A REPRESENTATION-FOCUSED TRAINING ALGORITHM FOR DEEP NETWORKS

Vincent Roulet[†], *Corinne Jones*[‡], *Zaid Harchaoui*[†]

[†]Department of Statistics, University of Washington, Seattle, USA

[‡]Swiss Data Science Center, École polytechnique fédérale de Lausanne, Lausanne, Switzerland

ABSTRACT

We present an optimization algorithm for the training of deep networks that focuses the optimization on the parameters of the feature representation while eliminating the optimization on the parameters of the prediction map. The proposed algorithm can be analyzed for least-squares objectives for which the implicit minimization can be obtained analytically on mini batches. We illustrate the numerical performance of the optimization algorithm with convolutional networks and explore a variant using Adam as a wrapper algorithm.

***Index Terms*—** deep neural networks, stochastic gradient algorithms.

1. INTRODUCTION

Training algorithms for deep networks have been the subject of an abundant literature. The stochastic gradient algorithm and its adaptive practice-informed offsprings are popular approaches to learn the parameters of deep networks. A deep network, however, involves two sets of different parameters: the parameters of the hidden layers defining the feature representation map, and the parameters of the ultimate layer defining the statistical prediction map. Common algorithms of the stochastic gradient type treat these two sets equally, normalization aside, by stacking all parameters into one large vector and performing the optimization over this parameter vector.

We explore in this paper an approach that allows one to drive the optimization process with the parameters of the feature representation by eliminating the parameters of the prediction map via analytical or inexact optimization over a mini-batch of data samples. We show that the resulting algorithm can give smoother training curves and faster amortized training times when learning convolutional networks. We provide a guarantee relating the accuracy of the gradients to the mini-batch size and the smoothness constants of the objective.

Related work. Variable elimination has been used to solve difficult nonlinear least-squares problems, by taking advantage of the simplicity of the optimization with respect to a subset of the variables [1]. This strategy is typically applied with a Gauss-Newton-type algorithm to estimate a set of parameters from noisy observations for inverse problems.

The parameters spared by the elimination strategy are the main parameters of interest in the scientific or engineering domain application. A popular example is the so-called Wiberg algorithm [2] in computer vision for matrix factorization [3]. Variable elimination also underlies the profile likelihood in stochastic modeling and semiparametric estimation, where the nuisance parameters are eliminated to facilitate the estimation of the main parameters [4, 5]. For problems with compositional structure, variable elimination, either exactly or inexactly, can also be cast in a general framework of numerical algorithms with implicit differentiation with reverse mode automatic differentiation [6, 7].

The algorithm we present departs from these algorithms by eliminating variables through mini-batch optimization, refreshing the mini-batches over the iterations by subsampling data, and by focusing the optimization on the subset of parameters defining the feature representation in a deep network. These distinctions are important in the context of machine learning, where the predictive accuracy can be of primary interest. The elimination of variables can be progressively amortized during the learning process in terms of predictive accuracy. The algorithm has first been used to compare convolutional neural networks and their kernelized counterparts [8]. We present additional results and extended analyses. Full proofs are available upon request to the authors.

Problem. We consider learning a feature representation $\phi(\cdot, \theta)$ on a dataset of input-output pairs $(x_i, y_i)_{i=1}^n$ by minimizing the average of a loss ℓ of an affine classifier parameterized by W, b computed on top of the feature representation, i.e., by solving

$$\min_{\theta, W, b} \frac{1}{n} \sum_{i=1}^{n} \ell(W^\top \phi(x_i, \theta) + b, y_i) + \Omega(\theta, W) \ ,$$

where $\Omega(\theta, W)$ encapsulates regularization terms such as $\Omega(\theta, W) = \lambda \|W\|_F^2/2 + \mu \|\theta\|_2^2/2$.

If we choose a squared loss and a ℓ_2^2 penalty, for a given feature representation parameterized by θ, the classifier can be obtained in closed form, and we can reduce our problem to $\min_\theta f(\theta)$ where

$$f(\theta) := \min_{W,b} \frac{1}{n} \sum_{i=1}^{n} \ell(W^\top \phi(x_i, \theta) + b, y_i) + \Omega(\theta, W).$$

The reduced objective $f(\theta)$ can be minimized by, e.g., gradient descent to converge to a solution θ^* of the problem. In the case where the feature representation $\phi(\cdot, \theta)$ is also linear with respect to θ the proposed approach reduces to the Wiberg algorithm used for matrix factorization [2, 3].

By considering the reduced objective, we eliminate the classifier variables from the optimization process, which may accelerate the optimization process as illustrated in Fig. 1. It can easily be verified that, if a θ^* approximates the minimum of the reduced objective, then the tuple $(\theta^*, W_{\theta^*}, b_{\theta^*})$, with W_θ, b_θ the corresponding classifier variables for a given θ, provide an approximate minimum for the original problem. The inconvenient aspect of considering the reduced objective is that it breaks the finite-sum structure of the problem, preventing us from using fast stochastic optimization algorithms.

To circumvent this issue, we consider mini-batch approximations of the reduced objective $f(\theta)$. Formally, for a mini-batch $S \subseteq \{1, \ldots, n\}$ of size $|S| = m$ we consider

$$f_S(\theta) = \min_{W,b} \frac{1}{m} \sum_{i \in S} \ell(W^\top \phi(x_i, \theta) + b, y_i) + \Omega(\theta, W),$$

and make gradient steps using this approximation, i.e., our iterates are of the form

$$\theta_{k+1} = \theta_k - \tau \nabla f_{S_k}(\theta_k), \tag{1}$$

for a given step size $\tau \geq 0$ and S_k a given subset of $\{1, \ldots, n\}$. By considering only mini-batches, our approach is akin to using a ridge ensemble method on top of the representation to assess the performance of the given representation. Ensemble methods over data subsets can lead to improved predictive accuracy compared to a single method over the dataset [9].

We shall look into the bias of approximating $\nabla f(\theta)$ by $\nabla f_S(\theta)$ for a squared loss. Our analysis leads to a worst-case convergence guarantee for our optimization algorithm in terms of convergence to a stationary point. We extend the approach for non-squared losses by considering Newton steps. We present numerical experiments that demonstrate the potential of our algorithm and an Adam based variant to train deep networks.

