

PAPER • OPEN ACCESS

Adaptive learning rate clipping stabilizes learning

To cite this article: Jeffrey M Ede and Richard Beanland 2020 *Mach. Learn.: Sci. Technol.* 1 015011

Recent citations

- [Partial Scanning Transmission Electron Microscopy with Deep Learning](#)
Jeffrey M. Ede and Richard Beanland

View the [article online](#) for updates and enhancements.



PAPER

Adaptive learning rate clipping stabilizes learning

OPEN ACCESS

RECEIVED
20 December 2019REVISED
26 February 2020ACCEPTED FOR PUBLICATION
20 March 2020PUBLISHED
28 April 2020

Original Content from this work may be used under the terms of the [Creative Commons Attribution 4.0 licence](#).

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Jeffrey M Ede¹ and Richard Beanland

Department of Physics, University of Warwick, Coventry CV4 7AL United Kingdom

E-mail: j.m.ede@warwick.ac.uk and r.beanland@warwick.ac.uk**Keywords:** machine learning, optimization, electron microscopy, learning stabilitySupplementary material for this article is available [online](#)**Abstract**

Artificial neural network training with gradient descent can be destabilized by ‘bad batches’ with high losses. This is often problematic for training with small batch sizes, high order loss functions or unstably high learning rates. To stabilize learning, we have developed adaptive learning rate clipping (ALRC) to limit backpropagated losses to a number of standard deviations above their running means. ALRC is designed to complement existing learning algorithms: Our algorithm is computationally inexpensive, can be applied to any loss function or batch size, is robust to hyperparameter choices and does not affect backpropagated gradient distributions. Experiments with CIFAR-10 supersampling show that ALRC decreases errors for unstable mean quartic error training while stable mean squared error training is unaffected. We also show that ALRC decreases unstable mean squared errors for scanning transmission electron microscopy supersampling and partial scan completion. Our source code is available at <https://github.com/Jeffrey-Ede/ALRC>.

1. Introduction

Loss spikes arise when artificial neural networks (ANNs) encounter difficult examples and can destabilize training with gradient descent [1, 2]. Examples may be difficult because an ANN needs more training to generalize, catastrophically forgot previous learning [3] or because an example is complex or unusual. Whatever the cause, applying gradients backpropagated [4] from high losses results in large perturbations to trainable parameters.

When a trainable parameter perturbation is much larger than others, learning can be destabilized while parameters adapt. This behaviour is common for ANN training with gradient descent where a large portion of parameters is perturbed at each optimization step. In contrast, biological networks often perturb small portions of neurons to combine new learning with previous learning. Similar to biological networks, ANN layers can become more specialized throughout training [5] and specialized capsule networks [6] are being developed. Nevertheless, ANN loss spikes during optimization are still a common reason for learning instability. Loss spikes are common for training with small batch sizes, high order loss functions, and unstably high learning rates.

During ANN training by stochastic gradient descent [1] (SGD), a trainable parameter, θ_t , from step t is updated to θ_{t+1} in step $t + 1$. The size of the update is given by the product of a learning rate, η , and the backpropagated gradient of a loss function with respect to the trainable parameter

$$\theta_{t+1} \leftarrow \theta_t - \eta \frac{\partial L}{\partial \theta}. \quad (1)$$

Without modification, trainable parameter perturbations are proportional to the scale of the loss function. Following gradient backpropagation, a high loss spike can cause a large perturbation to a learned parameter distribution. Learning will then be destabilized while subsequent iterations update trainable parameters back to an intelligent distribution.

¹ Author to whom any correspondence should be addressed.

Trainable parameter perturbations are often limited by clipping gradients to a multiple of their global L2 norm [7]. For large batch sizes, this can limit perturbations by loss spikes as their gradients will be larger than other gradients in the batch. However, global L2 norm clipping alters the distribution of gradients backpropagated from high losses and is unable to identify and clip high losses if the batch size is small. Clipping gradients of individual layers by their L2 norms has the same limitations.

Gradient clipping to a user-provided threshold [8] can also be applied globally or to individual layers. This can limit loss spike perturbations for any batch size. However, the clipping threshold is an extra hyperparameter to determine and may need to be changed throughout training. Further, it does not preserve distributions of gradients for high losses.

More commonly, destabilizing perturbations are reduced by selecting a low order loss function and stable learning rate. Low order loss functions, such as absolute and squared distances, are effective because they are less susceptible to destabilizingly high errors than higher-order loss functions. Indeed, loss function modifications used to stabilize learning often lower loss function order. For instance, Huberization [9, 10] reduces perturbations by losses, L , larger than h by applying the mapping $L \rightarrow \min(L, (hL)^{1/2})$.

2. Algorithm

Adaptive learning rate clipping (ALRC, algorithm 1) is designed to address the limitations of gradient clipping. Namely, to be computationally inexpensive, effective for any batch size, robust to hyperparameter choices and to preserve backpropagated gradient distributions. Like gradient clipping, ALRC also has to be applicable to arbitrary loss functions and neural network architectures.

Rather than allowing loss spikes to destabilize learning, ALRC applies the mapping $\eta L \rightarrow \text{stop_gradient}(L_{\max}/L)\eta L$ if $L > L_{\max}$. The function stop_gradient leaves its operand unchanged in the forward pass and blocks gradients in the backwards pass. ALRC adapts the learning rate to limit the effective loss being backpropagated to L_{\max} . The value of L_{\max} is non-trivial for ALRC to complement existing learning algorithms. In addition to training stability and robustness to hyperparameter choices, L_{\max} needs to adapt to losses and learning rates as they vary.

