DNNs from Theory to SoTA Practice

Optimisation and Generalisation

Diego Granziol*[1,2,3]

[1]Machine Learning Research Group
Information Engineering
University of Oxford

[2]Oxford-Man Institute of Quantitative Finance
University of Oxford

[3]AI Theory Team
Huawei, London

February 8, 2022

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

RMT approach to Minibatching

Statistical Learning Theory Intro

input, output pair $[\boldsymbol{x}, \boldsymbol{y}] \in [\mathbb{R}^{d_x}, \mathbb{R}^{d_y}]$

prediction function $h(\cdot;\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^{P} \to \mathbb{R}^{d_y}$

$\boldsymbol{w}$, i.e., $\mathcal{H} := \{h(\cdot; \boldsymbol{w}) : \boldsymbol{w} \in \mathbb{R}^{P}\}$

$\ell(h(\boldsymbol{x}; \boldsymbol{w}), \boldsymbol{y}) : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$

$$R_{true}(\boldsymbol{w}) = \int \ell(h(\boldsymbol{x}; \boldsymbol{w}), \boldsymbol{y}) d\psi(\boldsymbol{x}, \boldsymbol{y}), \tag{1}$$

$$R_{emp}(\boldsymbol{w}) = \sum_{i=1}^{N} \frac{1}{N} \ell(h(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{y}_i), \tag{2}$$

$$R_{batch}(\boldsymbol{w}) = \frac{1}{B} \sum_{i=1}^{B} \ell(h(\boldsymbol{x}_i; \boldsymbol{w}), \boldsymbol{y}_i), \tag{3}$$

RMT approach to Minibatching

Key Concept: Fluctuations Matrix

$$\boldsymbol{H}_{batch}(\boldsymbol{w}) = \boldsymbol{H}_{emp}(\boldsymbol{w}) + \epsilon(\boldsymbol{w}) \tag{4}$$

$$\epsilon(\boldsymbol{w}) = \left(\frac{1}{B} - \frac{1}{N}\right) \sum_{j=1}^{B} \nabla^2 \ell(\boldsymbol{x}_j, \boldsymbol{w}; \boldsymbol{y}_j) - \frac{1}{N} \sum_{i=B+1}^{N} \nabla^2 \ell(\boldsymbol{x}_i, \boldsymbol{w}; \boldsymbol{y}_i)$$

$$\text{thus } \mathbb{E}(\epsilon(\boldsymbol{w})_{j,k}) = 0 \text{ and } \mathbb{E}(\epsilon(\boldsymbol{w})_{j,k})^2 = \left(\frac{1}{B} - \frac{1}{N}\right) \mathrm{Var}[\nabla^2 \ell(\boldsymbol{x}, \boldsymbol{w}; \boldsymbol{y})_{j,k}]. \tag{5}$$

RMT approach to Minibatching

Lemma (1) Hessian elements are bounded)

For a Lipschitz-continuous empirical risk gradient and almost everywhere twice differentiable loss function $\ell(h(\boldsymbol{x};\boldsymbol{w}),\boldsymbol{y})$, the elements of the fluctuation matrix $\boldsymbol{\epsilon}(\boldsymbol{w})_{j,k}$ are strictly bounded in the range $-\sqrt{P}L \leq \boldsymbol{\epsilon}(\boldsymbol{w})_{j,k} \leq \sqrt{P}L$. Where P is the number of model parameters and L is a constant.

Lemma (2) Elements converge to normal random variable)

For independent samples drawn from the data generating distribution and an L-Lipschitz loss ℓ the difference between the empirical Hessian and Batch Hessian converges element-wise to a zero mean, normal random variable with variance $\propto \frac{1}{B} - \frac{1}{N}$ for large B, N.

RMT approach to Minibatching

Dependence in the Fluctuations Matrix

To derive analytic results, we employ the Kolmogorov limit

$P, B, N \to \infty$ but $P(\frac{1}{B} - \frac{1}{N}) = q > 0$

$\mathbb{E}(\epsilon(\boldsymbol{w})_{j,k}) = 0$ and $\mathbb{E}(\epsilon(\boldsymbol{w})^2_{j,k}) = \sigma^2_{j,k}$

To further account for dependence beyond the symmetry of the fluctuation matrix elements, we introduce the σ-algebras

$$\mathfrak{F}^{(i,j)} := \sigma\{\epsilon(\boldsymbol{w})_{kl} : 1 \leq k \leq l \leq P, (k,l) \neq (i,j)\}, \quad 0 \leq i \leq j \leq P \quad (6)$$

RMT approach to Minibatching

Theorem

Under the conditions of Lemmas 1 and 2 along with the following technical conditions:

(i) $\frac{1}{P^2}\sum_{i,j=1}^{P}\mathbb{E}|\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2|\mathfrak{F}^{i,j}) - \sigma_{i,j}^2| \to 0,$

(ii) $\frac{1}{P}\sum_{i=1}^{P}|\frac{1}{P}\sum_{j=1}^{P}\sigma_{i,j}^2 - \sigma_\epsilon^2| \to 0$

(iii) $\max_{1\leq i\leq P}\frac{1}{P}\sum_{j=1}^{P}\sigma_{i,j}^2 \leq C$

when $P \to \infty$, *the limiting spectra density* $p(\lambda)$ *of* $\boldsymbol{\epsilon}(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$ *satisfies the semi circle law* $p(\lambda) = \frac{\sqrt{4\sigma_\epsilon^2-\lambda^2}}{2\pi\sigma_\epsilon^2}$.

Where $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2|\mathfrak{F}^{i,j})$ *denotes the expectation conditioned on the sigma algebra, which is different to the unconditional expectation* $\mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2|\mathfrak{F}^{i,j}) \neq \mathbb{E}(\boldsymbol{\epsilon}(\boldsymbol{w})_{i,j}^2) = \sigma_{i,j}^2$.

RMT approach to Minibatching

Theorem (Key Concept: Noise Increases the Width of the Hessian)

Under the assumption that $\boldsymbol{H}_{emp}$ *is of low-rank* $r \ll P$*, the extremal eigenvalues* $[\lambda'_1, \lambda'_P]$ *of the matrix sum* $\boldsymbol{H}_{batch}(\boldsymbol{w}) = \boldsymbol{H}_{emp}(\boldsymbol{w}) + \boldsymbol{\epsilon}(\boldsymbol{w})$*, where* $\lambda'_1 \geq \lambda'_2 ... \geq \lambda'_P$ *and* $\boldsymbol{\epsilon}(\boldsymbol{w})$ *obeys the conditions set out in the previous slide, are given by*

$$\lambda'_1 = \left\{ \begin{array}{ll} \lambda_1 + \frac{P}{\mathfrak{b}} \frac{\sigma_\epsilon^2}{\lambda_1}, & \text{if } \lambda_1 > \sqrt{\frac{P}{\mathfrak{b}}} \sigma_\epsilon \\ 2\sqrt{\frac{P}{\mathfrak{b}}} \sigma_\epsilon, & \text{otherwise} \end{array} \right\} \tag{7}$$

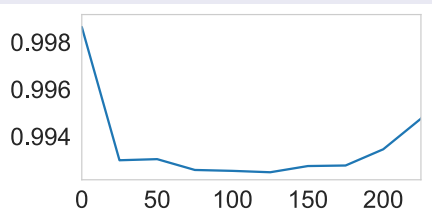
where $[\lambda_1, \lambda_P]$ *are the extremal eigenvalues of* $\boldsymbol{H}_{emp}(\boldsymbol{w})$*,* $\mathfrak{b} = B/(1 - B/N)$ *and* B *is the batch-size.*

Low rank assumption

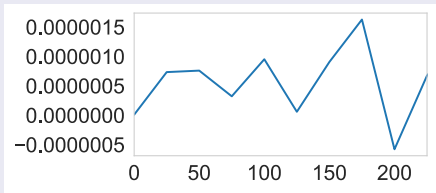
Assumption can be relaxed

Assumption can be alleviated under the assumption of non-interacting eigenspaces (free addition).

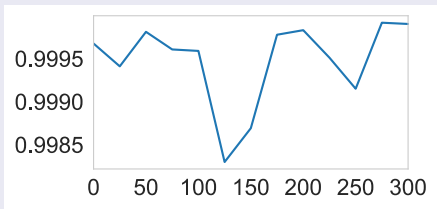
VGG-16 CIFAR100 does look low rank



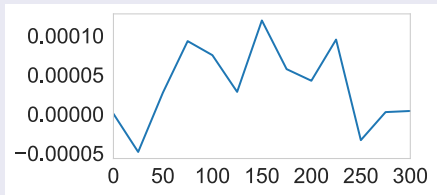
(a) GGN Degen



(b) GGN Ritz Val



(c) Hessian Degen



(d) $\boldsymbol{H}$ Ritz Val

Figure: Rank degeneracy (proportion of zero eigenvalues) evolution throughout training using the VGG-16 on the CIFAR-100 dataset, total training 225 epochs, the Ritz value corresponds to the value of the node which we assign to 0

Theory for Low Rank Assumption

We consider a neural network with a d_x dimensional input $\boldsymbol{x}$. Our network has $H-1$ hidden layers and we refer to the output as the H'th layer and the input as the 0'th layer. We denote the ReLU activation function as $f(x)$ where $f(x) = \max(0, x)$. Let $\boldsymbol{W}_i$ be the matrix of weights between the $(i-1)$'th and i'th layer. For a d_y dimensional output our q'th component of the output can be written as

$$\boldsymbol{z}(\boldsymbol{x}_i; \boldsymbol{w})_q = f(\boldsymbol{W}_H^T f(\boldsymbol{W}_{H-1}^T \ldots . f(\boldsymbol{W}_1 \boldsymbol{x}))) = \prod_{l=0}^{H} \sum_{n_{i,l}=1}^{N_l} \sum_{i}^{d_x} \boldsymbol{x}_i \boldsymbol{w}_{n_{i,l}, n_{i,l+1}} \quad (8)$$

where $\boldsymbol{w}_{n_{i,l}, n_{i,l+1}}$ denotes the weight of the path segment connecting node i in layer l with node i in layer $l+1$. layer l has N_l nodes. Where $n_{i,l_0} = x_i$.

Theory for Low Rank Assumption

The Hessian, in the small loss limit tends to

$$\frac{\partial^2 \ell(h(\boldsymbol{x}_i;\boldsymbol{w}),\boldsymbol{y}_i)}{\partial w_{\phi,\kappa}\partial w_{\theta,\nu}} \to -\sum_{m\neq c}\exp(h_m)\left[\frac{\partial^2 h_m}{\partial w_{\phi,\kappa}\partial w_{\theta,\nu}}+\frac{\partial h_m}{\partial w_{\phi,\kappa}}\frac{\partial h_m}{\partial w_{\theta,\nu}}\right]. \tag{9}$$

$$\begin{aligned}\left[\frac{\partial^2 h_m}{\partial w_{\phi,\kappa}\partial w_{\theta,\nu}}+\frac{\partial h_m}{\partial w_{\phi,\kappa}}\frac{\partial h_m}{\partial w_{\theta,\nu}}\right] &= \prod_{l=1}^{d-1}\sum_{n_{i,l}\neq[(\phi,\kappa),(\theta,\nu)]}^{N_{i,l}}\sum_{i}^{d_x}\boldsymbol{x}_i \boldsymbol{w}_{n_{i,l},n_{i,l+1}} \\ &+\left(\prod_{l=1}^{d-1}\sum_{n_{i,l}\neq(\theta,\nu)}^{N_{i,l}}\sum_{i}^{d_x}\boldsymbol{x}_i \boldsymbol{w}_{n_{i,l},n_{i,l+1}}\right)\left(\prod_{l=1}^{d-1}\sum_{n_{j,l}\neq(\phi,\kappa)}^{N_{j,l}}\sum_{i}^{d_x}\boldsymbol{x}_i \boldsymbol{w}_{n_{j,l},n_{j,l+1}}\right)\end{aligned} \tag{10}$$

Each product of weights contributes an object of rank-1 .Hence Equation 10 is rank bounded by $2(\sum_l N_l + d_x)$, where N_l is the total number of neurons in the network.

Theory for Low Rank Assumption

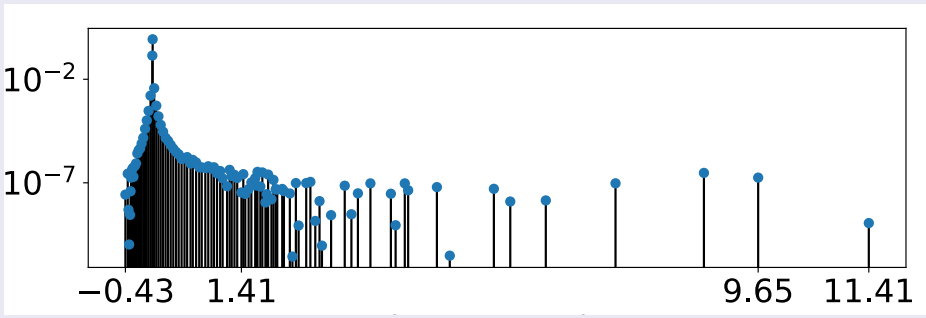
By rewriting the loss per-sample, repeating the same arguments and including the class factor, we obtain that

$$\frac{\partial^2 \ell}{\partial w_k \partial w_l} = -\frac{\partial^2 h_{q(i)}}{\partial w_k \partial w_l} + \frac{\sum_j \exp(h_j) \sum_i \exp(h_i)(\frac{\partial^2 h_i}{\partial w_k \partial w_l} + \frac{\partial h_i}{\partial w_k}\frac{\partial h_i}{\partial w_l}) - \sum_i \exp(h_i)\frac{\partial h_i}{\partial w_k}\sum_j \frac{\partial h_j}{\partial w_l}\exp(h_j)}{[\sum_j \exp(h_j)]^2}, \tag{11}$$

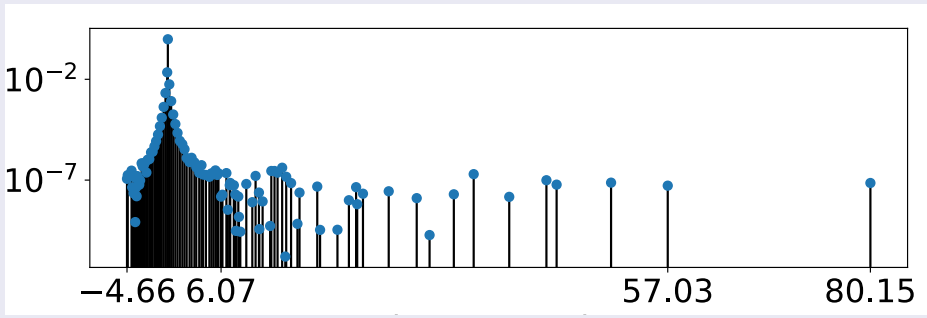
and thence a rank bound of $4d_y(\sum_l N_l + d_x)$. To give some context, along with a practical application of a real network and dataset, for the CIFAR-10 dataset, the VGG-16 [5] contains 1.6×10^7 parameters, the number of classes is 10 and the total number of neurons is $13,416$ and hence the bound gives us a spectral peak at the origin of at least $1 - \frac{577{,}600}{1.6 \times 10^7} = 0.9639$.

RMT approach to Minibatching

Seeing the Effect in Practice



(a) Empirical Hessian $p(\lambda), N = 50,000$



(b) Batch Hessian $p(\lambda), B = 128$

Figure: Spectral Density of the Hessian at epoch 200, for different sample sizes B, N on a VGG-16 on the CIFAR-100 dataset. The Y-axis corresponds to $p(\lambda)$ and the X-axis to λ.

