OPT2021: 13th Annual Workshop on Optimization for Machine Learning

Faking Interpolation Until You Make It

Alasdair Paren APAREN@ROBOTS.OX.AC.UK *Department of Engineering Science*
University of Oxford
Oxford, UK

Rudra P. K. Poudel RUDRA.POUDEL@CRL.TOSHIBA.CO.UK *Cambridge Research Laboratory,*
Toshiba Europe Ltd,
Cambridge, UK

M. Pawan Kumar PAWAN@ROBOTS.OX.AC.UK *Department of Engineering Science*
University of Oxford
Oxford, UK

Abstract

Deep over-parameterized neural networks exhibit the interpolation property on many data sets. That is, these models are able to achieve approximately zero loss on all training samples simultaneously. Recently, this property has been exploited to develop novel optimisation algorithms for this setting. These algorithms use the fact that the optimal loss value is known to employ a variation of a Polyak Step-size calculated on a stochastic batch of data. In this work, we introduce an algorithm that extends this idea to tasks where the interpolation property does not hold. As we no longer have access to the optimal loss values *a priori*, we instead estimate them for each sample online. To realise this, we introduce a simple but highly effective heuristic for approximating the optimal value based on previous loss evaluations. Through rigorous experimentation we show the effectiveness of our approach, which outperforms state of the art adaptive gradient and line search methods.

1. Introduction

Deep over-parameterized neural networks exhibit the interpolation property on many data sets [1, 18]. That is, these models are able to achieve close to zero loss on all samples simultaneously. Recently, the interpolation property has been exploited to prove new convergence rates [9, 12, 17, 18] and to develop novel optimisation algorithms for this setting. Examples include Adaptive learning rates for Interpolation with Gradients (ALI-G) [1], and Stochastic Polyak Step (SPS) [10]. Both ALI-G and SPS use the interpolation property to ensure that the optimal loss value for each data point will be zero. With this knowledge it is possible to employ a stochastic variation of the Polyak step-size [15]. This automatically scales a maximal step-size hyperparameter down to an appropriate value for each update, which removes the need for a painstakingly hand-designed learning rate schedule [1, 10]. ALI-G and SPS have been shown to produce highly competitive results, matching the generalisation performance of SGD with a manually tuned learning rate in many settings, and outperforming adaptive gradient methods by a large margin. While such techniques work well, the interpolation property does not hold on many interesting large-scale learning tasks, or in situations where the model size is limited.

In this work, we propose a novel optimisation method for non-interpolating problems inspired by algorithms for interpolation. Our approach is based on the observation that any non-interpolating

problem can be made to satisfy the interpolation property once a point that minimises the training objective is known. One simply modifies each loss to be the point-wise maximum of the loss function and its value at the optimal point. Moreover, one only requires the knowledge of this optimal loss value for every function and not the location in parameter space of the minimiser. Hence, if one is able to approximate the loss values at an optimal point with reasonable accuracy, one should be able to replicate the desirable characteristics of algorithms such as ALI-G and SPS. Specifically, we will be able to obtain an algorithm with a single fixed hyperparameter that is easy to tune, and a strong generalisation performance. We present an optimisation method that approximates the optimal function values online using a heuristic in combination with a Polyak step-size. We name our algorithm Adaptive ALI-G (ALIG+), as it makes use of ALI-G iteratively to update the parameters.

Related Works. Existing optimisation methods for supervised learning tasks that do not satisfy the interpolation property can be broadly classified into three categories: SGD with a manually tuned step size schedule, line search methods, and adaptive gradient methods. While SGD [16] has been used to produce state of the art performance for supervised learning tasks, the best performing models are trained with highly refined learning rate schedules. However, for settings where good learning rate schedules are not known in advance, finding such a schedule can be prohibitively expensive, in compute or practitioner time or both. Line search methods [5, 13, 18] seem appealing alternatives as they remove the need to find a learning rate schedule and instead run extra forward passes to select a step-size. However, these methods invariably introduce extra computation and hyperparameters governing how the line search is performed and whether a target point is accepted. Lastly, adaptive gradient methods such as Adagrad [4], Adam [7] and [11] use heuristics based on previous gradient evaluations to scale a learning rate for each parameter independently. While these methods are easy to use, their generalisation performance is typically poor for supervised learning tasks [19].

