FAKING INTERPOLATION UNTIL YOU MAKE IT

Anonymous authors
Paper under double-blind review

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. We introduce a novel extension of 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. This heuristic starts by setting the approximate optimal values to a known lower bound on the loss function, typically zero. It then updates them at fixed intervals through training in the direction of the best iterate visited so far. We provide rigorous experimentation on a wide range of problems including two natural language processing tasks, popular vision benchmarks and the challenging ImageNet classification data set. From our empirical analysis we demonstrate the effectiveness of our approach, which in the non-interpolating setting, outperforms state of the art baselines, namely adaptive gradient and line search methods.

1 INTRODUCTION

Deep over-parameterized neural networks exhibit the interpolation property on many data sets (Vaswani et al., 2019b; Berrada et al., 2020). That is, these models are able to achieve close to zero loss on all samples simultaneously. Recently, this property has been exploited to prove new convergence rates (Vaswani et al., 2019a;b; Ma et al., 2018; Liu & Belkin, 2019) and to develop novel optimisation algorithms for this setting. Examples include Adaptive learning rates for Interpolation with Gradients (ALI-G) (Berrada et al., 2020), and Stochastic Polyak Step (SPS) (Loizou et al., 2021). 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 (Polyak, 1969). This automatically scales a maximal step-size hyperparameter down to an appropriate value for each update, which removes the need for a learning rate schedule (Berrada et al., 2020; Loizou et al., 2021). 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 hyperparameters 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 (AALIG), as it makes use of ALI-G iteratively to update the parameters.

We conduct a thorough empirical evaluation of AALIG on a variety of tasks against strong baselines. We provide results for matrix factorisation, binary classification using RBF kernels, image classification on the SVHN, CIFAR, Tiny ImageNet and ImageNet datasets, and review classification and next character prediction. The majority of tasks are designed to provide a mix of non interpolating and interpolating settings. AALIG outperforms all line search and adaptive gradient methods on tasks where the interpolation assumption does not hold. These results demonstrate that estimating the optimal loss value online is an effective alternative approach for selecting the step-size.

2 Related Works

We discuss existing optimisation methods for supervised learning tasks that do not satisfy the interpolation property. The approaches can be broadly classified into three categories: SGD with a manually tuned learning rate schedule, line search methods, and adaptive gradient methods.

SGD with a Learning Rate Schedule. SGD (Robbins & Monro, 1951) has been used to produce state of the art performance for supervised learning tasks. However, the downside of SGD is that it requires the manual design and refinement of a learning rate schedule for best performance. Many forms of schedule have been proposed in the literature, including piecewise constant (Huang et al., 2017), geometrically decreasing (Szegedy et al., 2015) and warm starts with cosine annealing (Loshchilov & Hutter, 2017). Consequently, practitioners who wish to use SGD in a novel setting need to select which type of schedule to use for their learning task. To that end, they first need to choose the parameterization of the schedule and then tune the corresponding hyperparameters. For example, a piecewise linear scheme requires an initial learning rate value, a decay factor and a list or metric to determine at which points in training to decay the learning rate. This results in a large search space which increases exponentially in combination with other problem dependent quantities such as regularisation amount or batch size. As SGD can be sensitive to these hyperparameters, and their optimal values often are highly interdependent, the resulting cross-validation scheme necessary for best results can be prohibitively expensive.

Line Search Methods. Line search methods, such as those developed by Vaswani et al. (2019b); Mutschler & Zell (2020); Hao et al. (2021) offer an appealing alternative to SGD as they remove the need to find a learning rate schedule and instead run extra forward passes to select a step-size. While not specifically designed for settings where interpolation does not hold, Vaswani et al. (2019b) present algorithms based around the Armijo and Goldstein line-search methods, classically used for deterministic gradient descent. They also introduce heuristics with the aim of minimising the number of extra forward passes required, which reduces the average number required to one per batch. Mutschler & Zell (2020) and Hao et al. (2021) instead assume the loss function is approximately parabolic in the negative gradient direction, and thus use extra forward passes to construct a parabolic model of the loss that can then be minimised in closed form. Mutschler & Zell (2020) additionally provide empirical justification for the parabolic approximation. While line-search methods present strong performance they invariably introduce extra hyperparameters governing how points are selected in the line search or whether a target point is accepted. While these hyperparamters are held fixed over training and do not require a schedule they must be tuned for best results. Furthermore, line search methods require approximately twice the computation per batch of typical first order methods resulting in a far longer training time.