2. LEAST-SQUARES OBJECTIVES

We consider a feature representation that outputs d features and denote $\Phi(X, \theta) = (\phi(x_1, \theta), \ldots, \phi(x_n, \theta))^\top \in \mathbb{R}^{n \times d}$ the feature representation of the whole dataset. Similarly, we consider learning for k classes and concatenate the outputs in a matrix $Y = (y_1, \ldots, y_n)^\top \in \{0, 1\}^{n \times k}$, where each label y_i is a vector indicating the class to which input i belongs. The classifier consists of a matrix $W \in \mathbb{R}^{d \times k}$ with offsets $b \in \mathbb{R}^k$. Finally we consider $\theta \in \mathbb{R}^p$ to have p parameters.

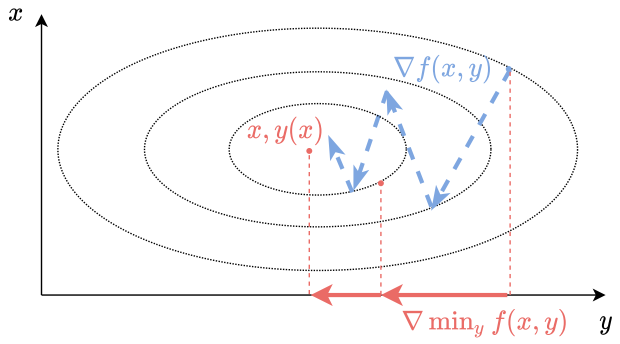


Fig. 1: The blue dashed line represents the path taken by gradient descent on a bivariate function. The plain red line represents the path taken by gradient descent on a reduced objective. Note that by eliminating one variable, we may circumvent oscillation issues from the variable y.

Gradient decomposition. For a squared loss and ℓ_2^2 penalties on the parameters, the objective can be written as

$$\min_{\theta, W, b} \frac{1}{2n} \|\Phi(X, \theta) W + \mathbf{1}_n b^\top - Y\|_F^2 + \frac{\lambda}{2} \|W\|_F^2 + \frac{\mu}{2} \|\theta\|_2^2.$$

for some $\lambda \geq 0, \mu \geq 0$. The reduced objective on a mini-batch S of size $|S| = m$ is given by $f_S(\theta) = h_S(Z) + \mu \|\theta\|_2^2 / 2$, for $Z = (z_1, \ldots, z_n)^\top = \Phi(X, \theta)$, with

$$\begin{aligned} h_S(Z) &= \frac{1}{2m} \|Z_S W_S - Y_S\|_F^2 + \frac{\lambda}{2} \|W_S\|_F^2, \\ W_S &= (\lambda \mathrm{I} + \Sigma_S)^{-1} C_S, \\ \Sigma_S &= \mathrm{Cov}_S(z, z), \quad C_S = \mathrm{Cov}_S(z, y), \\ Z_S^\top &= (\delta_{iS}(z_i - \mathrm{E}_S[z]))_{i=1}^n, \; Y_S^\top = (\delta_{iS}(y_i - \mathrm{E}_S[y]))_{i=1}^n, \end{aligned}$$

where $\delta_{iS} = 1$ if $i \in S$ and 0 otherwise and E_S, Cov_S denote the empirical mean and the empirical covariance on the subset S respectively, e.g., $\mathrm{E}_S[z] = \sum_{i \in S} z_i / m$, $\mathrm{Cov}_S(z, z) = \sum_{i \in S} (z_i - \mathrm{E}_S(z))(z_i - \mathrm{E}_S[z])^\top / m$.

We then have that

$$\nabla h_S(Z) = \frac{1}{m} (Z_S W_S - Y_S) W_S^\top,$$

and for $j \in \{1, \ldots, p\}$, denoting $g_{j,i} = \partial \phi(x_i, \theta) / \partial \theta_j$,

$$\frac{\partial f_S(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i \in S} (W_S^\top z_{S,i} - y_{S,i})^\top W_S^\top g_{j,i} + \mu \theta_j,$$

where $z_{S,i} = z_i - \mathrm{E}_S[z]$, $y_{S,i} = y_i - \mathrm{E}_S[y]$.

Gradient estimation. In the following, we assume that the feature representation is bounded and Lipschitz-continuous and define, for $\mathcal{X} = \{x_1, \ldots, x_n\}$,

$$r = \sup_{\theta \in \mathbb{R}^p, x \in \mathcal{X}} \|\phi(x, \theta)\|_2, \; \ell = \sup_{\theta \in \mathbb{R}^p, x \in \mathcal{X}} \|\nabla_\theta \phi(x, \theta)\|_2.$$

Consider mini-batches S to be subsets of size m chosen uniformly at random without replacement from $\{1, \ldots, n\}$. In that case, the mean-squared error of the approximation of the gradient can be upper-bounded as

$$\mathbb{E}[\|\nabla f_S(\theta) - \nabla f(\theta)\|_2^2] \leq O\left(\frac{q_m n^2 \ell^2 r^6}{\lambda^4}\right), \qquad (2)$$

where $q_m = (n-m)/((n-1)m)$. Bounds on the approximation error of the gradient may also be stated in high probability using the results of [10].

Convergence analysis. To ensure approximate convergence of the iterative algorithm (1), it remains to appropriately choose the step size. The latter depends on the smoothness of the reduced objective, which can be estimated from the smoothness properties of the representation. We then have the following convergence rate.

Theorem 2.1. *Assume that the reduced objective for a squared loss and a ℓ_2^2 penalty is L-smooth. If the mini-batches S are subsets of size m chosen uniformly at random without replacement from $\{1, \ldots, n\}$, then an approximate gradient descent using mini-batch approximations* (1) *with step size $\tau \leq 1/(2L)$ satisfies*

$$\min_{i \in \{0,\ldots,k-1\}} \mathbb{E}\|\nabla f(\theta_i)\|^2 \leq c\, \frac{f(\theta_0) - f^*}{\tau k} + O\left(\frac{q_m n^2 \ell^2 r^6}{\lambda^4}\right)$$

where $q_m = (n-m)/((n-1)m)$ and c is a universal constant.

Theorem 2.1 is similar to usual convergence results for non-convex stochastic optimization algorithms with here an error term that naturally decreases as $m \to n$. In particular in the full-batch case ($m = n$), the above convergence result matches the previously known convergence results for non-convex optimization on the reduced objective f.

Computational complexity. Compared to a stochastic gradient algorithm which computes the value of the original objective and the gradient associated to it by automatic differentiation, our algorithm computes the reduced objective by solving the least-squares problem associated to the prediction map. This computation induces an overhead of $O(\min\{d, m\}^3)$ elementary computations compared to computing the original objective. Once the optimal classifier is computed, the back-propagation can be computed by using the expression of the gradient of the reduced objective given above which induces no overhead compared to the computation of the gradient of the original objective. In our implementation, we code the reduced objective in a differentiable programming framework to access its gradient at a cost no larger than the cost of computing the reduced objective [11].

