Understanding the Training Speedup from Sampling with Approximate Losses

Rudrajit Das [1] **Xi Chen** [2] **Bertram Ieong** [2] **Parikshit Bansal** [1] **Sujay Sanghavi** [1 2]

Abstract

It is well known that selecting samples with large losses/gradients can significantly reduce the number of training steps. However, the selection overhead is often too high to yield any meaningful gains in terms of overall training time. In this work, we focus on the greedy approach of selecting samples with large *approximate losses* instead of exact losses in order to reduce the selection overhead. For smooth convex losses, we show that such a greedy strategy can converge to a constant factor of the minimum value of the average loss in fewer iterations than the standard approach of random selection. We also theoretically quantify the effect of the approximation level. We then develop SIFT which uses early exiting to obtain approximate losses with an intermediate layer's representations for sample selection. We evaluate SIFT on the task of training a 110M parameter 12 layer BERT base model, and show significant gains (in terms of training hours and number of backpropagation steps) without any optimized implementation over vanilla training. For e.g., to reach 64% validation accuracy, SIFT with exit at the first layer takes ~ 43 hours compared to ~ 57 hours of vanilla training.

1. Introduction

Stochastic Gradient Descent (SGD) and its variants are the algorithms of choice for solving large-scale optimization problems that arise in training machine learning models. These are problems of the form $\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w})$, where $F(.)$ is the expected population loss and $\boldsymbol{w} \in \mathbb{R}^d$ is the vector of model parameters. More specifically, if $f(\boldsymbol{w}, .)$ is the per-sample loss and $\mathcal{D}$ is the data distribution, then $F(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{w}, \boldsymbol{x})]$. The standard SGD update rule at $\boldsymbol{w}$ with step-size/learning rate η is:

$$\boldsymbol{w}_+ = \boldsymbol{w} - \eta\nabla f(\boldsymbol{w}, \boldsymbol{x}), \tag{1}$$

where the sample/example $\boldsymbol{x}$ is drawn from $\mathcal{D}$. In practice, the gradient over a single sample is replaced by the average gradient over a mini-batch of samples; for our theoretical results, we shall stick to batch-size = 1 (as in Equation (1)).

The choice of the sample $\boldsymbol{x}$ in Equation (1) significantly impacts the speed of convergence. There is copious work on selecting "important" samples for speeding up convergence. In this paradigm, the seminal idea is that of *importance sampling* which proposes to sample the examples such that resulting stochastic gradient is unbiased while having the minimum possible variance (Zhao & Zhang, 2015; Alain et al., 2015; Needell et al., 2014). It turns out that the optimal solution is to sample the examples with probability proportional to their per-sample gradient norms. While this has a clean theoretical solution, it is completely infeasible from the practical standpoint because of the high cost of computing the per-sample gradient norms. To address this shortcoming, several approximations to this exact solution have been proposed; we discuss these and related ideas in Section 2. In this space, one high-level idea is to use the per-sample loss valueas a proxy to the per-sample gradient norm, favoring samples with high losses for performing the update (Loshchilov & Hutter, 2015; Shrivastava et al., 2016; Katharopoulos & Fleuret, 2017; Kawaguchi & Lu, 2019; Zhang et al., 2019). We discuss the important works based on this idea in detail in Section 2. Since exact loss computation is itself expensive especially for large models, some works rely on *approximate* loss values for sample selection; for instance, using an auxiliary model to predict loss values of the actual model being trained (Katharopoulos & Fleuret, 2017; Zhang et al., 2019). So in summary, several approximations to the core idea of exact importance sampling have been proposed to make it practically usable.

Despite the myriad sample selection approaches, there is a paucity of theoretical results quantifying how much speedup we can obtain over random sampling and their limitations – especially for approaches involving the use of approximate quantities. In this work, we focus on the approach of choosing the sample with the highest loss for performing the gradient-based update. More specifically, suppose we are given $R > 1$ i.i.d. samples drawn from $\mathcal{D}$ and their **ap-**

[1]UT Austin [2]Amazon. Correspondence to: Rudrajit Das <rdas@utexas.edu>, Xi Chen <xichex@amazon.com>, Sujay Sanghavi <sanghavi@mail.utexas.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

proximate loss values with the constraint that we can pick only **one** sample by inspecting the *approximate* losses to perform the update. We consider the *greedy* approach of selecting the sample with the *largest approximate loss*, and we call this **greedy SGD (GSGD)**. Also, we will refer to the default approach of picking a sample uniformly at random as *(vanilla) SGD*. In Section 5, we theoretically compare the convergence rates of GSGD (with *approximate* losses) against SGD, characterizing the benefits and limitations of the former. We would like to clarify that *we are not claiming that GSGD is a novel algorithm*; in fact, it is very similar to OSGD (Kawaguchi & Lu, 2019) in spirit.[1] Rather, *we are claiming that our theoretical characterization of its benefits and limitations – with approximate losses – is the first of its kind*, to the best of our knowledge.

On the applied side, we propose to use *early exiting* (Teerapittayanon et al., 2016; Schwartz et al., 2020) as a light-weight way to obtain *approximate* losses for sample selection in training large ML models. To be clear, *our novelty is in the light-weight filtering process for training via early exiting*. Empirical results on a 110M parameter BERT base model and a ResNet-50 model show that early exit-based sample selection yields significant improvements.

We will now elaborate on our main **contributions**.
(a) In Section 5, we theoretically compare **greedy SGD (GSGD)** (i.e., pick the sample with the *highest approximate loss*) against vanilla SGD (i.e., pick a sample at random).

- Theorem 5.7 provides a convergence bound for GSGD on smooth convex losses assuming that the argmax of the approximate losses is equal to the argmax of the actual losses (Assumption 5.5). In Section 5.1, we consider a setting wherein Assumption 5.5 no longer holds. We quantify the degradation in the performance of GSGD in this setting; see Theorems 5.14 and 5.15.
- A **key insight** is that *GSGD can converge to a (problem-dependent) constant factor of* $F^* = \min_{\boldsymbol{w}} F(\boldsymbol{w})$ *in fewer iterations than SGD*; see Remark 5.11. This is of interest in training large ML models on extremely large datasets, where converging exactly to F^* is infeasible and instead converging *faster* to $\mathcal{O}(F^*)$ is desirable. On the negative side, our result indicates that GSGD may not converge to F^* asymptotically and hence it can be worse than SGD asymptotically (Remark 5.9).

(b) In Section 6, we propose *early exiting* as a way to cheaply obtain approximate losses with an intermediate layer's representations for sample selection in training large ML models such as *transformers*. Specifically, we propose to do backpropagation on only 50% of the samples in a batch with the *highest approximate losses obtained via early exiting*. To our knowledge, early exiting has not been studied for accelerating *training*. We call this early-exit based sample "sifting" process SIFT. This can be seamlessly integrated with other sample selection schemes; in particular, we also try large *entropy*-based sample selection with early exiting.

- In Section 7.1, we show the efficacy of SIFT in training a 12 layer BERT base model with 110M parameters from scratch. Specifically, to achieve 64% validation accuracy, loss-based and entropy-based SIFT with exit at the first layer take roughly 43 and 40 hours, respectively, compared to roughly 57 hours of vanilla training involving no sample selection. Further, in Section 7.2, we show that SIFT is also very effective in speeding up the training of a modified ResNet-50 model that is amenable to early exiting.
- In Section 6.1, we quantify the probability of correctly selecting the sample with the largest *actual loss* using early exiting for feedforward linear neural networks.

2. Related Work

There is a large body of work proposing several kinds of sample-selection schemes for accelerated training. These include importance sampling methods, sample reordering based approaches and algorithms focusing on samples with higher losses. Our approach falls in the last category.

Optimal importance sampling. Importance sampling asks how should the training examples be sampled so as to obtain an unbiased stochastic gradient with the minimum possible variance. Zhao & Zhang (2015); Alain et al. (2015); Needell et al. (2014) show that the optimal solution to this problem is to sample the examples with probability proportional to their per-sample gradient norms. Unfortunately, this is completely infeasible in practice because of the high cost of computing the per-sample gradient norms.

Approximate importance sampling. Several papers attempt to approximate the optimal importance sampling procedure so as to make it feasible. For convex settings, Zhao & Zhang (2015); Needell et al. (2014) propose sampling with probability proportional to the smoothness constant of the per-sample losses, while Borsos et al. (2018); Stich et al. (2017) propose adaptive sampling strategies. Going beyond convex settings, Alain et al. (2015) present a distributed approach for importance sampling, while Katharopoulos & Fleuret (2017) approximate the importance weights with loss values which are predicted by a smaller network. Katharopoulos & Fleuret (2018) derive an upper bound on the per-sample gradient norms for deep-learning networks which take essentially the same time to

[1] We do not call it OSGD because Kawaguchi & Lu (2019) use exact loss values whereas we use approximate loss values. Moreover, Kawaguchi & Lu (2019) consider the finite-sum (ERM) setting whereas we focus on the stochastic setting.

compute as the per-sample loss values, and propose using the upper bounds for approximating the importance weights. Johnson & Guestrin (2018) approximate the true sampling distribution by solving a robust optimization problem.

Sample reordering. We defer related works in this category to Appendix A.

Selection of samples with large losses. As mentioned in Section 1, GSGD is *not* a novel algorithm and there are prior approaches with the same underlying principle as GSGD, i.e., focus on the samples with the largest loss values. The two algorithms closest to GSGD in spirit are online hard example mining (OHEM) proposed by Shrivastava et al. (2016) and ordered SGD (OSGD) proposed by Kawaguchi & Lu (2019) – *except that both algorithms use exact losses* for sample selection. OHEM was proposed for training object detectors (in computer vision) and it involves back-propagating using the gradients of only the samples with the k largest losses in batch of size b with $k < b$. However, Shrivastava et al. (2016) do not provide any theoretical guarantees. OSGD (Kawaguchi & Lu, 2019) is essentially the same algorithm as OHEM (Shrivastava et al., 2016), but with convergence guarantees and generalization bounds. However, *unlike our theoretical results* for GSGD, Kawaguchi & Lu (2019) do *not* show how/when/to what extent ordered SGD is better than plain SGD. So even though GSGD is similar to OSGD in spirit, our theoretical results (for GSGD) are much more comprehensive. Loshchilov & Hutter (2015) propose a sampling strategy which favors picking examples with larger losses. Fan et al. (2017) introduce the average top-k loss and advocate minimizing this loss rather than the empirical average over all the samples.

Early Exiting. Early exiting (Teerapittayanon et al., 2016; Schwartz et al., 2020) is a promising approach to decreasing the computational cost of multilayered neural architectures by approximating the output of a model through its intermediate feature representations. This saves computational costs by dynamically deciding the number of layers/modules (attention-blocks in transformers) to use during inference by *exiting* based on some metric computed on the intermediate representations themselves. Initially used for ResNets (Teerapittayanon et al., 2016), early exiting is now widely popular even for transformer models, especially language models (Schwartz et al., 2020; Xin et al., 2020; Schuster et al., 2022; Rotem et al., 2023; Xin et al., 2021; Zhu, 2021). Recent work around early exiting for large language models also explores accelerated decoding (Schuster et al., 2022) and improving factuality (Chuang et al., 2023). In this work, we solely use early exiting as a light-weight way to obtain approximate losses with the intermediate layer representations for sample selection. To our knowledge, *early exiting has not been used in prior work for speeding up training*.

3. Notation

Vectors and matrices are in bold font. For a natural number n, we sometimes denote the set $\{1, \ldots, n\}$ by $[n]$. The (Gauss) error function and complementary error function are defined as:

$$\text{erf}(t) := \frac{2}{\sqrt{\pi}} \int_0^t e^{-z^2} dz \text{ and } \text{erfc}(t) := 1 - \text{erf}(t). \quad (2)$$

Note that $\lim_{t\to\infty} \text{erf}(t) = 1$. For any $z \in \mathbb{R}$, we define the sigmoid function as $\text{sig}(z) = \frac{1}{1+e^z}$.

4. Problem Setting

We briefly recap the optimization problem introduced in Section 1. Given access to a first-order optimization oracle, we would like to minimize $F(\boldsymbol{w}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{w}, \boldsymbol{x})]$, where $\boldsymbol{w} \in \mathbb{R}^d$ and $\mathcal{D}$ is the data distribution.

