `escape`, 40 envs: training. (e) `escape`, 40 envs: LR sweep. (f) `escape`, 40 envs: λ.

(g) `explore`, 20 envs: training. (h) `explore`, 20 envs: LR sweep. (i) `explore`, 20 envs: λ.

(j) `explore`, 40 envs: training. (k) `explore`, 40 envs: LR sweep. (l) `explore`, 40 envs: λ.

Figure 7: **POPGym labyrinth results.** Each row shows one environment and parallel-environment setting; columns show training dynamics, learning-rate sensitivity measured by the area-under-curve (AUC) of the success rate curve, and the adaptive mixing weight λ along steps. The more parallel environments, the more sampled trials the agent sees before each parameter update. Results show that synthetic gradient propagation helps the most in few-sample regime. Each model has different optimal learning rate, and the training dynamis show their optimal ones.

J Discussion: Gradient-Estimation MSE and Condition (B)

J.1 The Bridge from Gradient-Estimation MSE to Optimization Convergence

We present the core principles behind how lower gradient MSE leads to better optimization convergence under smoothness objectives. Suppose we apply stochastic gradient descent $\theta_{t+1} \leftarrow \theta_t - \eta \hat{g}_t^\theta$ with step size η on model parameters θ to a loss function $\ell(\theta)$, where $\hat{g}_t^\theta$ is the gradient estimator of interest. Suppose ℓ is β-smooth with respect to the model parameters θ, a standard assumption in optimization theory, meaning

$$\ell(\theta') \leq \ell(\theta) + \langle \nabla \ell(\theta), \theta' - \theta \rangle + \frac{\beta}{2} \|\theta' - \theta\|^2$$

for any parameters θ, θ'. Defining the gradient-estimation MSE as $\mathbb{E}\big[\|\hat{g}_t^\theta - \nabla\ell(\theta_t)\|^2\big] \le \Delta^2$, using step size $\eta \le 1/\beta$, we can obtain the one-step update inequality via standard algebraic manipulations

$$\mathbb{E}_t[\ell(\theta_{t+1})] \le \ell(\theta_t) - \frac{\eta}{2}\|\nabla\ell(\theta_t)\|^2 + \frac{\eta}{2}\Delta^2.$$

Thus, we can see that a stochastic-gradient step achieves the usual descent from the true gradient, up to an additive penalty proportional to the estimator MSE.

Beyond the principled one-step analysis above, gradient estimation MSE is shown to be useful in analyzing various SGD convergence in general. For an extended convergence analysis of SGD with biased gradient estimators, for example, see Theorem 1 in [28], section 3 and section 4 in [17], or section 4 in [16].

Moreover, we provide direct experimental diagnostics comparing gradient-estimation MSE and training performance, as shown in Section K.1.

J.2 Discussion on When We Can Expect Condition (B) to Hold in General

We illustrate that Condition (B) is similar to a network capability condition. Consider a layered directed computational graph and a node v. For each sample, the node receives a local input z_v from preceding nodes and produces an output $x_v = f_v(z_v)$. We ask the question: how capable can this node be in helping to optimize the overall loss? To measure this capability potential, we compare it to an idealized target: let Zcollect the additional upstream information that could, in principle, be used to choose a better x_v. Let ℓ be the loss term associated with this sample, which is a function of the node variables including x_v. As the node v aims to optimize its function $x_v = f_v(z_v)$, the loss landscape given this input z_v is

$$L_{\text{local}}(z_v, x_v) := \mathbb{E}[\ell \mid z_v],$$

whereas a hypothetical oracle node that also observes more information Z faces

$$L_{\text{oracle}}(z_v, Z, x_v) := \mathbb{E}[\ell \mid z_v, Z].$$

Therefore, for the current architecture to be capable of achieving near-oracle performance, the information available at node v should, in principle, be sufficient for it to select an action $x_v = f_v(z_v)$ close to the oracle-optimal one. This capability requirement can be expressed as

$$\arg\min_{x_v} L_{\text{local}}(z_v, x_v) \approx \arg\min_{x_v} L_{\text{oracle}}(z_v, Z, x_v). \tag{14}$$