Algorithm 1 Ultimate Layer Reversal step

```
1: Inputs: Mini-batch $S \subseteq \{1, \ldots, n\}$, with $|S| = m$,
   current parameters $\theta_k, W_k, b_k$, step-size $\tau$, regularization
   penalty $\Omega(\theta, W) = \lambda\|W\|_F^2/2 + \mu\|\theta\|_2^2/2$
2: Compute $q_{\ell_i}(\cdot; \hat{y}_i)$ the quadratic approx. of $\ell_i = \ell(\cdot, y_i)$
   around the current predictions $\hat{y}_i = \phi(x_i, \theta_k)^T W_k + b_k$
3: Compute
   $f_S(\theta) = \min_{W,b} \frac{1}{m} \sum_{i \in S} q_{\ell_i}(W^\top \phi(x_i, \theta) + b; \hat{y}_i) + \Omega(\theta, W)$
4: Update the parameters via $\theta_{k+1} = \theta_k - \tau \nabla f_S(\theta_k)$
5: Compute the corresponding classifiers from the quadratic
   approximation, i.e., compute $W_{k+1}, b_{k+1}$ as
   $\arg\min_{W,b} \frac{1}{m} \sum_{i \in S} q_{\ell_i}(W^\top \phi(x_i, \theta_{k+1}) + b; \hat{y}_i) + \Omega(\theta_{k+1}, W)$
6: Output: New parameters $\theta_{k+1}, W_{k+1}, b_{k+1}$
```

3. BEYOND LEAST-SQUARES OBJECTIVES

For non-squared losses we consider a reduced objective computed as the minimizer of a quadratic approximation of the objective. Namely, denoting θ_k, W_k, and b_k the current parameters of the training and $q_{\ell_i}(\cdot; \hat{y}_i)$ a quadratic approximation of the loss $\ell_i = \ell(\cdot, y_i)$ around $\hat{y}_i = W_k^\top \phi(x_i, \theta_k) + b_k$, our reduced objective consists in computing for a mini-batch S of size m

$$f_S(\theta) = \min_{W,b} \frac{1}{m} \sum_{i \in S} q_{\ell_i}(W^\top \phi(x_i, \theta) + b; \hat{y}_i) + \lambda \Omega(\theta, W).$$

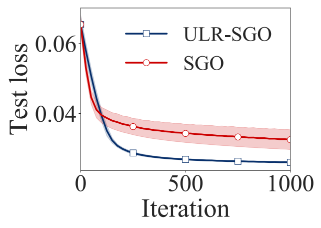
A gradient step is then performed on h_S and the corresponding optimal classifier parameters are stored to compute the next quadratic approximation. The overall algorithm is presented in Algorithm 1.

When implementing the algorithm, we consider adding a regularization κ to ensure that the classifiers remain stable along the iterations. Namely, we modify lines (3) and (5) by adding a term $\kappa\|W - W_k\|_2^2/2$ in the corresponding minimization problems for stability.

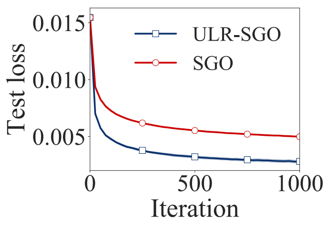
4. EXPERIMENTS

We train deep networks with Algorithm 1 (termed "ULR-SGO" for "Ultimate Layer Reversal-Stochastic Gradient Optimization") and compare to using mini-batch stochastic gradient optimization (SGO).

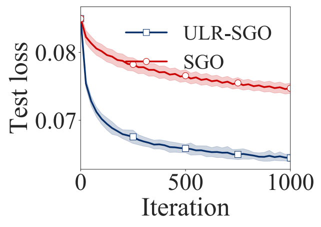
Experimental details. The tasks we consider are digit classification on MNIST [12] and image classification on CIFAR-10 [13]. The network we use for digit classification is a kernelized version of LeNet-5 [12], which we call the LeNet-5



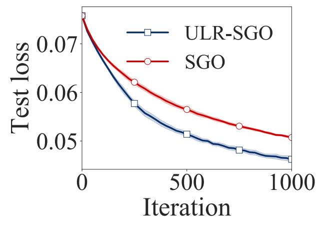
(a) LeNet-5 CKN on MNIST with 8 filters/layer



(b) LeNet-5 CKN on MNIST with 128 filters/layer

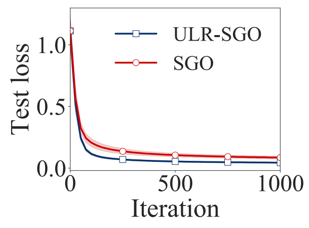


(c) All-CNN-C CKN on CIFAR-10 with 8 filters/layer

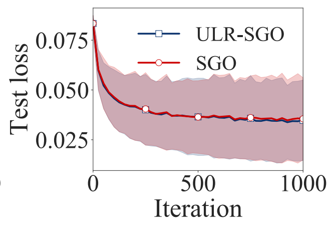


(d) All-CNN-C CKN on CIFAR-10 with 128 filters/layer

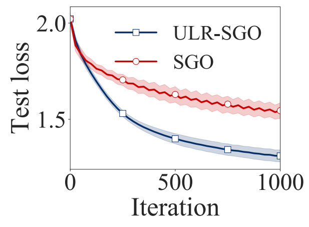
Fig. 2: Average test loss vs. iteration when using the square loss and training with the Ultimate Layer Reversal method (ULR-SGO) vs. stochastic gradient optimization (SGO).



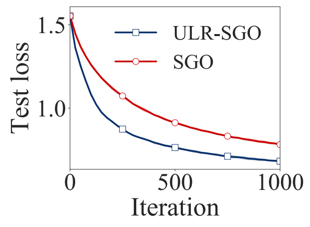
(a) LeNet-5 CKN on MNIST with 8 filters/layer



(b) LeNet-5 CKN on MNIST with 128 filters/layer



(c) All-CNN-C CKN on CIFAR-10 with 8 filters/layer



(d) All-CNN-C CKN on CIFAR-10 with 128 filters/layer

Fig. 3: Average test loss vs. iteration when using the multinomial logistic loss and training with the Ultimate Layer Reversal method (ULR-SGO) vs. stochastic gradient optimization (SGO).