Standard First-Order Stochastic Optimization Oracle. A query at $\boldsymbol{w}$ returns $\nabla_{\boldsymbol{w}} f(\boldsymbol{w}, \boldsymbol{x})$, where $\boldsymbol{x} \sim \mathcal{D}$.

We consider a more *generous* oracle. However, it can also perform **one gradient evaluation** per query (same as the standard oracle).

Definition 4.1 (Proposed First-Order Stochastic Optimization Oracle). A query at $\boldsymbol{w}$ first returns a set of $R > 1$ samples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(R)}\} = \mathcal{S}^{(R)}$ drawn i.i.d. from $\mathcal{D}$ and their *approximate* function[2] values $\{\widetilde{f}(\boldsymbol{w}, \boldsymbol{x}^{(1)}), \ldots, \widetilde{f}(\boldsymbol{w}, \boldsymbol{x}^{(R)})\}$. The user can then pick **one** $\widehat{\boldsymbol{x}}$ from $\mathcal{S}^{(R)}$, and the oracle will return $\nabla_{\boldsymbol{w}} f(\boldsymbol{w}, \hat{\boldsymbol{x}})$.

Later, we shall make an assumption on the relation between the approximate function value $\widetilde{f}(\boldsymbol{w}, \boldsymbol{x})$ and the actual function value $f(\boldsymbol{w}, \boldsymbol{x})$. Now that we have introduced the oracle that we consider in this work, we state the $\widehat{\boldsymbol{x}}$ chosen by vanilla SGD and greedy SGD (GSGD).

SGD choice: Pick $\widehat{\boldsymbol{x}}$ uniformly at random from $\mathcal{S}^{(R)}$.

GSGD choice: Pick $\widehat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}\in\mathcal{S}^{(R)}} \widetilde{f}(\boldsymbol{w}, \boldsymbol{x})$.

We state the GSGD update rule in more detail next.

4.1. Greedy SGD (GSGD) Algorithm

In the kth iteration, we observe a set of R i.i.d. samples drawn from $\mathcal{D}$, say $\mathcal{S}_k^{(R)} = \{\boldsymbol{x}_k^{(1)}, \ldots, \boldsymbol{x}_k^{(R)}\}$. We pick:

$$\widehat{\boldsymbol{x}}_k = \arg\max_{\boldsymbol{x}\in\mathcal{S}_k^{(R)}} \widetilde{f}(\boldsymbol{w}_k, \boldsymbol{x}). \quad (3)$$

The update of Greedy SGD (GSGD) with step-size η_k is:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta_k \nabla f(\boldsymbol{w}_k, \widehat{\boldsymbol{x}}_k). \quad (4)$$

[2]Throughout this work, we interchange "loss" value and "function" value freely. In most cases, we use "function" value in the context of optimization-based discussions and "loss" value in the context of ML-based discussions.

The update of vanilla SGD is the same as Equation (4), except with $\widehat{\boldsymbol{x}}_k$ being a random sample from $\mathcal{S}_k^{(R)}$.

5. GSGD vs. SGD for Smooth Convex Objectives

We begin by stating some assumptions and definitions.

Assumption 5.1 (Continuity). For any $\boldsymbol{x} \sim \mathcal{D}$, $f(\boldsymbol{w}, \boldsymbol{x})$ is continuous w.r.t. $\boldsymbol{w}$.

Assumption 5.2 (Convexity). For any $\boldsymbol{x} \sim \mathcal{D}$, $f(\boldsymbol{w}, \boldsymbol{x})$ is convex w.r.t. $\boldsymbol{w}$.

Assumption 5.3 (Smoothness). For any $\boldsymbol{x} \sim \mathcal{D}$, $f(\boldsymbol{w}, \boldsymbol{x})$ is L-smooth w.r.t. $\boldsymbol{w}$.

Assumption 5.4. For any $\boldsymbol{x} \sim \mathcal{D}$, $\min_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w}, \boldsymbol{x}) = 0$.

Let $\Phi_F := \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} F(\boldsymbol{w})$ and $F^* := \min_{\boldsymbol{w} \in \mathbb{R}^d} F(\boldsymbol{w})$. We restrict our attention to the case of Φ_F being closed and compact.

Let us first consider the case where the argmax of the approximate function values is the same as the argmax of the exact function values; we will relax this assumption later in Section 5.1.

Assumption 5.5 (Approximate function values preserve argmax). In the setting of Definition 4.1, $\widetilde{f}$ satisfies:

$$\arg\max_{\boldsymbol{x} \in \mathcal{S}^{(R)}} \widetilde{f}(\boldsymbol{w}, \boldsymbol{x}) = \arg\max_{\boldsymbol{x} \in \mathcal{S}^{(R)}} f(\boldsymbol{w}, \boldsymbol{x}).$$

Under Assumption 5.5, $\widehat{\boldsymbol{x}}_k$ in Equation (3) becomes the same as $\arg\max_{\boldsymbol{x} \in \mathcal{S}_k^{(R)}} f(\boldsymbol{w}_k, \boldsymbol{x})$.

Definition 5.6. For $R > 1$, let

$$\widehat{F}_R(\boldsymbol{w}) := \mathbb{E}_{\{\boldsymbol{x}^{(j)}\}_{j=1}^R \underset{\text{iid}}{\sim} \mathcal{D}} \Big[\max_{\boldsymbol{x} \in \{\boldsymbol{x}^{(j)}\}_{j=1}^R} f(\boldsymbol{w}, \boldsymbol{x}) \Big].$$

Define

$$\rho_R(\boldsymbol{w}) := \frac{\widehat{F}_R(\boldsymbol{w})}{F(\boldsymbol{w})} \text{ and } \rho_R^* := \inf_{\boldsymbol{w} \notin \Phi_F} \rho_R(\boldsymbol{w}).$$

Also, suppose

$$\sup_{\boldsymbol{w}^* \in \Phi_F} \widehat{F}_R(\boldsymbol{w}^*) \leq \Delta_R.$$

Except for the trivial case of $f(\boldsymbol{w}, \boldsymbol{x}_1) = f(\boldsymbol{w}, \boldsymbol{x}_2)\ \forall\ \boldsymbol{x}_1, \boldsymbol{x}_2$ which we disregard, ρ_R^* is *strictly* bigger than 1.

Further, consider a point $\widehat{\boldsymbol{w}}^* \notin \Phi_F$ that is ϵ-close to some $\boldsymbol{w}^* \in \Phi_F$ (i.e., $\|\widehat{\boldsymbol{w}}^* - \boldsymbol{w}^*\|_2 \leq \epsilon$) and let $\epsilon \to 0$. In that case, under Assumption 5.1 (and because the max operation preserves continuity) and Definition 5.6, $\widehat{F}_R(\widehat{\boldsymbol{w}}^*) \to \widehat{F}_R(\boldsymbol{w}^*) \leq \Delta_R$. Also, $F(\widehat{\boldsymbol{w}}^*) \geq F^*$. Thus, by definition, $\rho_R^* \leq \widehat{F}_R(\widehat{\boldsymbol{w}}^*)/F(\widehat{\boldsymbol{w}}^*) \leq (\Delta_R/F^*)$. Hence, we have that:

$$1 < \rho_R^* \leq \frac{\Delta_R}{F^*},\ \forall\ R > 1. \tag{5}$$

In Section 5.2, we quantify $\rho_R(\boldsymbol{w})$ and ρ_R^* for fitting a model with the squared loss.

For our convergence results, we assume that our initialization is $\boldsymbol{w}_0$ and let

$$D_0 := \min_{\boldsymbol{w}^* \in \Phi_F} \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|.$$

We are now ready to state our convergence results.

Theorem 5.7 (GSGD). *Suppose Assumptions 5.1, 5.2, 5.3, 5.4 and 5.5 hold. Set $\eta_k = \eta < \frac{1}{L}$ for all k. Then, GSGD has the following convergence guarantee after K iterations:*

$$\mathbb{E}\left[F\left(\frac{1}{K}\sum_{k=0}^{K-1} \boldsymbol{w}_k\right)\right] \leq \frac{D_0^2}{2\rho_R^* \eta(1-\eta L)K} + \frac{\Delta_R}{\rho_R^*(1-\eta L)}.$$

The proof of Theorem 5.7 can be found in Appendix C.

We now state a corresponding folklore result for SGD.

Theorem 5.8 (SGD). *Suppose Assumptions 5.2, 5.3 and 5.4 hold. Set $\eta_k = \eta < \frac{1}{L}$ for all k. Then, SGD has the following convergence guarantee after K iterations:*

$$\mathbb{E}\left[F\left(\frac{1}{K}\sum_{k=0}^{K-1} \boldsymbol{w}_k\right)\right] \leq \frac{D_0^2}{2\eta(1-\eta L)K} + \frac{\eta L F^*}{1-\eta L} + F^*.$$

From Theorem 5.7, observe that in general, GSGD *may not* converge to the minimum value F^* asymptotically (i.e., with $K \to \infty$). At best, we can show that GSGD converges to $\frac{\Delta_R}{\rho_R^*}$ which is $\geq F^*$ (this follows from Equation (5)). But SGD can indeed converge to F^* asymptotically by setting $\eta = \frac{1}{\eta L \sqrt{K}}$ for example in Theorem 5.8. Based on this, we make the following remark.

Remark 5.9. GSGD may be worse than SGD asymptotically.

However, **GSGD is better than SGD *early on*** as $\rho_R^* > 1$ and assuming $\Delta_R = \mathcal{O}(F^*)$ (and $F^* \neq 0$), **GSGD can converge to a constant factor of F^* in fewer iterations than SGD**. We formalize this next.

Corollary 5.10 (Up to what point is GSGD better than SGD?). *Suppose we run GSGD and SGD with constant step-size $\eta < \frac{1}{L}$. In that case, until $K = \frac{D_0^2(\rho_R^* - 1)}{2\eta(\Delta_R - \rho_R^* F^*)}$ iterations, the convergence bound of GSGD in Theorem 5.7 is better than that of SGD in Theorem 5.8.*

Thus, ***GSGD can converge to $\frac{\Delta_R - F^*}{(1-\eta L)(\rho_R^* - 1)}$ function value in fewer iterations than SGD.***

Remark 5.11. Based on Corollary 5.10, when $\Delta_R = \mathcal{O}(F^*)$, GSGD can converge to $\mathcal{O}(F^*)$ function value **in fewer iterations** than SGD. This is of particular interest in training large ML models such as transformers on extremely large datasets, *where minimizing the training loss exactly is infeasible* and converging *faster* to a constant factor of the minimum loss value is desirable.

5.1. Beyond Argmax-Preserving Approximate Function Values

Our previous results were under Assumption 5.5, i.e., the argmax of the approximate function values ($\widetilde{f}$) is always equal to the argmax of the actual function values (f). Here we relax this assumption by instead modeling $\widetilde{f}$ as a noisy version of f, and provide a convergence result for GSGD in such a setting. Modeling $\widetilde{f}$ as a noisy version of f is analogous to modeling the stochastic gradients as a noisy version of the actual gradient in vanilla stochastic optimization. Specifically, we make the following assumption.

Assumption 5.12 (Approximate function values). There exists $\mu(\boldsymbol{w}) \in \mathbb{R}$ and $\sigma \geq 0$ such that:

$$\widetilde{f}(\boldsymbol{w}, \boldsymbol{x}) = f(\boldsymbol{w}, \boldsymbol{x}) \exp\Big(\mu(\boldsymbol{w}) + \sigma\zeta(\boldsymbol{w}, \boldsymbol{x})\Big),$$

where $\zeta(\boldsymbol{w}, \boldsymbol{x})$ is i.i.d. random noise with mean 0 and variance 1.

Thus, the approximate function value is the actual function values times the exponential of random noise (i.e., $\exp\big(\sigma\zeta(\boldsymbol{w}, \boldsymbol{x})\big)$) times some other scaling (i.e., $\exp(\mu(\boldsymbol{w}))$). Assumption 5.12 is pretty mild as it does not involve any particular distributional assumptions on $\zeta(\boldsymbol{w}, \boldsymbol{x})$ (such as Gaussian, etc.).

The important thing to note is that under Assumption 5.12, $\widehat{\boldsymbol{x}}_k$ in Equation (3) is **not** always equal to $\arg\max_{\boldsymbol{x} \in \mathcal{S}_k^{(R)}} f(\boldsymbol{w}_k, \boldsymbol{x})$ – unlike Assumption 5.5.