Now recall that condition (B) says that the gradient g_v the node receives should be sufficiently entailed by its local observation: $\mathbb{E}[g_v \mid z_v] \approx \mathbb{E}[g_v \mid z_v, Z]$. Noting that $g_v := \nabla_{x_v}\ell$, Condition (B) is essentially

$$\nabla_x L_{\text{local}}(z_v, x_v) \approx \nabla_x L_{\text{oracle}}(z_v, Z, x_v).$$

Condition (B) can therefore be interpreted as a first-order version of the capability requirement in equation 14. Thus, architecture with no hope of good performance cannot satisfy (B).

We measure how well the condition (B) is satisfied in experiments in Section K.2.

K Ablation Studies and Additional Experiments

In this section, we present three additional ablation studies.

K.1 From Gradient-Estimation MSE to Learning Performance

We vary the quality of the synthetic gradient estimators and observe their corresponding gradient estimation MSE and training performance. Recall that synthetic gradient estimators are trained along with the regular model update. We can control the quality of the synthetic gradient estimator by varying how much effort and resources we put into the synthetic gradient estimator training per normal model update step. A good synthetic gradient estimator does reduce gradient estimation MSE and improve training performance.

We compare the following four gradient estimators on SparseNet in the MNIST bandit experiment.

- (**backprop**) gradient obtained by ordinary backpropagation.
- (**sgprop**) For each model update step, the synthetic gradient estimators are updated using the same batch of samples as used in the model update. This is the setting we used in our main experiments.
- (**sgprop-5**) For each model update step, the synthetic gradient estimators are updated using 5 fresh batches of samples.
- (**sgprop-10**) For each model update step, the synthetic gradient estimators are updated using 10 fresh batches of samples.

We note that sgprop-5 and sgprop-10 only serve for ablation purposes, as their synthetic gradient predictors use extra data samples than regular training. Nonetheless, the regular sgprop which uses the same samples as backprop shows improved gradient MSE and training performance. For synthetic gradient propagation, we fix the mixing weight to be 0.5. All other configurations are set to be the same: we train for 1000 steps under learning rate 1e-3 for 3 random seeds with the SparseNet architecture same as our main experiments.

We report evaluation reward and per-sample gradient-estimation MSE along the training trajectory for each method in Table 4, where uncertainty is measured by the standard error of the mean (SEM) across seeds. The gradient-estimation MSE is estimated as follows: sample a batch of validation images and obtain per-sample gradient estimators $\hat{\tilde{\boldsymbol{g}}}^\theta$ from each method; calculate the mean $\bar{\boldsymbol{g}}^\theta$ of using the ground truth backpropagated gradients; calculate the empirical mean of $||\hat{\tilde{\boldsymbol{g}}}^\theta - \bar{\boldsymbol{g}}^\theta||_2^2$ as the estimated per-sample gradient-estimation MSE.

Table 4: **Evaluation reward and gradient estimation MSE at different training steps**. All values in panel (b), including uncertainties, are multiplied by 10^{-3}. The evaluation-reward ranking of the four methods aligns with their gradient-estimation MSE: lower MSE corresponds to higher evaluation reward.

(a) Evaluation reward (↑; higher is better)

Method	Step 200	Step 400	Step 600	Step 800	Step 1000
backprop	0.186 ± 0.0043	0.252 ± 0.011	0.363 ± 0.0091	0.438 ± 0.012	0.521 ± 0.02
sgprop	0.223 ± 0.015	0.361 ± 0.0075	0.474 ± 0.0065	0.532 ± 0.013	0.602 ± 0.027
sgprop-5	0.453 ± 0.035	0.634 ± 0.017	0.730 ± 0.015	0.773 ± 0.01	0.800 ± 0.018
sgprop-10	0.503 ± 0.025	0.693 ± 0.012	0.753 ± 0.0028	0.796 ± 0.0058	0.822 ± 0.0062

(b) Gradient estimation MSE ($\times 10^{-3}$; ↓; lower is better)

Method	Step 200	Step 400	Step 600	Step 800	Step 1000
backprop	196 ± 9.6	222 ± 20	290 ± 15	335 ± 28	380 ± 15
sgprop	11.5 ± 1.9	13.6 ± 1.5	17.2 ± 1.9	17.4 ± 0.81	21.0 ± 0.73
sgprop-5	6.86 ± 0.46	6.30 ± 0.43	5.80 ± 0.64	6.04 ± 0.45	6.31 ± 0.27
sgprop-10	6.04 ± 0.061	5.15 ± 0.51	5.00 ± 0.52	4.79 ± 0.31	5.37 ± 0.78

K.2 how Well Condition (B) is Satisfied?