CKN. In contrast, for image classification we use a kernelized version of All-CNN-C [14], which we call the All-CNN-C CKN. When first introduced, LeNet5 and All-CNN-C were among the best-performing architectures on their respective tasks. In contrast to the original ConvNets, the CKNs are differentiable with respect to both their weights and their inputs. Moreover, it has been demonstrated that the ConvNets and the CKNs perform similarly [8]. For both dataset-architecture pairs we consider both 8 and 128 filters per layer, and we consider both the square loss and the multinomial logistic loss. The batch size is the largest batch size that fits on the GPU.[1]

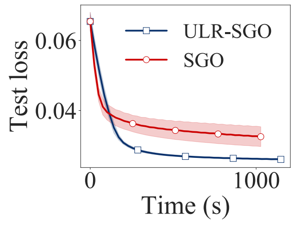
The datasets are preprocessed as follows: the MNIST images are standardized, while the CIFAR-10 images are standardized channel-wise and then ZCA whitened. The networks are initialized via the unsupervised spherical k-means procedure described in [8]. The classifier parameters are initialized by optimizing them for the initial feature representation on the whole dataset. The hyperparameters are tuned to minimize the loss on a hold-out set of size 10,000 images removed from the training set. The L_2 penalty on the classifier parameters is chosen from $2^{-40}, 2^{-39}, \ldots, 2^0$ based upon the parameter values at initialization. For ULR-SGO the step size τ and the regularization parameter κ are simultaneously determined from $2^{-10}, \ldots, 2^{-2}$ and $2^{-7}, \ldots, 2^{-2}$, respectively, on the trained networks. For SGO the step size is chosen in the same way. In all cases, we do not regularize the network parameters, i.e., $\mu = 0$. Following the hyperparameter tuning, each network is trained for 1000 iterations. In all plots, the error bands represent one standard deviation across 10 trials.

The code for the experiments was written using PyTorch [15], Faiss [16], and Scipy's L-BFGS [17]. The code for CKNs can be found here `https://github.com/cjones6/yesweckn`. The code to reproduce experimental results is available upon request to the authors.

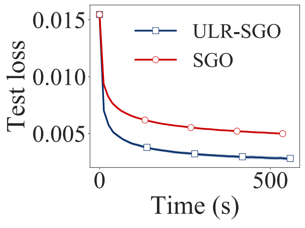
Results. Fig. 2 displays the test losses of the comparison for LeNet-5 CKN on MNIST and All-CNN-C CKN on CIFAR-10 with 8 and 128 filters/layer when using the square loss. ULR-SGO is usually better than SGO throughout the iterations. The final test loss from the ULR-SGO method after 1000 iterations ranges from being 9% lower, on average, when classifying CIFAR-10 images with the All-CNN-C CKN architecture with 128 filters/layer to being 44% lower, on average, when classifying MNIST digits with the LeNet-5 CKN with 128 filters/layer.

This performance difference can also be observed when using the multinomial logistic loss. Fig. 3 displays the results from training the same networks with ULR-SGO and SGO, but this time using the multinomial logistic loss. The final test losses when using ULR-SGO vary between being 2% lower, on average, when training the LeNet-5 CKN on MNIST with 128 filters/layer to being 47% lower, on average, when training the LeNet-5 CKN on MNIST with 8 filters/layer.

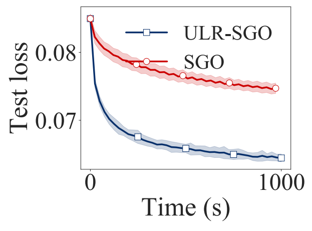
[1]For LeNet-5 CKN on MNIST with 8 (128) filters/layer this is 16,384 (1024). For All-CNN-C CKN with 8 (128) filters/layer this is 4096 (256).



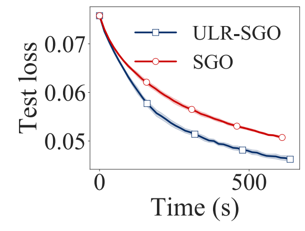
(a) LeNet-5 CKN on MNIST with 8 filters/layer



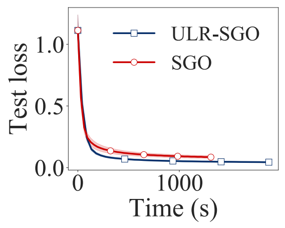
(b) LeNet-5 CKN on MNIST with 128 filters/layer



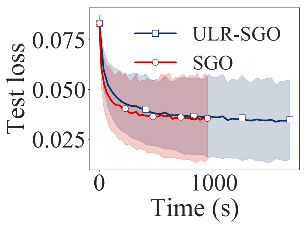
(c) All-CNN-C CKN on CIFAR-10 with 8 filters/layer



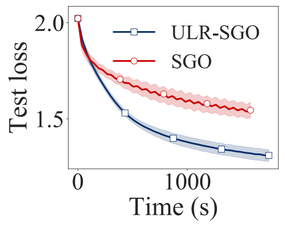
(d) All-CNN-C CKN on CIFAR-10 with 128 filters/layer



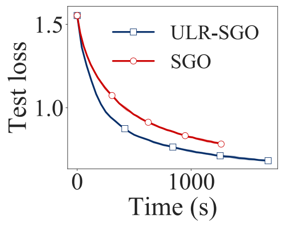
(e) LeNet-5 CKN on MNIST with 8 filters/layer



(f) LeNet-5 CKN on MNIST with 128 filters/layer



(g) All-CNN-C CKN on CIFAR-10 with 8 filters/layer



(h) All-CNN-C CKN on CIFAR-10 with 128 filters/layer

Fig. 4: Average test loss vs. time when using the squared (top 4 panels) or the multinomial (4 bottom panels) logistic loss and training with the Ultimate Layer Reversal method (ULR-SGO) vs. stochastic gradient optimization (SGO).

While it is clear that ULR-SGO dominates SGO in terms of its performance across iterations on the harder tasks, it is also important to ensure that this is true in terms of time. Fig. 4 provides the same plots as Fig. 2 and 3, except that the x-axis is now time. The experiments with the multinomial loss on architectures with 128 filters/layer were performed with Nvidia GeForce 1080Ti GPUs. The remainder were performed with Nvidia Titan Xp GPUs. With the exception of the LeNet-5 experiment with 128 filters/layer and the multinomial logistic loss, the ULR-SGO method still outperforms the SGO method in terms of the test loss vs. time.