RMT approach to Minibatching

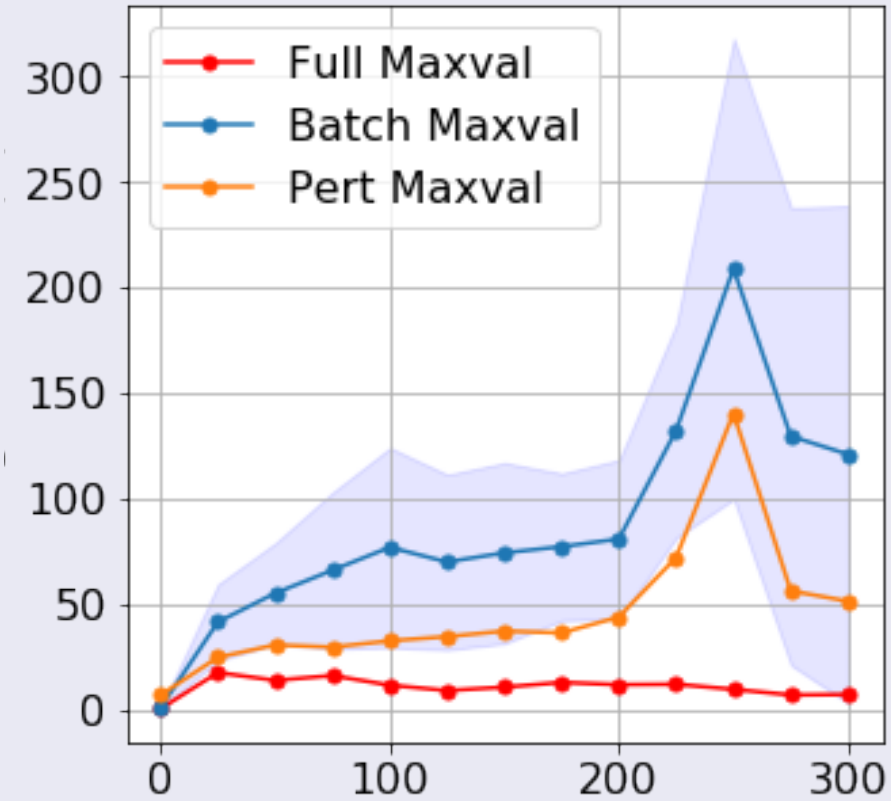
Key Concept: Fluctuations Matrix

Algorithm 1 Calculate Hessian Variance

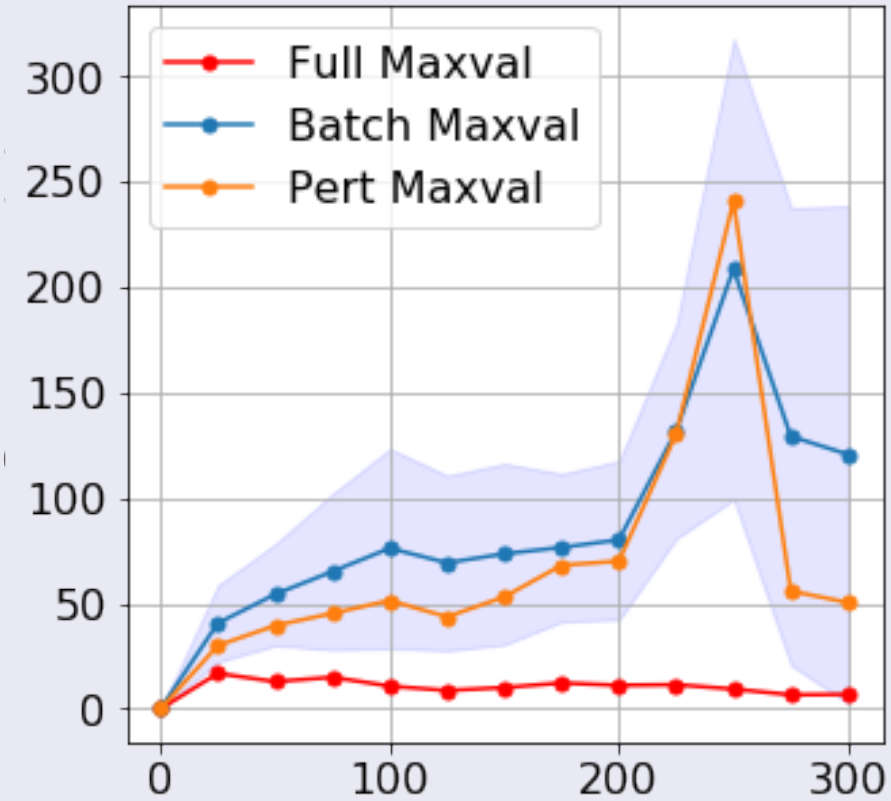
Input: Sample Hessian $\boldsymbol{H}_i \in \mathbb{R}^{P \times P}$
Output: Hessian Variance σ^2
$\boldsymbol{v} \in \mathbb{R}^{1 \times P} \sim \mathcal{N}(0, \boldsymbol{I})$
Initialise $\sigma^2 = 0, i = 0,\ \boldsymbol{v} \leftarrow \boldsymbol{v}/||\boldsymbol{v}||$
for $i < N$ **do**
    $\sigma^2 \leftarrow \sigma^2 + \boldsymbol{v}^T \boldsymbol{H}_i^2 \boldsymbol{v}$
    $i \leftarrow i + 1$
end for
$\sigma^2 \leftarrow \sigma^2 - [\boldsymbol{v}^T (1/N \sum_{j=1}^{N} \boldsymbol{H}_j) \boldsymbol{v}]^2$

RMT approach to Minibatching

Key Concept: Fluctuations Matrix



(a) SGD $\lambda_1(\boldsymbol{H})$

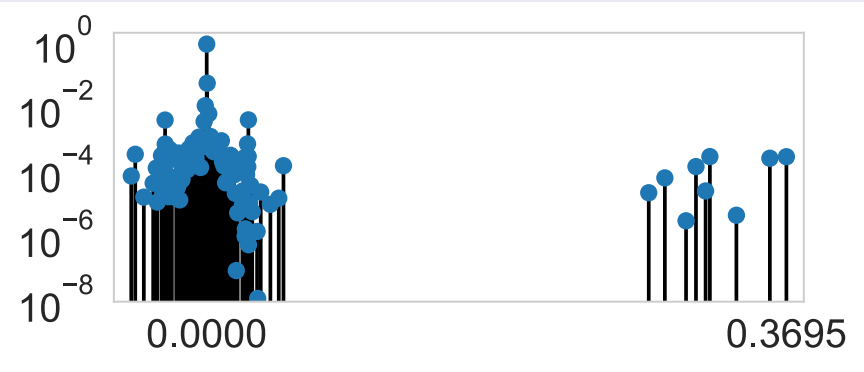


(b) SGD $\lambda_1(\boldsymbol{G})$

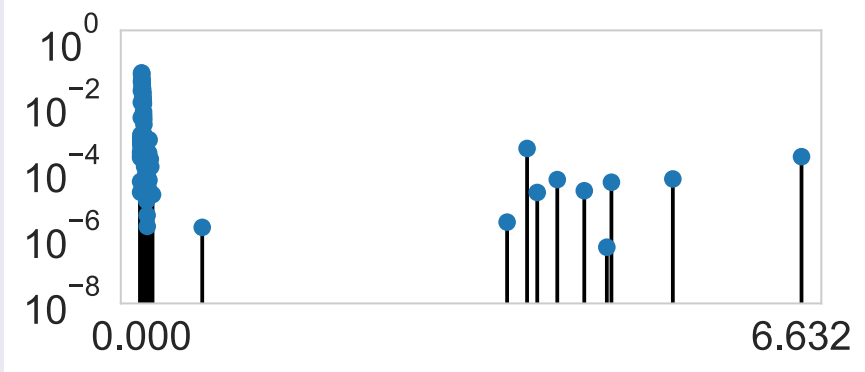
Figure: Evolution of the Variance σ and maximal eigenvalue λ_1 for both the Hessian H and the GGN matrix G, during SGD training.

Outliers and class Seperability

Single Layer MLP GMM Model



(a) Spectrum at Initialisation

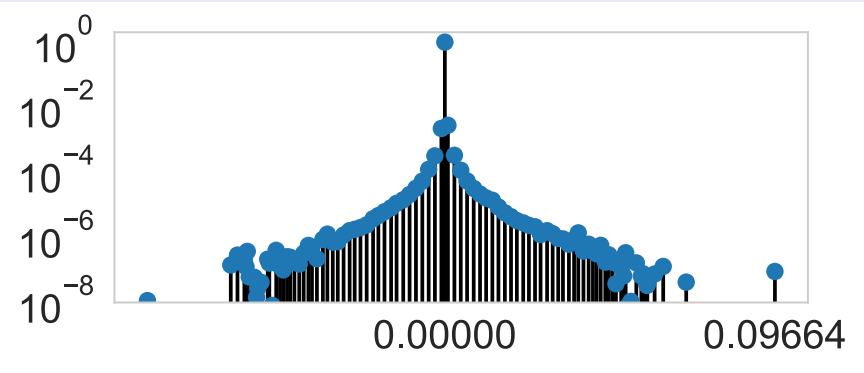


(b) Spectrum at Training End

Figure: Outlier persistence: Spectrum of a 1-Layer MLP on a 10 class Gaussian Mixture model at Initialisation (where performance is random, at 10%) and End of Training for 100 Epochs with $\alpha = 0.01$ and linear learning rate decay (where the performance is over 95%).

What does it mean to have no outliers?

VGG-16 example



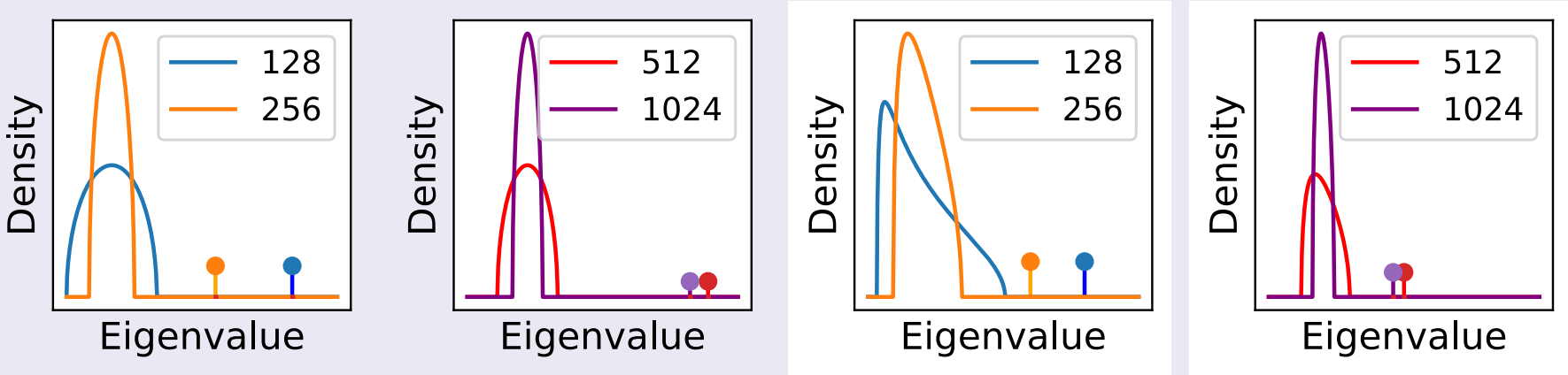
(a) Spectrum at Initialisation

(b) Spectrum at Training End

Figure: **VGG Network Always contains outlayers**. Spectrum of a 16-Layer VGG Network on the 10 class CIFAR-100 dataset, at Initalisation (where performance is random at 1%) and end of Training for 100 Epochs with $\alpha = 0.2, \gamma = 0$ and linear learning rate decay (where the performance remains random and is irrecoverable even with a learning rate drop).

What does it mean to have no outliers?

VGG-16 example



(a) Wigner Linear (b) Wigner Thresh (c) MP Linear (d) MP Threshold

Figure: **Variation of the spectral norm with batch size. Spectral norm decreases linearly until a threshold with batch size increase for both the Wigner and Machenko-Pastur noise models.** The continuous region (bulk) corresponds to the fluctuation matrix induced by mini-batching, shown as a Wigner semicircle (a & b) or Marchenko-Pastur (MP - c & d), whose width depends on the square root of the batch size). The largest eigenvalue of the batch Hessian is shown as a single peak, which decreases in magnitude as the batch size increases.

How does this affect learning rate choices?

Key Concept: Fluctuations Matrix Pushes out the Spectral Norm

$$\delta L(\boldsymbol{w} - \alpha \nabla L) = -\alpha ||\boldsymbol{g}(\boldsymbol{w})||^2 \left(1 - \frac{\alpha \sum_i^P \lambda_i ||\hat{\phi}_i \hat{\boldsymbol{g}}(\boldsymbol{w})||^2}{2}\right)$$
$$\mathbb{E}(\delta L(\boldsymbol{w} - \alpha \nabla L)) \leq -\alpha ||\boldsymbol{g}(\boldsymbol{w})||^2 \left(1 - \frac{\alpha \lambda_1(\boldsymbol{H}_{batch})}{2}\right) \tag{12}$$

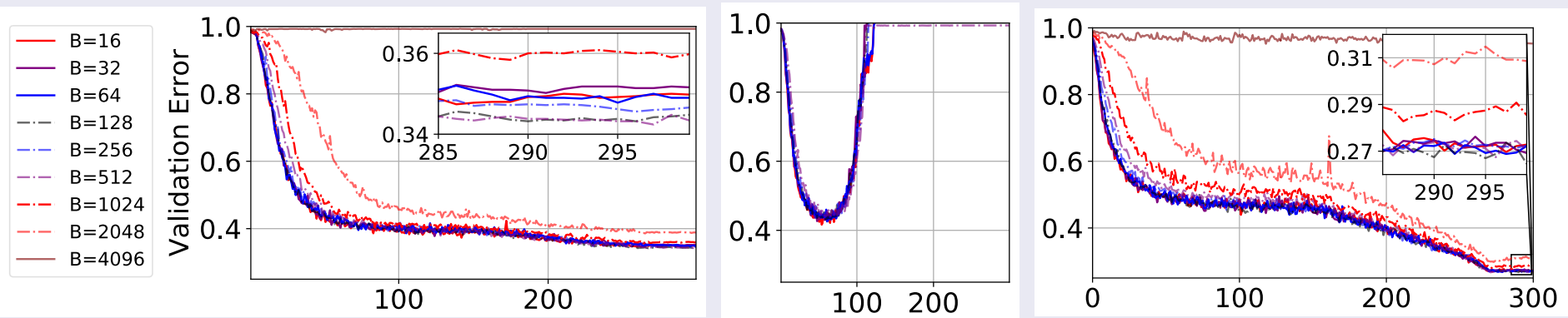
If most of the gradient is aligned in the outlier directions

$$1 - \frac{\alpha |\boldsymbol{g}(\boldsymbol{w})|^2}{2} \sum_i \beta_i^2 \left(\lambda_i + \frac{P\sigma^2}{\mathfrak{b}\lambda_i}\right) \tag{13}$$

How does this affect learning rate choices?

Key Concept: Fluctuations Matrix Pushes out the Spectral Norm

$$\delta L(\boldsymbol{w} - \alpha\nabla L) = -\alpha||\boldsymbol{g}(\boldsymbol{w})||^2\left(1 - \frac{\alpha\sum_i^P \lambda_i||\hat{\phi}_i\hat{\boldsymbol{g}}(\boldsymbol{w})||^2}{2}\right) \leq -\alpha||\boldsymbol{g}(\boldsymbol{w})||^2\left(1 - \frac{\alpha\lambda_1}{2}\right) \tag{14}$$

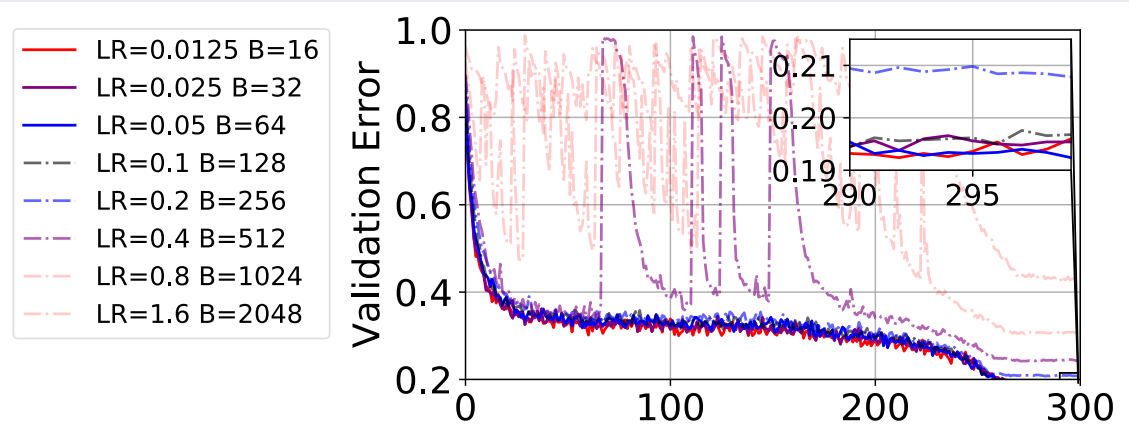


(a) No-BN $\alpha_0 = \frac{0.01B}{128}$, $\gamma = 0$ (b) $\alpha_0 = \frac{0.02B}{128}$ (c) BN $\alpha_0 = \frac{0.1B}{128}$, $\gamma = 5e^{-4}$

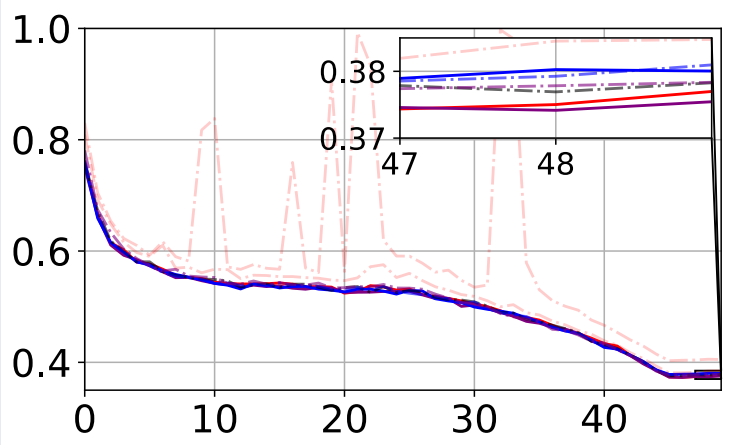
Figure: Validation error of the VGG-16 architechutre, with and without batch normalisation (BN) on CIFAR-100, with corresponding weight decay γ and initial learning rate α_0.