Adaptive Gradient Methods Adaptive gradient methods such as Adagrad (Duchi et al., 2011), Adam (Kingma & Welling, 2014) or more recently Adabound (Luo et al., 2019) use heuristics based on previous gradient evaluations to scale a learning rate for each parameter independently. These algorithms are easy to use as they require a single fixed learning rate hyperparameter that tends to provide decent results over a wide range of values Sivaprasad et al. (2020). However, once tuned, non-adaptive optimisation algorithms such as ALI-G and SGD provide superior generalisation performance over adaptive gradient methods on a range of supervised learning benchmarks (Berrada et al., 2020; Wilson et al., 2017).

3 PRELIMINARIES

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

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

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 feasible vector $\boldsymbol{w}_\star \in \Omega$ that minimises f:

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

where λ controls the regularisation amount. We use weight decay for convenience as it allows for simple comparison with other algorithms, however AALIG can easily be use with other forms of regularisation. For unconstrained problems, like those considered in this paper, we set to $\Omega = \mathbb{R}^p$.

The ALI-G Algorithm. AALIG is inspired by the ALI-G algorithm (Berrada et al., 2020) and hence we formally introduce this method here. ALI-G was designed for the optimisation of interpolating problems, that is, problems where $f(\boldsymbol{w}_\star) = B = 0$. This condition also trivially implies that $\ell_z(\boldsymbol{w}_\star) = 0, \forall z \in \mathbb{Z}$. In order for the interpolation assumption to hold for ($\mathcal{P}$) when regularisation is used, ALI-G does not apply it in the conventional way, and hence $\lambda = 0$. Instead, ALI-G makes use of a constraint based regularisation, where the feasible set is defined as $\Omega = \left\{\boldsymbol{w} \in \mathbb{R}^p : \ \|\boldsymbol{w}\|_2^2 \leq r\right\}$ and r controls the regularisation level. In order to ensure that only feasible solutions are found the iterate is projected back onto the set Ω after each update. At time step t a sample z_t, or in practice a mini-batch, is sampled from $\mathbb{Z}$ and the loss and gradient is evaluated at the current interate $\boldsymbol{w}_t$. ALI-G then selects $\boldsymbol{w}_{t+1}$ as the solution to following the proximal problem:

$$\operatorname*{argmin}_{\boldsymbol{w} \in \Omega} \left\{ \frac{1}{2\eta}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + \max\{0, \ell_{z_t}(\boldsymbol{w}_t) + \nabla_{\boldsymbol{w}} \ell_{z_t}(\boldsymbol{w}_t)^\top (\boldsymbol{w} - \boldsymbol{w}_t)\} \right\}, \tag{2}$$

where η is the step-size hyperparameter. This proximal problem is identical to that solved in closed form by the SGD update, with a minor modification. The problem (2) additionally includes a point-wise maximum between the linear approximation of the loss and the known lower bound ($\ell_z(\boldsymbol{w}_\star) = 0$). The dual of (2) is a maximisation over a concave function in one dimension constrained to the interval $[0, 1]$. Hence, one can obtain the optimal point by projecting the unconstrained solution onto the feasible region. After some simplification this results in the following closed form solution:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t \nabla_{\boldsymbol{w}} \ell_{z_t}(\boldsymbol{w}_t), \quad \text{where} \quad \gamma_t \triangleq \max\left\{ \min\left\{ \eta, \frac{\ell_{z_t}(\boldsymbol{w}_t) - \ell_z(\boldsymbol{w}_\star)}{\|\nabla_{\boldsymbol{w}} \ell_{z_t}\|^2} \right\}, 0 \right\}. \tag{3}$$