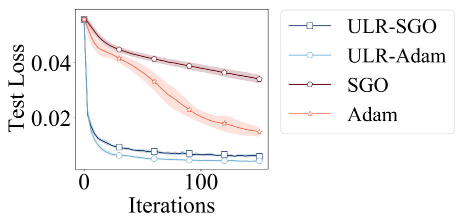


Fig. 5: Average test loss vs iterations when using the squared loss on LeNet5 ConvNet on MNIST by using gradients of the reduced objective with stochastic gradient descent (ULR-SGO) or with the Adam optimizer (ULR-Adam) compared to the same optimization algorithms on the original objective (SGO and Adam).

The gradient of the reduced objective can also be substituted in place of a regular gradient of a classical objective in the update rule of a momentum based stochastic gradient algorithm such as Adam [18]. This leads to an interesting variant of our approach suggested by the reviewers in which Adam is used as a wrapper algorithm. To explore this variant, we considered the original ConvNet architecture of LeNet5 [12] with a squared loss and fixed regularization parameters ($\lambda = \mu = 10^{-3}$). The mini-batch size is fixed to 4096 and the learning rate parameters are tuned on a log10 basis for this experiment. In Fig. 5, we observe that the resulting algorithm, called ULR-Adam, is competitive compared to ULR-SGD, and the gap is comparable to the gap between Adam and SGD in the usual setting.

5. CONCLUSION

We presented an algorithm that allows one to eliminate the optimization over the parameters of the prediction map and focus the optimization on the parameters of the feature map. This can result in faster training of deep networks in terms of the number of iterations and the total amortized time. The proposed algorithm can be further extended in a number of interesting directions. For example, adaptive sampling strategies could potentially further improve the behavior of the algorithm. Extensions to prediction problems with heteroskedastic noise could also be interesting to pursue.

Acknowledgments. This work was supported by NSF DMS-2023166, CCF-2019844, DMS-1839371, CIFAR-LMB, and faculty research awards.

6. REFERENCES

[1] Axel Ruhe and Per Åke Wedin, "Algorithms for separable nonlinear least squares problems," *SIAM review*, vol. 22, no. 3, pp. 318–337, 1980.

[2] T Wiberg, "Computation of principal components when data are missing," in *Proc. Second Symp. Computational Statistics*, 1976, pp. 229–236.

[3] Pei Chen, "Heteroscedastic low-rank matrix approximation by the wiberg algorithm," *IEEE transactions on signal processing*, vol. 56, no. 4, pp. 1429–1439, 2008.

[4] Julian Besag, "Statistical analysis of non-lattice data," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 24, no. 3, pp. 179–195, 1975.

[5] Susan A Murphy and Aad W Van der Vaart, "On profile likelihood," *Journal of the American Statistical Association*, vol. 95, no. 450, pp. 449–465, 2000.

[6] Pierre Ablin, Gabriel Peyré, and Thomas Moreau, "Super-efficiency of automatic differentiation for functions defined as a minimum," in *International Conference on Machine Learning*. PMLR, 2020, pp. 32–41.

[7] Kaiyi Ji, Junjie Yang, and Yingbin Liang, "Bilevel optimization: Convergence analysis and enhanced design," in *Proceedings of the 38th International Conference on Machine Learning*, 2021, pp. 4882–4892.

[8] Corinne Jones, *Representation Learning for Partitioning Problems*, Ph.D. thesis, University of Washington, 2020.

[9] Katuwal Rakesh and Ponnuthurai N Suganthan, "An ensemble of kernel ridge regression for multi-class classification," *Procedia Computer Science*, vol. 108, pp. 375–383, 2017.

[10] Rémi Bardenet and Odalric-Ambrym Maillard, "Concentration inequalities for sampling without replacement," *Bernoulli*, vol. 21, no. 3, pp. 1361–1385, 2015.

[11] Walter Baur and Volker Strassen, "The complexity of partial derivatives," *Theoretical computer science*, vol. 22, no. 3, pp. 317–330, 1983.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Intelligent Signal Processing*. 2001, pp. 306–351, IEEE Press.

[13] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.

[14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *International Conferenceon Learning Representations (Workshop Track)*, 2015.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035. 2019.

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[17] Dong C. Liu and Jorge Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 3 (B), pp. 503–528, 1989.

[18] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[19] Olivier Devolder, François Glineur, and Yurii Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *Mathematical Programming*, vol. 146, no. 1, pp. 37–75, 2014.

A. PROOFS

To decompose the error of approximation of the gradient of the reduced objective by using a mini-batch, we define for $\theta \in \mathbb{R}^p$, $j \in \{1, \ldots, p\}$, $g_j = (\partial\phi(x_i, \theta)/\partial\theta_j)_{i=1}^n$ and $S \subseteq \{1, \ldots, n\}$, $U_{j,S} = \mathrm{Cov}_S(g_j, z)$, $\quad V_{j,S} = \mathrm{Cov}_S(g_j, y)$, and the associated tensors $\mathcal{U}_S = (U_{j,S})_{j=1}^p$, $\mathcal{V}_S = (V_{j,S})_{j=1}^p$. We can then bound the approximation error from computing the gradient on a mini-batch as follows.

Lemma A.1. *Consider some parameters $\theta \in \mathbb{R}^p$, and some mini-batch $S \subseteq \{1, \ldots, n\}$ with $|S| = m$. Define the following constants associated to θ, S,*

$$\begin{aligned} a &= \| \mathrm{E}_S[z] - \mathrm{E}_{[n]}[z] \|_2, & b &= \| \mathrm{E}_S[y] - \mathrm{E}_{[n]}[y] \|_2, \\ s &= \|\Sigma_S - \Sigma_{[n]}\|_2, & c &= \|C_S - C_{[n]}\|_2, \\ u &= \|\mathcal{U}_S - \mathcal{U}_{[n]}\|_{*,2}, & v &= \|\mathcal{V}_S - \mathcal{V}_{[n]}\|_{*,2}, \end{aligned}$$

where $\|\mathcal{U}\|_{*,2} = \sup_{\|z\|_2 \leq 1} \| \sum_{j=1}^p z_j U_j \|_*$, with $\|\cdot\|_*$ the nuclear norm. The approximation error of the gradient by the reduced objective is bounded as $\|\nabla f_S(\theta) - \nabla f(\theta)\|_2 \leq \varepsilon_{\theta,S} = \delta_{\theta,S} + \eta_{\theta,S}$, where

$$\begin{aligned} \delta_{\theta,S} &= \frac{2\ell c}{\lambda} + \sqrt{\frac{m}{4\lambda}}\ell b + \frac{m}{4\lambda}\ell a + \sqrt{\frac{n}{\lambda}}\frac{\ell(2rc + s)}{\lambda} + \frac{n}{\lambda}\frac{\ell s r}{\lambda} \\ \eta_{\theta,S} &= \sqrt{\frac{m}{4\lambda}}(v + \ell b) + \frac{m}{4\lambda}(u + \ell a). \end{aligned} \tag{3}$$

The approximation error of the gradient of the reduced objective $\nabla f(\theta)$ by its approximation $\nabla f_S(\theta)$ on a mini-batch is decomposed into (i) $\delta_{\theta,S}$ the error of computing the classifier W_S on the mini-batch instead of computing the classifier W over the whole dataset; and (ii) $\eta_{\theta,S}$ the error of using only a subset of the points to back-propagate the gradient instead of using the whole dataset.