In our implementation, L_{\max} and L_{\min} are numbers of standard deviations of the loss above and below its mean, respectively. ALRC has six hyperparameters; however, it is robust to their values. There are two decay rates, β_1 and β_2 , for exponential moving averages used to estimate the mean and standard deviation of the loss and a number, n , of standard deviations. Similar to batch normalization [11], any decay rate close to 1 is effective e.g. $\beta_1 = \beta_2 = 0.999$. Performance does vary slightly with n_{\max} ; however, we found that any $n_{\max} \approx 3$ is effective. Varying n_{\min} is an optional extension and we default to one-sided ALRC above i.e. $n_{\min} = \infty$. Initial values for the running means, μ_1 and μ_2 , where $\mu_1^2 < \mu_2$ also have to be provided. However, any sensible initial estimates larger than their true values are fine as μ_1 and μ_2 will decay to their correct values.

ALRC can be extended to any loss function or batch size. For batch sizes above 1, we apply ALRC to individual losses, while μ_1 and μ_2 are updated with mean losses. ALRC can also be applied to loss summands, such as per pixel errors between generated and reference images, while μ_1 and μ_2 are updated with the mean errors.

Algorithm 1 Two-sided adaptive learning rate clipping (ALRC) of loss spikes. Sensible parameters are $\beta_1 = \beta_2 = 0.999$, $n_{\min} = \infty$, $n_{\max} = 3$, and $\mu_1^2 < \mu_2$.

Initialize running means, μ_1 and μ_2 , with decay rates, β_1 and β_2 .

Choose number, n , of standard deviations to clip to.

While Training is not finished **do**

Infer forward-propagation loss, L .

$\sigma \leftarrow (\mu_2 - \mu_1^2)^{1/2}$

$L_{\min} \leftarrow \mu_1 - n_{\min}\sigma$

$L_{\max} \leftarrow \mu_1 + n_{\max}\sigma$

If $L < L_{\min}$ **then**

$L_{\text{dyn}} \leftarrow \text{stop_gradient}(L_{\min}/L)L$

else if $L > L_{\max}$ **then**

$L_{\text{dyn}} \leftarrow \text{stop_gradient}(L_{\max}/L)L$

else

$L_{\text{dyn}} \leftarrow L$

end if

Optimize network by back-propagating L_{dyn} .