This update can be viewed as a stochastic analog of the Polyak step-size (Polyak, 1969), with the addition of a maximal value η. From the interpolation assumption, we have $\ell_z(\boldsymbol{w}_\star) = 0, \forall z \in \mathbb{Z}$ and hence the numerator of the fraction in (3) can be simplified to $\ell_{z_t}(\boldsymbol{w}_t)$. Additionally the maximum with zero is redundant as both numerator and denominator of the fraction will always be positive due to the non-negative nature of the loss function. We show both redundant pieces of notation here as it allows us to clearly specify our modified version in the next section. The ALI-G update is computationally cheap with the evaluation of the norm of the gradients being the only extra computation required over SGD. Importantly, ALI-G removes the need for a learning rate and performs comparably on many benchmarks (Berrada et al., 2020).

4 TRAINING IN NON-INTERPOLATING SETTINGS

While the interpolation setting has received a lot of attention from recent work, many interesting problems do not satisfy this assumption. This could be for any of the following reasons: i) the

model size could be limited due to hardware or power constraints, such as for embedded devices; ii) the data set is very large, for example, the vast majority of models trained on the ImageNet data set (Deng et al., 2009) do not achieve zero training loss; iii) complexity of the loss function, such as in adversarial training; iv) label noise can make interpolation impossible by creating one to two mappings between inputs and labels. Thus, we think this setting is deserving of bespoke optimisation algorithms that are easy to use and produce strong generalisation performance.

Motivation. Our algorithm is motivated by trying to approximate $\ell_z(\boldsymbol{w}_\star)$ 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 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). We describe these steps in detail in the following two sections.

Updating the Parameters. AALIG uses the same stochastic version of the Polyak step-size as ALI-G (Berrada et al., 2020), However, we replace the optimal loss value $\ell_z(\boldsymbol{w}_\star) = 0$ with its approximation $\tilde{\ell}_z^k$. Hence, at time t the AALIG algorithm uses the following weight update:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t \boldsymbol{g}_t, \quad \text{where} \quad \gamma_t \triangleq \max\left\{\min\left\{\eta, \frac{\ell_{z_t}(\boldsymbol{w}_t) - \tilde{\ell}_z^k}{\|\nabla_{\boldsymbol{w}} \ell_{z_t}\|^2}\right\}, 0\right\}. \tag{4}$$

We define $\boldsymbol{g}_t \triangleq (\nabla_{\boldsymbol{w}} \ell_{z_t}(\boldsymbol{w}_t) + \lambda \boldsymbol{w}_t)$, where 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 AALIG, and can simply make use of weight decay. AALIG can use other forms of regularisation, we use weight decay for convenience as it allows for easy comparison with other algorithms. It is worth noting here that the max with 0 is no longer redundant as there is no guarantee that $(\ell_{z_t}(\boldsymbol{w}_t) - \tilde{\ell}_z^k)$ will be positive. Without this positivity constraint a negative step size could be used resulting in a gradient ascent step. Moreover if $\ell_{z_t}(\boldsymbol{w}_t)$ is already lower than its AOV $\tilde{\ell}_z^k$ we have achieved the approximate optimal value for this sample and no more progress is needed until the AOV is updated, which we describe in the next section. The full procedure for updating the parameters given the AOVs is outlined in Algorithm 1.

Algorithm 1 ALI-G with AOV's

Input: time horizon T, initial point $\boldsymbol{w}_0$, maximum step-size η, AOVs $\tilde{\boldsymbol{\ell}}^k$ and λ.
for $t = 0, ..., T-1$ **do**
  Sample $z_t \in \mathbb{Z}$, $\ell_{z_t}(\boldsymbol{w}_t)$, $\nabla_{\boldsymbol{w}} \ell_{z_t}(\boldsymbol{w}_t)$
  Set $\gamma_t = \max\left\{\min\left\{\eta, \frac{\ell_{z_t}(\boldsymbol{w}_t) - \tilde{\ell}_z^k}{\|\nabla_{\boldsymbol{w}} \ell_{z_t}\|^2}\right\}, 0\right\}$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t(\nabla_{\boldsymbol{w}} \ell_{z_t}(\boldsymbol{w}_t) + \lambda \boldsymbol{w}_t)$
end for
Return $\bar{\boldsymbol{w}} \approx \operatorname{argmin}_{t \in \{1,...,T\}} \{f(\boldsymbol{w}_t)\}$