How does this affect learning rate choices?

Key Concept: Fluctuations Matrix Pushes out the Spectral Norm



(a) CIFAR-100 , $\gamma = 5e^{-4}$



(b) ImageNet-32, $\gamma = 5e^{-5}$

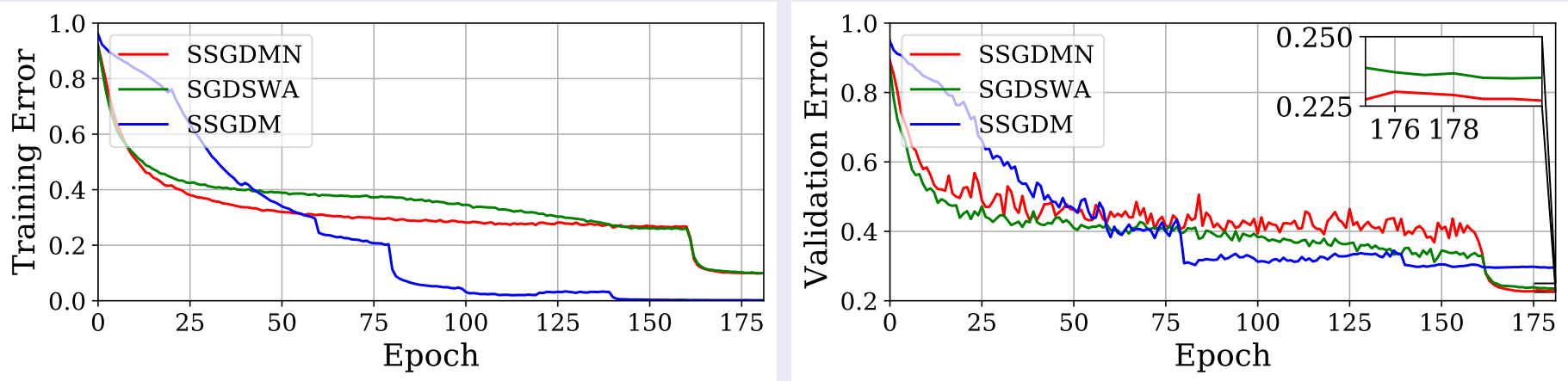
Figure: Validation error of the WideResNet-28 $\times$ 10 on the CIFAR-100 and ImageNet32 dataset, with initial learning rate $\alpha_0 = \frac{0.1B}{128}$ and weight decay γ.

Practical Application

Learning the Learning Rates

$$\alpha_{Polyak} = \frac{2}{\sqrt{\lambda_1} + \sqrt{\lambda_P}}, \; \alpha_{Nesterov} = \sqrt{\frac{\lambda_P}{\lambda_1}} \tag{15}$$

$$\rho_{Polyak} = \left(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_P}}{\sqrt{\lambda_1} + \sqrt{\lambda_P}} \right)^2, \rho_{Nesterov} = \left(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_P}}{\sqrt{\lambda_1} + \sqrt{\lambda_P}} \right). \tag{16}$$



(a) Training Error (b) Validation Error

Figure: **Learned Learning Rates seem Competitive with Fine Tuned** PreResNet-110 on the CIFAR-100 dataset, with weight decay $\gamma = 5e^{-4}$.

Application to Adam

Linear learning rate scaling rule for Adam looks unconvincing.

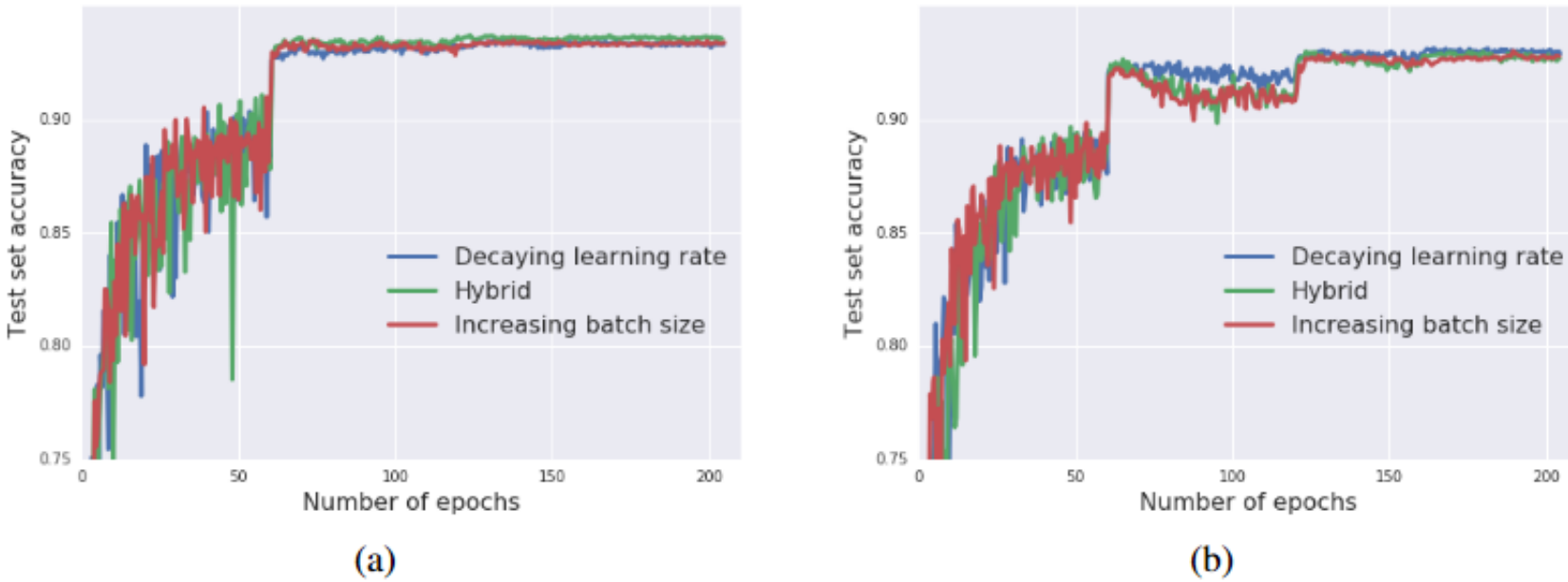


Figure 4: Wide ResNet on CIFAR10. The test set accuracy during training, for vanilla SGD (a) and Adam (b). Once again, all three schedules result in equivalent test set performance.

Figure: "Don't Decay the Learning Rate, Increase the Batch size" - Adam test accuracy clearly diverges from decayed learning rate (keeps decreasing) before final LR drop

What about Adam?

Extra Complication

$$L(\boldsymbol{w}_k - \alpha \boldsymbol{B}^{-1}\nabla L(\boldsymbol{w}_k)) - L(\boldsymbol{w}) = \alpha \nabla L(\boldsymbol{w}_k)^T \boldsymbol{B}^{-1}\nabla L(\boldsymbol{w}_k) + \frac{\alpha^2}{2}\nabla L(\boldsymbol{w}_k)^T \boldsymbol{B}^{-1}\boldsymbol{H}\boldsymbol{B}^{-1}\nabla L(\boldsymbol{w}_k). \tag{17}$$

Writing $\boldsymbol{H}_{emp} = \sum_i \lambda_i \boldsymbol{\psi}_i \boldsymbol{\psi}^T$ and $\boldsymbol{B} = \sum_j \eta_j \boldsymbol{\phi}_j \boldsymbol{\phi}_j^T$

$$L(\boldsymbol{w}_{k+1}) - L(\boldsymbol{w}) = \sum_i^P \frac{\alpha_0 |\boldsymbol{\phi}_i^T \nabla L(\boldsymbol{w})|^2}{\eta_i + \delta}\left(1 - \frac{\alpha_0}{2(\eta_i+\delta)}\sum_\mu \lambda_\mu |\boldsymbol{\psi}_u^T \boldsymbol{\phi}_i|^2)\right). \tag{18}$$

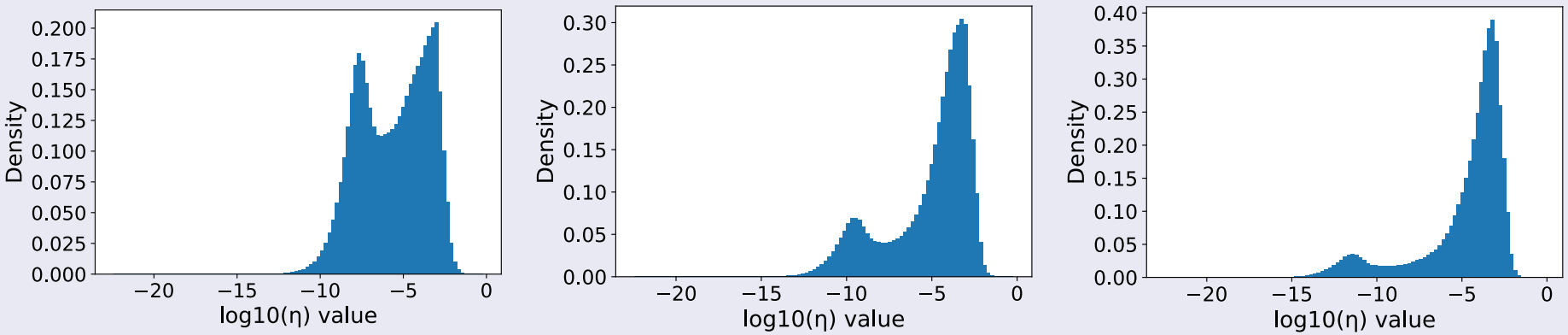
assume $|\boldsymbol{\psi}_u^T \boldsymbol{\phi}_i|^2 = \delta_{u,i}$, the for small δ we might incur greater loss at the edge of the bulk than outlier. This happens i.f.f

$$\frac{\sqrt{P}\sigma}{\sqrt{\mathfrak{b}}(\eta_i+\delta)} > \frac{\lambda_j + \frac{P\sigma^2}{\mathfrak{b}\lambda_j}}{(\eta_j+\delta)} \quad \text{i.e} \quad \frac{\eta_j+\delta}{\eta_i+\delta} > \left(\frac{\lambda_j\sqrt{\mathfrak{b}}}{\sqrt{P}\sigma} + \frac{\sqrt{P}\sigma}{\sqrt{\mathfrak{b}}\lambda_j}\right). \tag{19}$$

$$\left(1 - \frac{\alpha_0\sqrt{P}\sigma}{(\eta_i+\delta)\sqrt{\mathfrak{b}}}\right) > 0 \therefore \alpha_0 < \frac{\sqrt{\mathfrak{b}}\kappa}{\sqrt{P}\sigma} \leq \frac{\sqrt{\mathfrak{b}}(\eta_i+\delta)}{\sqrt{P}\sigma}, \tag{20}$$

Big Idea

Bulk Edge eigenvalue/eigenvector pairs estimated worse than outliers



(a) Epoch 25 (b) Epoch 50 (c) Epoch 75

Figure: **Huge Variation in Scaling Coefficients for Adam.** Density of pseudo eigenvalues η_i learned during training a VGG-16 on CIFAR-100 using the Adam optimiser for different epoch values.

Big Idea

Bulk Edge eigenvalue/eigenvector pairs estimated worse than outliers

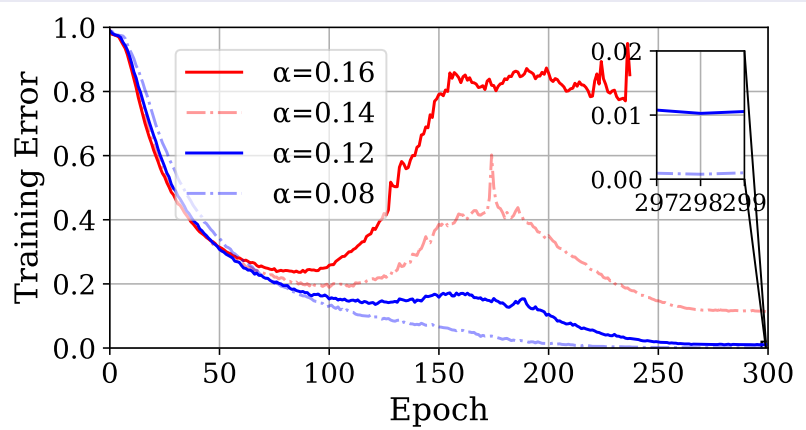


Figure: Adam $\delta = 1$

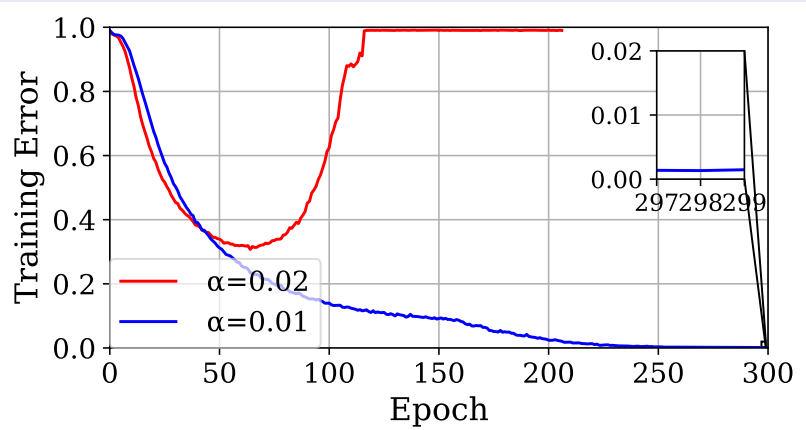
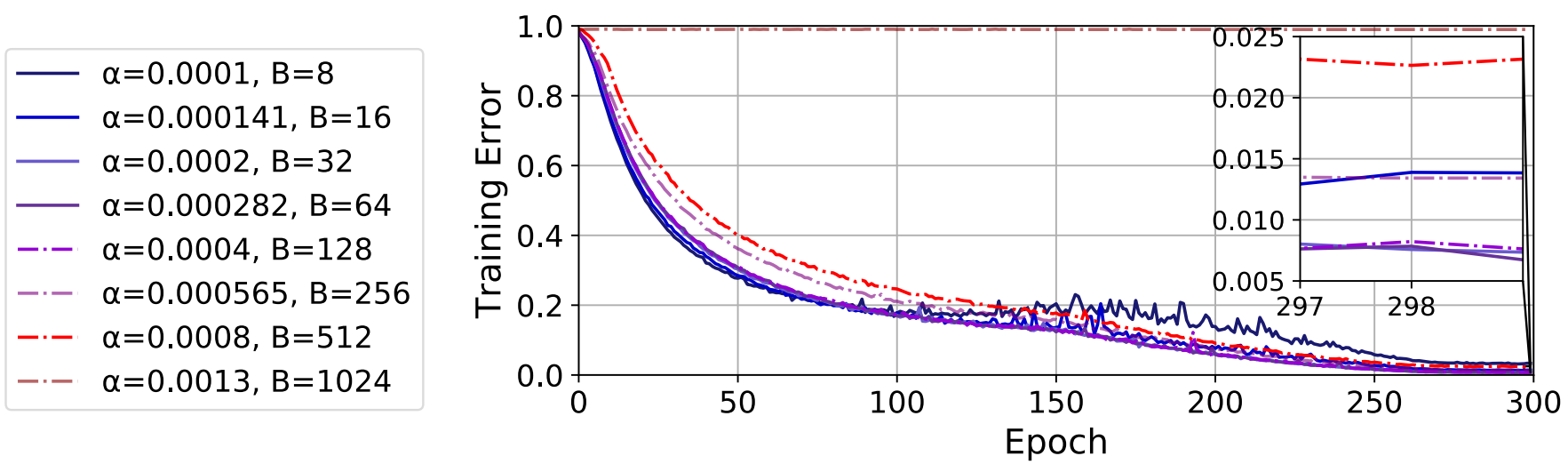
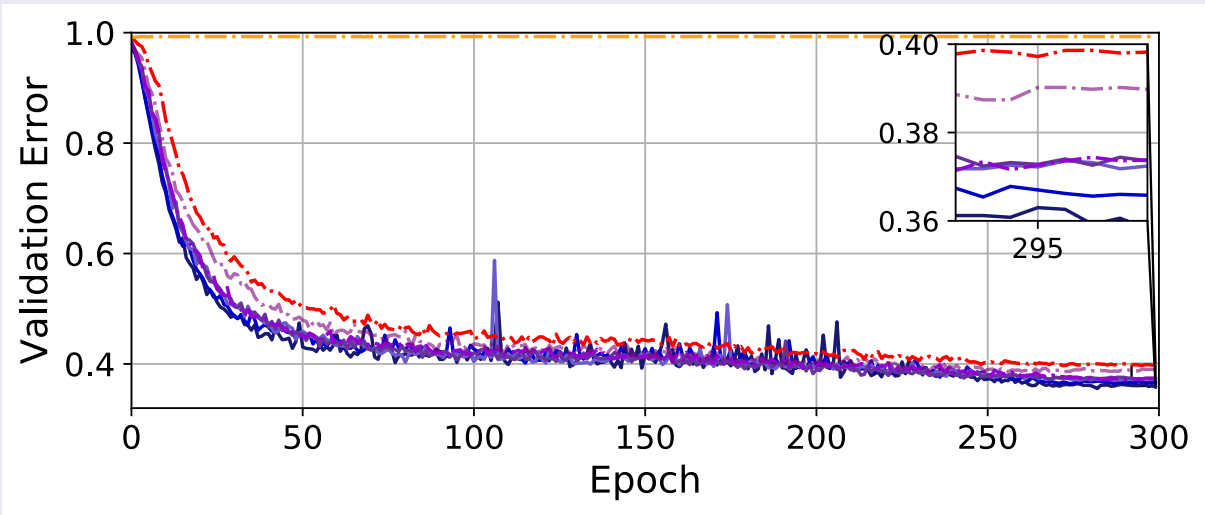