2. Problem Formulation

Loss Function. As is standard for supervised learning, we consider tasks where the model is parameterized by $\mathbf{w} \in \mathbb{R}^p$. We assume the objective function can be expressed as an expectation over $z \in \mathcal{Z}$, where z is a random variable indexing the samples of the training set $\mathcal{Z}$:

$$f(\mathbf{w}) \triangleq \mathbb{E}_{z \in \mathcal{Z}}[\ell_z(\mathbf{w})]. \tag{1}$$

Here ℓ_z is the loss function associated with the sample z. We assume that each ℓ_z admits a known lower bound B. For the large majority of loss functions used in machine learning, such as cross-entropy or hinge losses, the lower bound is $B = 0$. However, we do not assume this lower bound is reached during training. In other words, the interpolation property does not hold.

Learning Task. We consider the task of finding a vector $\mathbf{w}_\star$ that minimises f:

$$\mathbf{w}_\star \in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^p} f(\mathbf{w}) + \frac{\lambda}{2}||\mathbf{w}||^2, \tag{$\mathcal{P}$}$$

where λ controls the regularisation amount. We used weight decay for convenience as it allows for simple comparison with other algorithms. However, ALIG+ can easily be used with other forms of regularisation.

3. Algorithm

Our algorithm is motivated by trying to approximate $\ell_z(\mathbf{w}_\star)$ online, and as a result recover interpolation. Thus, we introduce a scalar $\tilde{\ell}_z^k$ to store our estimate for each example in the training set. We refer to these scalars as approximate optimal values (AOVs) and the superscript k indicates how many times the approximation has been updated. Our algorithm alternates between two "steps" i) using the current approximation of the optimal loss $\tilde{\ell}_z^k$ to inform the step-size, (see Algorithm 1); and ii) improving the approximations based on the best previous iterates, (see Algorithm 2).

Updating the Parameters. ALIG+ uses the same stochastic version of the Polyak step-size as ALI-G [1], However, we replace the optimal loss value which is set to zero $\ell_z(\mathbf{w}_\star) = 0$ with its current approximation $\ell_z(\mathbf{w}_\star) = \tilde{\ell}_z^k$. Hence, at time t the ALIG+ algorithm uses the following weight update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma_t \boldsymbol{g}_t', \qquad \gamma_t \triangleq \max\left\{\min\left\{\eta, \frac{\ell_{z_t}(\mathbf{w}_t) - \tilde{\ell}_z^k}{\|\nabla_{\mathbf{w}} \ell_{z_t}\|^2}\right\}, 0\right\}. \tag{2}$$

We define $\boldsymbol{g}_t' \triangleq (\nabla_{\mathbf{w}} \ell_{z_t}(\mathbf{w}_t) + \lambda \mathbf{w}_t)$. Additionally η and λ are the hyperparameters controlling the maximum step-size and weight decay amount, respectively. As we do not require the interpolation assumption to hold, we do not need to use the constraint based regularisation of ALIG+ . It is worth noting that the max with 0 is important as there is no guarantee that $(\ell_{z_t}(\mathbf{w}_t) - \tilde{\ell}_z^k)$ will be positive. Without this positivity constraint a negative step size could be used resulting in a gradient ascent step.

Algorithm 1: ALI-G with AOVs

Input: time horizon T, initial point $\mathbf{w}_0$, maximum step size η, AOVs $\tilde{\boldsymbol{\ell}}^k$ and λ.

for $t = 0, ..., T-1$ **do**
- Sample $z_t \in \mathcal{Z}$, $\ell_{z_t}(\mathbf{w}_t)$, $\nabla \ell_{z_t}(\mathbf{w}_t)$
- Set $\gamma_t \triangleq \max\left\{\min\left\{\eta, \frac{\ell_{z_t}(\mathbf{w}_t) - \tilde{\ell}_z^k}{\|\nabla_{\mathbf{w}} \ell_{z_t}\|^2}\right\}, 0\right\}$
- $\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma_t(\nabla \ell_{z_t}(\mathbf{w}_t) + \lambda \mathbf{w}_t)$

end

Return $\bar{\mathbf{w}} \approx \text{argmin}_{t \in \{1,...,T\}} \{f(\mathbf{w}_t)\}$