Updating the AOVs. To replicate the performance of algorithms for interpolation we want the approximation $\tilde{\ell}_z^k$ to tend towards $\ell_z(\boldsymbol{w}_\star)$ 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 5. This scheme is motivated by the intuition that easy examples typically both experience a decrease in loss value earlier in training and also finish training with lower final loss. While hard examples which finish with high loss rarely decrease far past this value during training. The left half of figure 1 shows losses over training of an "easy" sample in blue and a "hard" sample in green. Of course not every example in the training set satisfies one of these characterisations, such as those shown in the right half of figure 1.

The AOV update scheme is as follows: we store the vectors $\tilde{\boldsymbol{\ell}}^k, \tilde{\boldsymbol{\ell}}^{k-1}, \boldsymbol{\ell}(\bar{\boldsymbol{w}})$ containing $(\tilde{\ell}_z^k, \tilde{\ell}_z^{k-1}, \ell_z(\bar{\boldsymbol{w}}))\ \forall z \in \mathbb{Z}$, where $\bar{\boldsymbol{w}} = \operatorname{argmin}_{t \in \{0,...,T\}} \{f(\boldsymbol{w}_t)\}$ in memory at all time . The AOVs $\tilde{\ell}_z^k$,

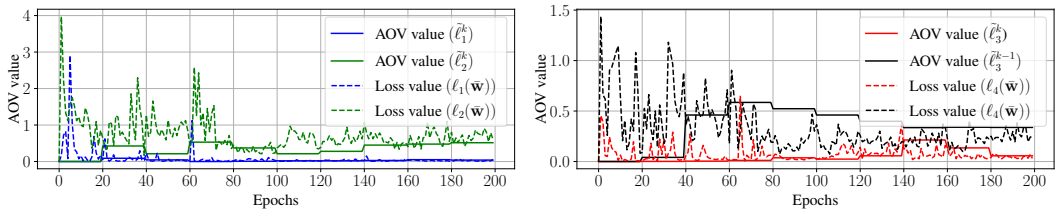

Figure 1: Figure showing four AOVs (solid lines) and their corresponding loss values (dashed lines) evolving during training. **Left:** In blue we show the behaviour of an "easy" sample that quickly reaches zero loss and remains there for the rest of training. In green we show a "hard" example that has high loss throughout training. **Right:** Here we demonstrate the adaptability of our AOV update heuristic, where the red sample increases in value toward the end of training when initially low. Conversely, the black sample is initially too high and decreases in value from epoch 80 to 140.

$\tilde{\ell}_z^{k-1}$ 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 $\ell_z(\bar{\boldsymbol{w}})$ 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"; if $(\ell_z(\bar{\boldsymbol{w}}) \leq \tilde{\ell}_z^k)$ is true. In both cases we are optimistic that the loss can be decreased further from its current value. Hence, if an AOV hasn't been reached it is updated by simply averaging $\ell_z(\bar{\boldsymbol{w}})$ 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 $(\ell_z(\bar{\boldsymbol{w}}) \leq \tilde{\ell}_z^k)$ is true, we instead try decreasing $\tilde{\ell}_z^k$ halfway to the last value that was reached $\tilde{\ell}_z^{k-1}$. An example of this behaviour is shown in green in figure 1 at epoch 40. 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 $\ell_z(\boldsymbol{w}_\star)$ it can easily be corrected by consecutive reductions. An example of this behaviour is shown in black in figure 1 between epochs 80 and 140.