Definition 5.13. Let

$$\widehat{F}_{R,\text{approx}}(\boldsymbol{w}) := \mathbb{E}_{\{\boldsymbol{x}^{(j)}\}_{j=1}^{R} \underset{\text{iid}}{\sim} \mathcal{D}, \zeta}\Big[f\big(\boldsymbol{w}, \boldsymbol{x}^{(j^*)}\big) \Big| j^* = \arg\max_{j \in [R]} \widetilde{f}(\boldsymbol{w}, \boldsymbol{x}^{(j)})\Big].$$

Define

$$\rho_{R,\text{approx}}(\boldsymbol{w}) := \frac{\widehat{F}_{R,\text{approx}}(\boldsymbol{w})}{F(\boldsymbol{w})} \text{ and}$$

$$\rho^*_{R,\text{approx}} := \inf_{\boldsymbol{w} \notin \Phi_F} \rho_{R,\text{approx}}(\boldsymbol{w}).$$

Clearly, $\widehat{F}_{R,\text{approx}}(\boldsymbol{w}) \leq \widehat{F}_R(\boldsymbol{w})$ (as defined in Definition 5.6). So we also have:

$$\sup_{\boldsymbol{w}^* \in \Phi_F} \widehat{F}_{R,\text{approx}}(\boldsymbol{w}^*) \leq \sup_{\boldsymbol{w}^* \in \Phi_F} \widehat{F}_R(\boldsymbol{w}^*) \leq \Delta_R.$$

It is easy to extend the proof of Theorem 5.7 to obtain the following result for GSGD under Assumption 5.12.

Theorem 5.14 (GSGD). *Suppose Assumptions 5.1, 5.2, 5.3, 5.4 and 5.12 hold. Set $\eta_k = \eta < \frac{1}{L}$ for all k. Then, GSGD has the following convergence guarantee after K iterations:*

$$\mathbb{E}\left[F\left(\frac{1}{K}\sum_{k=0}^{K-1} \boldsymbol{w}_k\right)\right] \leq \frac{D_0^2}{2\rho^*_{R,\text{approx}}\eta(1-\eta L)K} + \frac{\Delta_R}{\rho^*_{R,\text{approx}}(1-\eta L)}.$$

The expectation in Theorem 5.14 also includes the randomness due to $\zeta(.)$ (i.e., the noise in the approximate function values). The proof of Theorem 5.14 is almost identical to the proof of Theorem 5.7 (see Appendix C) and is obtained by replacing $\widehat{F}_R(\boldsymbol{w})$ with $\widehat{F}_{R,\text{approx}}(\boldsymbol{w})$ and ρ^*_R with $\rho^*_{R,\text{approx}}$.

For the subsequent results in this subsection, we shall consider $R = 2$. Specifically, we will provide a lower bound for $\widehat{F}_{2,\text{approx}}(\boldsymbol{w})$, $\rho_{2,\text{approx}}(\boldsymbol{w})$ and $\rho^*_{2,\text{approx}}$ in terms of $\widehat{F}_2(\boldsymbol{w})$, $\rho_2(\boldsymbol{w})$ and ρ^*_2, respectively, as a function of the noise level σ.

Theorem 5.15. *Suppose Assumption 5.12 holds with $\sigma \leq \frac{1}{2\sqrt{2}}$. Define $p(\sigma) := \big(1 - 0.72\big(1 - e^{-\sqrt{2}\sigma}\big)\big)$. Then:*

$$\widehat{F}_{2,\text{approx}}(\boldsymbol{w}) \geq p(\sigma)\widehat{F}_2(\boldsymbol{w}),$$

$$\rho_{2,\text{approx}}(\boldsymbol{w}) \geq p(\sigma)\rho_2(\boldsymbol{w}) \text{ and } \rho^*_{2,\text{approx}} \geq p(\sigma)\rho^*_2.$$

The proof of Theorem 5.15 is in Appendix D. It is worth pointing out that our result is independent of $\mu(\boldsymbol{w})$; intuitively, this is because a constant scaling (w.r.t. the samples) does not change the argmax. Regarding the dependence w.r.t. σ, notice that $p(\sigma)$ is a decreasing function of σ. So as the noise level σ increases, the lower bound becomes worse; this happens because the quality of the approximate function value worsens as σ increases. Also, as a quick sanity check, observe that $p(0) = 1$. This makes sense because $\sigma = 0$ means no effective noise, i.e., the approximate function values match the actual function values modulo the constant scaling $\exp(\mu(\boldsymbol{w}))$.

In the following corollary, we specify a bound for the noise level σ below which $\rho^*_{2,\text{approx}} > 1$ and thus, we can obtain a speed-up over SGD.

Corollary 5.16. *As long as $\sigma < \sigma_{\max} := \frac{1}{\sqrt{2}} \log\big(\frac{18\rho^*_2}{25 - 7\rho^*_2}\big)$, $p(\sigma)\rho^*_2 > 1$ and therefore, $\rho^*_{2,\text{approx}} > 1$. Hence, for $\sigma < \sigma_{\max}$, GSGD with $R = 2$ using approximate function values for sample selection can be faster than SGD.*

5.2. Quantifying $\rho_R(\boldsymbol{w})$

Here we shall quantify $\rho_R(\boldsymbol{w})$ for a particular case. Let us consider the problem of learning a parameterized model $\mathcal{M}(\boldsymbol{w}^*, .)$ with the squared loss. In this case, our per-sample objective function f is:

$$f(\boldsymbol{w}, \boldsymbol{x}) = \Big(\mathcal{M}(\boldsymbol{w}, \boldsymbol{x}) - \mathcal{M}(\boldsymbol{w}^*, \boldsymbol{x})\Big)^2. \tag{6}$$

We make the following assumption.

Assumption 5.17. For $\boldsymbol{w} \neq \boldsymbol{w}^*$, $\mathcal{M}(\boldsymbol{w}, \boldsymbol{x}) - \mathcal{M}(\boldsymbol{w}^*, \boldsymbol{x}) \underset{\text{iid}}{\sim} \mathcal{N}\big(\varepsilon(\boldsymbol{w}), \delta^2(\boldsymbol{w})\big)$, i.e., the per-sample prediction error is a Gaussian random variable.

Modeling the per-sample prediction error as a Gaussian random variable is fairly reasonable and a similar assumption has been made in (Pennington & Bahri, 2017). We provide a lower bound for $\rho(\boldsymbol{w})$ in this setting.

Theorem 5.18. *Suppose Assumption 5.17 holds. Let* $\nu(R) := \sqrt{\frac{\pi}{2} \log \frac{R}{4 \log R}}$. *Then:*

$$\rho_R(\boldsymbol{w}) \geq \frac{\big(\varepsilon^2(\boldsymbol{w}) + \nu^2(R)\delta^2(\boldsymbol{w}) + 2\nu(R)\varepsilon(\boldsymbol{w})\delta(\boldsymbol{w})\big)\big(1 - \frac{1}{R}\big)}{\varepsilon^2(\boldsymbol{w}) + \delta^2(\boldsymbol{w})}.$$

The proof of Theorem 5.18 is in Appendix E.

Corollary 5.19. *In the setting of Theorem 5.18, if* $\varepsilon(\boldsymbol{w}) \leq \mathcal{O}(\delta(\boldsymbol{w}))$ *for all* $\boldsymbol{w}$ *in our region of optimization, then:*

$$\rho_R^* \geq \left(1 + \Omega\left(\log \frac{R}{\log R} + \sqrt{\log \frac{R}{\log R}}\right)\right)\Big(1 - \frac{1}{R}\Big).$$

6. Approximate Losses via Early Exiting in Neural Networks

For sample selection in GSGD-like algorithms, we propose to use an approximation of the actual loss which is computed on the "early" predictions obtained by applying the linear head after the final layer to an intermediate layer's representation (instead of the final layer's representation). Specifically, suppose $\boldsymbol{\theta}$ is the linear head and for a sample $\boldsymbol{x}$ with label y, $\mathcal{R}(\boldsymbol{x})$ and $\widetilde{\mathcal{R}}(\boldsymbol{x})$ are the final layer's and some intermediate layer's representation with the same dimension as $\boldsymbol{\theta}$, respectively. Then the approximate and actual losses with the cross-entropy loss function denoted by ℓ are $\ell(y, \text{softmax}(\boldsymbol{\theta}^\top \widetilde{\mathcal{R}}(\boldsymbol{x})))$[3] and $\ell(y, \text{softmax}(\boldsymbol{\theta}^\top \mathcal{R}(\boldsymbol{x})))$, respectively; we use the former to select samples for performing the gradient-based update.

We shall now consider a feedforward linear neural network (Kawaguchi, 2016) and quantify the probability of correctly picking the sample with the largest actual loss when using early exiting.

6.1. Probability of Correctly Selecting the Sample with the Largest Actual Loss

We consider a binary classification problem, where each sample $\boldsymbol{x} \sim \mathcal{N}(\vec{0}_d, \mathrm{I}_d)$ has a binary label $y \in \{0, 1\}$. Our model is a k-layer linear feed-forward network parameterized by $\{\boldsymbol{W}_i\}_{i=1}^d \in \mathbb{R}^{d \times d}$ and $\boldsymbol{\theta} \in \mathbb{R}^d$, where the soft prediction $\hat{y}$ for $\boldsymbol{x}$ is:

$$\hat{y} = \text{sig}\Big(\boldsymbol{\theta}^\top \big(\boldsymbol{W}_k \times \boldsymbol{W}_{k-1} \times \ldots \times \boldsymbol{W}_1\big)\boldsymbol{x}\Big). \quad (7)$$

In Equation (7), sig(.) is the sigmoid function[4] as defined in Section 3. For any $j \in [k]$, let us define:

$$\boldsymbol{A}_j := \boldsymbol{W}_j \times \ldots \times \boldsymbol{W}_1 \text{ and } \boldsymbol{B}_j := \boldsymbol{W}_k \times \ldots \times \boldsymbol{W}_{j+1}. \quad (8)$$

Then,

$$\hat{y}_j = \text{sig}\big(\boldsymbol{\theta}^\top \boldsymbol{A}_j \boldsymbol{x}\big) \quad (9)$$

is the "early prediction" for $\boldsymbol{x}$ at the jth layer. Note that $\hat{y} = \hat{y}_k$, where:

$$\hat{y}_k = \text{sig}\big(\boldsymbol{\theta}^\top \boldsymbol{B}_j \boldsymbol{A}_j \boldsymbol{x}\big). \quad (10)$$

In the context of GSGD, we once again focus on the case of $R = 2$. For any two i.i.d. samples $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$, let $y^{(1)}$ and $y^{(2)} \in \{0, 1\}$ be the corresponding ground truth labels and let $\hat{y}_j^{(1)}$ and $\hat{y}_j^{(2)}$ be the corresponding early predictions at the jth layer. Further, let $\ell_j^{(1)}$ and $\ell_j^{(2)}$ be the corresponding cross-entropy losses of the early predictions at the jth layer, i.e.,

$$\ell_j^{(i)} = -y^{(i)} \log\big(\hat{y}_j^{(i)}\big) - (1 - y^{(i)}) \log\big(1 - \hat{y}_j^{(i)}\big), \quad (11)$$

for $i \in \{1, 2\}$. In the context of GSGD, $\ell_j^{(1)}$ and $\ell_j^{(2)}$ are the approximate function values, whereas $\ell_k^{(1)}$ and $\ell_k^{(2)}$ are the actual function values. In this section, we are interested in quantifying the probability (over the randomness of data) that early exiting at the jth layer preserves the argmax; specifically, we wish to quantify

$$p_j := \mathbb{P}_{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}}\Big(\arg\max_{i \in [1,2]} \ell_j^{(i)} = \arg\max_{i \in [1,2]} \ell_k^{(i)}\Big). \quad (12)$$

Note that $p_k = 1$. We have the following result on this.