Updating the AOVs. To replicate the performance of algorithms for interpolation we want the approximation $\tilde{\ell}_z^k$ to tend towards $\ell_z(\mathbf{w}_\star)$ throughout training. Due to the stochastic and non-convex nature of training neural networks it is impossible to guarantee this behaviour. However, we present a simple scheme for updating the AOVs that demonstrates strong empirical performance as shown in section 4. The AOV update scheme is designed to be both reactive and optimistic. By reactive we mean that if a specific sample returns a constant loss its AOV will tend toward this value. By optimistic we mean that the AOVs are updated to a lower value than the current best loss value for this sample, in the hope that a further decrease in loss is possible. In practice the loss values of specific samples can fluctuate throughout training especially when data augmentation is used. However, the step-size is calculated on a batch of data, and hence is relatively robust to this noise. The AOV update scheme is as follows: we store the vectors $\tilde{\boldsymbol{\ell}}^k, \tilde{\boldsymbol{\ell}}^{k-1}, \boldsymbol{\ell}(\bar{\mathbf{w}})$ that contain $(\tilde{\ell}_z^k, \tilde{\ell}_z^{k-1}, \ell_z(\bar{\mathbf{w}}))\ \forall z \in \mathcal{Z}$, where $\bar{\mathbf{w}} = \text{argmin}_{t \in \{0,...,T\}} \{f(\mathbf{w}_t)\}$ in memory at all time. The AOVs $\tilde{\ell}_z^k$, $\tilde{\ell}_z^{k-1}$ are initialised to our known lower bound on the loss B. The training duration is split into K equal

sections each with length T. During each of these sections we keep the AOVs fixed and try to get a good estimate of $\ell_z(\bar{\mathbf{w}})$ for each example. After each of the K sections we update all AOVs simultaneously. Each AOV is updated depending on whether it has been "reached", that is, if $(\ell_z(\bar{\mathbf{w}}) \leq \tilde{\ell}_z^k)$ is true. In both cases we are optimistic that the loss can be decreased further from its current value. If an AOV hasn't been reached it is updated by simply averaging $\ell_z(\bar{\mathbf{w}})$ and $\tilde{\ell}_z^k$. This increases this AOV to halfway between the loss at the best point visited and its current value. However, if $(\ell_z(\bar{\mathbf{w}}) \leq \tilde{\ell}_z^k)$ is true, we instead try decreasing $\tilde{\ell}_z^k$ halfway to the last AOV that was reached $\tilde{\ell}_z^{k-1}$. If the z^{th} AOV is reached again in successive sections we reduce $\tilde{\ell}_z^k$ each time by the same magnitude. Thus, even if an AOV is incorrectly updated to a value far higher than $\ell_z(\mathbf{w}_\star)$ it can easily be corrected by consecutive reductions. Lastly, we ensure AOVs are never decreased below the lower bound B. Therefore when using a non-negative loss the AOVs are always positive.

Algorithm 2: ALIG+ Algorithm

Input: Input: time horizon T_{max}, $K = 5$, initial point $\bar{\mathbf{w}}_0$, η $\tilde{\boldsymbol{\ell}}^1, \tilde{\boldsymbol{\ell}}^0 = B$, and λ.

1 **for** $k = 1, ..., K$ **do**
2 Run Algorithm 1 with $\bar{\mathbf{w}}^{k-1}$, $\frac{T_{max}}{K}$, $\tilde{\boldsymbol{\ell}}^k$, η and λ to obtain $\bar{\mathbf{w}}^k$
3 **for** $z \in \mathcal{Z}$ **do**
4 **if** $\ell_z(\bar{\mathbf{w}}) \leq \tilde{\ell}_z^k$ **then**
5 $\tilde{\ell}_z^{k+1} \leftarrow \max\{\frac{\tilde{\ell}_z^k + \tilde{\ell}_z^{k-1}}{2}, 0\}, \quad \tilde{\ell}_z^k \leftarrow \tilde{\ell}_z^{k+1} - \tilde{\ell}_z^{k-1}$
6 **else**
7 $\tilde{\ell}_z^{k+1} \leftarrow \frac{\tilde{\ell}_z^k + \ell_z(\bar{\mathbf{w}})}{2}, \quad \tilde{\ell}_z^k \leftarrow \tilde{\ell}_z^{k-1}$
8 **end**
9 **end**
10 **end**
11 Return $\bar{\mathbf{w}}_K$