Algorithm 2 AALIG Algorithm

1: **Input**: time horizon T_{max}, $K = 10$, $\bar{\boldsymbol{w}}_0$ and $\tilde{\ell}_z^1, \tilde{\ell}_z^0 = B, \ \ \forall z \in \mathbb{Z}$ and λ.
2: **for** epoch $k = 1, ..., K$ **do**
3: Run Algorithm 1 with $\bar{\boldsymbol{w}}^{k-1}$, $\frac{T_{max}}{K}$, $\tilde{\boldsymbol{\ell}}^k$, η and λ to obtain $\bar{\boldsymbol{w}}^k$.
4: **for** $z \in \mathbb{Z}$ **do**
5: **if** $\ell_z(\bar{\boldsymbol{w}}) \leq \tilde{\ell}_z^k$ **then**
6: $\tilde{\ell}_z^{k+1} \leftarrow \frac{\tilde{\ell}_z^k + \tilde{\ell}_z^{k-1}}{2}, \qquad \tilde{\ell}_z^k \leftarrow \max\{\tilde{\ell}_z^{k+1} - \tilde{\ell}_z^{k-1}, B\}$
7: **else**
8: $\tilde{\ell}_z^{k+1} \leftarrow \frac{\tilde{\ell}_z^k + \ell_z(\bar{\boldsymbol{w}})}{2}, \qquad \tilde{\ell}_z^k \leftarrow \tilde{\ell}_z^{k-1}$
9: **end if**
10: **end for**
11: **end for**
12: Return $\bar{\boldsymbol{w}}_K$

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 $\ell_z(\bar{\boldsymbol{w}})$ too fast. However, it is also important that the AOVs are updated a sufficient number of times so they can approximate $\ell_z(\boldsymbol{w}_\star)$, if $\ell_z(\boldsymbol{w}_\star)$ is large. We find $K = 10$ provides a good balance between these considerations and fix K to this value. Furthermore, to save computation we i) avoid calculating $f(\boldsymbol{w}_t)$ exactly and instead approximate this online after each epoch and ii) we use $\boldsymbol{w}_T^{k-1}$ in the place of $\bar{\boldsymbol{w}}_k$ in line 2 of Algorithm 2. This results in AALIG having a similar run time to SGD, where the only extra computation is the updating of the vectors $\tilde{\boldsymbol{\ell}}^k, \tilde{\boldsymbol{\ell}}^{k-1}, \boldsymbol{\ell}(\bar{\boldsymbol{w}})$ 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 AALIG 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 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. To support this claim we calculate the loss value of all possible crops for a subset of 5000 images chosen from a selection of common data sets. We find empirically, at the start of training that the variance between loss values is on average 20 times lower for the different crops of the same image compared to randomly chosen images. Over training this ratio typically drops to 5 times lower.

5 EXPERIMENTS

We introduced a very simple heuristic for computing the AOVs. We now show through rigorous experiments that this modification is sufficient to produce state of the art generalisation performance for single hyperparameter optimisation algorithms on a large variety of non-interpolating problems. Additionally AALIG shows strong performance in settings where interpolation holds. We start with relatively simple problems such as matrix factorisation and binary classification using RBF kernels. We then consider the training of deep neural networks on popular image classification benchmarks. To show that our approach scales to large problems we also provide results on the ImageNet data set. Furthermore, we do not just show results for computer vision data sets as is common in the literature but also for two NLP tasks, highlighting the flexibility of our approach. All experiments are conducted in PyTorch (Paszke et al., 2017) and are performed on a single GPU except for the ImageNet experiments that use two.

5.1 SIMPLE OPTIMISATION BENCHMARKS

Setting. We first demonstrate the performance of AALIG on the matrix factorisation and the RBF Binary Classification using tasks detailed in Vaswani et al. (2019b). The matrix factorisation task can be expressed as:

$$\min_{\boldsymbol{W}_1, \boldsymbol{W}_2} \mathbb{E}_{\mathbf{x} \in \mathbb{X}}[||\boldsymbol{W}_1, \boldsymbol{W}_2 \mathbf{x} - \boldsymbol{A}\mathbf{x}||^2], \tag{5}$$