Theorem 6.1. *Define*

$$\beta_j := \frac{\langle \boldsymbol{A}_j^\top \boldsymbol{\theta}, \boldsymbol{A}_j^\top \boldsymbol{B}_j^\top \boldsymbol{\theta} \rangle}{\|\boldsymbol{A}_j^\top \boldsymbol{\theta}\|_2 \|\boldsymbol{A}_j^\top \boldsymbol{B}_j^\top \boldsymbol{\theta}\|_2}.$$

We restrict our attention to the case of $\beta_j \geq 0$. *Then:*

$$p_j = 1 - \frac{1}{2\sqrt{\pi}} \int_0^\infty \exp\Big(-\frac{y^2}{4}\Big) \text{erfc}\left(\frac{\beta_j y}{2\sqrt{1 - \beta_j^2}}\right) dy, \quad (13)$$

where erfc(.) *is as defined in Equation (2). We also have the following lower bound for* p_j*:*

$$p_j \geq 1 - \sqrt{\frac{2 - 2\beta_j^2}{2 - \beta_j^2}}. \quad (14)$$

[3] For a vector $\boldsymbol{v} = [v_1, \ldots, v_d]$, $\text{softmax}(\boldsymbol{v}) = [\hat{v}_1, \ldots, \hat{v}_d]$ with $\hat{v}_i = \exp(v_i)/\sum_{j=1}^d \exp(v_j)$.

[4] For binary classification problems, the sigmoid function essentially plays the role of the softmax function mentioned earlier.

The proof of Theorem 6.1 can be found in Appendix F. Note that β_j is the (normalized) correlation between $\boldsymbol{A}_j^\top \boldsymbol{\theta}$ and $\boldsymbol{A}_j^\top \boldsymbol{B}_j^\top \boldsymbol{\theta}$. As per Theorem 6.1, $p_j = 1$ when $\beta_j = 1$; this makes sense because $\beta_j = 1$ implies $\boldsymbol{A}_j^\top \boldsymbol{\theta}$ and $\boldsymbol{A}_j^\top \boldsymbol{B}_j^\top \boldsymbol{\theta}$ are parallel and so the argmax operation is unaffected if we use $\hat{y}_j$ instead of $\hat{y}_k$. It is worth mentioning that the lower bound for p_j in Equation (14) is loose for small β_j, but we believe it is tight up to constant factors for $\beta_j \approx 1$; in particular, it is exact in the case of $\beta_j = 1$. Specifically, we have the following simple corollary for $\beta_j \approx 1$.

Corollary 6.2. *In the setting of Theorem 6.1, suppose* $\beta_j = 1 - \tau_j$ *where* $\tau_j \to 0$. *Then,* $p_j \geq 1 - \mathcal{O}(\sqrt{\tau_j})$.

In simple words, if $\boldsymbol{A}_j^\top \boldsymbol{\theta}$ is strongly correlated with $\boldsymbol{A}_j^\top \boldsymbol{B}_j^\top \boldsymbol{\theta}$, then early exiting at the jth layer preserves the argmax with high probability.

In the next section, we empirically demonstrate the efficacy of early exiting in training a transformer model as well as a ResNet model.

7. Empirical Evaluation

We demonstrate the efficacy of early exiting in accelerating the training of a BERT base model and a slightly modified version of ResNet-50 from scratch. Specifically, we apply early exiting for selecting samples in a mini-batch version of *greedy* AdamW which is a simple extension of the greedy SGD idea to AdamW. We refer to this practical early exit-based sample selection or "sifting" strategy as *SIFT*. We describe SIFT in more detail later but at a high level, we propose to backpropagate on only 50% of the examples in a batch with the *highest approximate losses obtained via early exiting*. We would like to emphasize that *our novelty is in the light-weight sifting process for training via early exiting* and not the idea of backpropagating on the samples with large losses.[5]

7.1. BERT Base Model

Here we consider the task of pretraining a BERT base model from scratch with the masked language modeling (MLM) loss. The BERT base model used in our experiments consists of 12 layers with a hidden dimension of 768; *the total number of parameters is 110M*. We train on BookCorpus (Zhu et al., 2015) and English Wikipedia which are two diverse and extensive standard corpora. The validation set for assessing the model's performance is derived from the development partition of the training corpus. This partition ensures that the validation set represents a diverse and unbiased subset of the overall data. In our training, we use 512 as the maximum sequence length. To process the input data, we use the "bert-base-uncased" tokenizer from the Hugging Face model repository. The masking is applied after tokenization with a uniform masking rate of 15%. Our experiments were conducted on AWS p4d.24xlarge instances (8 NVIDIA A100 Tensor core GPUs).

Our baseline algorithm is the vanilla approach with no kind of sample filtering. The micro-batch per GPU is set to 32 sequences (so there are 32 sequences per GPU * 8 GPUs * 512 tokens = 131072 tokens per batch). We will now elaborate on SIFT which does greedy selection based on early exit loss.

Loss-based SIFT: We select 50% sequences per batch (for training) with the largest MLM losses computed on the early predictions obtained from an intermediate layer in the way described in the beginning of Section 6. Specifically, we show results for the first, second, third, sixth and last (i.e., twelfth) layer. The micro-batch per GPU is set to 64 sequences. Since we select half of the sequences for training, the effective batch size per GPU is 32 which is the same as that of baseline.

The early-exit based filtering idea is pretty general in the sense that it can be seamlessly integrated with other sample selection schemes. In particular, we also tried greedy selection based on *early exit entropy* (instead of loss). We describe it below.

Entropy-based SIFT: Everything is the same as loss-based SIFT except that we select 50% sequences per batch whose early predictions obtained from an intermediate layer have the largest entropies instead of MLM losses. Prediction entropy has been used a measure of model uncertainty for active learning; see for e.g., Ren et al. (2021); Gal et al. (2017). So our proposal of entropy-based SIFT may also be of interest in active learning with large-scale models.

It is worth mentioning that in our implementation, the forward propagation for selecting samples in SIFT is implemented naively before full forward and backward propagation for the update step; one could potentially make this more efficient. However, even without any optimization of the initial forward propagation for sample selection, we obtain significant gains as we discuss later. For the experiments with SIFT, we do not perform any sample filtering for the first 20K steps; this is to provide an initial warm-up period and is aligned with the suggestion of Kawaguchi & Lu (2019) for GSGD. The total number of update steps for each algorithm is 800K. We use the hyper-parameters suggested in the pretraining part of the original BERT paper (Devlin et al., 2018); we defer these details to Appendix G.

We would like to point out that the goal of our experiments is to only show the efficacy of early exiting for approximately selecting samples with large losses/entropies and *not* propose a SOTA sample selection scheme.

[5] This is the same idea as OSGD (Kawaguchi, 2016) and OHEM (Shrivastava et al., 2016).

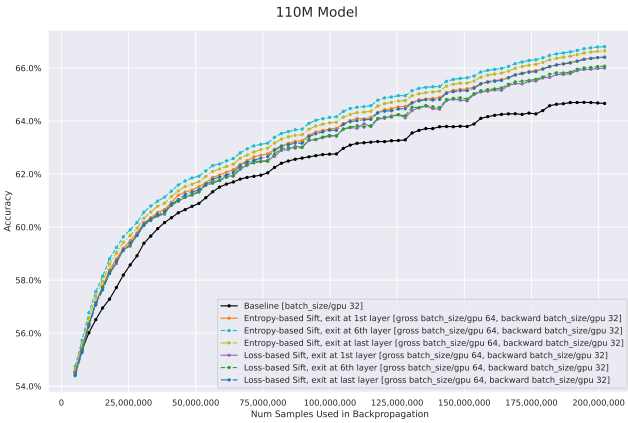

Figure 1. **Validation accuracy vs. backpropagation sample complexity.** In terms of performance, entropy-based SIFT > loss-based SIFT > baseline. All the accuracies are listed in Table 1 but for quick reference, loss-based and entropy-based SIFT with exit at the first layer are better than the baseline by 1.33% and 1.75%, respectively. For loss-based SIFT, exit at the last layer has the best performance while for entropy-based SIFT, exit at the sixth layer has the best performance.

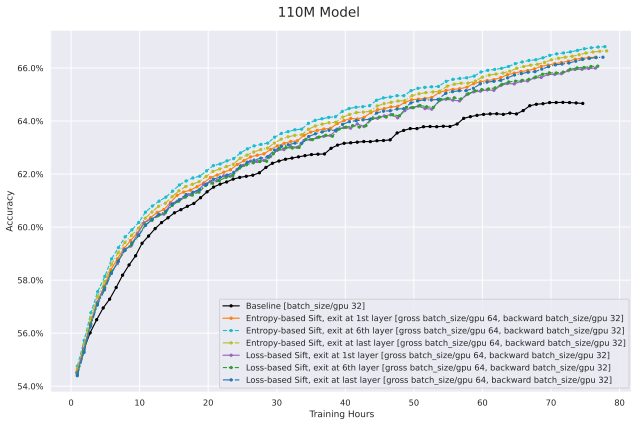

Figure 2. **Validation accuracy vs. wall-clock time.** The observations and trends are the same as Figure 1; please see the discussion therein.

Results. Figures 1 and 2 show the performance of SIFT with exits at the first, sixth and last layers[6] and baseline w.r.t. the number of samples used for backpropagation (i.e., backpropagation sample complexity) and the number of training hours, respectively. We report the validation accuracy at the last step with early exit at all the layers we considered in Table 1. Please see the figure and table captions for detailed discussion but the key takeaways are as follows:

- SIFT is better than the baseline in terms of backpropagation sample complexity as well as wall-clock time.
- Entropy-based SIFT does better than loss-based SIFT.
- For entropy-based SIFT, exit at the sixth layer has the best performance while for loss-based SIFT, exit at the last layer has the best performance.

In Table 6 (Appendix G), we report the total time spent in the forward pass for selecting samples in SIFT as a function of the layer number.

Overall our results here show that SIFT can yield significant gains for BERT pretraining.

[6]We do not show the plots for exits at the second and third layers here to avoid congestion; we report the validation accuracies of all these layers in Table 1.

Table 1. **Validation accuracy at the last step.** For loss-based SIFT, exit at the last layer has the best performance while the performance of other layers is nearly the same. However for entropy-based SIFT, exit at the sixth layer has the best performance.

Algorithm	SIFT Layer #	Val. Accuracy
Baseline	N/A	0.6466
Loss-based SIFT	1	0.6599
Loss-based SIFT	2	0.6597
Loss-based SIFT	3	0.6604
Loss-based SIFT	6	0.6607
Loss-based SIFT	12	0.6640
Entropy-based SIFT	1	0.6641
Entropy-based SIFT	2	0.6672
Entropy-based SIFT	3	0.6678
Entropy-based SIFT	6	0.6680
Entropy-based SIFT	12	0.6664

7.2. ResNet-50

Here we consider training a slightly modified version of ResNet-50 on CIFAR-100 and Food-101 (Bossard et al., 2014) which is a harder dataset than CIFAR-100 consisting of 101 classes. The modification has been made to make early exiting feasible in ResNet-50; we describe the modification and early exit details in Appendix H. Importantly, here we will *tune the learning rates* for both SIFT and the baseline (i.e., the vanilla approach with no sample filtering). We use the one-cycle learning rate schedule proposed for *fast training* in (Smith & Topin, 2019) and available in PyTorch (`https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html`). Specifically, we tune the maximum learning rate ("max_lr") for the schedule and thus show results with different learning

rates. We use default values for all other hyper-parameters of the schedule. We consider a scenario with limited training budget where we can train each algorithm for 100 epochs. Training is done with the standard cross-entropy loss function. Other empirical details are mentioned in Appendix H.

Tables 2 and 4 show the comparisons of SIFT and baseline with the *same training time* (analogous to Figure 2), whereas Tables 3 and 5 show comparisons with the same *backpropagation sample complexity or number of gradient updates* (analogous to Figure 1) for CIFAR-100 and Food-101, respectively. In these experiments, we saw that loss-based sampling worked better than entropy-based sampling; so we only report the results for loss-based sampling here.

Please refer to the table captions for detailed discussion but in summary, these results show that **SIFT is better even if we tune the learning rate**. Overall our results here show that SIFT can yield significant gains in training a ResNet-50 (appropriately modified to make early exiting feasible) from scratch.