Implementation Details. For the above scheme to work well, it is important that the AOVs are not updated too frequently as this can lead to them trending towards $\ell_z(\bar{\mathbf{w}})$ too fast. However, it is also important that the AOVs are updated a sufficient number of times so they can approximate $\ell_z(\mathbf{w}_\star)$, if $\ell_z(\mathbf{w}_\star)$ is large. We find $K = 5$ provides a good balance between these considerations and fix K to this value. Furthermore, to save computation we i) avoid calculating $f(\mathbf{w}_t)$ exactly and instead approximate this online after each epoch and ii) we use $\mathbf{w}_T^{k-1}$ in the place of $\bar{\mathbf{w}}_k$ in line 2 of Algorithm 2. This results in ALIG+ having a similar run time to SGD, where the only extra computation is the updating of the vectors $\tilde{\boldsymbol{\ell}}^k, \tilde{\boldsymbol{\ell}}^{k-1}, \boldsymbol{\ell}(\bar{\mathbf{w}})$ and evaluating the norm of the gradients.

Data Augmentation. Data augmentation can be thought of in two ways. First, it increases the size of the data-set by adding new examples that are simply transformed versions of others. Second, it makes online alterations to the original number of examples. As ALIG+ is designed for the optimisation of non-interpolating problems, which often have large data sets, we choose to view data augmentation in the second way and save only a single AOV for all possible augmentations. When viewing data augmentation in the first way, training regimes where the number of epochs is less than the number of possible transformations would only visit each example less than once on average. Hence, approximating the optimal value would be challenging. Moreover, for many common data

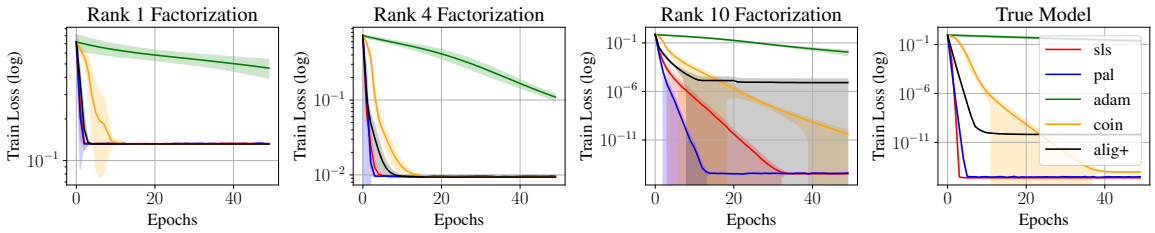

Figure 1: Training performance on the matrix factorisation problem of [18].

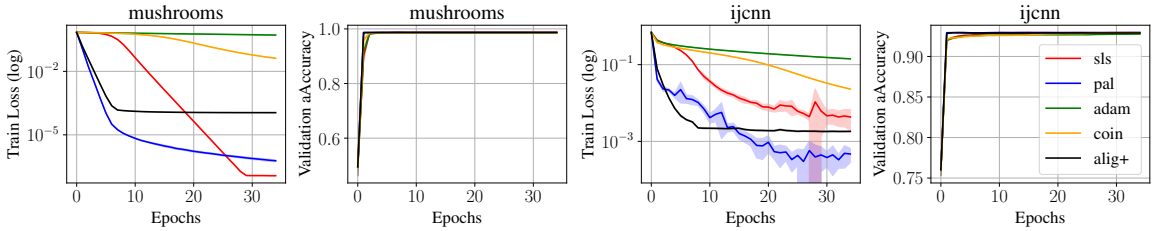

Figure 2: Training and validation performance on mushrooms and ijcnn data sets [2].

augmentation transforms, such as random crops of images, we would expect the optimal loss value to be highly correlated between the same example under different versions of the transform.

4. Experiments

4.1. Simple Optimisation Benchmarks

Setting and Method. We first demonstrate the performance of ALIG+ on the matrix factorisation and RBF Binary Classification tasks detailed in [18]. We compare ALIG+ against Parabolic Approximation Line Search (PAL) [13] and a selection of the optimisation methods used in [18].