$\mu_1 \leftarrow \beta_1\mu_1 + (1 - \beta_1)L$

$\mu_2 \leftarrow \beta_2\mu_2 + (1 - \beta_2)L^2$

end while

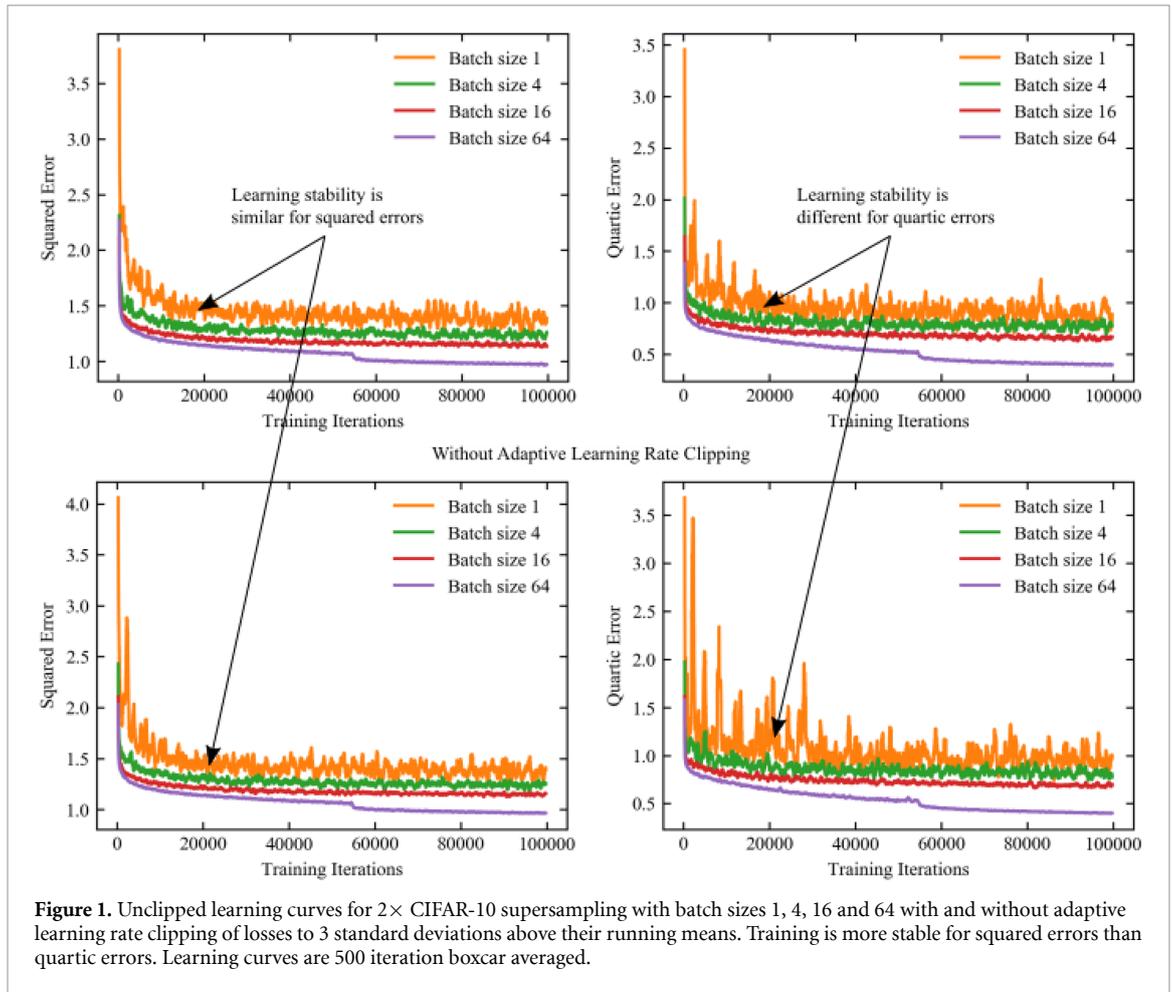


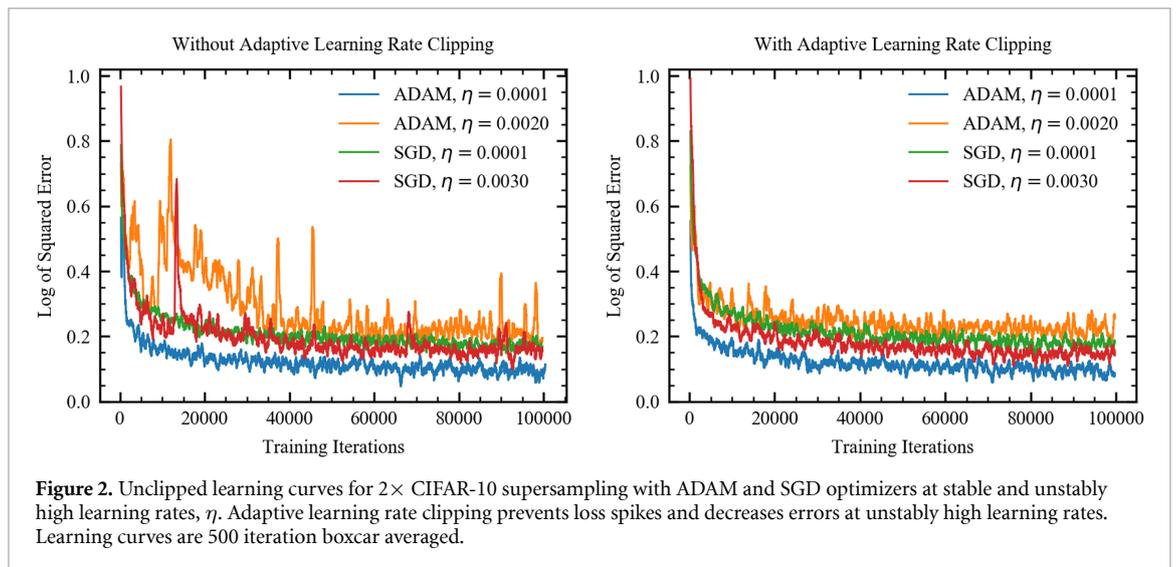
Figure 1. Unclipped learning curves for $2\times$ CIFAR-10 supersampling with batch sizes 1, 4, 16 and 64 with and without adaptive learning rate clipping of losses to 3 standard deviations above their running means. Training is more stable for squared errors than quartic errors. Learning curves are 500 iteration boxcar averaged.

Table 1. Adaptive learning rate clipping (ALRC) for losses 2, 3, 4 and ∞ running standard deviations above their running means for batch sizes 1, 4, 16 and 64. ALRC was not applied for clipping at ∞ . Each squared and quartic error mean and standard deviation is for the means of the final 5000 training errors of 10 experiments. ALRC lowers errors for unstable quartic error training at low batch sizes and otherwise has little effect. Means and standard deviations are multiplied by 100.

Squared Errors									
Threshold	Batch Size 1		Batch Size 4		Batch Size 16		Batch Size 64		
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	
2	5.55	0.048	4.96	0.016	4.58	0.010	—	—	
3	5.52	0.054	4.96	0.029	4.58	0.004	3.90	0.013	
4	5.56	0.048	4.97	0.017	4.58	0.007	3.89	0.016	
∞	5.55	0.041	4.98	0.017	4.59	0.006	3.89	0.014	
Quartic Errors									
Threshold	Batch Size 1		Batch Size 4		Batch Size 16		Batch Size 64		
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	
2	3.54	0.084	3.02	0.023	2.60	0.012	1.65	0.011	
3	3.59	0.055	3.08	0.024	2.61	0.014	1.58	0.016	
4	3.61	0.054	3.13	0.023	2.64	0.016	1.57	0.016	
∞	3.88	0.108	3.32	0.037	2.74	0.020	1.61	0.008	

3. Experiments: CIFAR-10 supersampling

To investigate the ability of ALRC to stabilize learning and its robustness to hyperparameter choices, we performed a series of toy experiments with networks trained to supersample CIFAR-10 [12, 13] images to $32\times 32\times 3$ after downsampling to $16\times 16\times 3$.



Data pipeline: In order, images were randomly flipped left or right, had their brightness altered, had their contrast altered, were linearly transformed to have zero mean and unit variance and bilinearly downsampled to $16 \times 16 \times 3$.

Architecture: Images were upsampled and passed through a convolutional neural network [14, 15] shown in figure 5. Each convolutional layer is followed by ReLU [16] activation, except the last.

Initialization: All weights were Xavier [17] initialized. Biases were zero initialized.

Learning policy: ADAM optimization was used with the hyperparameters recommended in [18] and a base learning rate of $1/1280$ for 100 000 iterations. The learning rate was constant in batch size 1, 4, 16 experiments and decreased to $1/12800$ after 54 687 iterations in batch size 64 experiments. Networks were trained to minimize mean squared or quartic errors between restored and ground truth images. ALRC was applied to limit the magnitudes of losses to either 2, 3, 4 or ∞ standard deviations above their running means. For batch sizes above 1, ALRC was applied to each loss individually.