Table 2. **CIFAR-100 with same training time.** We run the baseline for 100 epochs and SIFT with early exit and SIFT with last layer exit *till the time it takes to run 100 epochs of the baseline* (analogous to Figure 2) with different learning rates. The corresponding *test accuracies* are reported. The best test accuracy of each method is in bold font. So with the *same training time*, the **best test accuracy of SIFT with early exit** $>$ **best test accuracy of SIFT with last layer exit** $>$ **best test accuracy of baseline**.

max_lr	Baseline	SIFT w/ early exit	SIFT w/ last layer exit
$5e-2$	67.68	66.64	63.92
$1e-2$	**69.04**	**73.45**	70.13
$5e-3$	69.02	73.26	**72.91**
$1e-3$	58.54	66.85	71.09

Table 3. **CIFAR-100 with same backpropagation sample complexity.** We run all algorithms for 100 epochs so that the *backpropagation sample complexity (i.e., # of gradient updates) is the same for all algorithms* (analogous to Figure 1) with different learning rates. The corresponding *test accuracies* are reported. The best test accuracy of each method is in bold font. So with the *same gradient complexity*, the **best test accuracy of SIFT with last layer exit** $>$ **best test accuracy of SIFT with early exit** $>$ **best test accuracy of baseline**.

max_lr	Baseline	SIFT w/ early exit	SIFT w/ last layer exit
$5e-2$	67.68	68.16	68.12
$1e-2$	**69.04**	**75.41**	74.48
$5e-3$	69.02	75.06	**76.93**
$1e-3$	58.54	68.35	75.14

Table 4. **Food-101 with same training time.** All the details are the same as in Table 2 above. '–' indicates non-convergence. Here, with the *same training time*, the **best test accuracy of SIFT with last layer exit** $>$ **best test accuracy of SIFT with early exit** $>$ **best test accuracy of baseline**.

max_lr	Baseline	SIFT w/ early exit	SIFT w/ last layer exit
$5e-2$	57.50	–	–
$1e-2$	**59.52**	**64.03**	64.01
$5e-3$	58.88	63.23	**66.47**
$1e-3$	44.04	59.29	63.52

Table 5. **Food-101 with same backpropagation sample complexity.** All the details are the same as in Table 3 above. '–' indicates non-convergence. In this case, with the *same gradient complexity*, the **best test accuracy of SIFT with last layer exit** $>$ **best test accuracy of SIFT with early exit** $>$ **best test accuracy of baseline**.

max_lr	Baseline	SIFT w/ early exit	SIFT w/ last layer exit
$5e-2$	57.50	–	–
$1e-2$	**59.52**	**64.04**	64.19
$5e-3$	58.88	63.22	**66.64**
$1e-3$	44.04	59.27	63.68

8. Conclusion and Limitations

In this work, we theoretically characterized the benefits (as well as limitations) of the greedy approach of selecting samples with large approximate losses instead of exact losses. We also showed the promise of early exiting in speeding up the training of a transformer model.

We will mention some limitations in our current work which we hope to explore and address in future work. As we mentioned, we did not pipeline the early exit forward propagation step for sample selection with the full forward and backward propagation steps for model update in our current implementation; doing so can yield bigger gains. In our current work, we have only shown the efficacy of SIFT on BERT and a modified version of ResNet. In the future, we hope to test SIFT on much larger transformer models. On the theory side, our current convergence result is for convex functions; we would like to derive a similar result for non-convex functions too.

Acknowledgements

The authors are grateful to anonymous reviewers for their feedback which helped in improving this manuscript.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Bertsekas, D. P. et al. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010(1-38):3, 2011.

Borsos, Z., Krause, A., and Levy, K. Y. Online variance reduction for stochastic optimization. *arXiv preprint arXiv:1802.04715*, 2018.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pp. 446–461. Springer, 2014.

Bulatov, Y. Mathematics stack exchange: Proving $1 - \exp(-4x^2/\pi) \geq \mathrm{erf}(x)^2$, https://math.stackexchange.com/questions/6908/proving-1-exp-4x2-pi-ge-texterfx2.

Chuang, Y.-S., Xie, Y., Luo, H., Kim, Y., Glass, J., and He, P. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv: 2309.03883*, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Fan, Y., Lyu, S., Ying, Y., and Hu, B. Learning with average top-k loss. In *Advances in neural information processing systems*, pp. 497–505, 2017.

Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. In *International conference on machine learning*, pp. 1183–1192. PMLR, 2017.

Gurbuzbalaban, M., Ozdaglar, A., and Parrilo, P. A. Convergence rate of incremental gradient and incremental newton methods. *SIAM Journal on Optimization*, 29(4): 2542–2565, 2019.

Gurbuzbalaban, M., Ozdaglar, A., and Parrilo, P. A. Why random reshuffling beats stochastic gradient descent. *arXiv preprint arXiv:1510.08560*, 2022.

Haochen, J. and Sra, S. Random shuffling beats sgd after finite epochs. In *International Conference on Machine Learning*, pp. 2624–2633. PMLR, 2019.

Jiang, A. H., Wong, D. L.-K., Zhou, G., Andersen, D. G., Dean, J., Ganger, G. R., Joshi, G., Kaminksy, M., Kozuch, M., Lipton, Z. C., et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

Johnson, T. B. and Guestrin, C. Training deep models faster with robust, approximate importance sampling. In *Advances in Neural Information Processing Systems*, pp. 7265–7275, 2018.

Kamath, G. Bounds on the expectation of the maximum of samples from a gaussian. http://www.gautamkamath.com/writings/gaussian_max.pdf.

Katharopoulos, A. and Fleuret, F. Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*, 2017.

Katharopoulos, A. and Fleuret, F. Not all samples are created equal: Deep learning with importance sampling. *arXiv preprint arXiv:1803.00942*, 2018.

Kawaguchi, K. Deep learning without poor local minima. *Advances in neural information processing systems*, 29, 2016.

Kawaguchi, K. and Lu, H. Ordered sgd: A new stochastic optimization framework for empirical risk minimization. *arXiv preprint arXiv:1907.04371*, 2019.

Kim, T.-H. and Choi, J. Screenernet: Learning self-paced curriculum for deep neural networks. *arXiv preprint arXiv:1801.00904*, 2018.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

Loshchilov, I. and Hutter, F. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

Lu, Y., Meng, S. Y., and De Sa, C. A general analysis of example-selection for stochastic gradient descent. In *International Conference on Learning Representations (ICLR)*, volume 10, 2022.

Mishchenko, K., Khaled, A., and Richtárik, P. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33: 17309–17320, 2020.

Mohtashami, A., Stich, S., and Jaggi, M. Characterizing & finding good data orderings for fast convergence of sequential gradient methods. *arXiv preprint arXiv:2202.01838*, 2022.

Needell, D., Ward, R., and Srebro, N. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in neural information processing systems*, pp. 1017–1025, 2014.

Nesterov, Y. et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

Pennington, J. and Bahri, Y. Geometry of neural network loss surfaces via random matrix theory. In *International conference on machine learning*, pp. 2798–2806. PMLR, 2017.

Recht, B. and Ré, C. Toward a noncommutative arithmetic-geometric mean inequality: Conjectures, case-studies, and consequences. In *Conference on Learning Theory*, pp. 11–1. JMLR Workshop and Conference Proceedings, 2012.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Rotem, D., Hassid, M., Mamou, J., and Schwartz, R. Finding the sweet spot: Analysis and improvement of adaptive inference in low resource settings. *Annual Meeting of the Association for Computational Linguistics*, 2023. doi: 10.48550/arXiv.2306.02307.

Safran, I. and Shamir, O. How good is sgd with random shuffling? In *Conference on Learning Theory*, pp. 3250–3284. PMLR, 2020.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V., Tay, Y., and Metzler, D. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Schwartz, R., Stanovsky, G., Swayamdipta, S., Dodge, J., and Smith, N. A. The right tool for the job: Matching model and instance complexities. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6640–6651, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.593. URL https://aclanthology.org/2020.acl-main.593.

Shrivastava, A., Gupta, A., and Girshick, R. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 761–769, 2016.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pp. 369–386. SPIE, 2019.

Stich, S. U., Raj, A., and Jaggi, M. Safe adaptive importance sampling. In *Advances in Neural Information Processing Systems*, pp. 4381–4391, 2017.

Teerapittayanon, S., McDanel, B., and Kung, H. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016. doi: 10.1109/ICPR.2016.7900006.

Tsvetkov, Y., Faruqui, M., Ling, W., MacWhinney, B., and Dyer, C. Learning the curriculum with bayesian optimization for task-specific word representation learning. *arXiv preprint arXiv:1605.03852*, 2016.

Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. DeeBERT: Dynamic early exiting for accelerating BERT inference. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2246–2251, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.204. URL https://aclanthology.org/2020.acl-main.204.

Xin, J., Tang, R., Yu, Y., and Lin, J. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In Merlo, P., Tiedemann, J., and Tsarfaty, R. (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 91–104, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.8. URL https://aclanthology.org/2021.eacl-main.8.

Zhang, J., Yu, H.-f., and Dhillon, I. S. Autoassist: A framework to accelerate training of deep neural networks. *arXiv preprint arXiv:1905.03381*, 2019.

Zhao, P. and Zhang, T. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pp. 1–9, 2015.

Zhu, W. LeeBERT: Learned early exit for BERT with cross-level optimization. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2968–2980, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.231. URL https://aclanthology.org/2021.acl-long.231.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

Appendix

A. Other Related Work

Sample reordering. Another related line of work attempts to improve the way/order in which samples are presented while training. Bengio et al. (2009) propose curriculum learning wherein the key idea is to present easier examples before harder ones but this requires prior information about the training set. Several improved modifications of this idea are out there (Tsvetkov et al., 2016; Kumar et al., 2010; Jiang et al., 2017; Kim & Choi, 2018; Zhang et al., 2019; Jiang et al., 2019). There is also a long line of papers that theoretically analyze conventional sample ordering schemes (such as shuffle once, random reshuffling, etc.) as well as improved sample ordering schemes (Bertsekas et al., 2011; Recht & Ré, 2012; Gurbuzbalaban et al., 2019; 2022; Haochen & Sra, 2019; Safran & Shamir, 2020; Mishchenko et al., 2020; Lu et al., 2022; Mohtashami et al., 2022).

B. Some Preliminaries for the Proofs

Fact B.1 (Nesterov et al. (2018)). For an L-smooth function $g : \mathbb{R}^p \to \mathbb{R}$, $\|\nabla g(\boldsymbol{y})\|_2^2 \leq 2L(g(\boldsymbol{y}) - \min_{\boldsymbol{z}\in\mathbb{R}^p} g(\boldsymbol{z}))\ \forall\ \boldsymbol{y} \in \mathbb{R}^p$.

Fact B.2 (**Cantelli's inequality**). For any random variable Y and any $t > 0$:

$$\mathbb{P}\Big(Y - \mathbb{E}[Y] \geq t\Big) \leq \frac{\text{Var}(Y)}{\text{Var}(Y) + t^2}.$$

Fact B.3 (Bulatov). We have:

$$\text{erf}(t) \leq \sqrt{1 - \exp\Big(-\frac{4t^2}{\pi}\Big)}.$$

The above bound has been also used by Kamath.

Fact B.4. It holds that:

$$\text{erfc}(t) \leq 2\exp\Big(-\frac{t^2}{2}\Big).$$

Proof. Let $Z \sim \mathcal{N}(0, 1)$. It can be checked that $\text{erf}(t) = 2\mathbb{P}(Z \in (0, t))$. Thus,

$$\text{erfc}(t) = 1 - \text{erf}(t) = 2\Big(\frac{1}{2} - \mathbb{P}\big(Z \in (0, t)\big)\Big) = 2\mathbb{P}(Z \geq t). \tag{15}$$

But using the Gaussian tail bound, we have $\mathbb{P}(Z \geq t) \leq e^{-t^2/2}$. Using this in Equation (15) gives us the desired result. □

Fact B.5. For any $t > 0$:

$$\int_0^\infty \exp\Big(-\frac{y^2}{2t^2}\Big)dy = \sqrt{\frac{\pi}{2}}t.$$

Proof. Let $Z \sim \mathcal{N}(0, t^2)$. We have:

$$\int_0^\infty \exp\Big(-\frac{y^2}{2t^2}\Big)dy = \sqrt{2\pi t^2}\underbrace{\Bigg(\frac{1}{\sqrt{2\pi t^2}}\int_0^\infty \exp\Big(-\frac{y^2}{2t^2}\Big)dy\Bigg)}_{=\mathbb{P}(Z>0)=\frac{1}{2}} = \sqrt{\frac{\pi}{2}}t. \tag{16}$$

□

C. Proof of Theorem 5.7

Proof. Consider some $\boldsymbol{w}^* \in \Phi_F$. With a constant step-size, say η, we have for any iteration k:

$$\mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\|^2\right] = \|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - 2\eta\langle \mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\nabla f(\boldsymbol{w}_k, \widehat{\boldsymbol{x}}_k)\right], \boldsymbol{w}_k - \boldsymbol{w}^*\rangle + \eta^2 \mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\nabla f(\boldsymbol{w}_k, \widehat{\boldsymbol{x}}_k)\|^2\right]. \tag{17}$$

Using Definition 5.6 in Equation (17), we get:

$$\mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\|^2\right] = \|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - 2\eta\langle \nabla \widehat{F}_R(\boldsymbol{w}_k), \boldsymbol{w}_k - \boldsymbol{w}^*\rangle + \eta^2 \mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\nabla f(\boldsymbol{w}_k, \widehat{\boldsymbol{x}}_k)\|^2\right]. \tag{18}$$

Using the convexity of $f(\boldsymbol{w}, \boldsymbol{x})$ w.r.t. $\boldsymbol{w}$ (Assumption 5.2) and the fact that pointwise maximum of convex functions is also convex, we conclude that $\widehat{F}_R(\boldsymbol{w})$ is convex. Thus:

$$\langle \nabla \widehat{F}_R(\boldsymbol{w}_k), \boldsymbol{w}_k - \boldsymbol{w}^*\rangle \geq \widehat{F}_R(\boldsymbol{w}_k) - \widehat{F}_R(\boldsymbol{w}^*) \geq \widehat{F}_R(\boldsymbol{w}_k) - \Delta_R, \tag{19}$$

where the last step follows from the fact that $\sup_{\boldsymbol{w}^* \in \Phi_F} \widehat{F}_R(\boldsymbol{w}^*) \leq \Delta_R$.