Results. The results of these experiments are shown in Figures 1 and 2. On the non-interpolating tasks rank 1 and rank 4 matrix factorisation and ijcnn, ALIG+ performs comparably to the best algorithms, PAL [13] and SLS [18]. On the interpolating tasks ALIG+ attains the same validation performance as these methods, however it fails to minimise the training loss to machine precision.

4.2. Deep Learning Experiments

Setting. For deep learning experiments we train a small 8 layer ResNet [6] on SVHN [14], CIFAR10, CIFAR100 [8] and Tiny ImageNet. The small ResNet contains 90K parameters with 16 channels in the first layer. We also train a ResNet 18 [6] on ImageNet [3] containing 11.7M parameters. The SVHN and CIFAR data sets are comprised of 32x32 pixel RBG images. For the SVHN data set we use the split proposed in [1]. SVHN and CIFAR10 both have 10 classes and CIFAR100 has 100. The Tiny ImageNet data set is more challenging and contains 100K training examples of 64x64 pixels split over 200 classes. The ImageNet data set [3] contains 1.2M large RGB images of various sizes split over 1000 classes. For our experiments we use the following data augmentation. All images are normalised per channel, randomly cropped to 224x224 pixels. For the Tiny ImageNet and ImageNet data set the ground truth labels are not freely available so we report validation scores instead. All images are centred and normalised per channel and when

data augmentation is used we apply standard random flips and crops. For the majority of data sets we present results with and without data augmentation. The exceptions being SVHN, which is not designed for data augmentation and ImageNet where we only report results with augmentation. These tasks were chosen to give examples of i) interpolation (SVHN); ii) near interpolation (CIFAR10); iii) non-interpolation resulting from limited model size (CIFAR100 and Tiny ImageNet) and non-interpolation resulting from data set size (ImageNet).

Method. We compare ALIG+ against PAL [13], ALI-G [1], SPS [10], the optimisation methods used in [18] and SGD with a step learning rate schedule. For these problems the step-size or maximum step-size hyperparameter is cross validated as powers of ten and the regularisation hyperparameter is selected from $\lambda \in \{1^{-3}, 1^{-4}, 1^{-5}, 0\}$. ALI-G uses a constraint based regularisation [1], and r was selected from $r \in \{50, 100, 200, \infty\}$. All other hyperparameters are left at their default values. For PAL we keep the maximum step-size hyperparameter within the suggested interval $[1, 10]$. For SGD we use the learning rate schedules detailed in [6]. We reuse the schedule for the CIFAR data sets for SVHN and Tiny ImageNet, reducing the learning rate by a factor of 10 both half way and three quarters through training. A fixed batch size of 128 and a epoch budget of 200 are used for all experiments. We accelerate SGD, ALIG and ALIG+ with a Nesterov momentum of 0.9. SLS_{Polyak}, Adam and Adabound also include momentum like terms which we leave at their default settings. Due to computational constraints we use the best hyperparameters from Tiny ImageNet for ImageNet. However, the batch size is increased to 256 and the epoch budget is reduced to 90.

Results. The accuracy of the best performing model for each optimisation method is shown in Table 1. On the tasks considered, ALIG+ outperforms all line search and adaptive gradient methods by a significant margin. The exception being on CIFAR10 with data augmentation where PAL produced similar test accuracy. The dominant performance of ALIG+ shows the lack of strong algorithms for non-interpolating settings. The performance benefit of ALIG+ is most notable when the interpolation property is far from satisfied or when data augmentation is not used. For example on the challenging Tiny Imagenet data set ALIG+ produces validation accuracy 4% higher that the next best. Empirically we observe that ALIG+ is almost twice as fast as the line search methods, significantly faster than Adam and has a run time within a couple of percent of SGD.

4.3. Worse Case Bound

In this section we provide a worst case bound on the performance of ALIG+ for smooth and convex functions. As ALIG+ and ALI-G are equivalent before the first AOV update, we provide a result based on the work of Berrada et al.[1]. Due to space constraints, we refer the reader to [1] for a proof.