Results: Example learning curves for mean squared and quartic error training are shown in figure 1. Training is more stable and converges to lower losses for larger batch sizes. However, learning is less stable for quartic errors than squared errors, allowing ALRC to be examined for loss functions with different stability. Training was repeated 10 times for each combination of ALRC threshold and batch size. Means and standard deviations of the means of the last 5000 training losses for each experiment are tabulated in table 1. ALRC has no effect on mean squared error (MSE) training, even for batch size 1. However, it decreases errors for batch sizes 1, 4 and 16 for mean quartic error training.

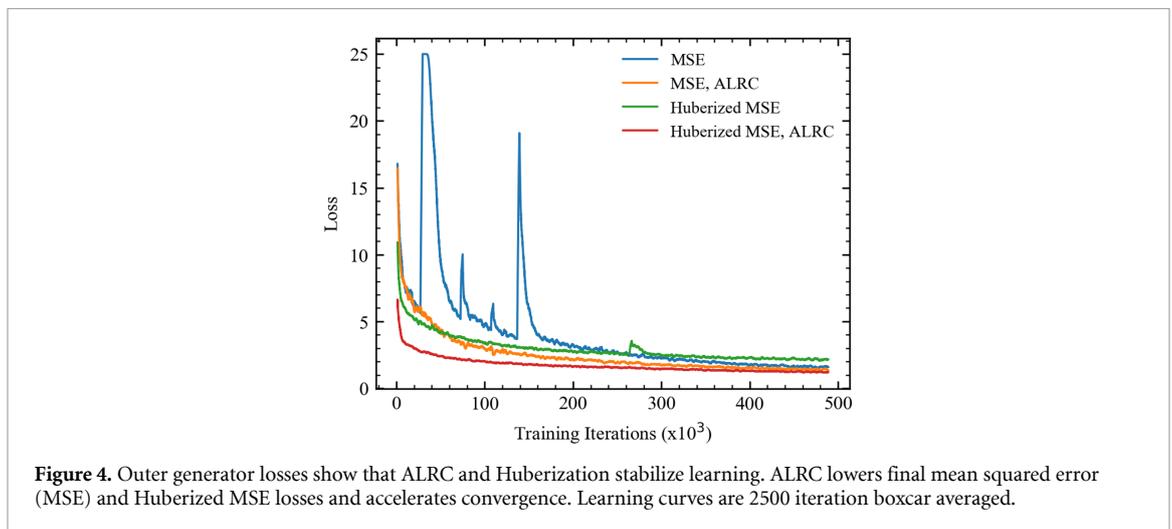
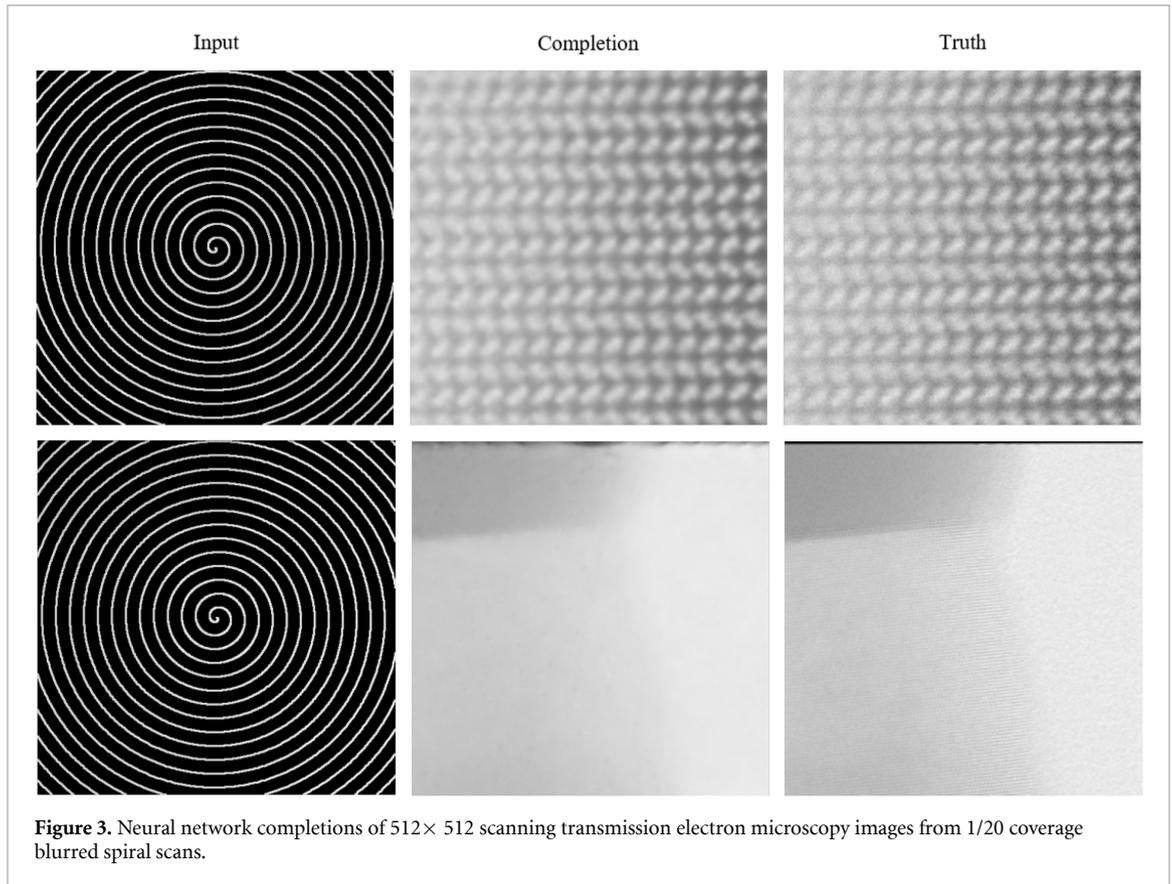
Additional learning curves are shown in figure 2 for both ADAM and SGD optimizers to showcase the effect of ALRC on unstably high learning rates. Experiments are for a batch size of 1. ALRC has no effect at stable learning rates where learning is unaffected by loss spikes. However, ALRC prevents loss spikes and decreases errors at unstably high learning rates. In addition, these experiments show that ALRC is effective for different optimizers.