Figure: SGD

Square Root Scaling For Adam

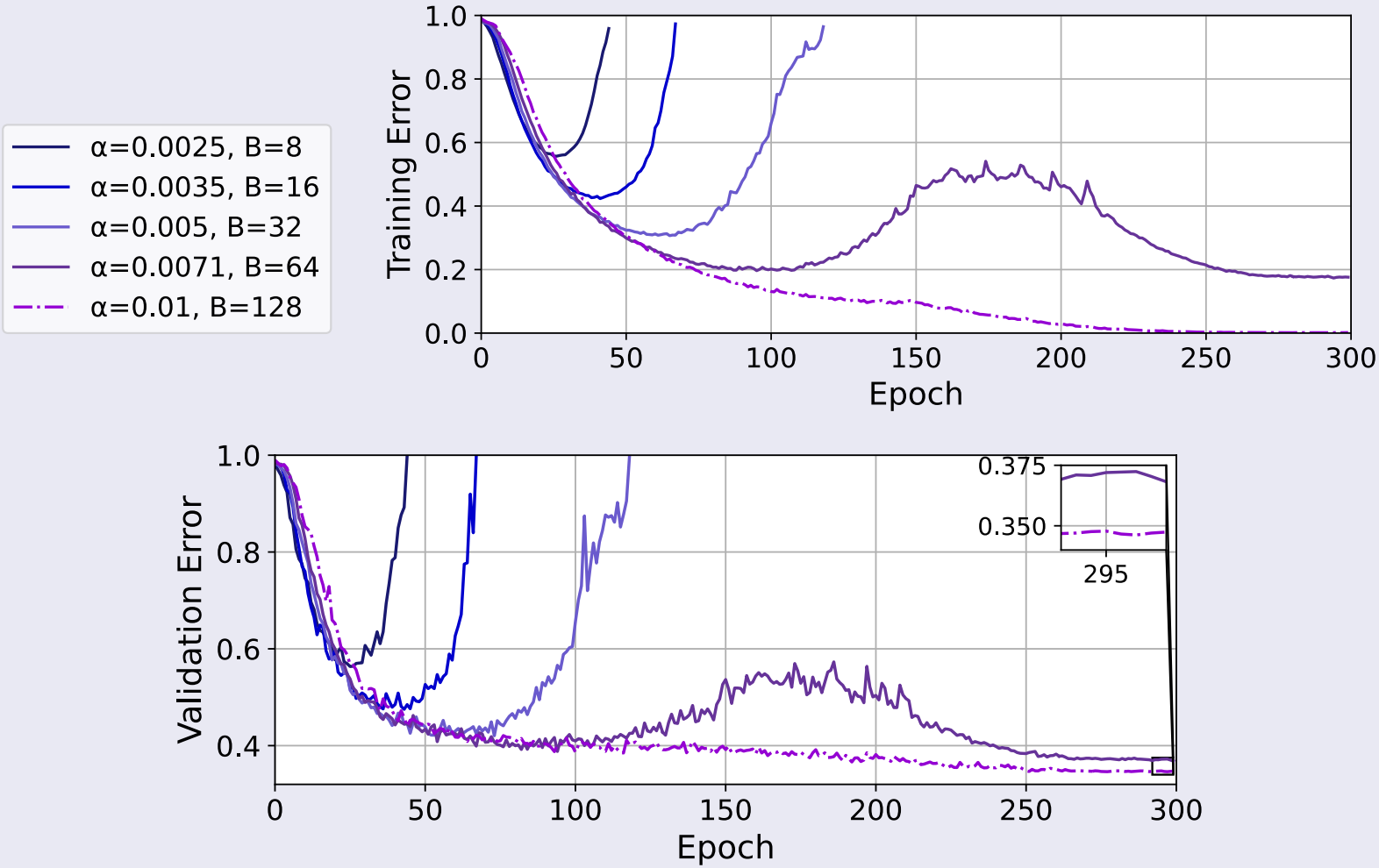
Consistent up to a threshold.





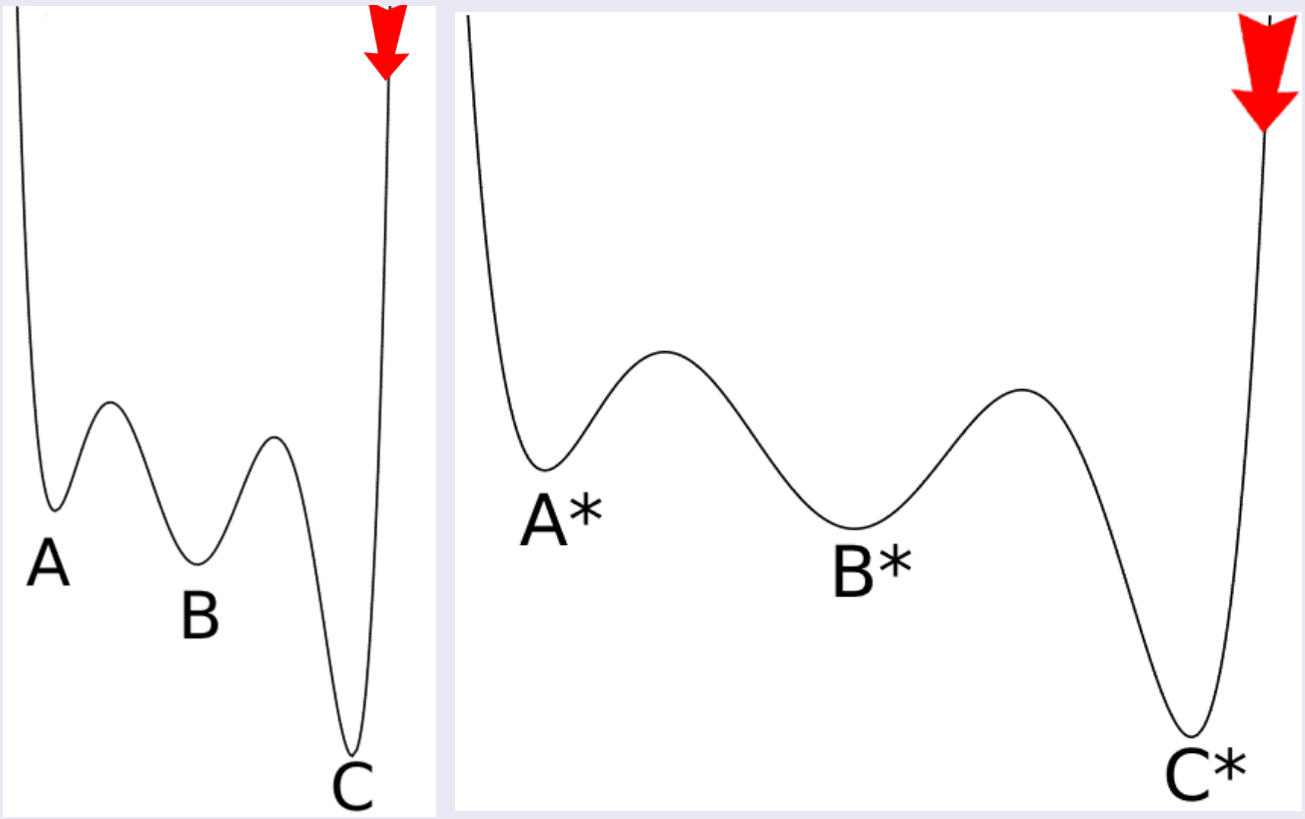
Square Root Scaling For SGD

Square root scaling for SGD does not hold.



Validation Curves

Intuition on Generalisation Error



(a) Small B (b) Large B

Figure: Transformed Test Set Surface, going from sharper to flatter with an increase in Batch Size. Larger learning rates are more able to escape local poor quality mimima which are close to the initialisation.

Validation Curves

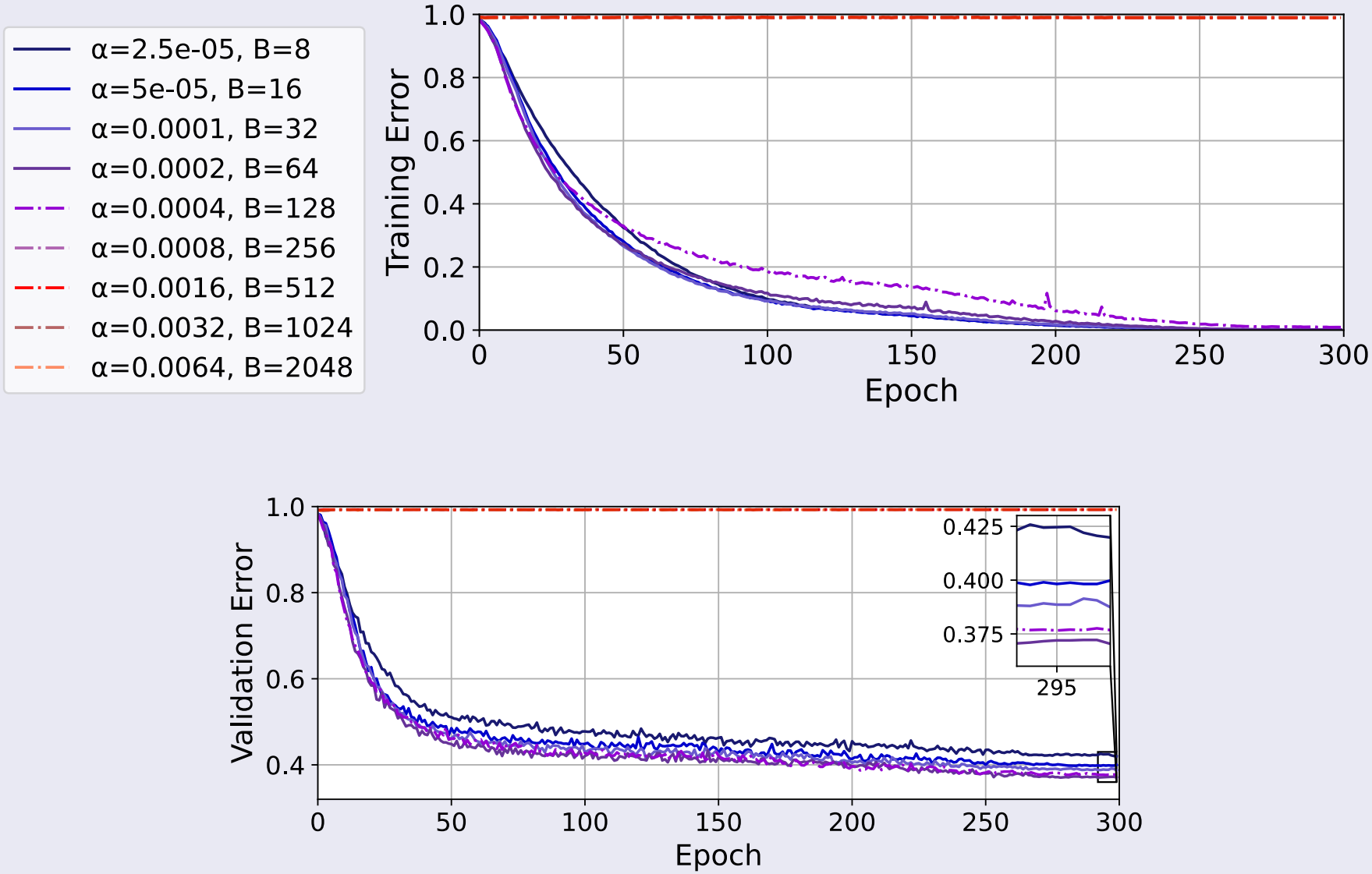
Intuition on Generalisation Error

α	Train Acc	Val Acc	Test Acc	$\|\|\Delta_{init}\|\|$
0.08	99.7%	64.35%	65.36%	145.76
0.028	99.89%	61.08%	62.45%	67.44

Table: **Larger distances from initalisation in weight space give solutions of greater generalisation.** Various Learning Rates α for $B = 1024$, with corresponding val/test accuracies and L_2 distance from initialisation for CIFAR-100 on the VGG-16.

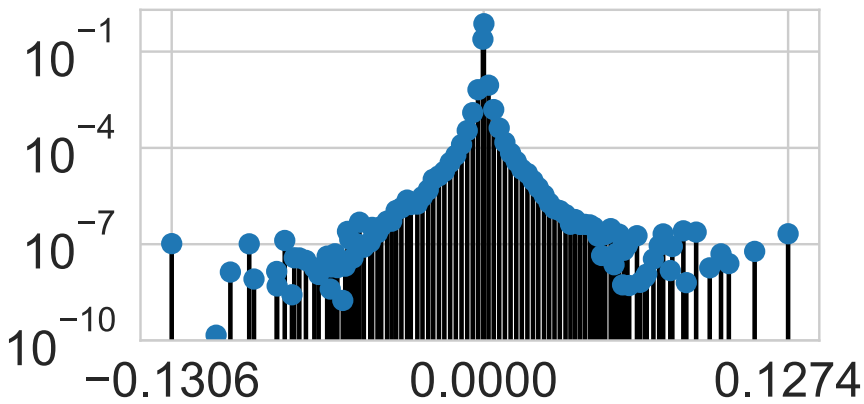
Linear Scaling For Adam

Linear rate does not hold for Adam

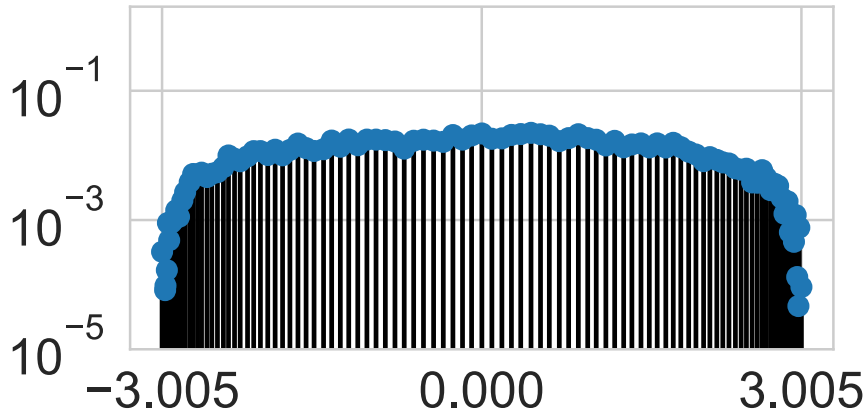


Beyond RMT

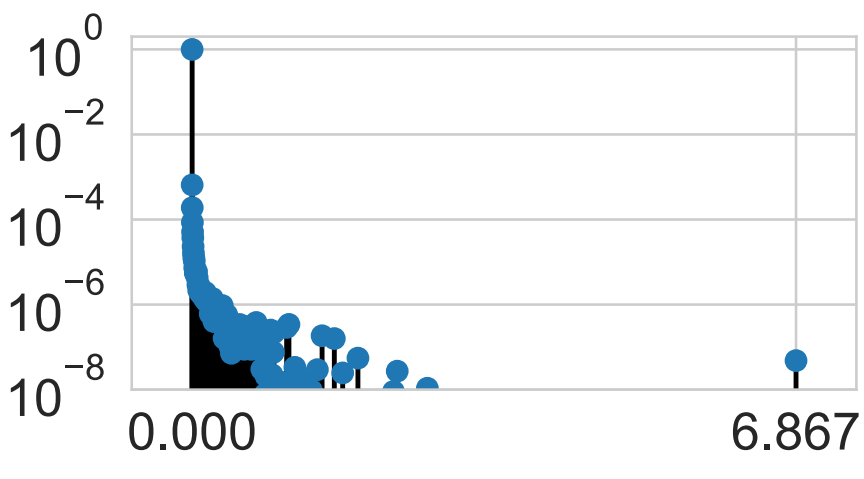
A snag in the plot: We never see the semi-circle law of MP