where $\mathbb{X}$ is a data set of 1000 examples drawn from $\mathcal{N}(\mathbf{x}; 0, \boldsymbol{I})$, $\boldsymbol{A} \in \mathbb{R}^{10 \times 6}$ is randomly generated to have condition number 10^{10}, $\boldsymbol{W}_1 \in \mathbb{R}^{10 \times k}$, $\boldsymbol{W}_1 \in \mathbb{R}^{k \times 6}$, $\boldsymbol{A}$. The rank of the factorisation k is selected to be one of four different values resulting in two problems where interpolation holds and two where it does not. The binary classification with radial basis functions tasks use the mushrooms and ijcnn dataset from the LIBSVM library of SVM problems (Chang & Lin, 2011). The mushrooms dataset satisfies the interpolation assumption, whereas ijcnn does not.

Method. We compare AALIG against Parabolic Approximation Line Search (PAL) (Mutschler & Zell, 2020) and a selection of the optimisation methods used in Vaswani et al. (2019b), and we reuse their code for the baselines. These optimisation algorithms contain a collection of strong line search and adaptive gradient methods, all of which do not require a learning rate schedule. Additionally, the majority have a single step-size hyperparameter which makes for fair comparison with AALIG .

Results. The results of these experiments are shown in Figures 2 and 3. On the non-interpolating tasks rank 1 and rank 4 matrix factorisation and ijcnn, at all time. On the interpolating tasks AALIG fails to minimise the training loss to machine precision like PAL (Mutschler & Zell, 2020) and SLS (Vaswani et al., 2019b), however it attains the same validation performance.

5.2 SMALL IMAGE CLASSIFICATION EXPERIMENTS

Setting. We run experiments on a broad range of image classification benchmarks. Specifically we use the SVHN (Netzer et al., 2011), CIFAR10, CIFAR100 (Krizhevsky, 2009) and Tiny Ima-

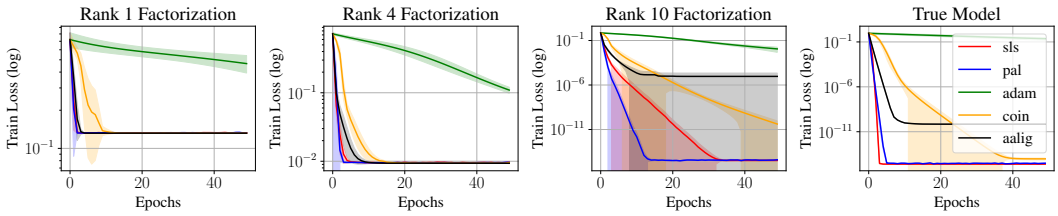

Figure 2: Training performance on the matrix factorisation problem of Vaswani et al. (2019b). In the settings where interpolation does not hold, namely the Rank 1 and Rank 4 problems, AALIG quickly achieves the loss floor. For the Rank 10 and True model problems AALIG does not minimise the loss to machine precision, such as SLS (Vaswani et al., 2019b) and PAL (Mutschler & Zell, 2020), however, it still provides rapid optimisation to at worst $> 10^{-4}$.

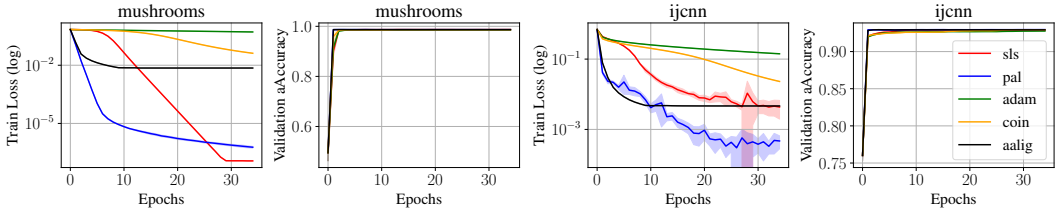

Figure 3: Training and validation performance on the mushrooms and ijcnn data sets (Chang & Lin, 2011). On the mushroom data set, where interpolation holds, AALIG fails to achieve the same training loss as the line search methods. However, in both non-interpolating and interpolating settings AALIG obtains equally good validation performance as the best baseline.