4. Experiments: partial STEM

To test ALRC in practice, we applied our algorithm to neural networks learning to complete 512×512 scanning transmission electron microscopy (STEM) images [19] from partial scans [20] with $1/20$ coverage. Example completions are shown in figure 3.

Data pipeline: In order, each image was subject to a random combination of flips and 90° rotations to augment the dataset by a factor of 8. Next, each STEM image was blurred, and a path described by a $1/20$ coverage spiral was selected. Finally, artificial noise was added to scans to make them more difficult to complete.

Architecture: Our network can be divided into three subnetworks shown in figure 6: an inner generator, outer generator and an auxiliary inner generator trainer. The auxiliary trainer [21, 22] is introduced to provide a more direct path for gradients to backpropagate to the inner generator. Each convolutional layer is followed by ReLU activation, except the last.



Initialization: Weights were initialized from a normal distribution with mean 0.00 and standard deviation 0.05. There are no biases.

Weight normalization: All generator weights are weight normalized [23] and a weight normalization initialization pass was performed after weight initialization. Following [23, 24], running mean-only batch normalization was applied to the output channels of every convolutional layer except the last. Channel means were tracked by exponential moving averages with decay rates of 0.99. Similar to [25], running mean-only batch normalization was frozen in the second half of training to improve stability.

Loss functions: The auxiliary inner generator trainer learns to generate half-size completions that minimize MSEs from half-size blurred ground truth STEM images. Meanwhile, the outer generator learns to produce full-size completions that minimize MSEs from blurred STEM images. All MSEs were multiplied by 200. The inner generator cooperates with the auxiliary inner generator trainer and outer generator.

Table 2. Means and standard deviations of 20 000 unclipped test set MSEs for STEM supersampling networks trained with various learning rate clipping algorithms and clipping hyperparameters, n^\downarrow and n^\uparrow , above and below, respectively.

Algorithm	n^\downarrow	n^\uparrow	Mean	Std
Unchanged	∞	∞	0.95	1.33
ALRC	∞	3	0.89	1.68
ALRC	3	3	0.92	1.77
CLRC ^(\downarrow) , ALRC ^(\uparrow)	1	3	0.95	2.30
DALRC	3	3	0.93	1.57
DALRC	∞	2	0.89	1.51
DALRC	2	2	0.91	1.34
DALRC	1	2	0.91	1.54

To benchmark ALRC, we investigated training with MSEs, Huberized ($h = 1$) MSEs, MSEs with ALRC and Huberized ($h = 1$) MSEs with ALRC before Huberization. Training with both ALRC and Hubarization showcases the ability of ALRC to complement another loss function modification.

Learning policy: ADAM optimization [18] was used with a constant generator learning rate of 0.0003 and a first moment of the momentum decay rate, $\beta_1 = 0.9$, for 250 000 iterations. In the next 250 000 iterations, the learning rate and β_1 were linearly decayed in eight steps to zero and 0.5, respectively. The learning rate for the auxiliary inner generator trainer was two times the generator learning rate; β_1 were the same. All training was performed with batch size 1 due to the large model size needed to complete 512×512 scans.

Results: Outer generator losses in figure 4 show that ALRC and Huberization stabilize learning. Further, ALRC accelerates MSE and Huberized MSE convergence to lower losses. To be clear, learning policy was optimized for MSE training so direct loss comparison is uncharitable to ALRC.

Algorithm 2 Two-sided constant learning rate clipping (CLRC) to effective losses in $[L_{\min}, L_{\max}]$.

Choose effective loss bounds, L_{\min} and L_{\max} .

While Training is not finished **do**

If $L < L_{\min}$ **then**

$L_{\text{dyn}} \leftarrow \text{stop_gradient}(L_{\min}/L)L$

else if $L > L_{\max}$ **then**

$L_{\text{dyn}} \leftarrow \text{stop_gradient}(L_{\max}/L)L$

else

$L_{\text{dyn}} \leftarrow L$

end if

 Optimize network by back-propagating L_{dyn} .

end While

Algorithm 3 Two-sided doubly adaptive learning rate clipping (DALRC) of loss spikes. Sensible parameters are $\beta_1 = \beta^\downarrow = \beta^\uparrow = 0.999$, and $n^\downarrow = n^\uparrow = 2$.

Initialize running means, μ_1 , μ^\downarrow and μ^\uparrow , with decay rates, β_1 , β^\downarrow and β^\uparrow .

Choose numbers, n , of standard deviations to clip to.

While Training is not finished **do**

 Infer forward-propagation loss, L .