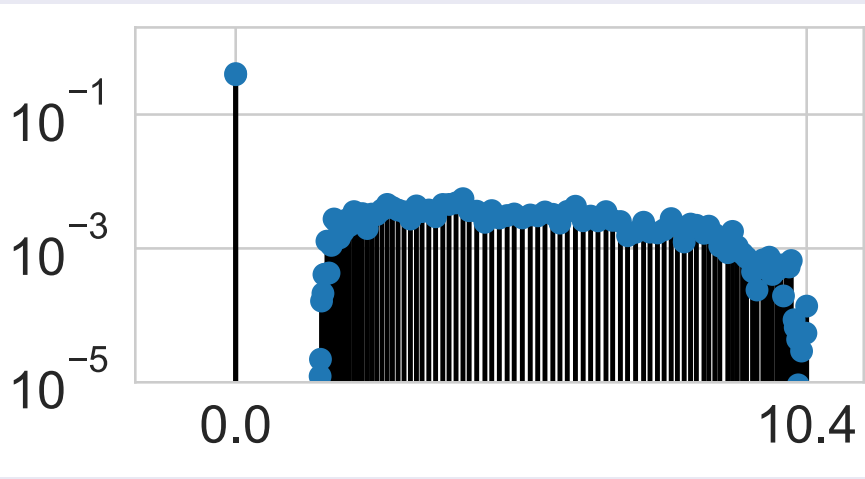
(a) r VGG-16 Epoch 300



(b) Wigner



(c) $\boldsymbol{G}$ VGG-16 Epoch 300

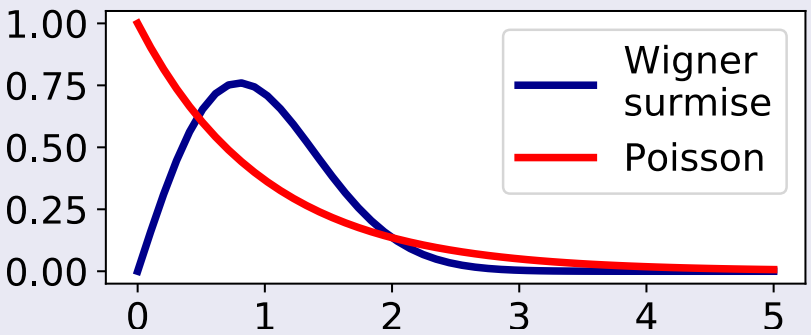


(d) MP $q = 6$

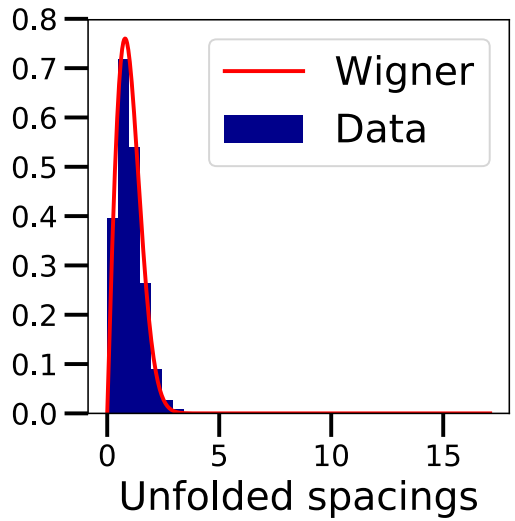
The Winger Surmise

Any k point correlation function attainable by marginalisation $\mathbb{E}\rho^{(P)}(\lambda) = \int p(\lambda, \lambda_2, \ldots, \lambda_P) d\lambda_2 \ldots d\lambda_P$.

For the 2x2 GOE, $p(\lambda_1, \lambda_2) \propto |\lambda_1 - \lambda_2| e^{-\frac{1}{2}(\lambda_1^2 + \lambda_2^2)}$ $\nu_2, \nu_1 = \lambda_1 \pm \lambda_2$, $s = |\nu_1|$ $\rho_{Wigner}(s) = \frac{\pi s}{2} e^{-\frac{\pi}{4} s^2}$,



(a) Wigner Surmise



(b) MNIST MLP

Figure: Local Spectral Statistics match GOE

Beyond RMT

The Wigner Surmise

For larger matrices, the j.p.d.f must include an indicator function $1\{\lambda_1 \leq \lambda_2 \leq \ldots \lambda_P\}$

studying pairs of *adjacent* eigenvalues

Wigner surmise holds to high accuracy for the NNSD of GOE matrices of any size provided that the eigenvalues have been scaled to give mean spacing 1

The Wigner surmise density vanishes at 0, capturing "repulsion" between eigenvalues that is characteristic of RMT statistics

In contrast to the distribution of entirely independent eigenvalues given by the *Poisson law* $\rho_{Poisson}(s) = e^{-s}$.

The Wigner surmise is universal in that the same density formula applies to all real-symmetric random matrices, not just the GOE or Wigner random matrices.

SWA/ASGD/etc.. i.e variants of iterate averaging

Long known [1] often written off in Deep Learning [6, 2]

$$\begin{aligned} \boldsymbol{w}_{i+1} &= \boldsymbol{w}_i - \alpha \nabla L(\boldsymbol{w}_i) \\ \boldsymbol{w}_{avg} &= \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{w}_k \end{aligned} \tag{21}$$

instead exponential moving average or final point used instead

rediscovered as *NEW* technique for generalisation in SWA and ASGD [3, 4, 1]

Limited theoretical justification for *Generalisation*

Not combined with Adaptive Methods (*SGD generalises best**)

Iterate Averaging

A true risk model

simplest model

High dimensional quadratic of true risk

each gradient perturbed by i.i.d normal noise

Iterate Averaging

Theorem

Assume the aforementioned quadratic loss and i.i.d. Gaussian gradient noise model. Assume further that $\alpha\lambda_i \ll 1$ *for all* i *and* $\lambda_i > 0$ *for all* i. *Then there exists a constant* $c > 0$ *such that for all* $t > 0$, *as* $n \to \infty$

$$\mathbb{P}\left(\left|\sqrt{\sum_{i}^{P}\left(w_{n,i} - w_{0,i}\mathrm{e}^{-n\alpha\lambda_i}(1+o(1))\right)^2} - \sqrt{P\frac{\alpha\sigma^2}{B}\left\langle\frac{1}{\lambda(2-\alpha\lambda)}\right\rangle}\right| \geq t\right) \leq \nu(t),$$

$$\mathbb{P}\left(\left|\sqrt{\sum_{i}^{P}\left(w_{\mathrm{avg},i} - \frac{w_{0,i}}{\lambda_i n\alpha}(1+o(1))\right)^2} - \sqrt{\frac{P\sigma^2}{Bn}\left\langle\frac{1}{\lambda}\right\rangle}\right| \geq t\right) \leq \nu(t), \tag{22}$$

where $\nu(t) = 2\exp(-ct^2)$ *and* B *is the batch size.*

Iterate Averaging

Importance of Dimension

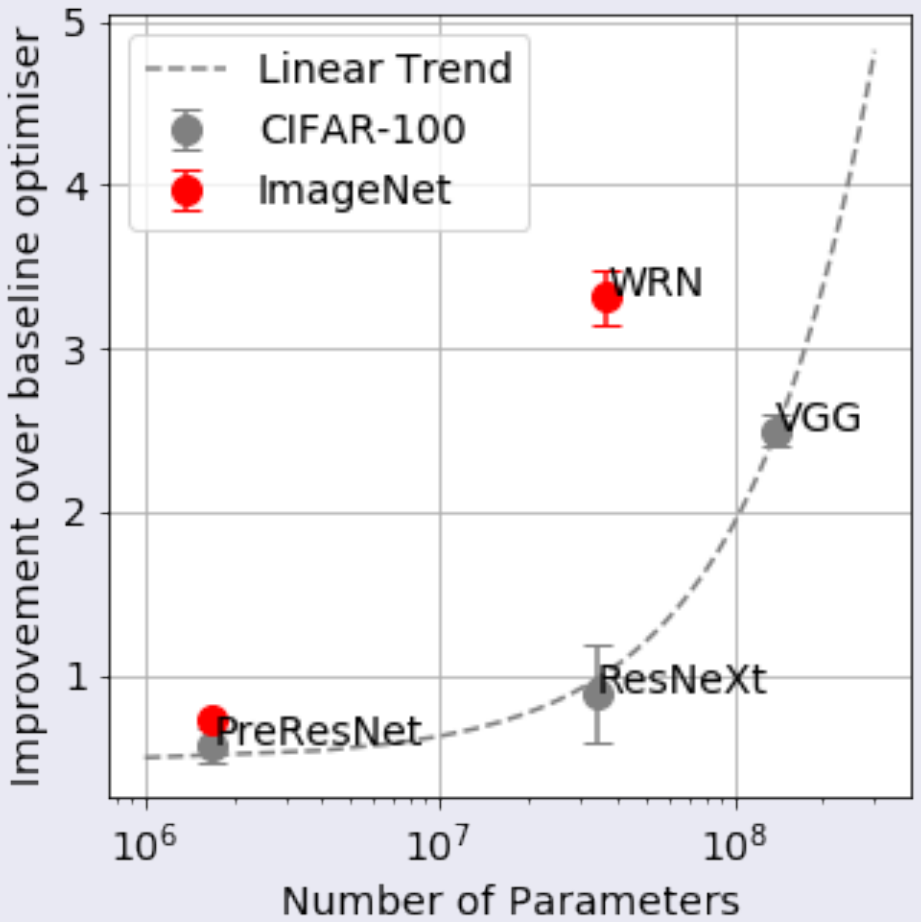


Figure: Improvement vs. P. We expect greater improvement for larger models.

Iterate Averaging

Perturbation as a function of Distance

$$Cov(\epsilon_i(\boldsymbol{w}), \epsilon_j(\boldsymbol{w}')) = \partial_{w_i}\partial_{w_j'} k(\boldsymbol{w}, \boldsymbol{w}') \tag{23}$$

and further, assuming a stationary kernel $k(\boldsymbol{w}, \boldsymbol{w}') = k\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}||_2^2\right)$ (note the slight abuse of notation):

$$(w_i - w_i')(w_j' - w_j)k''\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}'||_2^2\right) + \delta_{ij}k'\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}'||_2^2\right).$$

Thus we have a non-trivial covariance between gradient noise at different points in weight-space. This covariance structure can be used to prove the following variance reduction result.

Depenedence needs a large learning rate

Theorem (Dependent Perturbation between True & Empirical Risk)

Let $\boldsymbol{w}_n$ and $\boldsymbol{w}_{avg}$ be defined as in Theorem 5 and let the gradient noise be given by the covariance structure in (24). Assume that the kernel function k is such that $k'(-x^2)$ and $x^2 k''(-x^2)$ decay as $x \to \infty$, and define $\sigma^2 B^{-1} = k'(0)$ Assume further that $P \gg \log n$. Let $\delta = o(P^{1/2})$. Then $\boldsymbol{w}_n$ and $\boldsymbol{w}_{avg}$ are multivariate Gaussian random variables and, with probability which approaches unity as $P, n \to \infty$ the iterates $\boldsymbol{w}_t$ are all mutually at least δ apart

$$\mathbb{E} w_{n,i} \sim e^{-\alpha \lambda_i n} w_{0,i}, \frac{1}{P} \operatorname{Tr} Cov(\boldsymbol{w}_n) \sim \frac{1}{P} \sum_{i=1}^{P} \frac{\alpha \sigma^2}{B \lambda_i (2 - \alpha \lambda_i)}, \mathbb{E} w_{avg,i} \sim \frac{1 - \alpha \lambda_i}{\alpha \lambda_i n} w_{0,i} \tag{24}$$

$$\frac{1}{P} \operatorname{Tr} Cov(\boldsymbol{w}_{avg}) \leq \frac{\sigma^2}{Bn} \left\langle \frac{1}{\lambda^2} \right\rangle + \mathcal{O}(1) \left(k'(-\frac{\delta^2}{2}) + P^{-1} \delta^2 k''(-\frac{\delta^2}{2}) \right). \tag{25}$$

How does the picture change if we stride?

Corollary

Let $\boldsymbol{w}_{avg}$ now be a strided iterate average with stride κ, i.e.

$$\boldsymbol{w}_{avg} = \frac{\kappa}{n} \sum_{i=1}^{\lfloor n/\kappa \rfloor} \boldsymbol{w}_i. \tag{26}$$

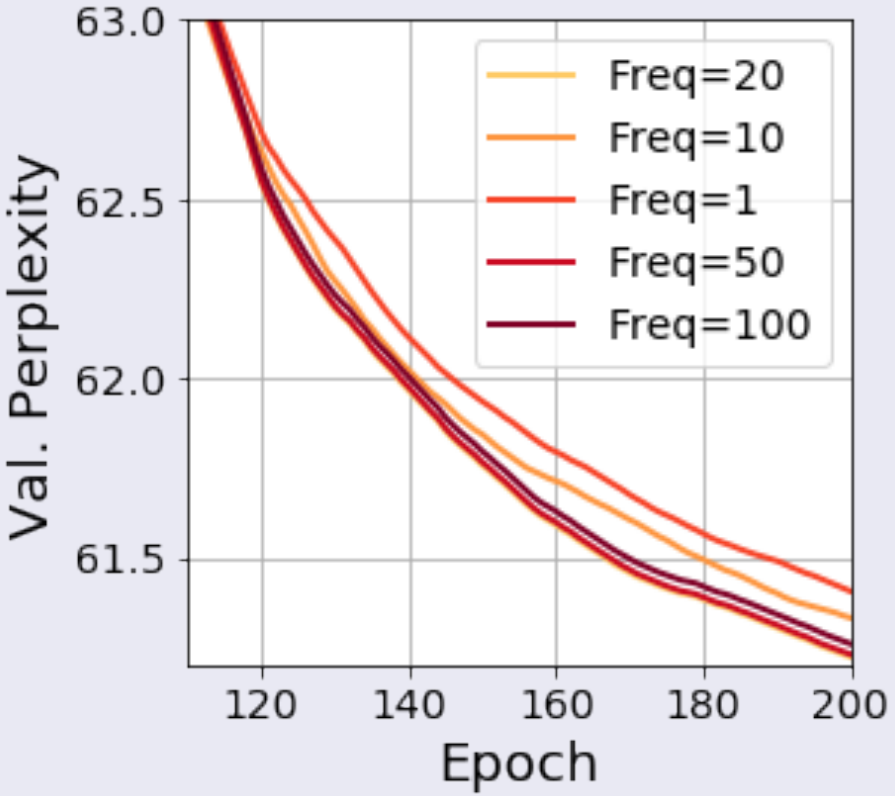
Then, under the same conditions as Theorem 2 (or Theorems 3 or 4)

$$\mathbb{E} w_{avg,i} = \frac{\kappa(1-\alpha\lambda_i)^{\kappa}}{n(1-(1-\alpha\lambda_i)^{\kappa})}(1+o(1))w_{0,i}, \tag{27}$$

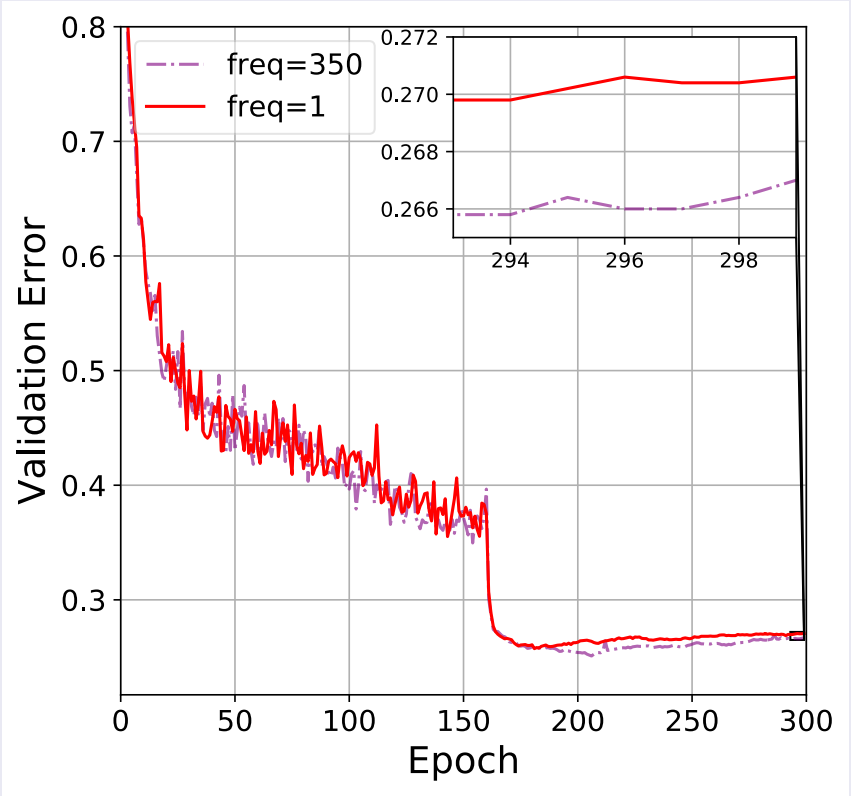
$$\frac{1}{P}\,\mathrm{Tr}\,Cov(\boldsymbol{w}_{avg}) \leq \frac{\sigma^2\alpha^2\kappa}{Bn}\left\langle \frac{1}{(1-(1-\alpha\lambda)^{\kappa})^2}\,\frac{1-(1-\alpha\lambda)^{2\kappa}}{1-(1-\alpha\lambda)^2}\right\rangle + \mathcal{O}(1)\left(k'(-\frac{\delta^2}{2}) + P^{-1}\delta^2 k''(-\frac{\delta^2}{2})\right) \tag{28}$$

where the constant $\mathcal{O}(1)$ coefficient of the second term in (28) is independent of κ.