Condition (B) in Definition 5 requires that a node's local observation $\boldsymbol{z}_v$ be sufficient for the conditional mean of its incoming gradient: $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v] = \mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_{<l}]$, where $\boldsymbol{Z}_{<l}$ is any information in previous layers, e.g., maximally the global input $\boldsymbol{Z}_0$. In this ablation study, we use the global input, since with our deterministic neural networks, any $\boldsymbol{Z}_{<l}$ can be derived from $\boldsymbol{Z}_0$.

To evaluate how well this condition is approximately satisfied, we need to estimate the two terms: $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v]$ and $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_0]$.

For the first term, we train each synthetic gradient predictor from $\boldsymbol{z}_v$ to $\boldsymbol{g}_v$ to its best using an extensive amount of samples. For the second term, we compute the ground truth conditional mean gradient $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_0]$ by analytically averaging over all actions under the fixed policy. We then compare the l2 distance and cosine similarity between the two gradients, averaged over nodes. The l2 distance between the two terms provides a conservative measure of how closely Condition (B) is satisfied, since it includes both condition (B) violation and the synthetic gradient predictor error.

For SparseNet, we use the same configuration as the main MNIST bandit experiment, freeze a randomly initialized policy, and train its synthetic-gradient predictors for 1000 steps on clean MNIST samples. Then, for each validation image $\boldsymbol{Z}_0$, each node v gets its local observation $\boldsymbol{z}_v$ and makes a prediction of its gradient that approximates $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v]$, and we compute the ground truth conditional mean gradient $\mathbb{E}[\boldsymbol{g}_v \mid \boldsymbol{z}_v, \boldsymbol{Z}_0]$ by analytically averaging over all actions under the fixed policy. Results are averaged across nodes and five seeds, with standard error of mean (SEM) reported, as shown in Table 5.

Table 5: **Layer-wise L2 distance (ℓ_2; lower is better) and cosine similarity (Cosine; higher is better)**. L2 distances and their uncertainties are multiplied by 10^{-2}. At the layer interface of width $d = 2000$, cosine similarities of approximately 0.5–0.6 reflect substantial positive alignment, compared with a chance-level absolute cosine similarity of about $O(1/\sqrt{d})$. The predicted gradients are meaningfully aligned with the ground-truth gradients, showing that condition B is reasonably approximately satisfied in SparseNet on the MNIST bandit task.

Metric	Layer 0	Layer 1	Layer 2	Layer 3	Layer 4
ℓ_2 ($\times 10^{-2}$)	1.51 ± 0.00428	1.28 ± 0.00637	1.18 ± 0.00604	1.10 ± 0.00555	1.02 ± 0.00446
Cosine	0.515 ± 0.0049	0.599 ± 0.00446	0.605 ± 0.0046	0.608 ± 0.00491	0.610 ± 0.00514

K.3 Preliminary Extension to Transformers

We conduct preliminary experiments that apply synthetic gradients in Mixture-of-Experts (MoE) layers in a small GPT-style transformer. We consider a next-token prediction experiment on TinyStories [18] using a standard three-block GPT-style Transformer. For synthetic gradient prediction, each expert in the MoE layer does a linear transformation from its hidden representations to a gradient prediction. The expert's predictions are then weighted and averaged using the same expert scores computed by the router. The transformer uses a standard GPT-style architecture with MoE FFN: it has 3 transformer blocks, a 50258 token vocabulary, each block's FFN is a 32-expert MoE with top-8 activation, and it amounts to about 17M parameters in total. To simulate the scenario where condition (A) presents, we create a noisy dataset from TinyStories in which the next token target has a 0.2 probability of being the true one, while the rest are random tokens.