Proof of Lemma 7.1. Let $j \in \{1, \ldots, p\}$. Denote $g_{j,i} = \partial\phi(x_i, \theta)/\partial\theta_j$, $\mu_S = \mathrm{E}_S[z]$ and $\nu_S = \mathrm{E}_S[y]$ for $S \subseteq \{1, \ldots, n\}$. Recall that $W_S = (\lambda \mathrm{I} + \Sigma_S)^{-1} C_S$. Note that by using the singular value decomposition of Z_S/m, we have that $\|W_S\|_2 \leq \|Y_S\|_2/(2\sqrt{\lambda}) \leq \sqrt{m}/(2\sqrt{\lambda})$, and similarly $\|W_{[n]}\|_2 \leq \sqrt{n}/(2\sqrt{\lambda})$. We have

$$\begin{aligned} (\nabla f_S(\theta))_j - (\nabla f(\theta))_j &= \underbrace{\frac{1}{m}\sum_{i \in S}(W_S^\top(z_i - \mu_S) - (y_i - \nu_S))^\top W_S^\top g_{j,i} - \frac{1}{n}\sum_{i=1}^n (W_S^\top(z_i - \mu_S) - (y_i - \nu_S))^\top W_S^\top g_{j,i}}_{\eta_j} \\ &+ \frac{1}{n}\sum_{i=1}^n \underbrace{\left[(W_S^\top(z_i - \mu_S) - (y_i - \nu_S))^\top W_S^\top - (W_{[n]}^\top(z_i - \mu_{[n]}) - (y_i - \nu_{[n]}))^\top W_{[n]}^\top\right]}_{\delta_i^\top} g_{j,i}, \end{aligned}$$

We can then rewrite η_j as

$$\begin{aligned} \eta_j &= \frac{1}{m}\sum_{i \in S}(W_S^\top(z_i - \mu_S) - (y_i - \nu_S))^\top W_S^\top (g_{j,i} - \mathrm{E}_S[g_j]) - \frac{1}{n}\sum_{i=1}^n (W_S^\top(z_i - \mu_S) - (y_i - \nu_S))^\top W_S^\top (g_{j,i} - \mathrm{E}_{[n]}[g_j]) \\ &\quad + (W_S^\top(\mu_{[n]} - \mu_S) + \nu_{[n]} - \nu_S)^\top W_S^\top \mathrm{E}_{[n]}[g_j] \\ &= \mathbf{Tr}\left(\left((U_{j,S} - U_{j,[n]})W_S - (V_{j,S} - V_{j,[n]})\right) W_S^\top\right) + (W_S^\top(\mu_{[n]} - \mu_S) + \nu_{[n]} - \nu_S)^\top W_S^\top \mathrm{E}_{[n]}[g_j]. \end{aligned}$$

Denoting then $\eta = (\eta_j)_{j=1}^p$, we have $\|\eta\|_2 \leq \left((u + \ell a)\frac{1}{2}\sqrt{\frac{m}{\lambda}} + v + \ell b\right)\frac{1}{2}\sqrt{\frac{m}{\lambda}}$. On the other hand,

$$\begin{aligned} &\delta_i = W_S(W_S^\top(\mu_{[n]} - \mu_S) - (\nu_{[n]} - \nu_S)) + W_S\left((W_S^\top - W_{[n]}^\top)(z_i - \mu_{[n]})\right) + (W_S - W_{[n]})\left(W_{[n]}^\top(z_i - \mu_{[n]}) - (y_i - \nu_{[n]})\right), \\ &W_S - W_{[n]} = (\lambda \mathrm{I} + \Sigma_S)^{-1}(C_S - C_{[n]}) + (\lambda \mathrm{I} + \Sigma_S)^{-1}(\Sigma_{[n]} - \Sigma_S)(\lambda \mathrm{I} + \Sigma_{[n]})^{-1}C_{[n]}, \\ &\text{so} \quad \|\delta_i\|_2 \leq \frac{1}{2}\sqrt{\frac{m}{\lambda}}\left(\frac{1}{2}\sqrt{\frac{m}{\lambda}}a + b\right) + \frac{2}{\lambda}\left(\frac{\sqrt{n} + \sqrt{m}}{2\sqrt{\lambda}}r + 1\right)\left(c + \frac{1}{2}\sqrt{\frac{n}{\lambda}}s\right), \end{aligned}$$

where we used that $\|z_i\|_2 \leq r$, $\|y_i\|_2 \leq 1$ and $\|(\lambda \mathrm{I} + \Sigma_{[n]})^{-1}C_{[n]}\|_2 \leq \sqrt{n}/(2\sqrt{\lambda})$ by eigendecomposition of $Z_{[n]}$. Since $\nabla_\theta\phi(x_i, \theta) = (g_{j,i})_{j=1}^p$, we have $\|\nabla f_S(\theta) - \nabla f(\theta)\|_2 \leq \|\eta\|_2 + \frac{1}{n}\|\sum_{i=1}^n \nabla_\theta\phi(x_i, \theta)\delta_i\|_2 \leq \|\eta\|_2 + \ell\frac{1}{n}\sum_{i=1}^n \|\delta_i\|_2$ and the result follows. □

Lemma A.2. *Consider mini-batches S to be subsets of size m chosen uniformly at random without replacement from $\{1, \ldots, n\}$. In that case, the mean-squared error of the approximation of the gradient can be upper-bounded as*

$$\mathbb{E}[\varepsilon_{\theta,S}^2] \leq q_m \left(\frac{\ell^2 r^2}{\lambda^2} + \frac{m\ell^2}{\lambda} + \frac{m^2\ell^2 r^2}{\lambda^2} + \frac{n\ell^2 r^4}{\lambda^3} + \frac{n^2\ell^2 r^6}{\lambda^4} + \frac{m\ell^2}{\lambda} + \frac{m^2\ell^2 r^2}{\lambda^2}, \right)$$

where $q_m = \kappa(n-m)/((n-1)m)$ for κ a dimension-dependent constant.