Empirical Averaging Frequency Importance



(a) LSTM on PTB

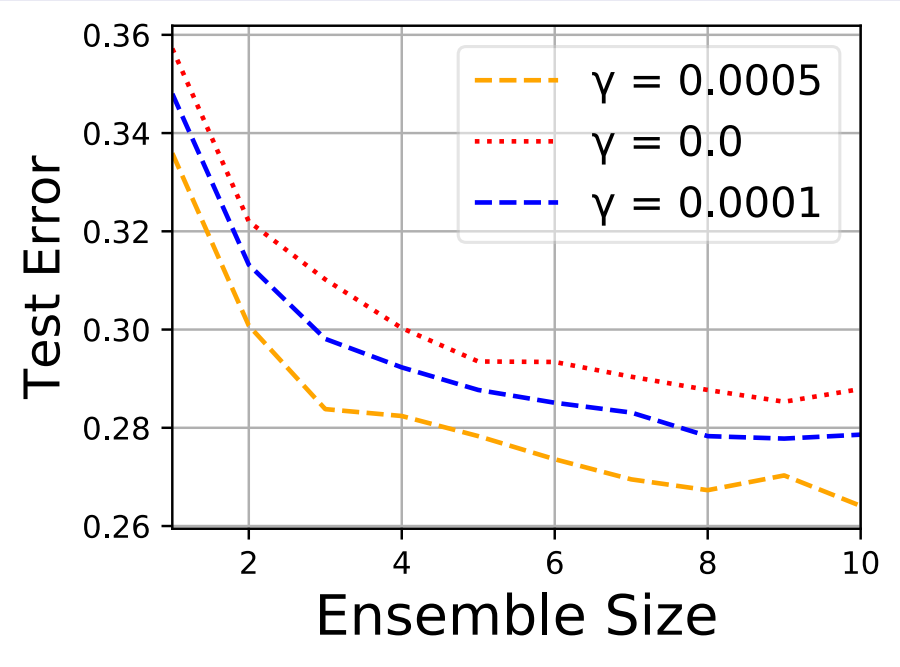


(b) VGG-16 on CIFAR-100

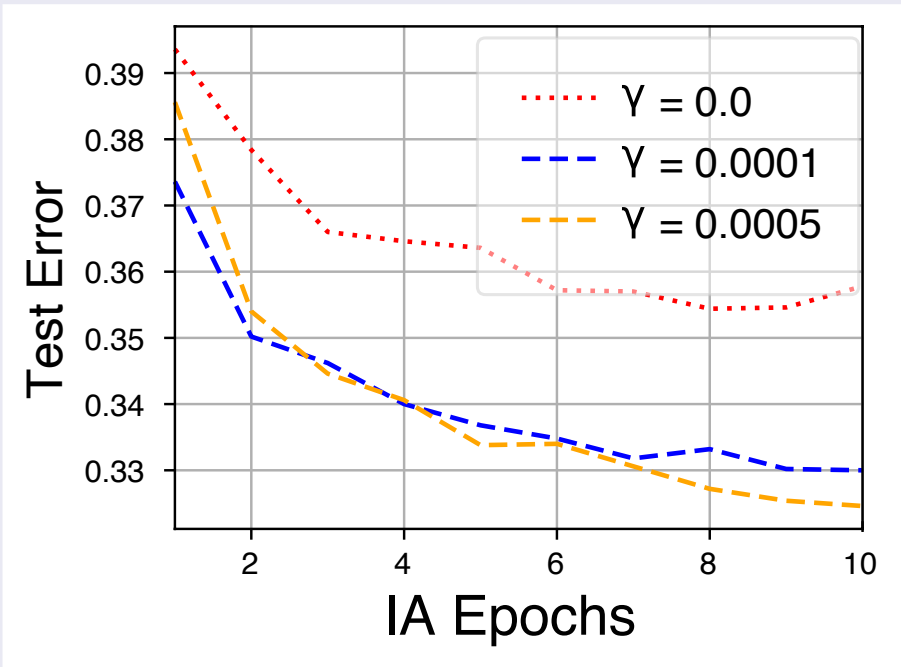
Figure: Effect of different averaging frequencies on validation perplexity of Gadam on representative (a) Language and (b) Image classification tasks. `Freq=`n suggests averaging once per n iterations. `freq=350` in (b) is equivalently averaging once per *epoch*.

Iterate Averaging and Ensembling?

Importance of Weight Decay for IA



(a) Network Ensemble



(b) Iterate Averaging

Iterate Averaging and Regularisation

Increasing weight decay reduces epoch distance in weight space

For $\gamma = \{0, 0.0001, 0.0005\}$

$|\boldsymbol{d}|^2 \downarrow 17.7, 14.9, 13.9,$ $|\boldsymbol{d}|^2/||\boldsymbol{w}||^2 \uparrow$ increases $0.11, 0.13, 0.22$

Theorem (Extension to Relative Distance in Weight Space)

Let $\boldsymbol{w}_n$ and $\boldsymbol{w}_{avg}$ be defined as in Theorem 6 and let the gradient noise be given by the basic covariance structure in (23). Let the kernel function be of the form

$$k(\boldsymbol{w}, \boldsymbol{w}') = k\left(-\frac{||\boldsymbol{w} - \boldsymbol{w}'||_2^2}{||\boldsymbol{w}||_2^2}\right)$$

and assume that the kernel function k is such that $k'(-x^2)$ and $x^2 k''(-x^2)$ decay as $x \to \infty$, and define $\sigma^2 B^{-1} = k'(0)$ Assume further that $P \gg \log n$. Then the result of Theorem 6 holds.

Iterate Averaging for Adaptive Optimisers

Theorem (for Adaptive Optimisers, hp = high probability)

Fix some $\zeta > 0$, assume that $|\tilde{\lambda}_i^{(t)} - \lambda_i| < \zeta$ for all $t \geq n_0$, for some fixed $n_0(\zeta)$, with hp. λ_i are bounded away from zero and $\min_i \lambda_i > \zeta$. Assume $c(\gamma + \varepsilon + \zeta) < 1$, where the constant is independent of $\varepsilon, \zeta, \gamma$ defined below. Let everything else be as in Theorem 6. Then there exist constants $c, c_1, c_2, c_3 > 0$ such that, with hp,

$$|\mathbb{E} w_{n,i} - w_i^*| \leq e^{-\alpha(1+\gamma-c(\varepsilon+\zeta))n} w_{0,i} + c_1(\varepsilon + \zeta + \gamma) \tag{29}$$

$$\left| \frac{1}{P} \operatorname{Tr} Cov(\boldsymbol{w}_n) - \frac{\alpha\sigma^2}{B(2-\alpha)} \right| \leq c_2(\varepsilon + \zeta + \gamma) + o(1), \tag{30}$$

$$|\mathbb{E} w_{avg,i} - w_i^*| \leq \frac{1 - \alpha(1 + \gamma - c(\varepsilon + \zeta))}{\alpha(1 + \gamma - c(\varepsilon + \zeta))n}(1 + o(1)) w_{0,i} + c_3(\varepsilon + \zeta + \gamma) \tag{31}$$

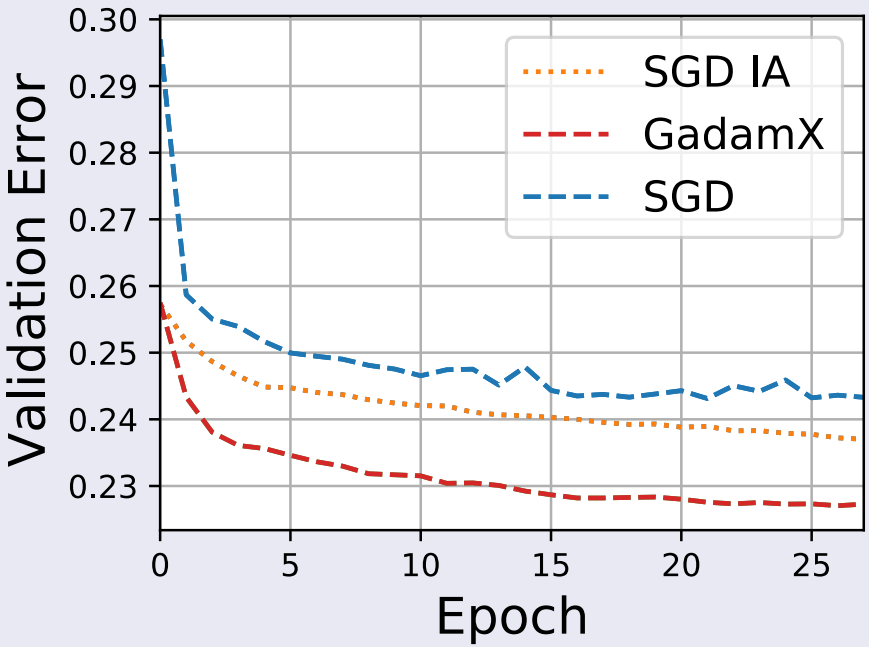
$$\frac{1}{P} \operatorname{Tr} Cov(\boldsymbol{w}_{avg}) = o(1). \tag{32}$$

Iterate Averaging for the Win

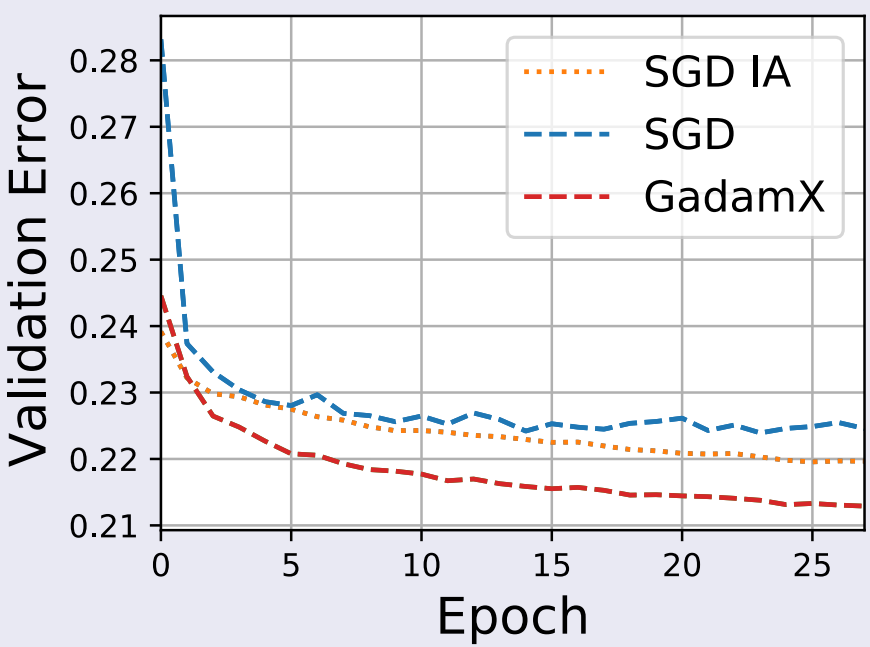
Squeezing out the Best Test Error with Adaptive Iterate Averaging

Significant gains shown on Image Problems

over SGD & SGD+IA (SWA)



(a) ResNet-50



(b) ResNet-101

Iterate Averaging for the Win

Squeezing out the Best Test Error with Adaptive Iterate Averaging

Architecture	Optimiser	Top-1	Top-5
ResNet-50	SGD(step)	75.63	92.67
	SWA	76.32	93.15
	AdamW(lin)	74.04	91.57
	Ranger	75.64	92.53
	Gadam	76.79	93.21
	GadamX	**77.31**	**93.47**
ResNet-101	SGD (step)	77.37	93.78
	SWA	78.08	93.92
	AdamW(lin)	74.48	91.82
	Ranger	75.62	92.42
	Gadam	78.53	**94.29**
	GadamX	**78.72**	94.18

Iterate Averaging for the Win

Squeezing out the Best Test Error with Adaptive Iterate Averaging

Have a hunch this will work well for NLP tasks

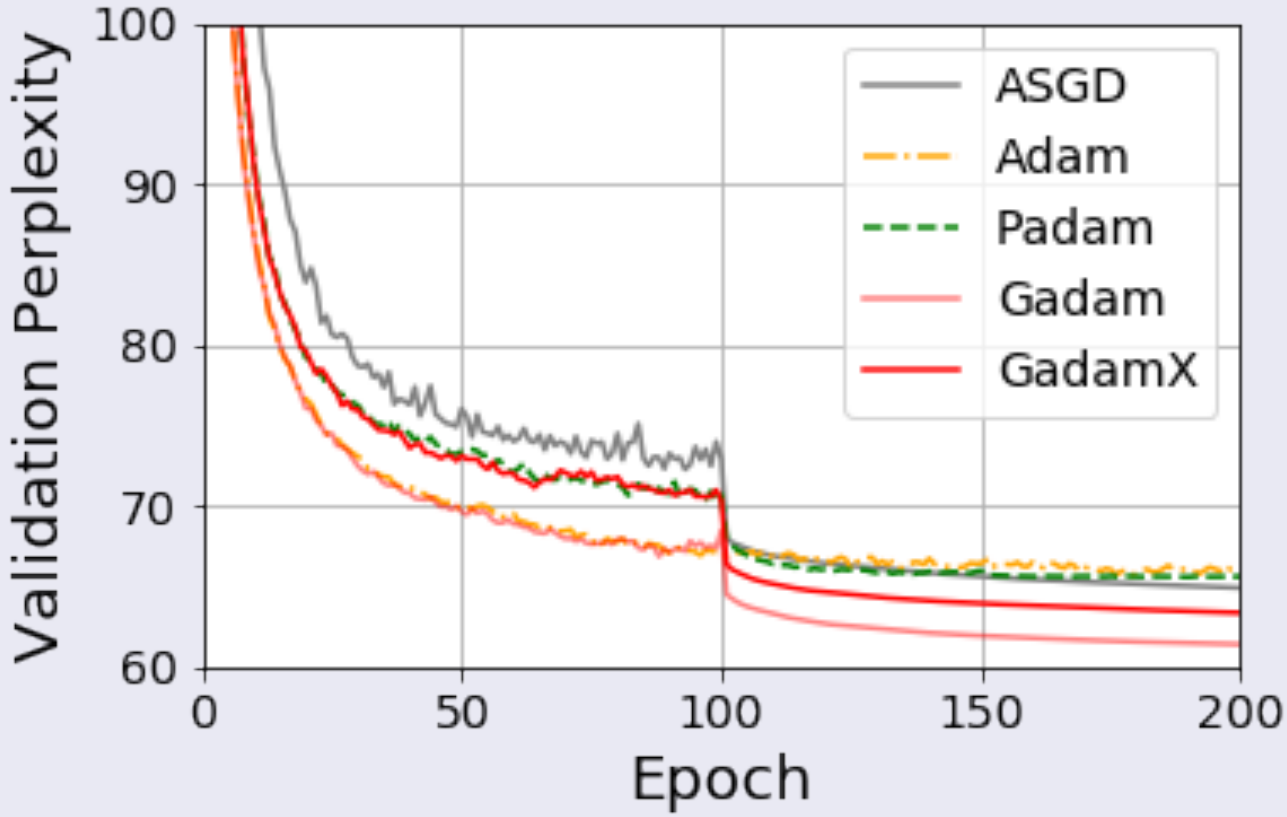
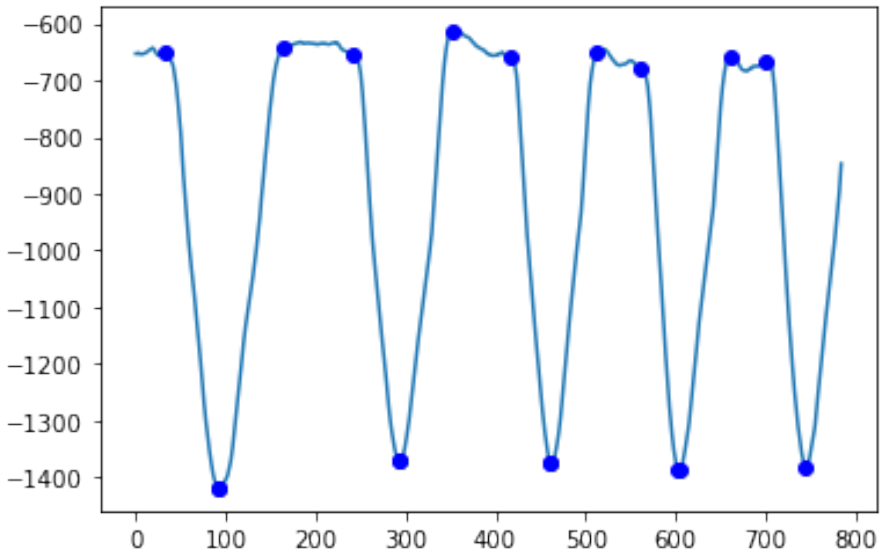
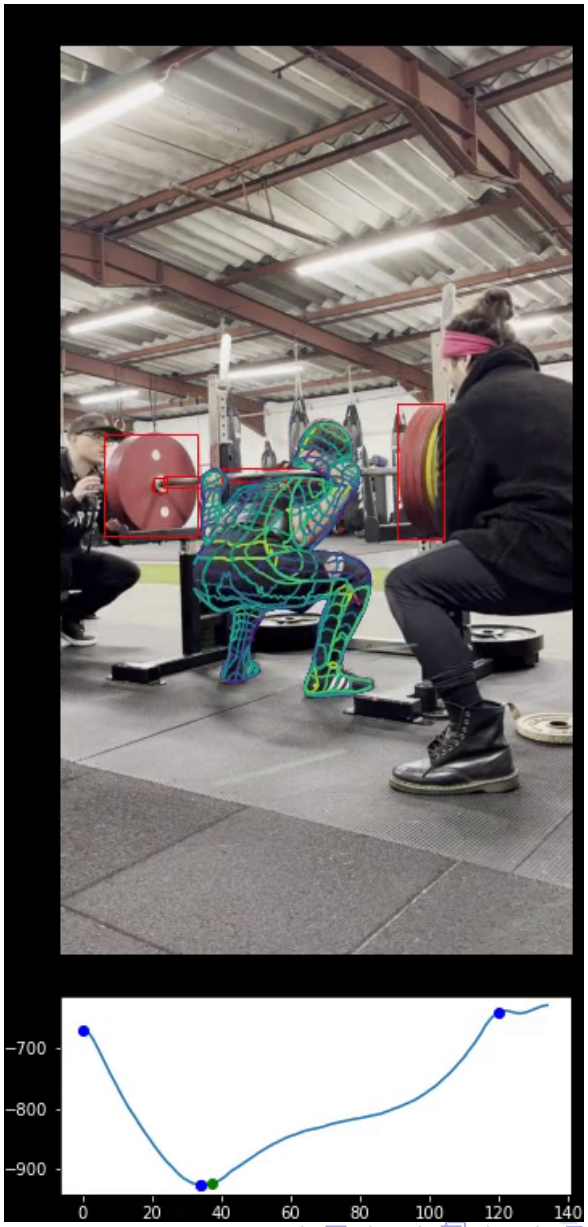