$L_{\min} \leftarrow \mu_1 - n^\downarrow \mu^\downarrow$

$L_{\max} \leftarrow \mu_1 + n^\uparrow \mu^\uparrow$

if $L < L_{\min}$ **then**

$L_{\text{dyn}} \leftarrow \text{stop_gradient}(L_{\min}/L)L$

else if $L > L_{\max}$ **then**

$L_{\text{dyn}} \leftarrow \text{stop_gradient}(L_{\max}/L)L$

else

$L_{\text{dyn}} \leftarrow L$

end if

 Optimize network by back-propagating L_{dyn} .

if $L > \mu_1$ **then**

$\mu^\uparrow \leftarrow \beta^\uparrow \mu^\uparrow + (1 - \beta^\uparrow)(L - \mu_1)$

else if $L < \mu_1$ **then**

$\mu^\downarrow \leftarrow \beta^\downarrow \mu^\downarrow + (1 - \beta^\downarrow)(\mu_1 - L)$

end if

$\mu_1 \leftarrow \beta_1 \mu_1 + (1 - \beta_1)L$

end While

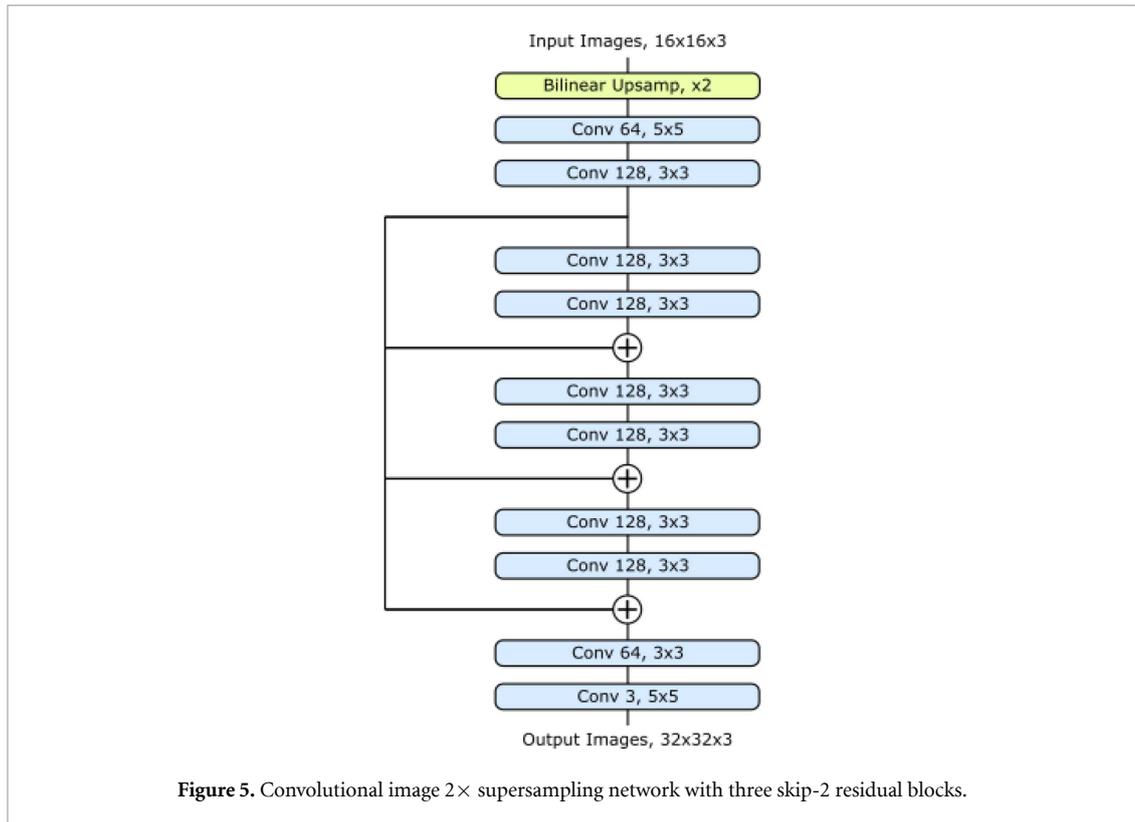


Figure 5. Convolutional image $2\times$ supersampling network with three skip-2 residual blocks.

5. Experiments: ALRC variants

ALRC was developed to limit perturbations by loss spikes. Nevertheless, ALRC can also increase parameter perturbations for low losses, possibly improving performance on examples that an ANN is already good at. To investigate ALRC variants, we trained a generator to supersample STEM images to 512×512 after nearest neighbour downsampling to 103×103 . Network architecture and learning protocols are the same as those for partial STEM in section 4, except training iterations are increased from 5×10^5 to 10^6 .