geNet data sets. We next give a brief description of each of these data sets. The SVHN and CIFAR data sets are comprised of 32x32 pixel RBG images. For the SVHN data set we use the split proposed in Berrada et al. (2020) resulting in 598k training, 6k validation and 26k test samples. 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. For the Tiny 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. For all data sets we make use of the cross entropy loss to train a small 8 layer ResNet (He et al., 2016) containing 90K parameters with 16 channels in the first layer. These tasks were chosen to give examples of i) interpolation (SVHN); ii) near interpolation (CIFAR10) and iii) non-interpolation resulting from limited model size (CIFAR100 and Tiny ImageNet).

Method. We again compare AALIG against PAL (Mutschler & Zell, 2020), the optimisation methods used in Vaswani et al. (2019b) 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 $\lambda \in \{1^{-3}, 1^{-4}, 1^{-5}, 0\}$. 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 He et al. (2016). 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. As is common for deep learning experiments we accelerate SGD and AALIG with a Nesterov momentum of 0.9. SLS$_{Polyak}$, Adam and Adabound also include momentum like terms which we leave at their default values.

Results. The accuracy of the best performing model for each optimisation method is shown in Table 1. On the tasks considered, AALIG 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 AALIG shows the lack of strong

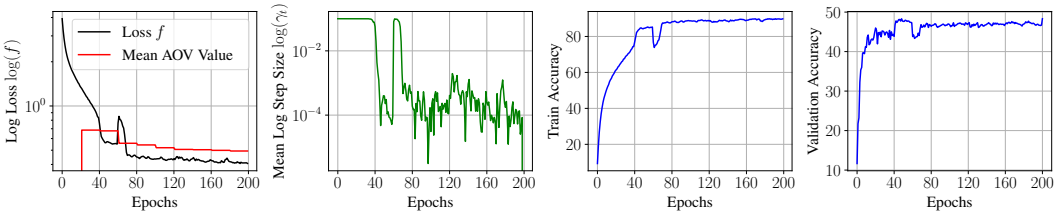

Figure 4: Curves produced by training a small ResNet on CIFAR100 with the AALIG optimiser. The AOVs are updated every 20 epochs. At the first update the mean AOV value increases significantly, however, by the second update it remains almost constant due to a portion AOVs increasing in value and others decreasing. Until epoch 40 the loss for each sample is significantly higher than the AOVs and thus the maximum step-size ($\eta = 0.1$) is used for all updates. Shortly after epoch 40 the mean loss becomes larger than the AOV value, and hence the step size used for many batches becomes zero, causing a lower mean value. At epoch 60 many of the AOVs are reduced as they have been "reached" resulting in the step size increasing for roughly 10 epochs, until roughly epoch 70. For the rest of training the AOVs begin to stabilise and the loss is slowly decreased as the optimiser focuses on samples where the current AOV has not been reached.

algorithms for non-interpolating settings. The performance benefit of AALIG is most notable when the interpolation property is far from satisfied and when data augmentation is not used. For example on the challenging Tiny Imagenet data set AALIG produces validation accuracy 4% higher that the next best. Empirically we observe that AALIG is almost twice as fast as the line search methods and has the same run time as SGD and adaptive gradient methods. Typical training curves for AALIG are shown in figure 4.

	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
$Adabound$	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
$AALIG$	**95.0**	**85.0**	**86.8**	**50.4**	**58.0**	**39.9**	**42.3**	**67.1**

Table 1: *Accuracy of single hyperparameter optimisation methods.*

5.3 Large Image Classification Experiments

Setting. The ImageNet data set (Deng et al., 2009) 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 and horizontal flips are applied with probability 0.5. For validation a centre crop is used and no flips are performed. For ImageNet the ground truth labels are not freely available so we report validation scores instead. We train a ResNet18 containing 11.7M parameters (He et al., 2016). Due to the large number of images and data augmentation the interpolation assumption does not hold.