Figure: Validation perplexity of 3-layer LSTM on PTB word-level modelling

PureStrength - Computer Vision for Powerlifting





PureStrength - Computer Vision for Powerlifting

Customer Obsession

- Velocity training for programming - requested
- Creating video content for beginners - requested

Ownership

- Collected the data and labelled it myself
- Lots of Software and Environment Engineering
- Layered product driven algorithm development

Frugality

- Trained the weight detection model on my $4GB$ GPU laptop overnight

Learn and be Curious

- Complete novice at Pose Detection, Model Compression, NAS

Insist on the Highest Standards & Think Big

Invent and Simplify

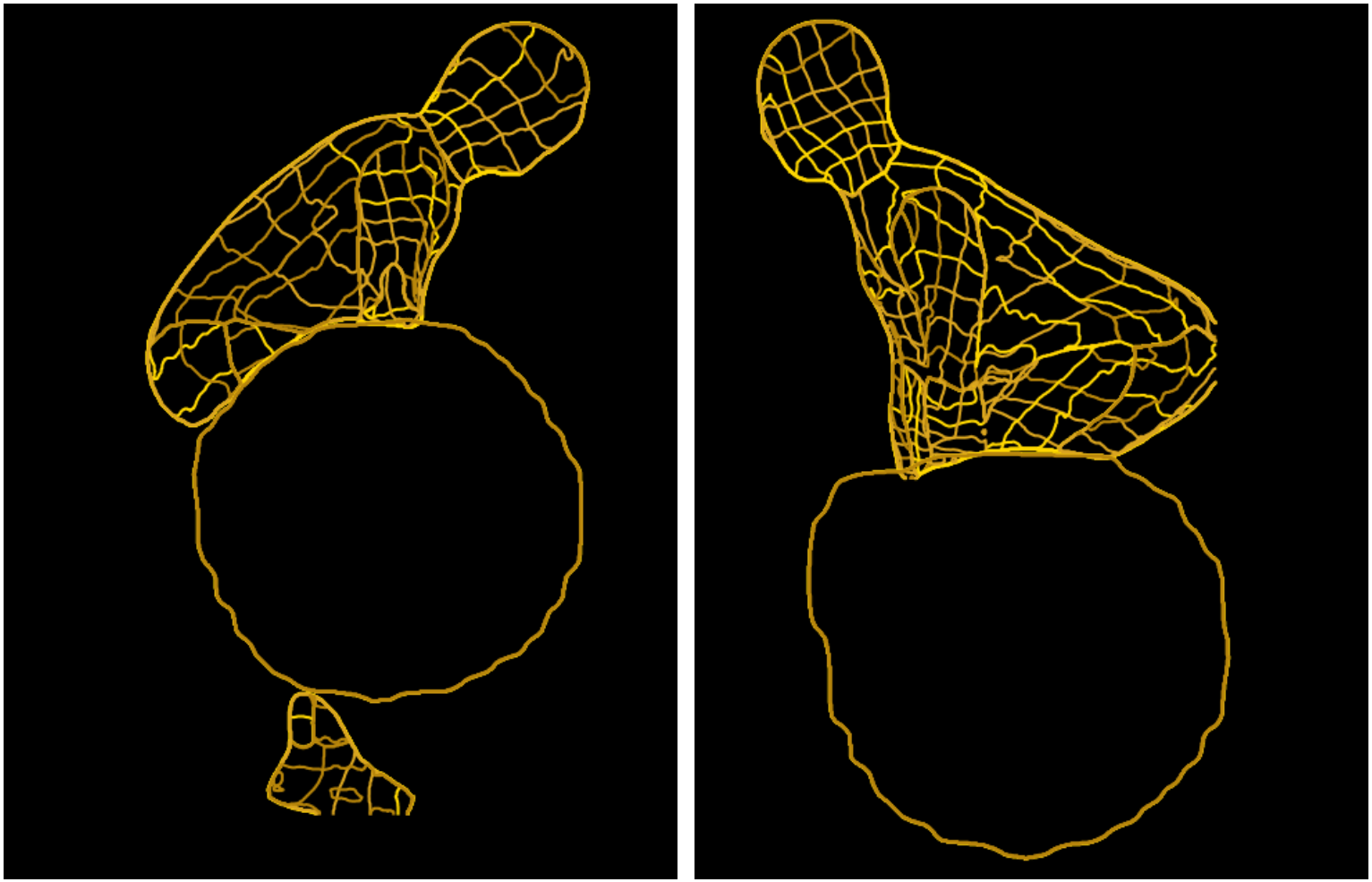
Excercise induced lower backpain

- Incorrect technique increases the probability of injury

Example Invent and Simplify

Excercise induced lower backpain

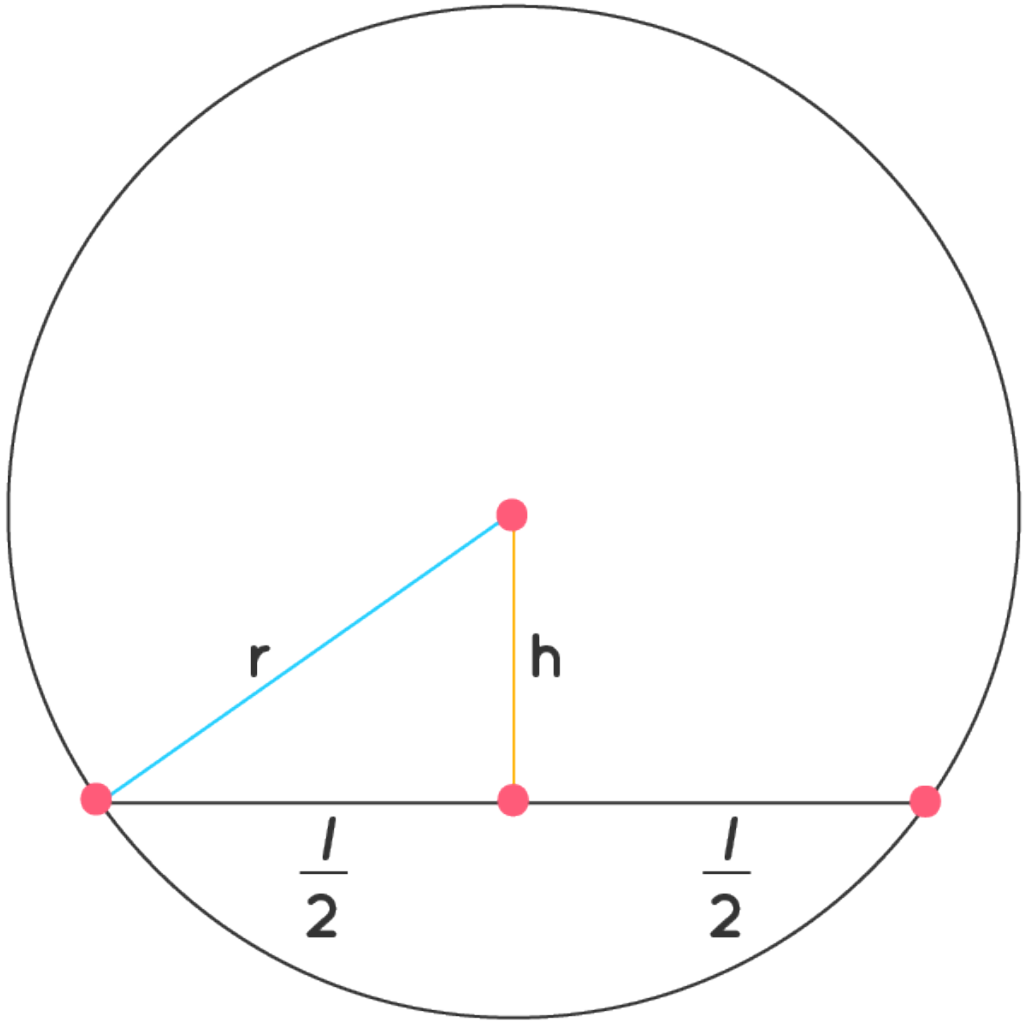
- Extract curvature from outline - Engineering Task



Example Invent and Simplify

Creating an "invariant" metric for back curvature

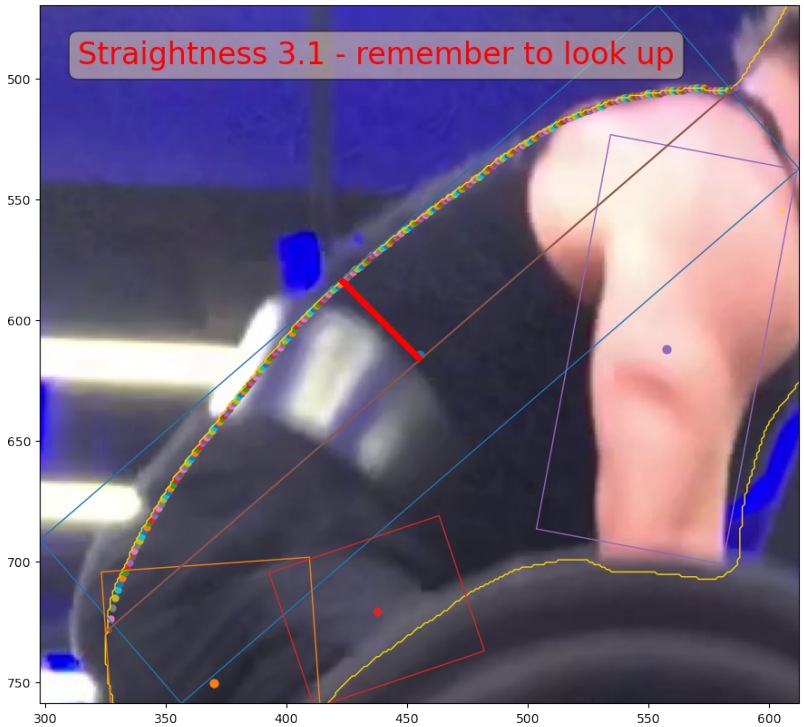
- What if we can map every outer back contour to a circle segment?

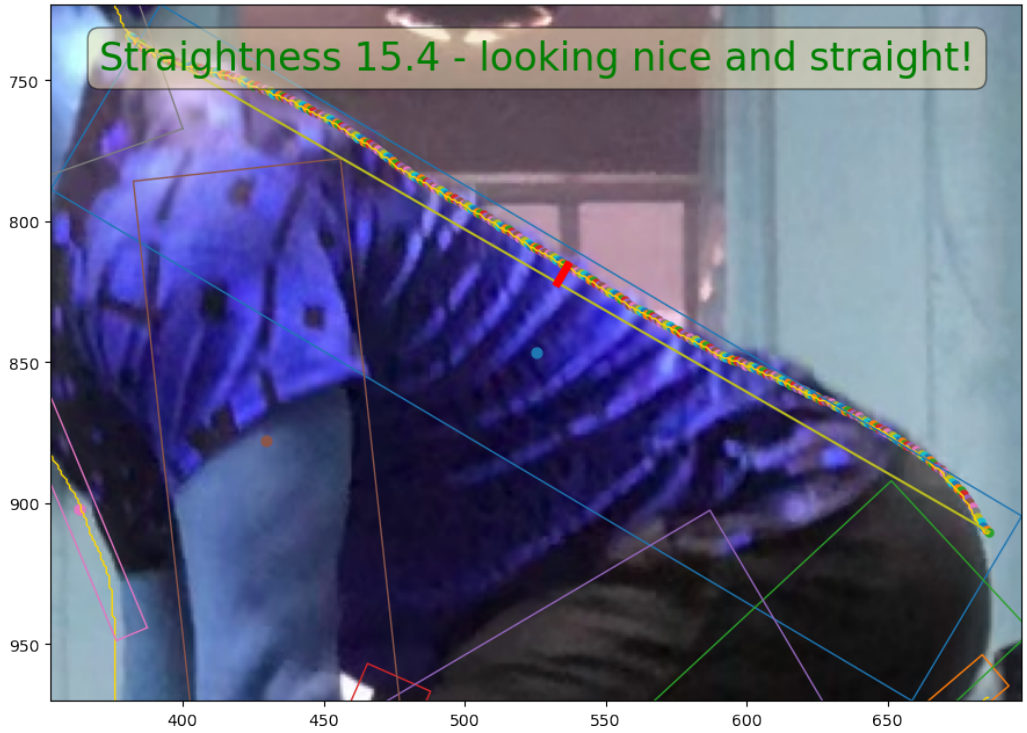


Example Invent and Simplify

Creating an "invariant" metric for back curvature

- Ask the Audience





[1] M. Baity-Jesi, L. Sagun, M. Geiger, S. Spigler, G. B. Arous, C. Cammarota, Y. LeCun, M. Wyart, and G. Biroli. Comparing dynamics: Deep neural networks versus glassy systems. In *International Conference on Machine Learning*, pages 314–323. PMLR, 2018.

[2] L. Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[3] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[4] A. Defazio and L. Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In *Advances in Neural Information Processing Systems*, pages 1753–1763, 2019.

[5] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[6] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*, 2017.

[7] S. Trivedi and R. Kondor. Cmsc 35246: Deep learning. lecture 6: Optimization for deep neural networks. 2017.