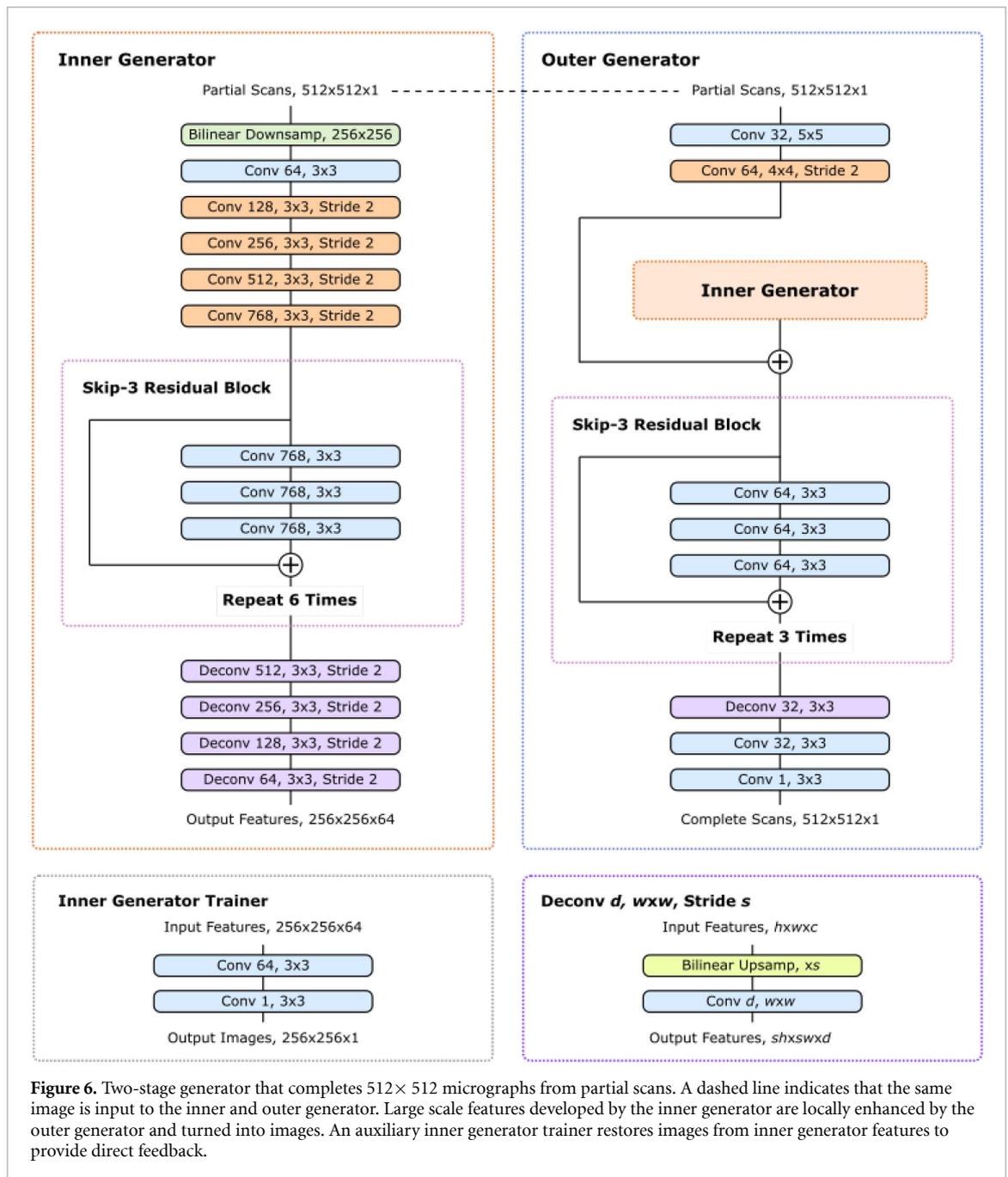
Means and standard deviations of 20 000 unclipped test set MSEs for possible ALRC variants are tabulated in table 2. Variants include constant learning rate clipping (CLRC) in algorithm 2; where the effective loss is kept between constant values, and doubly adaptive learning rate clipping (DALRC) in algorithm 3; where moments above and below a running mean are tracked separately. ALRC has the lowest test set MSEs whereas DALRC has lower variance. Both ALRC and DLRC outperform no learning rate clipping for all tabulated hyperparameters and may be a promising starting point for future research on learning rate clipping.

6. Discussion

Taken together, our CIFAR-10 supersampling results show that ALRC improves stability and lowers losses for learning that would be destabilized by loss spikes and otherwise has little effect. Loss spikes are often encountered when training with high learning rates, high order loss functions or small batch sizes. For example, a moderate learning rate was used in MSE experiments so that losses did not spike enough to destabilize learning. In contrast, training at the same learning rate with quartic errors is unstable so ALRC stabilizes learning and lowers losses. Similar results are confirmed at unstably high learning rates, for partial STEM and for STEM supersampling, where ALRC stabilizes learning and lowers losses.

ALRC is designed to complement existing learning algorithms with new functionality. It is effective for any loss function or batch size and can be applied to any neural network trained with gradient descent. Our algorithm is also computationally inexpensive, requiring orders of magnitude fewer operations than other layers typically used in neural networks. As ALRC either stabilizes learning or has little effect, this means that it is suitable for routine application to arbitrary neural network training with gradient descent. In addition, we note that ALRC is a simple algorithm that has a clear effect on learning.

Nevertheless, ALRC can replace other learning algorithms in some situations. For instance, ALRC is a computationally inexpensive alternative to gradient clipping in high batch size training where gradient



clipping is being used to limit perturbations by loss spikes. However, it is not a direct replacement as ALRC preserves the distribution of backpropagated gradients whereas gradient clipping reduces large gradients. Instead, ALRC is designed to complement gradient clipping by limiting perturbations by large losses while gradient clipping modifies gradient distributions.

The implementation of ALRC in algorithm 1 is for positive losses. This avoids the need to introduce small constants to prevent divide-by-zero errors. Nevertheless, ALRC can support negative losses by using standard methods to prevent divide-by-zero errors. Alternatively, a constant can be added to losses to make them positive without affecting learning.

ALRC can also be extended to limit losses more than a number of standard deviations below their mean. This had no effect in our experiments. However, preemptively reducing loss spikes by clipping rewards between user-provided upper and lower bounds can improve reinforcement learning [26]. Subsequently, we suggest that clipping losses below their means did not improve learning because losses mainly spiked above their means; not below. Some partial STEM losses did spike below; however, they were mainly for blank or otherwise trivial completions.

7. Conclusions

We have developed ALRC to stabilize the training of ANNs by limiting backpropagated loss perturbations. Our experiments show that ALRC accelerates convergence and lowers losses for learning that would be destabilized by loss spikes and otherwise has little effect. Further, ALRC is computationally inexpensive, can be applied to any loss function or batch size, does not affect the distribution of backpropagated gradients and has a clear effect on learning. Overall, ALRC complements existing learning algorithms and can be routinely applied to arbitrary neural network training with gradient descent.