Method. For SGD we use the learning rate schedule shown in (He et al., 2016). Due to computational constraints for all other methods we reuse 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 validation accuracy of each optimisation method is shown in the last column of table 1. On this task, AALIG outperforms all line search and adaptive gradient methods by at least 3.5%. Additionally AALIG was significantly quicker to train than the next two best performing methods. SLS_{Armijo} and PAL took 20 and 12 hours respectively longer to train than AALIG . This result demonstrates the advantage of AALIG for training on large data sets over comparable techniques.

5.4 NLP EXPERIMENTS

Setting. For NLP Experiments we consider two tasks. The first is binary classification of reviews in the IMDB data set using a bi-directional LSTM. The second is the training of a Recurrent Neural Network for character-level language modelling on the Tolstoi War and Peace data set which forms part of the DeepOBS benchmark (Schneider et al., 2019). The bi-directional LSTM has 1 layer and the RNN has 2 layers. Both models have 128 hidden units per layer. The result of these models is the interpolation condition is satisfied on the IMDB data set but not on the Tolstoi data set.

Method. We compare AALIG against the majority of the algorithms used in section 5.2. However, we use a slightly modified cross validation scheme; each optimiser's step-size or maximum step-size hyperparameter is cross validated as powers of ten. The weight decay amount λ was selected from $\{0.01, 0.001\}$ for the IMDB classification task, and was not applied to biases. For this task a batch size of 128 and an epoch budget of 100 was used. In contrast, no regularisation, a batch size of 50 and an epoch budget of 150 was used for Tolstoi character prediction.

Results. On the easy IMDB review classification task a large number of the optimisation methods achieved close to zero training loss and similar accuracies. The best performing of these was Adam that resulted in a test accuracy of 87.9%, where as AALIG attained 87.7% test. For the harder character prediction task using the Tolstoi data set AALIG was the best performing algorithm by over 1%. While these two results are not exceptionally significant on their own, they do reinforce i) that AALIG consistently achieves highly competitive results in a wide range of settings; and ii) in the non-interpolation setting AALIG is particularly effective compared to alternatives.

	IMDB	Tolstoi
Model	Small ResNet	Small ResNet
SGD_{const}	87.5	49.4
$Adabound$	82.7	41.7
$Adam$	**87.9**	57.9
$Coin$	87.6	56.9
SLS_{Armijo}	73.4	30.0
$SLS_{Goldstein}$	78.6	39.2
SLS_{Polyak}	67.1	31.3
PAL	85.7	52.9
$AALIG$	87.7	**59.4**

Table 2: *Test accuracy of single hyperparameter optimisation methods on NLP tasks .*

6 DISCUSSION

We have introduced AALIG , an optimisation algorithm designed for the non-interpolating setting, and demonstrated its effectiveness on many standard benchmarks. However, due to stochasticity it is theoretically possible to design simple convex and Lipshitz continuous problems where, in the worst case, AALIG will perform arbitrarily poorly. Thus, we provide no guarantee of convergence. We leave to future work the characterisation of the conditions where AALIG offers provable convergence. Additionally, we provide two orthogonal directions for future work. The first being an extension to AALIG when data augmentation is used. Here one could make use of a convenient distribution, to model the uncertainty of $\ell_z(\bar{\boldsymbol{w}})$. When updating the AOVs one could then use a lower confidence bound on this distribution. The second direction would be the application of AALIG to the distillation setting where the teacher network could be used to generate AOVs for the student.

REFERENCES

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

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

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.

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

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.

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

Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. *International Conference on Computer Aided Verification*, 2017.

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

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

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

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.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations*, 2017.

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

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.

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

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS Autodiff Workshop*, 2017.

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

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

Frank Schneider, Lukas Balles, and Philipp Hennig. DeepOBS: A deep learning optimizer benchmark suite. *International Conference on Learning Representations*, 2019.

Prabhu Teja Sivaprasad, Florian Mai, Thijs Vogels, Martin Jaggi, and Francois Fleuret. Optimizer benchmarking needs to account for hyperparameter tuning. *International Conference on Machine Learning*, 2020.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Conference on Computer Vision and Pattern Recognition*, 2015.

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*, 2019a.

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*, 2019b.

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.