Proof. Denote by $a_S, b_S, \ldots, v_S$ the quantities defined in Lemma 7.1 where, e.g.. $a_S = a$. Denote $q_m = (n-m)/((n-1)m)$. From Lemma 7.3, we have that, for S a mini-batch of size m chosen uniformly at random without replacement that $\mathbb{E}[a_S^2] \leq q_m r^2$, $\mathbb{E}[b_S^2] \leq q_m$. We have that $\Sigma_S = \frac{1}{m}\sum_{i\in S}(z_i - \mu_{[n]})(z_i - \mu_{[n]})^\top + (\mu_{[n]} - \mu_S)(\mu_{[n]} - \mu_S)^\top$. Hence denoting $\zeta_i = (z_i - \mu_{[n]})(z_i - \mu_{[n]})^\top$, we have $\|\Sigma_S - \Sigma\|_2^2 \leq 2\|\zeta_S - \zeta_{[n]}\|_F^2 + 4r^2\|\mu_S - \mu_{[n]}\|_2^2$. Hence we get that $\mathbb{E}[s_S^2] \leq 4q_m r^4$. Similarly, we have that $\mathbb{E}[c_S^2] \leq 4q_m r^2$. For a tensor $\mathcal{U} = (U_1, \ldots, U_p)$, we have that $\|\mathcal{U}\|_{*,2} \leq \sum_{j=1}^p \|U_j\|_*$. Hence $\mathbb{E}[u_S^2] \leq p\sum_{j=1}^p d\mathbb{E}_S[\|U_{[S],j} - U_{[n,j]}\|_F^2] \leq 4p^2 d q_m r^2\ell^2$ and similarly $\mathbb{E}[v_S^2] \leq 4p^2\min\{d,k\} q_m \ell^2$. □

Lemma A.3. *Let $x_1, \ldots, x_n \in \mathcal{X}$ with $\mathcal{X}$ equipped with a scalar product $\langle\cdot,\cdot\rangle$ and associated norm $\|\cdot\|$ and denote $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$. Let S be a subset of size m chosen uniformly at random without replacement from $\{x_1, \ldots, x_n\}$. Then*

$$\mathbb{E}_S\left\|\frac{1}{m}\sum_{i\in S} x_i - \bar{x}\right\|^2 \leq \frac{n-m}{n-1}\frac{1}{mn}\sum_{i=1}^n \|x_i - \bar{x}\|^2 .$$

Proof. Consider without loss of generality $\bar{x} = 0$ s.t. $\sum_{j\neq i} x_j = -x_i$. Let $\mathcal{S}$ the set of subsets of size m in $\{1, \ldots n\}$. We have

$$\mathrm{E}_S\left\|\frac{1}{m}\sum_{i\in S} x_i\right\|^2 = \frac{1}{m^2\binom{n}{m}}\sum_{S\in\mathcal{S}}\left(\sum_{i\in S}\|x_i\|_2^2 + \sum_{\substack{i,j\in S\\ i\neq j}}\langle x_i, x_j\rangle\right) = \frac{1}{m^2\binom{n}{m}}\sum_{i=1}^n\left(\sum_{S\in\mathcal{S}:S\ni i}\|x_i\|_2^2 + \sum_{j\neq i}\sum_{S\in\mathcal{S}:i,j\ni S}\langle x_i, x_j\rangle\right)$$

The right hand side is then equal to $(\binom{n-1}{m-1} - \binom{n-2}{m-2})/(m^2\binom{n}{m})\sum_i \|x_i\|_2^2$ and the result follows. □

Lemma A.4. *Assume that ϕ is bounded, Lipschitz-continuous and smooth such that*

$$r = \sup_{\theta\in\mathbb{R}^p, x\in\mathcal{X}}\|\phi(x,\theta)\|_2, \quad \ell = \sup_{\theta\in\mathbb{R}^p, x\in\mathcal{X}}\|\nabla_\theta\phi(x,\theta)\|_2, \quad L = \sup_{\theta,\theta'\in\mathbb{R}^p, x\in\mathcal{X}}\frac{\|\nabla_\theta\phi(x,\theta) - \nabla_\theta\phi(x,\theta')\|_2}{\|\theta-\theta'\|_2}$$

are finite for $\mathcal{X} = \{x_1, \ldots, x_n\}$. Then the reduced objective $f(\theta)$ defined in (**??**) *for a squared loss and a squared penalty is $\tilde{L}$ smooth with*

$$\tilde{L} \leq \sqrt{\frac{n}{\lambda}}\left(L + \frac{\ell^2(r(2+\sqrt{n}) + 2\sqrt{\lambda^{-1}})}{\lambda}\right) + \frac{2\ell^2\sqrt{n}}{\lambda} + \frac{n}{2\lambda}\left(Lr + \ell^2\left(\frac{2b^2}{\lambda} + 1\right)\right).$$

Estimation of the smoothness constant. Let $\theta, \theta' \in \mathbb{R}^p$, denote $\nabla h_{[n]}(\Phi(X,\theta)) = (v_1, \ldots, v_n)^\top \in \mathbb{R}^{n\times d}$, $g_{j,i} = \partial\phi(x_i,\theta)/\partial\theta_j$ and denote with superscripts $'$ quantities corresponding to computations with θ'. We have that $(\nabla f(\theta))_j - (\nabla f(\theta'))_j = \frac{1}{n}\sum_{i=1}^n v_i^\top(g_{j,i} - g'_{j,i}) + \frac{1}{n}\sum_{i=1}^n(v_i - v'_i)^\top g'_{j,i}$, hence $\|\nabla f(\theta) - \nabla f(\theta')\|_2 \leq L\frac{1}{n}\sum_{i=1}^n\|v_i\|_2\|\theta-\theta'\|_2 + \ell\frac{1}{n}\sum_{i=1}^n\|v_i - v'_i\|_2$. We have $v_i = W_{[n]}(W_{[n]}^\top(z_i - \mu_{[n]}) - (y_i - \nu_{[n]}))$ where $W_{[n]}, \mu_{[n]}, z_i$ are defined from $\Phi(X,\theta)$. Hence $\|v_i\|_2 \leq \sqrt{\frac{n}{\lambda}}\left(\frac{1}{2}\sqrt{\frac{n}{\lambda}}b + 1\right)$. Now we have, denoting for simplicity $\mu = \mu_{[n]}$, $\nu = \nu_{[n]}$,