Further, using Assumption 5.3, Fact B.1 and Assumption 5.4, we get:

$$\mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\nabla f(\boldsymbol{w}_k, \widehat{\boldsymbol{x}}_k)\|^2\right] \leq 2L\mathbb{E}_{\mathcal{S}_k^{(R)}}\left[f(\boldsymbol{w}_k, \widehat{\boldsymbol{x}}_k)\right] \leq 2L\widehat{F}_R(\boldsymbol{w}_k). \tag{20}$$

Plugging Equation (19) and Equation (20) into Equation (17), we get:

$$\mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\|^2\right] = \|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - 2\eta\big(1 - \eta L\big)\widehat{F}_R(\boldsymbol{w}_k) + 2\eta\Delta_R. \tag{21}$$

Let us impose $\eta < \frac{1}{L}$ so that $1 - \eta L > 0$. From Definition 5.6, we have $\widehat{F}_R(\boldsymbol{w}_k) \geq \rho_R^* F(\boldsymbol{w}_k)$. Using this above, we get:

$$\mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\|^2\right] = \|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - 2\rho_R^*\eta\big(1 - \eta L\big)F(\boldsymbol{w}_k) + 2\eta\Delta_R. \tag{22}$$

After some rearrangement, we get:

$$F(\boldsymbol{w}_k) \leq \frac{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - \mathbb{E}_{\mathcal{S}_k^{(R)}}\left[\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\|^2\right]}{2\rho_R^*\eta\big(1 - \eta L\big)} + \frac{\Delta_R}{\rho_R^*\big(1 - \eta L\big)}. \tag{23}$$

Next, we sum the above for all $k \in \{0, \ldots, K-1\}$ while taking expectation throughout. After dividing both sides of the resultant inequality by $\frac{1}{K}$, we get:

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[F(\boldsymbol{w}_k)] \leq \frac{\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2}{2\rho_R^*\eta\big(1 - \eta L\big)K} + \frac{\Delta_R}{\rho_R^*\big(1 - \eta L\big)}. \tag{24}$$

Let $\overline{\boldsymbol{w}}_K = \frac{1}{K}\sum_{k=0}^{K-1} \boldsymbol{w}_k$. Applying Jensen's inequality, we get $\mathbb{E}[F(\overline{\boldsymbol{w}}_K)] \leq \frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[F(\boldsymbol{w}_k)]$. Using this in Equation (24) gives us:

$$\mathbb{E}\big[F\big(\overline{\boldsymbol{w}}_K\big)\big] \leq \frac{\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2}{2\rho_R^*\eta\big(1 - \eta L\big)K} + \frac{\Delta_R}{\rho_R^*\big(1 - \eta L\big)}. \tag{25}$$

Observe that this analysis holds for any $\boldsymbol{w}^* \in \Phi_F$, including the one that is closest to $\boldsymbol{w}_0$. Using this in the above bound gives us:

$$\mathbb{E}\big[F\big(\overline{\boldsymbol{w}}_K\big)\big] \leq \frac{D_0^2}{2\rho_R^*\eta\big(1 - \eta L\big)K} + \frac{\Delta_R}{\rho_R^*\big(1 - \eta L\big)}, \tag{26}$$

where $D_0 := \min_{\boldsymbol{w}^* \in \Phi_F} \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|$. □

D. Proof of Theorem 5.15

Proof. We have for $j \in \{1, 2\}$, $\widetilde{f}(\boldsymbol{w}, \boldsymbol{x}^{(j)}) = f(\boldsymbol{w}, \boldsymbol{x}^{(j)}) \exp\Big(\mu(\boldsymbol{w}) + \sigma\zeta(\boldsymbol{w}, \boldsymbol{x}^{(j)})\Big)$, where $\zeta(\boldsymbol{w}, \boldsymbol{x}^{(j)})$ is i.i.d. random noise with mean 0 and variance 1. For brevity of notation, let:

$$\mu = \mu(\boldsymbol{w}) \text{ and } \widetilde{f}^{(j)} = \widetilde{f}(\boldsymbol{w}, \boldsymbol{x}^{(j)}),\ f^{(j)} = f(\boldsymbol{w}, \boldsymbol{x}^{(j)}) \text{ and } \zeta^{(j)} = \zeta(\boldsymbol{w}, \boldsymbol{x}^{(j)}) \text{ for } j \in \{1, 2\}.$$

Without loss of generality, let $f^{(1)} \geq f^{(2)}$.

Let $j^* = \arg\max_{j \in [2]} \widetilde{f}^{(j)}$. Let us consider the case of $j^* = 1$. This happens when $f^{(2)} \exp(\mu + \sigma\zeta^{(2)}) \leq f^{(1)} \exp(\mu + \sigma\zeta^{(1)})$; this is equivalent to:

$$\sigma(\zeta^{(2)} - \zeta^{(1)}) \leq \log \frac{f^{(1)}}{f^{(2)}}. \tag{27}$$

Let $Z = \zeta^{(2)} - \zeta^{(1)}$. Note that Z is random variable with mean 0 and variance 2. As per the above discussion, we have:

$$\mathbb{P}\big(j^* = 1\big) = \mathbb{P}\left(Z \leq \frac{1}{\sigma} \log \frac{f^{(1)}}{f^{(2)}}\right) = 1 - \mathbb{P}\left(Z > \frac{1}{\sigma} \log \frac{f^{(1)}}{f^{(2)}}\right). \tag{28}$$

Thus,

$$\mathbb{P}\big(j^* = 2\big) = \mathbb{P}\left(Z > \frac{1}{\sigma} \log \frac{f^{(1)}}{f^{(2)}}\right). \tag{29}$$

Then, we have:

$$\begin{aligned}
\mathbb{E}_{\{\zeta^{(1)}, \zeta^{(2)}\}}\Big[f^{(j^*)}\Big] &= f^{(1)}\mathbb{P}\big(j^* = 1\big) + f^{(2)}\mathbb{P}\big(j^* = 2\big) && (30)\\
&= f^{(1)}\left(1 - \mathbb{P}\left(Z > \frac{1}{\sigma} \log \frac{f^{(1)}}{f^{(2)}}\right)\right) + f^{(2)}\mathbb{P}\left(Z > \frac{1}{\sigma} \log \frac{f^{(1)}}{f^{(2)}}\right) && (31)\\
&= f^{(1)} - \big(f^{(1)} - f^{(2)}\big)\mathbb{P}\left(Z > \frac{1}{\sigma} \log \frac{f^{(1)}}{f^{(2)}}\right). && (32)
\end{aligned}$$

Since $\mathbb{E}[Z] = 0$ and $\text{Var}(Z) = 2$, using Cantelli's inequality (Fact B.2), we have:

$$\mathbb{P}\left(Z > \frac{1}{\sigma} \log \frac{f^{(1)}}{f^{(2)}}\right) \leq \frac{2}{2 + \frac{1}{\sigma^2} \log^2 \frac{f^{(1)}}{f^{(2)}}}. \tag{33}$$

Using this in Equation (32), we get:

$$\mathbb{E}_{\{\zeta^{(1)}, \zeta^{(2)}\}}\Big[f^{(j^*)}\Big] \geq f^{(1)} - \big(f^{(1)} - f^{(2)}\big)\left(\frac{2}{2 + \frac{1}{\sigma^2} \log^2 \frac{f^{(1)}}{f^{(2)}}}\right). \tag{34}$$

Let us define the function $h(t; \sigma) := \frac{2(1 - e^{-t})}{2 + \frac{t^2}{\sigma^2}}$ for $t \geq 0$. Then, note that:

$$\big(f^{(1)} - f^{(2)}\big)\left(\frac{2}{2 + \frac{1}{\sigma^2} \log^2 \frac{f^{(1)}}{f^{(2)}}}\right) = f^{(1)} \times h\left(\log \frac{f^{(1)}}{f^{(2)}}; \sigma\right).$$

From Lemma D.1, we have $h(t; \sigma) \leq 0.72\big(1 - e^{-\sqrt{2}\sigma}\big)$ for all $t \geq 0$, when $\sigma \leq \frac{1}{2\sqrt{2}}$. Using this above gives us:

$$\big(f^{(1)} - f^{(2)}\big)\left(\frac{2}{2 + \frac{1}{\sigma^2} \log^2 \frac{f^{(1)}}{f^{(2)}}}\right) \leq 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big) f^{(1)}, \tag{35}$$

when $\sigma \leq \frac{1}{2\sqrt{2}}$. Plugging this into Equation (34) gives us:

$$\mathbb{E}_{\{\zeta^{(1)},\zeta^{(2)}\}}\Big[f^{(j^*)}\Big] \geq \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big) f^{(1)}. \tag{36}$$

Rewriting the above in terms of the full notation, we obtain:

$$\mathbb{E}_{\{\zeta(\boldsymbol{w},\boldsymbol{x}^{(j)})\}_{j=1}^2}\Big[f\big(\boldsymbol{w},\boldsymbol{x}^{(j^*)}\big)\Big| j^* = \arg\max_{j\in[2]} \widetilde{f}(\boldsymbol{w},\boldsymbol{x}^{(j)})\Big] \geq \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big) \max_{j\in[2]} f(\boldsymbol{w},\boldsymbol{x}^{(j)}). \tag{37}$$

Thus:

$$\widehat{F}_{2,\text{approx}}(\boldsymbol{w}) \geq \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big)\mathbb{E}_{\{\boldsymbol{x}^{(j)}\}_{j=1}^2}\Big[\max_{j\in[2]} f(\boldsymbol{w},\boldsymbol{x}^{(j)})\Big] = \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big)\widehat{F}_2(\boldsymbol{w}). \tag{38}$$

Hence, we also have:

$$\rho_{2,\text{approx}}(\boldsymbol{w}) = \frac{\widehat{F}_{2,\text{approx}}(\boldsymbol{w})}{F(\boldsymbol{w})} \geq \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big)\frac{\widehat{F}_2(\boldsymbol{w})}{F(\boldsymbol{w})} = \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big)\rho_2(\boldsymbol{w}), \tag{39}$$

and:

$$\rho^*_{2,\text{approx}} = \inf_{\boldsymbol{w}\notin\Phi_F} \rho_{2,\text{approx}}(\boldsymbol{w}) \geq \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big) \inf_{\boldsymbol{w}\notin\Phi_F} \rho_2(\boldsymbol{w}) = \Big(1 - 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big)\Big)\rho^*_2. \tag{40}$$