Data Availability

The data that support the findings of this study are openly available. Source code based on TensorFlow[27] is provided for CIFAR-10 supersampling[28] and partial STEM[29], and both CIFAR-10[12] and STEM[19] datasets are available. For additional information contact the corresponding author (J M E).

8. Network architecture

ANN architecture for CIFAR-10 experiments is shown in figure 5, and architecture for STEM partial scan and supersampling experiments is shown in figure 6. The components in our networks are

Bilinear Downsamp, $w \times w$: This is an extension of linear interpolation in one dimension to two dimensions. It is used to downsample images to $w \times w$.

Bilinear Upsamp, $\times s$: This is an extension of linear interpolation in one dimension to two dimensions. It is used to upsample images by a factor of s .

Conv d , $w \times w$, Stride, x : Convolutional layer with a square kernel of width, w , that outputs d feature channels. If the stride is specified, convolutions are only applied to every x th spatial element of their input, rather than to every element. Striding is not applied depthwise.

⊕: Circled plus signs indicate residual connections[30] where tensors are added together. Residual connections help reduce signal attenuation and allow networks to learn perturbative transformations more easily.

Acknowledgment

J M E and R B acknowledge EPSRC grant EP/N035437/1 for financial support. In addition, J M E acknowledges EPSRC Studentship 1917382.

ORCID iD

Jeffrey M Ede  <https://orcid.org/0000-0002-9358-5364>

References

- [1] Ruder S 2016 An overview of gradient descent optimization algorithms arXiv:1609.04747
- [2] Zou D, Cao Y, Zhou D and Gu Q 2018 Stochastic gradient descent optimizes over-parameterized deep ReLU networks arXiv:1811.08888
- [3] Pfüll B, Gepperth A, Abdullah S and Kilian A 2018 Catastrophic forgetting: still a problem for DNNs *Int. Conf. on Artificial Neural Networks* pp 487–97 Springer
- [4] Boué L 2018 Deep learning for pedestrians: backpropagation in CNNs arXiv:1811.11987
- [5] Qin Z, Yu F, Liu C and Chen X 2018 How convolutional neural network see the world-A survey of convolutional neural network visualization methods arXiv:1804.11191
- [6] Sabour S, Frosst N and Hinton G E 2017 Dynamic routing between capsules *Advances in Neural Information Processing Systems* pp 3856–66
- [7] Bengio Y and Pascanu R 2012 On the difficulty of training recurrent neural networks arXiv:1211.5063
- [8] Mikolov T 2012 Statistical language models based on neural networks *PhD thesis* Brno University of Technology
- [9] Huber P J 1964 Robust estimation of a location parameter *The Annals of Mathematical Statistics* pp 73–101
- [10] Meyer G P 2019 An alternative probabilistic interpretation of the Huber loss arXiv:1911.02088
- [11] Ioffe S and Szegedy C 2015 Batch normalization accelerating deep network training by reducing internal covariate shift arXiv:1502.03167
- [12] Krizhevsky A, Nair V and Hinton G 2014 The CIFAR-10 dataset Online (www.cs.toronto.edu/~Kriz/Cifar.html) vol 55
- [13] Krizhevsky A and Hinton G 2009 Learning multiple layers of features from tiny images *Technical Report TR-2009* University of Toronto
- [14] McCann M T, Jin K H and Unser M 2017 Convolutional neural networks for inverse problems in imaging: A review *IEEE Signal Process. Mag.* **34** 85–95
- [15] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* pp 1097–105

- [16] Nair V and Hinton G E 2010 Rectified linear units improve restricted Boltzmann machines in *Proc. of the 27th Int. Conf. on Machine Learning (ICML-10)* pp 807–14
- [17] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. of the Thirteenth Int. Conf. on Artificial Intelligence and Statistics* pp 249–56
- [18] Kingma D P and Ba J 2014 ADAM: A method for stochastic optimization arXiv:1412.6980
- [19] Ede J M and STEM Crops Dataset 2019 Online (<https://warwick.ac.uk/fac/sci/physics/research/condensedmatt/microscopy/research/machinelearning>)
- [20] Ede J M and Beanland R 2020 Partial scanning transmission electron microscopy with deep learning arXiv:1910.10467
- [21] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2015 Going deeper with convolutions *Proc. of the Conf. on Computer Vision and Pattern Recognition* pp 1–9
- [22] Szegedy C, Vanhoucke V, Ioffe S, Shlens J and Wojna Z 2016 Rethinking the inception architecture for computer vision *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* pp 2818–26
- [23] Salimans T and Kingma D P 2016 Weight normalization: A simple reparameterization to accelerate training of deep neural networks *Advances in Neural Information Processing Systems* pp 901–9
- [24] Hoffer E, Banner R, Golan I and Soudry D 2018 Norm matters: efficient and accurate normalization schemes in deep networks *Advances in Neural Information Processing Systems* pp 2160–70
- [25] Chen L-C, Papandreou G, Schroff F and Adam H 2017 Rethinking atrous convolution for semantic image segmentation arXiv:1706.05587
- [26] Mnih V *et al et al* 2015 Human-Level control through deep reinforcement learning *Nature* **518** 529
- [27] Abadi M *et al et al* 2016 Tensor flow: A system for large-scale machine learning. *OSDI* **16** 265–83
- [28] Ede J M and ALRC 2020 Online: (<https://github.com/Jeffrey-Ede/ALRC>)
- [29] Ede J M and Partial STEM 2020 Online: (<https://github.com/Jeffrey-Ede/partial-STEM>)
- [30] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. of the Conf. on Computer Vision and Pattern Recognition* pp 770–8