$$v_i - v'_i = (W - W')(y_i - \nu) + WW^\top(z_i - \mu) - W'W'^\top(z'_i - \mu') ,$$
$$WW^\top(z_i - \mu) - W'W'^\top(z'_i - \mu') = (W - W')^\top W^\top(z_i - \mu) + W'(W - W')^\top(z_i - \mu) + W'W'^\top(z_i - \mu + z'_i - \mu') ,$$
$$W - W' = (\lambda\,\mathrm{I} + \Sigma)^{-1}(C - C') + (\lambda\,\mathrm{I} + \Sigma)^{-1}(\Sigma' - \Sigma)(\lambda\,\mathrm{I} + \Sigma')^{-1}C' ,$$

hence by bounding each term we get that the smoothness of the reduced objective can be estimated as $\sup_{\theta,\theta'\in\mathbb{R}^p}\frac{\|\nabla f(\theta) - \nabla f(\theta')\|_2}{\|\theta-\theta'\|_2} \leq \tilde{L}$ with $\tilde{L}$ defined in the claim. □

Theorem A.5. *Assume that the reduced objective f defined in (**??**) for a squared loss and a squared penalty is L-smooth. Then for any choice of mini-batches, an approximate gradient descent using mini-batch approximations* (1) *with step size $\tau \leq 1/(2L)$ satisfies*

$$\min_{i \in \{0,\dots,k-1\}} \|\nabla f(\theta_i)\|^2 \leq c_1 \frac{f(\theta_0) - f^*}{\tau k} + \frac{c_2}{k} \sum_{i=0}^{k-1} \varepsilon^2_{\theta_i, S_i},$$

where $f^* = \min_{\theta \in \mathbb{R}^p} f(\theta)$, c_1, c_2 are universal constants and $\varepsilon_{\theta,S}$ is defined in Lemma 7.1.

If the mini-batches S are subsets of size m chosen uniformly at random without replacement from $\{1,\dots,n\}$, then an approximate gradient descent using mini-batch approximations (1) *with step size $\tau \leq 1/(2L)$ satisfies*

$$\min_{i \in \{0,\dots,k-1\}} \mathbb{E}\|\nabla f(\theta_i)\|^2 \leq c_1 \frac{f(\theta_0) - f^*}{\tau k} + O\left(\frac{q_m n^2 \ell^2 r^6}{\lambda^4}\right)$$

where $q_m = \kappa(n-m)/((n-1)m)$ and c_1 is a universal constant.

Proof. The first claim is a corollary of Lemma 7.6 akin to the results of [19]. For the second claim, it suffices to note that, following the proof of Lemma 7.6 in Eq. (4),

$$\mathbb{E}[f(\theta_{i+1}) - f(\theta_i)] \leq -c_1 \tau \mathbb{E}[\|\nabla f(\theta_i)\|_2^2] + c_2 \tau \mathbb{E}_{S_1,\dots,S_{i-1}}[\mathbb{E}_{S_i}[\varepsilon^2_{\theta_i,S_i} | S_1, \dots, S_{i-1}]]$$

for c_1, c_2 some universal constants and the second term can be bounded using (2). Summing over i for $i \in \{0,\dots,k-1\}$ and rearranging the terms gives the result. □

Lemma A.6. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth function. Consider an ε-approximate gradient descent on f with step size $\tau \leq 1/(2L)$, i.e., $x_{k+1} = x_k - \tau \widehat{\nabla} f(x_k)$, where $\|\widehat{\nabla} f(x_k) - \nabla f(x_k)\|_2 \leq \varepsilon_k$. After k iterations, this method satisfies, for c_1, c_2 two universal constants,*

$$\min_{i \in \{0,\dots,k-1\}} \|\nabla f(x_i)\|_2^2 \leq c_1 \frac{f(x_0) - \min_{x \in \mathbb{R}^d} f(x)}{\tau k} + \frac{c_2}{k} \sum_{i=0}^{k-1} \varepsilon_i^2.$$

Proof. Denote $g_k = \widehat{\nabla} f(x_k) - \nabla f(x_k)$ for all $k \geq 0$. By L-smoothness of the objective, the iterations of the approximate gradient descent satisfy

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \nabla f(x_k)^\top (x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|_2^2 \\ &= f(x_k) - \tau \|\nabla f(x_k)\|_2^2 - \tau \nabla f(x_k)^\top g_k + \frac{L\tau^2}{2}\|\nabla f(x_k) + g_k\|_2^2 \\ &= f(x_k) - \tau\left(1 - \frac{L\tau}{2}\right)\|\nabla f(x_k)\|_2^2 + \frac{L\tau^2}{2}\|g_k\|_2^2 + \tau(L\tau - 1)\nabla f(x_k)^\top g_k \\ &\leq f(x_k) - \tau\left(1 - \frac{L\tau}{2}\right)\|\nabla f(x_k)\|_2^2 + \frac{L\tau^2}{2}\|g_k\|_2^2 + \tau(1 - L\tau)\|\nabla f(x_k)\|_2 \|g_k\|_2 \ , \end{aligned}$$

where in the last inequality we bounded the absolute value of the last term and used that $\tau L \leq 1$. Now we use that for any $a, b \in \mathbb{R}$ and $\theta > 0$, $2ab \leq \theta a^2 + \theta^{-1} b^2$, which gives for $\theta > 0$, $a = \sqrt{\tau(1 - L\tau)/2}\|\nabla f(x_k)\|_2$ and $b = \sqrt{\tau(1 - L\tau)/2}\|g_k\|_2$,

$$f(x_{k+1}) \leq f(x_k) - \tau\left(1 - \frac{L\tau + \theta(1 - L\tau)}{2}\right)\|\nabla f(x_k)\|_2^2 + \frac{L\tau^2 + \theta^{-1}\tau(1 - L\tau)}{2}\|g_k\|_2^2 \ .$$

Using $0 \leq L\tau \leq 1/2$, $\theta = 1/4$ and $\|g_k\|_2^2 \leq \varepsilon_k^2$, we get

$$f(x_{k+1}) \leq f(x_k) - \frac{11}{16}\tau\|\nabla f(x_k)\|_2^2 + 2\tau\varepsilon_k^2. \tag{4}$$

Rearranging the terms, summing from $i = 0, \dots, k-1$, taking the minimum, and dividing by k, we get the result. □