Theorem 1 *We assume that for every $z \in \mathcal{Z}$, ℓ_z is convex and β-smooth. Let $\mathbf{w}_\star$ be a minimum of f such that $\forall z \in \mathcal{Z}$, $\ell_z(\mathbf{w}_\star) \leq \varepsilon$. Further assume that $\eta \leq \frac{1}{2\beta}$ and $\lambda = 0$. Then if we apply ALIG+ with a maximal learning-rate of η to f, we have:*

$$\left[f\left(\frac{1}{T+1} \sum_{t=0}^{T} \mathbf{w}_t \right) \right] - f_\star \leq \frac{\|\mathbf{w}_0 - \mathbf{w}_\star\|^2}{\eta(T+1)} + \varepsilon.$$

Note here $T = \frac{T_{max}}{K}$ and does not have the same meaning as in [1].

	SVHN Test Acc (%)	Cifar10 Test Acc (%)		Cifar100 Test Acc (%)		Tiny ImageNet Val Acc (%)		ImageNet Val Acc (%)
Model	Small ResNet							ResNet18
Data Aug	No	No	Yes	No	Yes	No	Yes	Yes
SGD_{Step}	95.5	84.2	88.1	51.0	59.6	39.8	43.2	71.1
$ALIG$	93.7	80.8	86.2	47.5	57.9	35.6	41.8	63.7
SPS	93.9	81.1	86.8	43.7	53.1	20.5	23.8	63.9
$Adabound$	93.1	75.6	85.2	44.0	55.4	34.3	40.1	62.9
$Adam$	94.3	79.7	85.8	48.1	56.2	35.5	41.2	62.6
$Coin$	92.4	76.2	84.1	42.4	54.0	30.8	36.3	61.5
SLS_{Armijo}	92.9	81.2	85.8	31.6	42.0	9.4	10.0	63.2
$SLS_{Goldstein}$	92.1	78.2	86.4	45.5	57.2	32.9	40.4	62.6
SLS_{Polyak}	93.6	79.9	85.9	43.6	54.0	31.4	38.0	62.7
PAL	93.0	81.5	86.7	39.8	57.0	35.3	40.8	63.6
$ALIG+$	**95.3**	**86.0**	**87.3**	**56.3**	**59.6**	**39.8**	**42.8**	**67.7**

Table 1: *Accuracies of optimisation methods on a selection of standard image classification data sets. The model and dataset combinations have been chosen to include both interpolating and non-interpolating tasks. SGD_{step} is the only method to benefit from a manually designed step size schedule. All other methods have at most one fixed step size hyperparameter. On these tasks ALIG+ out performs all other single hyperparameter methods, often by a large margin.*

5. Discussion

We have introduced ALIG+ , an optimisation algorithm designed for the non-interpolating setting and demonstrated its superiority over linear search and adaptive gradient methods on standard benchmarks. However, due to stochasticity it is theoretically possible to design simple convex and Lipshitz continuous problems where the AOVs will not converge to $\ell_z(\mathbf{w}_\star)$ in the worst case. An interesting direction for future work would be to characterise the conditions where ALIG+ offers provable convergence to the global optimum.

References

[1] Leonard Berrada, Andrew Zisserman, and M Pawan Kumar. Training neural networks for and by interpolation. *International Conference on Machine Learning*, 2020.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2011.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *Conference on Computer Vision and Pattern Recognition*, 2009.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

[5] Zhiyong Hao, Yixuan Jiang, Huihua Yu, and Hsiao-Dong Chiang. Adaptive learning rate and momentum for training deep neural networks. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*, 2021.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition*, 2016.

[7] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

[8] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[9] Chaoyue Liu and Mikhail Belkin. Accelerating sgd with momentum for over-parameterized learning. *International Conference on Learning Representations*, 2019.

[10] Nicolas Loizou, Sharan Vaswani, Issam Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. *International Conference on Artificial Intelligence and Statistics*, 2021.

[11] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *International Conference on Learning Representations*, 2019.

[12] Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. *International Conference on Machine Learning*, 2018.

[13] Maximus Mutschler and Andreas Zell. Parabolic approximation line search for dnns. *Neural Information Processing Systems*, 2020.

[14] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *Neural Information Processing Systems*, 2011.

[15] Boris Teodorovich Polyak. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, 1969.

[16] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, 1951.

[17] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. *International Conference on Artificial Intelligence and Statistics*, 2019.

[18] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. *arXiv preprint*, 2019.

[19] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Neural Information Processing Systems*, 2017.