We compare the training performance of three variants.

- (**gpt-moe sgprop**) GPT with MoE FFNs trained with synthetic-gradient propagation.
- (**gpt-moe backprop**) GPT with MoE FFNs trained by backpropagation.
- (**gpt backprop**) GPT with MLP FFN trained by backpropagation.

This comparison separates the effect of the MoE architecture from the effect of synthetic gradients. Mixing weight is fixed at 0.5. Training is done with AdamW, with a learning rate of 1e-3, and each batch of samples corresponds to 1k tokens. We report clean validation loss mean ± SEM across three seeds in Table 6.

Table 6: **Validation loss at different training token counts (lower is better)**. In this noisy dataset training study, synthetic gradient propagation achieves lower loss than backprop on the GPT with MoE. The standard GPT with MLP serves as a baseline.

Method	1k tokens	50k tokens	100k tokens	150k tokens	200k tokens
gpt-moe sgprop	10.834 ± 0.001	8.591 ± 0.098	7.395 ± 0.114	7.009 ± 0.072	6.722 ± 0.073
gpt-moe backprop	10.838 ± 0.001	9.140 ± 0.069	7.658 ± 0.029	7.203 ± 0.048	6.830 ± 0.030
gpt backprop	10.626 ± 0.007	8.264 ± 0.247	7.914 ± 0.112	7.554 ± 0.110	7.189 ± 0.095

To evaluate how condition (B) is satisfied in the GPT-MoE architecture. We train the SG predictors within the MoE layers for 100 fresh batches of 16 sequences at SG LR 1e-5. We treat the backpropagated gradient on the clean next token target as the ground truth, and measure the synthetic gradient prediction. This accounts for a conservative measure of Condition (B) as it combines both condition (B) and the prediction error.

Evaluation is done on two kinds of models: randomly initialized; and the same model after five model training warm-up steps. Evaluation uses 32 clean sequences of 256 tokens and computes per-token ℓ_2 distance and cosine similarity at each layer. Values are averaged over tokens and then across seeds, with SEM reported across five random seeds. The results are presented in Table 7 as follows.

Table 7: **Layer-wise L2 distance (ℓ_2; lower is better) and cosine similarity (Cosine; higher is better)** for warmup and random initialization. ℓ_2 distances and their uncertainties are multiplied by 10^{-3}. At the MoE layer interface of width $d = 256$, meaningful gradient alignment (a conservative proxy for Condition (B)) presents across the GPT-MoE layers at random initialization and improves after warming up for five training updates.

(a) Model after warmup

Metric	Layer 0	Layer 1	Layer 2
ℓ_2 ($\times 10^{-3}$)	6.85 ± 0.61	4.38 ± 0.35	3.69 ± 0.23
Cosine	0.291 ± 0.013	0.257 ± 0.011	0.199 ± 0.008

(b) Model at random initialization

Metric	Layer 0	Layer 1	Layer 2
ℓ_2 ($\times 10^{-3}$)	13.8 ± 0.070	8.54 ± 0.054	6.27 ± 0.040
Cosine	0.147 ± 0.0021	0.161 ± 0.0028	0.176 ± 0.0037

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our results show that backpropagation is not necessarily the minimum-MSE estimator of the true gradient when gradient feedback contains uncertainty. We identify conditions under which synthetic gradients reduce gradient-estimation MSE. These claims are proven as shown in our theorems in Section 3 and Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Limitation discussed in Section 7.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All proofs are in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Experiment details are provided in the appendix.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is released.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed experimental settings are presented in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars are plotted from random trials.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: This is primarily a theory work, and all experiments should be easy to run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: the research is conducted with the NeurIPS Code of Ethics

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This work is not closely related to societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer:[N/A]

    Justification: no data or model is released.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: the dataset and baseline methods used in this paper are cited as shown in Section 6.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No assets are introduced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: no crowdsourcing in involved.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: no IRB is involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is only used for writing and language check.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.