□

Lemma D.1. *Consider the function* $h(t;\sigma) := \frac{2(1-e^{-t})}{\left(2+\frac{t^2}{\sigma^2}\right)}$ *for* $t \geq 0$ *and* $\sigma \leq \frac{1}{2\sqrt{2}}$. *Then, we have:*

$$\max_{t\geq 0} h(t;\sigma) \leq 0.72\Big(1 - e^{-\sqrt{2}\sigma}\Big).$$

Proof. Let $t^* = \arg\max_{t\geq 0} h(t;\sigma)$. Setting $\frac{dh}{dt}\big|_{t=t^*} = 0$, we obtain the following equation:

$$e^{t^*} = 1 + \frac{\sigma^2}{t^*} + \frac{t^*}{2}. \tag{41}$$

Using the series expansion of the exponential function above, we get:

$$1 + \sum_{j=1}^{\infty} \frac{(t^*)^j}{j!} = 1 + \frac{\sigma^2}{t^*} + \frac{t^*}{2} \implies (t^*)^2 \underbrace{\Big(1 + 2\sum_{j=1}^{\infty} \frac{(t^*)^j}{(j+1)!}\Big)}_{:=\nu(t^*)} = 2\sigma^2. \tag{42}$$

Since $\nu(t^*) \geq 1$, we conclude that:

$$t^* \leq \sqrt{2}\sigma. \tag{43}$$

Further, since $\sigma \leq \frac{1}{2\sqrt{2}}$, we also have $t^* \leq \frac{1}{2}$. Note that $\nu(t^*)$ is an increasing function of t^*. So $\nu(t^*) \leq \nu\big(\frac{1}{2}\big)$. Also, by using the series expansion of the exponential function, it can be verified that $\nu(t^*) = \frac{2(e^{t^*}-1)}{t^*} - 1$. Thus, we have:

$$\nu(t^*) \leq \nu\Big(\frac{1}{2}\Big) \leq 4e^{1/2} - 5. \tag{44}$$

Using this in Equation (42), we get:

$$t^* \geq \frac{\sqrt{2}\sigma}{\sqrt{4e^{1/2} - 5}} \geq 0.62\big(\sqrt{2}\sigma\big). \tag{45}$$

Combining the bounds of Equation (43) and Equation (45), we deduce that $t^* = c\big(\sqrt{2}\sigma\big)$, where $c \in [0.62, 1]$. Thus, $(1 - e^{-t^*}) \leq 1 - e^{-\sqrt{2}\sigma}$ and $\frac{2}{2+\frac{t^2}{\sigma^2}} \leq \frac{2}{2(1+0.62^2)} \leq 0.72$. Using all of this, we obtain:

$$\max_{t\geq 0} h(t;\sigma) = h(t^*;\sigma) \leq 0.72\big(1 - e^{-\sqrt{2}\sigma}\big). \tag{46}$$

□

E. Proof of Theorem 5.18

We restate Theorem 5.18 before proving it.

Theorem E.1. *Suppose Assumption 5.17 holds. Then:*

$$\rho_R(\boldsymbol{w}) \geq \frac{\left(\varepsilon^2(\boldsymbol{w}) + \left(\frac{\pi}{2}\log\frac{R}{4\log R}\right)\delta^2(\boldsymbol{w}) + \sqrt{2\pi\log\frac{R}{4\log R}}\varepsilon(\boldsymbol{w})\delta(\boldsymbol{w})\right)\left(1-\frac{1}{R}\right)}{\varepsilon^2(\boldsymbol{w})+\delta^2(\boldsymbol{w})}.$$

Proof. Per Assumption 5.17, $\mathcal{M}(\boldsymbol{w},\boldsymbol{x}) - \mathcal{M}(\boldsymbol{w}^*,\boldsymbol{x}) \underset{\text{iid}}{\sim} \mathcal{N}\big(\varepsilon(\boldsymbol{w}), \delta^2(\boldsymbol{w})\big)$ for $\boldsymbol{w} \neq \boldsymbol{w}^*$. Clearly,

$$F(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}}\big[f(\boldsymbol{w},\boldsymbol{x})\big] = \varepsilon^2(\boldsymbol{w}) + \delta^2(\boldsymbol{w}), \tag{47}$$

and

$$\widehat{F}_R(\boldsymbol{w}) = \mathbb{E}_{\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(R)}\}}\left[\max_{\boldsymbol{x}\in\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(R)}\}} f(\boldsymbol{w},\boldsymbol{x})\right]. \tag{48}$$

For conciseness, we shall denote $\varepsilon(\boldsymbol{w})$ and $\delta(\boldsymbol{w})$ by just ε and δ, respectively. Also, let $Z_i = \mathcal{M}(\boldsymbol{w},\boldsymbol{x}^{(i)}) - \mathcal{M}(\boldsymbol{w}^*,\boldsymbol{x}^{(i)})$ for $i \in [R]$. As per our concise notation, note that each $Z_i \underset{\text{iid}}{\sim} \mathcal{N}(\varepsilon,\delta^2)$. Hence:

$$\widehat{F}_R(\boldsymbol{w}) = \mathbb{E}\Big[\max_{i\in[R]} Z_i^2\Big]. \tag{49}$$

We shall obtain a lower bound for $\mathbb{E}\big[\max_{i\in[R]} Z_i^2\big]$. Let $Y \sim \mathcal{N}(0,1)$. For $t > 0$, we have:

$$\begin{aligned}
\mathbb{P}\Big(\max_{i\in[R]}(Z_i-\varepsilon) \leq t\Big) = \mathbb{P}\Big(\cap_{i\in[R]}\,(Z_i-\varepsilon) \leq t\Big) &= \Big(\mathbb{P}\big(Y \leq t/\delta\big)\Big)^R && (50)\\
&= \Big(\frac{1}{2} + \frac{1}{2}\mathrm{erf}\Big(\frac{t}{\sqrt{2}\delta}\Big)\Big)^R && (51)\\
&\leq \left(\frac{1}{2} + \frac{1}{2}\sqrt{1-\exp\Big(-\frac{2t^2}{\pi\delta^2}\Big)}\right)^R && (52)\\
&\leq \left(\frac{1}{2} + \frac{1}{2}\left(1-\frac{1}{2}\exp\Big(-\frac{2t^2}{\pi\delta^2}\Big)\right)\right)^R && (53)\\
&= \left(1-\frac{1}{4}\exp\Big(-\frac{2t^2}{\pi\delta^2}\Big)\right)^R && (54)\\
&\leq \exp\Big(-\frac{R}{4}\exp\Big(-\frac{2t^2}{\pi\delta^2}\Big)\Big). && (55)
\end{aligned}$$

In Equation (51), the error function is as defined in Equation (2). Equation (52) follows from Fact B.3. Equation (53) is obtained by using the fact that $\sqrt{1-a} \leq 1-\frac{a}{2}$ for all $a \in [0,1]$. Equation (55) follows from the fact that $1-a \leq e^{-a}$ for all $a \in \mathbb{R}$. So, $\max_{i\in[R]}(Z_i-\varepsilon) \leq t$ with a probability of at most $\exp\Big(-\frac{R}{4}\exp\Big(-\frac{2t^2}{\pi\delta^2}\Big)\Big)$. Let us choose $t = \delta\sqrt{\frac{\pi}{2}\log\frac{R}{4\log R}}$. With this choice, we get:

$$\max_{i\in[R]} Z_i \geq \varepsilon + \delta\sqrt{\frac{\pi}{2}\log\frac{R}{4\log R}} \;\text{ w.p. }\; \geq 1-\frac{1}{R}.$$

Thus,

$$\mathbb{E}\Big[\max_{i\in[R]} Z_i^2\Big] \geq \left(\varepsilon + \delta\sqrt{\frac{\pi}{2}\log\frac{R}{4\log R}}\right)^2\Big(1-\frac{1}{R}\Big) = \left(\varepsilon^2 + \frac{\pi}{2}\log\frac{R}{4\log R}\delta^2 + \sqrt{2\pi\log\frac{R}{4\log R}}\varepsilon\delta\right)\Big(1-\frac{1}{R}\Big). \tag{56}$$

Plugging this into Equation (49) and replacing ε and δ with their complete notations, i.e., $\varepsilon(\boldsymbol{w})$ and $\delta(\boldsymbol{w})$, we get:

$$\widehat{F}_R(\boldsymbol{w}) \geq \left(\varepsilon^2(\boldsymbol{w}) + \Big(\frac{\pi}{2}\log\frac{R}{4\log R}\Big)\delta^2(\boldsymbol{w}) + \sqrt{2\pi\log\frac{R}{4\log R}}\varepsilon(\boldsymbol{w})\delta(\boldsymbol{w})\right)\Big(1-\frac{1}{R}\Big). \tag{57}$$

Hence:

$$\rho_R(\boldsymbol{w}) = \frac{\widehat{F}_R(\boldsymbol{w})}{F(\boldsymbol{w})} \geq \frac{\left(\varepsilon^2(\boldsymbol{w}) + \Big(\frac{\pi}{2}\log\frac{R}{4\log R}\Big)\delta^2(\boldsymbol{w}) + \sqrt{2\pi\log\frac{R}{4\log R}}\varepsilon(\boldsymbol{w})\delta(\boldsymbol{w})\right)\Big(1-\frac{1}{R}\Big)}{\varepsilon^2(\boldsymbol{w})+\delta^2(\boldsymbol{w})}. \tag{58}$$

□

F. Proof of Theorem 6.1

Proof. For $i \in \{1,2\}$, we have:

$$\hat{y}_j^{(i)} = \mathrm{sig}\big(\boldsymbol{\theta}^\top \boldsymbol{A}_j \boldsymbol{x}^{(i)}\big) \text{ and } \hat{y}^{(i)} = \hat{y}_k^{(i)} = \mathrm{sig}\big(\boldsymbol{\theta}^\top \boldsymbol{B}_j \boldsymbol{A}_j \boldsymbol{x}^{(i)}\big). \tag{59}$$

For the i.i.d. samples $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ with ground truth labels $y^{(1)}$ and $y^{(2)}$, recall that $\ell_j^{(1)}$ and $\ell_j^{(2)}$ are the corresponding cross-entropy losses of the early predictions at the jth layer, i.e.,

$$\ell_j^{(i)} = -y^{(i)}\log\big(\hat{y}_j^{(i)}\big) - (1-y^{(i)})\log\big(1-\hat{y}_j^{(i)}\big) \text{ for } i \in \{1,2\}. \tag{60}$$

We are interested in:

$$p_j := \mathbb{P}_{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)}}\Big(\text{arg max}_{i\in[1,2]}\ell_j^{(i)} = \text{arg max}_{i\in[1,2]}\ell_k^{(i)}\Big).$$

Using symmetry, we get:

$$p_j := \mathbb{P}_{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)}}\Big(\ell_j^{(1)} \geq \ell_j^{(2)}\Big|\ell_k^{(1)} \geq \ell_k^{(2)}\Big). \tag{61}$$

Note that $p_k = 1$. Henceforth, we shall drop the subscript $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}$ for conciseness.

Let $\bar{y}^{(1)} = 2y^{(1)}-1$ and $\bar{y}^{(2)} = 2y^{(2)}-1$ be the centered ground truth labels (i.e., $\in \{-1,1\}$) of $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$, respectively. It can be verified that:

$$\ell_j^{(1)} \geq \ell_j^{(2)} \iff \boldsymbol{\theta}^\top \boldsymbol{A}_j\Big(\bar{y}^{(1)}\boldsymbol{x}^{(1)} - \bar{y}^{(2)}\boldsymbol{x}^{(2)}\Big) \geq 0 \text{ and } \ell_k^{(1)} \geq \ell_k^{(2)} \iff \boldsymbol{\theta}^\top \boldsymbol{B}_j\boldsymbol{A}_j\Big(\bar{y}^{(1)}\boldsymbol{x}^{(1)} - \bar{y}^{(2)}\boldsymbol{x}^{(2)}\Big) \geq 0. \tag{62}$$

Let $\boldsymbol{z} := \bar{y}^{(1)}\boldsymbol{x}^{(1)} - \bar{y}^{(2)}\boldsymbol{x}^{(2)}$. Since $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)} \underset{\text{iid}}{\sim} \mathcal{N}(\vec{0}_d, \mathrm{I}_d)$, we have that $\boldsymbol{z} \sim \mathcal{N}(\vec{0}_d, 2\mathrm{I}_d)$.

Using Equation (62) and the definition of $\boldsymbol{z}$ followed by the application of Bayes' theorem in Equation (61), we obtain:

$$p_j = \mathbb{P}\Big(\boldsymbol{\theta}^\top \boldsymbol{A}_j\boldsymbol{z} \geq 0\Big|\boldsymbol{\theta}^\top \boldsymbol{B}_j\boldsymbol{A}_j\boldsymbol{z} \geq 0\Big) = \frac{\mathbb{P}\big(\boldsymbol{\theta}^\top \boldsymbol{A}_j\boldsymbol{z} \geq 0, \boldsymbol{\theta}^\top \boldsymbol{B}_j\boldsymbol{A}_j\boldsymbol{z} \geq 0\big)}{\mathbb{P}\big(\boldsymbol{\theta}^\top \boldsymbol{B}_j\boldsymbol{A}_j\boldsymbol{z} \geq 0\big)}. \tag{63}$$

Since $\boldsymbol{z} \sim \mathcal{N}(\vec{0}_d, 2\mathrm{I}_d)$, $\mathbb{P}\big(\boldsymbol{\theta}^\top \boldsymbol{B}_j\boldsymbol{A}_j\boldsymbol{z} \geq 0\big) = \frac{1}{2}$. Using this above, we get:

$$p_j = 2\mathbb{P}\Big(\boldsymbol{\theta}^\top \boldsymbol{A}_j\boldsymbol{z} \geq 0, \boldsymbol{\theta}^\top \boldsymbol{B}_j\boldsymbol{A}_j\boldsymbol{z} \geq 0\Big). \tag{64}$$

For ease of notation, let $u_1 = \boldsymbol{\theta}^\top \boldsymbol{A}_j\boldsymbol{z}$ and $u_2 = \boldsymbol{\theta}^\top \boldsymbol{B}_j\boldsymbol{A}_j\boldsymbol{z}$ and $\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$. Note $\boldsymbol{u}$ is a multivariate Gaussian random variable with $\mathbb{E}[\boldsymbol{u}] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and

$$\mathbb{E}[\boldsymbol{u}\boldsymbol{u}^\top] := \boldsymbol{\Sigma} = 2\begin{bmatrix} \|\boldsymbol{A}_j^\top\boldsymbol{\theta}\|_2^2 & \big\langle \boldsymbol{A}_j^\top\boldsymbol{\theta}, \boldsymbol{A}_j^\top\boldsymbol{B}_j^\top\boldsymbol{\theta}\big\rangle \\ \big\langle \boldsymbol{A}_j^\top\boldsymbol{\theta}, \boldsymbol{A}_j^\top\boldsymbol{B}_j^\top\boldsymbol{\theta}\big\rangle & \|\boldsymbol{A}_j^\top\boldsymbol{B}_j^\top\boldsymbol{\theta}\|_2^2 \end{bmatrix}. \tag{65}$$

Let $\alpha_1 = \|\boldsymbol{A}_j^\top\boldsymbol{\theta}\|_2$, $\alpha_2 = \|\boldsymbol{A}_j^\top\boldsymbol{B}_j^\top\boldsymbol{\theta}\|_2$ and

$$\beta = \frac{\big\langle \boldsymbol{A}_j^\top\boldsymbol{\theta}, \boldsymbol{A}_j^\top\boldsymbol{B}_j^\top\boldsymbol{\theta}\big\rangle}{\|\boldsymbol{A}_j^\top\boldsymbol{\theta}\|_2\|\boldsymbol{A}_j^\top\boldsymbol{B}_j^\top\boldsymbol{\theta}\|_2}.$$

In the theorem statement, we shall denote β by β_j to indicate the dependence on the layer number j; we drop the subscript j here for conciseness. With this notation, we have:

$$\boldsymbol{\Sigma} = 2\begin{bmatrix} \alpha_1^2 & \beta\alpha_1\alpha_2 \\ \beta\alpha_1\alpha_2 & \alpha_2^2 \end{bmatrix}. \tag{66}$$

The pdf of $\boldsymbol{u}$ is therefore:

$$\phi(\boldsymbol{u}) = \frac{1}{2\pi\sqrt{\det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}\boldsymbol{u}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{u}\right) \tag{67}$$

$$= \frac{1}{4\pi\alpha_1\alpha_2\sqrt{1-\beta^2}} \exp\left(-\frac{1}{4(1-\beta^2)}\Big(\frac{u_1^2}{\alpha_1^2} - \frac{2\beta u_1 u_2}{\alpha_1\alpha_2} + \frac{u_2^2}{\alpha_2^2}\Big)\right). \tag{68}$$

Using this in Equation (64), we get:

$$p_j = 2\int_0^\infty \int_0^\infty \phi(\boldsymbol{u}) du_1 du_2 \tag{69}$$

$$= \frac{1}{2\pi\alpha_1\alpha_2\sqrt{1-\beta^2}} \int_0^\infty \int_0^\infty \exp\left(-\frac{1}{4(1-\beta^2)}\Big(\frac{u_1^2}{\alpha_1^2} - \frac{2\beta u_1 u_2}{\alpha_1\alpha_2} + \frac{u_2^2}{\alpha_2^2}\Big)\right) du_1 du_2 \tag{70}$$

$$= \frac{1}{2\pi\alpha_1\alpha_2\sqrt{1-\beta^2}} \int_0^\infty \exp\Big(-\frac{u_1^2}{4\alpha_1^2}\Big)\left(\int_0^\infty \exp\left(-\frac{\big(u_2 - \big(\frac{\beta\alpha_2}{\alpha_1}\big)u_1\big)^2}{4\alpha_2^2(1-\beta^2)}\right) du_2\right) du_1. \tag{71}$$

With some simple change of variables and some simplification, the above equation becomes:

$$p_j = \frac{1}{\pi\alpha_1}\int_0^\infty \exp\Big(-\frac{u_1^2}{4\alpha_1^2}\Big)\left(\int_{-\frac{\beta u_1}{2\sqrt{1-\beta^2}\alpha_1}}^\infty \exp(-t^2)dt\right) du_1 \tag{72}$$

$$= \frac{1}{2\sqrt{\pi}\alpha_1}\int_0^\infty \exp\Big(-\frac{u_1^2}{4\alpha_1^2}\Big)\left(\frac{2}{\sqrt{\pi}}\int_0^{\frac{\beta u_1}{2\sqrt{1-\beta^2}\alpha_1}} \exp(-t^2)dt + \frac{2}{\sqrt{\pi}}\int_0^\infty \exp(-t^2)dt\right) du_1 \tag{73}$$

$$= \frac{1}{2\sqrt{\pi}\alpha_1}\int_0^\infty \exp\Big(-\frac{u_1^2}{4\alpha_1^2}\Big)\left(\mathrm{erf}\Big(\frac{\beta u_1}{2\sqrt{1-\beta^2}\alpha_1}\Big) + \underbrace{\lim_{y\to\infty}\mathrm{erf}(y)}_{=1}\right) du_1 \tag{74}$$

$$= \frac{1}{2\sqrt{\pi}\alpha_1}\int_0^\infty \exp\Big(-\frac{u_1^2}{4\alpha_1^2}\Big)\mathrm{erf}\Big(\frac{\beta u_1}{2\sqrt{1-\beta^2}\alpha_1}\Big) du_1 + \underbrace{\frac{1}{2\sqrt{\pi}\alpha_1}\int_0^\infty \exp\Big(-\frac{u_1^2}{4\alpha_1^2}\Big) du_1}_{=\frac{1}{2}\text{ using Fact B.5}} \tag{75}$$

$$= \frac{1}{2\sqrt{\pi}\alpha_1}\int_0^\infty \exp\Big(-\frac{u_1^2}{4\alpha_1^2}\Big)\mathrm{erf}\Big(\frac{\beta u_1}{2\sqrt{1-\beta^2}\alpha_1}\Big) du_1 + \frac{1}{2}. \tag{76}$$

Replacing (u_1/α_1) by y above gives us:

$$p_j = \frac{1}{2\sqrt{\pi}}\int_0^\infty \exp\Big(-\frac{y^2}{4}\Big)\mathrm{erf}\Big(\frac{\beta y}{2\sqrt{1-\beta^2}}\Big) dy + \frac{1}{2} \tag{77}$$

$$= \underbrace{\frac{1}{2\sqrt{\pi}}\int_0^\infty \exp\Big(-\frac{y^2}{4}\Big) dy}_{=\frac{1}{2}\text{ using Fact B.5}} - \frac{1}{2\sqrt{\pi}}\int_0^\infty \exp\Big(-\frac{y^2}{4}\Big)\mathrm{erfc}\left(\frac{\beta y}{2\sqrt{1-\beta^2}}\right) dy + \frac{1}{2} \tag{78}$$

$$= 1 - \frac{1}{2\sqrt{\pi}}\int_0^\infty \exp\Big(-\frac{y^2}{4}\Big)\mathrm{erfc}\left(\frac{\beta y}{2\sqrt{1-\beta^2}}\right) dy. \tag{79}$$

This is the exact final expression for p_j.

Using Fact B.4 fact above yields:

$$p_j \geq 1 - \frac{1}{\sqrt{\pi}} \int_0^\infty \exp\Big(-\frac{y^2}{4}\Big) \exp\Big(-\frac{\beta^2 y^2}{8(1-\beta^2)}\Big) dy \tag{80}$$

$$= 1 - \frac{1}{\sqrt{\pi}} \int_0^\infty \exp\left(-\frac{y^2}{8}\Big(\frac{2-\beta^2}{1-\beta^2}\Big)\right) dy \tag{81}$$

$$= 1 - \sqrt{\frac{2-2\beta^2}{2-\beta^2}}. \tag{82}$$

Equation (82) follows from Fact B.5.

Finally, the theorem statement follows by adding the subscript j to β in Equation (79) and Equation (82) (recall that we omitted the subscript j earlier for conciseness). □

F.1. Proof of Corollary 6.2

Proof. Plugging in $\beta_j = 1 - \tau_j$ into the lower bound for p_j in Theorem 6.1, we get:

$$p_j \geq 1 - \sqrt{\frac{2-2(1-\tau_j)^2}{2-(1-\tau_j)^2}} = 1 - \sqrt{\frac{2\tau_j(2-\tau_j)}{1+\tau_j(2-\tau_j)}} = 1 - \mathcal{O}(\sqrt{\tau_j}),$$

for $\tau_j \to 0$. □

G. Remaining Experimental Details for Section 7.1

For AdamW, we used the following hyper-parameter values: learning rate = 1e-4, ℓ_2 weight decay = 0.01, β_1 = 0.9 and β_2 = 0.999. The learning rate warmup was over the first 0.2% of total steps followed by linear decay. We used the GELU activation and a dropout probability of 0.1 on all the layers. The training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood.

Table 6. Sampling time of SIFT as a function of early exit layer. For comparison, the full forward and backward propagation times (for the update) in all the cases is $\sim$ 3.8 hours and $\sim$ 7.9 hours, respectively.

SIFT Criterion	SIFT Layer #	Sampling Time (hrs)
Loss-based	1	0.8452
Loss-based	2	1.1662
Loss-based	3	1.4764
Loss-based	6	2.5901
Loss-based	12	4.5086
Entropy-based	1	0.9121
Entropy-based	2	1.2181
Entropy-based	3	1.5398
Entropy-based	6	2.6195
Entropy-based	12	4.5148

H. Remaining Details for Section 7.2

Modified ResNet architecture: The vanilla ResNet architecture is not really amenable to the early exit idea we used for BERT. This is because, unlike BERT, the intermediate layer representations of vanilla ResNet are not of the same size as the final layer representations due to which we cannot use the linear classifier at the head for computing "early loss/entropy" like we did for BERT. So we slightly modify the vanilla ResNet architecture as follows: the output of our modified architecture = linear classifier at head (L_f) $\times$ final layer's representation (R_f) + another linear classifier of appropriate size (L_i) $\times$ some intermediate layer's representation (R_i) instead of just $L_f \times R_f$ which is the output of vanilla ResNet; note that L_i

is also trained. To compute the early loss/entropy, we use $L_i \times R_i$. For our experiments in Section 7.2, we modify the vanilla ResNet-50 architecture as described above and the intermediate layer we use is the final output of the second block consisting of 128 filters. The modification described here for ResNets does indeed use an additional linear classifier (L_i) increasing the number of parameters (compared to the vanilla architecture), but we think it is worthwhile given the amount of improvement we get with SIFT. Moreover, the number of extra parameters is not too much for datasets like CIFAR-100, Food-101, etc., wherein the number of classes is of the order of 100.

Other empirical details: Here, the baseline batch size is 125. The gross batch size of SIFT is 250, while the forward/backward batch size of SIFT is 125 (i.e., we perform the SIFT update on top 50% of the samples in a batch just like Section 7.1). We use the default values of β_1 and β_2 for AdamW and set the weight decay $= 5e - 4$.