Gadam/GadamX

Require: weights θ_0; $\alpha_t = \alpha(t)$; $\{\beta_1, \beta_2\}$; $p \in [0, 0.5]$ Default to $\{0.125, 0.5\}$ for {GadamX, Gadam}; decoupled wd γ; T_{avg}; ϵ (10^{-8})

Ensure: Optimised weights $\tilde{\theta}$

Set $\boldsymbol{m}_0 = 0, \boldsymbol{v}_0 = 0, \hat{\boldsymbol{v}}_0 = 0, n_{\text{models}} = 0.$

for t = 1, ... T **do**

$\alpha_t = \alpha(t)$

$\boldsymbol{g}_t = \nabla f_t(\theta_t)$

$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t/(1 - \beta_1^t)$

$\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2/(1 - \beta_2^t)$

$\theta_t = (1 - \alpha_t \gamma)\theta_{t-1} - \alpha_t \frac{\hat{\boldsymbol{m}}_t}{(\hat{\boldsymbol{v}}_t + \epsilon)^p}$

if $T \geq T_{\text{avg}}$ **then**

$n_{\text{models}} = n_{\text{models}} + 1$

$\theta_{\text{avg}} = \frac{\theta_{\text{avg}} \cdot n_{\text{models}} + \theta_t}{n_{\text{models}} + 1}$

else

$\theta_{\text{avg}} = \theta_t$

end if

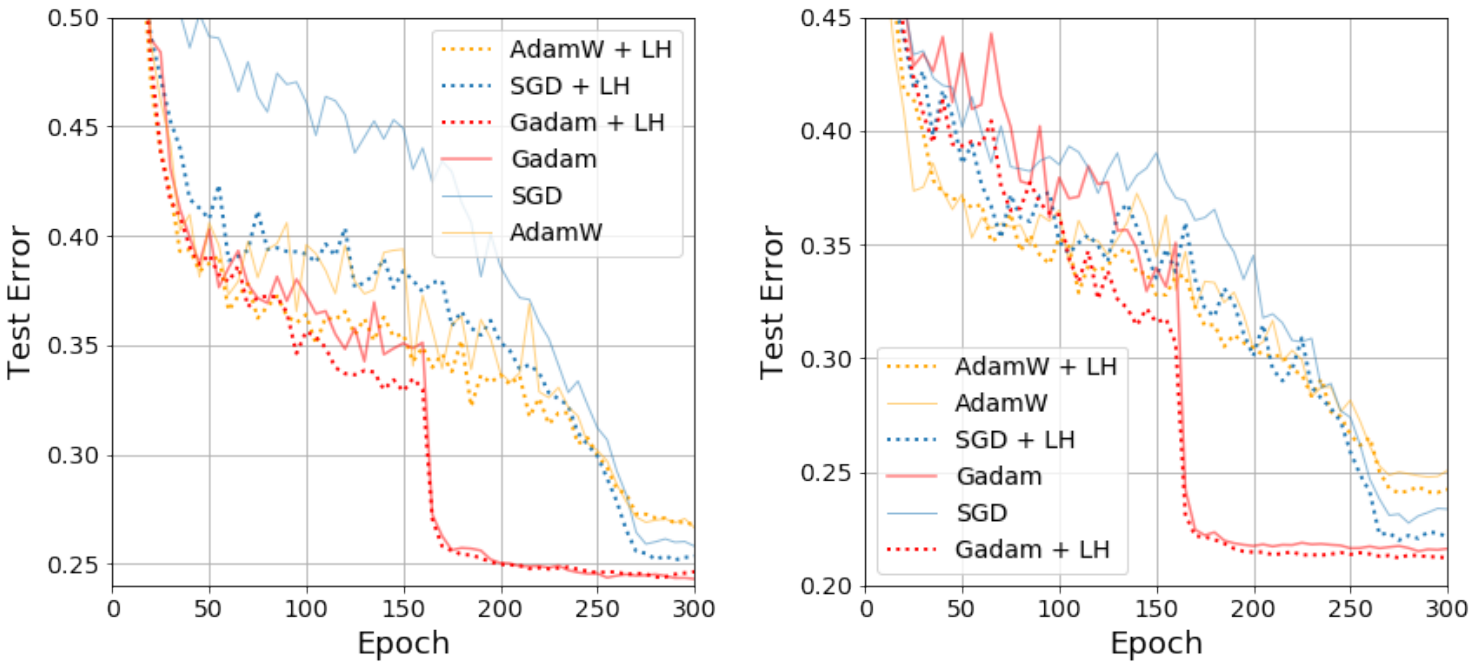
end for

Comparison to other Averaging Algorithms

Lookahead (EMA had a lot of press)

Performs badly with Adam on Image tasks

EMA performs a significantly reduced variance reduction



(a) VGG-16 (b) PRN-110

Figure: Test accuracy of Lookahead in CIFAR-100 against number of epochs.

Future of Gadam?

Could this be useful for Pangu-Alpha? Vision Transformers, or other architectures?

VAE, GANs?

Table 5: The detailed settings for training PanGu-α models.

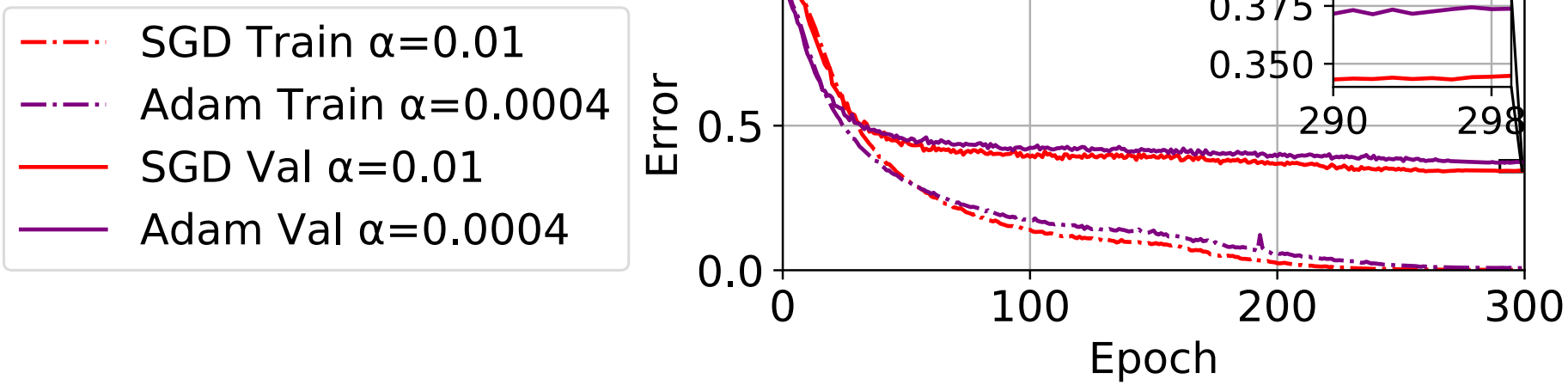
Models	#Training Steps	#Ascend processors	Adam Betas	Learning Rate	Weight Decay
PanGu-α 2.6B	0∼70,000	512	β_1=0.9 ,β_2=0.999	1e-4	0.01
PanGu-α 13B	0∼84,000	1024	β_1=0.9 ,β_2=0.98	5e-5	0.01
PanGu-α 200B	0∼130,000 130,000∼260,000	2048 1024	β_1=0.9 ,β_2=0.95	2e-5	0.1

(a) VGG-16

Figure: Many algorithms are only trained with Adam or AdamW. We have consistently shown superior performance with Gadam/GadamX and are curious to learn about more challenging problems.

Adaptive Generalisation Gap

Generally observed that Adam generalises worse than SGD, why?

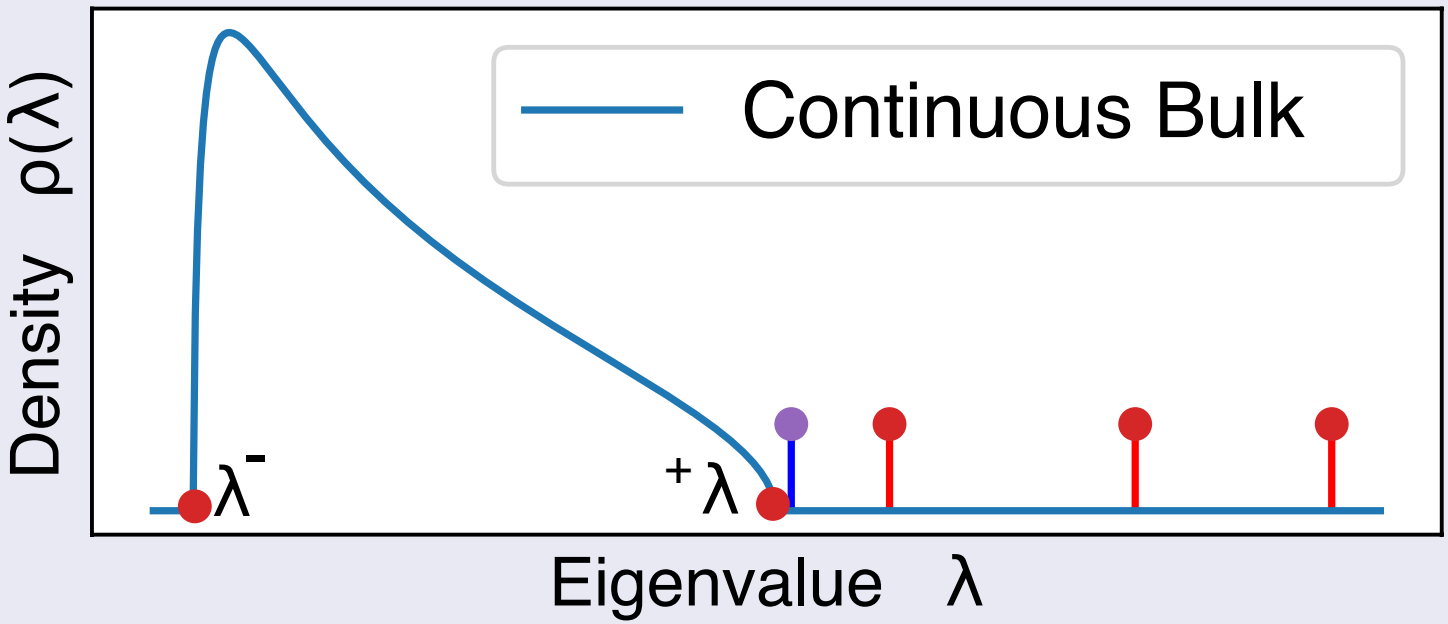


(a) SGD quickly outgeneralises Adam, even with no regularisation $\gamma = 0$

Figure: **Adaptive Generalisation Gap and its extent are clearly visible without regularisation.** Train/Val Error on CIFAR-100 using VGG-16 without batch normalisation and weight decay.

RMT approach to Minibatching

Bulk and Outliers



(a) Hypothetical $\rho(\lambda)$

Figure: (a) Hypothetical spectral density plot with a sharply supported continuous bulk region, a finite size fluctuation shown in blue corresponding to the Tracy-Widom region and three well-separated outliers shown in red.

Theorem (Overlap between True and Batch Hessian eigenvectors)

The eigenvector overlap between the eigenvectors $\phi_i \in \mathbb{R}^{P\times 1}$ of the Hessian of the mini-batch $\nabla^2 L_{\text{batch}}(\boldsymbol{w}_k)$ of batch size B and the eigenvectors $\boldsymbol{\theta}_i$ of the Hessian under the expectation of the true data distribution $\nabla^2 L_{\text{true}}(\boldsymbol{w}_k)$, where $L_{\text{true}}(\boldsymbol{w}_k) = \int \ell(\boldsymbol{w}_k; \boldsymbol{x}, \boldsymbol{y}) d\psi_{\boldsymbol{x},\boldsymbol{y}}$

$$|\boldsymbol{\theta}_i^T \boldsymbol{\phi}_i|^2 = \begin{cases} 1 - \frac{P\sigma^2}{B\lambda_i^2} & \text{if } |\lambda_i| > \sqrt{\frac{P}{B}}\sigma, \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

where λ_i are true Hessian eigenvalues and σ is the sampling noise per Hessian element.

RMT approach to Minibatching

Damping can be seen as Linear Shrinkage + Learning Rate Scaling

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \alpha \boldsymbol{B}^{-1} \nabla L_{\text{batch}}(\boldsymbol{w}_k) \tag{31}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \sum_{i=1}^{P} \frac{\alpha}{\lambda_i + \delta} \phi_i \phi_i^T \nabla L_{\text{batch}}(\boldsymbol{w}_k). \tag{32}$$

$$\frac{1}{\lambda_i + \delta} = \frac{1}{\kappa(\beta\lambda_i + (1-\beta))} \tag{33}$$

RMT approach to Minibatching

Ablation Experiment Lanczos MNIST

$$\boldsymbol{w}_k - \alpha\left(\frac{1}{\eta}\sum_{i=1}^{k}\frac{1}{\lambda_i+\delta}\phi_i\phi_i^T\nabla L(\boldsymbol{w}_k) + \sum_{k+1}^{P}\frac{1}{\delta}\phi_i\phi_i^T\nabla L(\boldsymbol{w}_k)\right) \tag{34}$$

Convex experiment (no bad "sharp" minima)

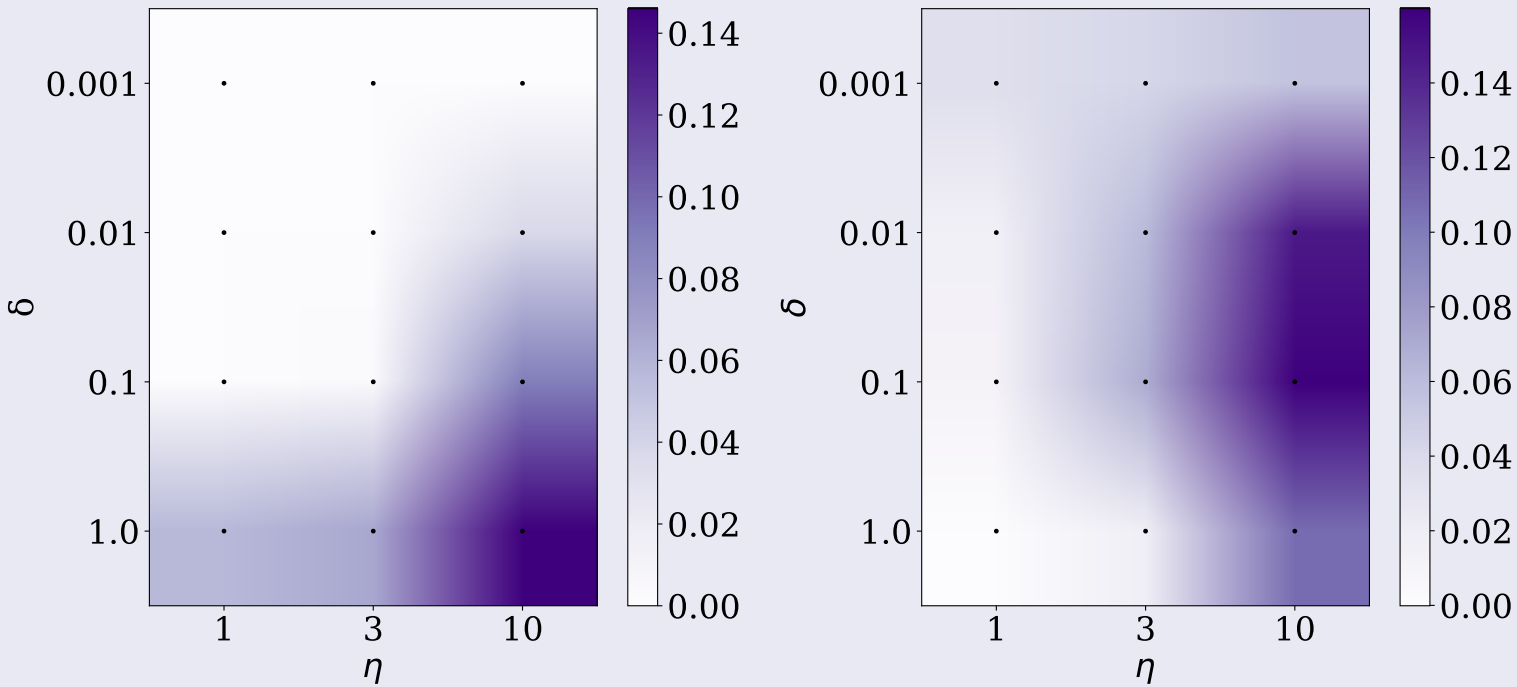
Small so can do Hessian vector products

can actively perturb "sharp" directions $\eta \in [1, 3, 10]$

directly test hypothesis, *do sharp directions matter for generalisation*

RMT approach to Minibatching

Key Concept: Fluctuations Matrix



(a) $\Delta(\delta, \eta)$ Training (b) $\Delta(\delta, \eta)$ Testing

Figure: Error change with damping/sharp direction perturbation δ, η in LanczosOPT, relative to the single best run. Darker regions indicate higher error.

RMT approach to Minibatching

Key Concept: Fluctuations Matrix

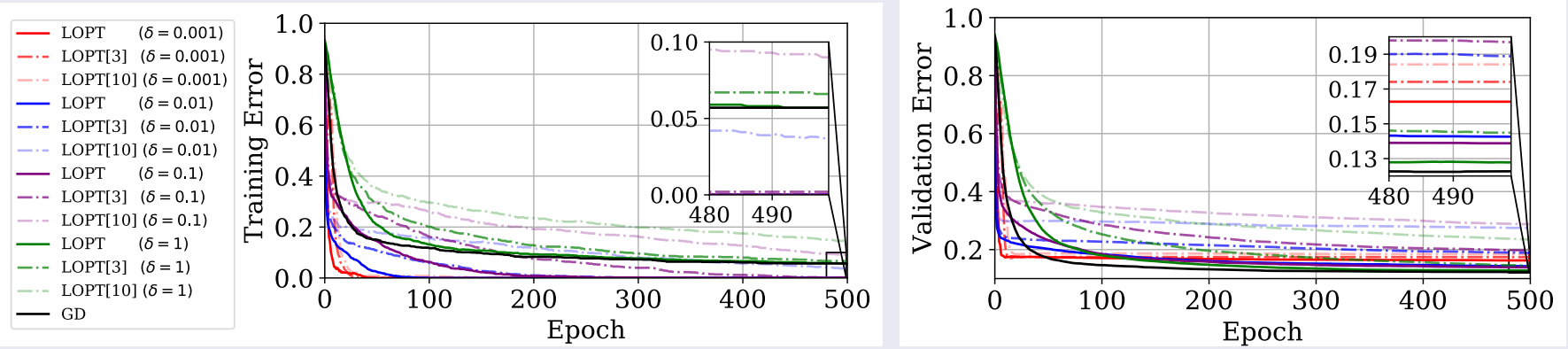
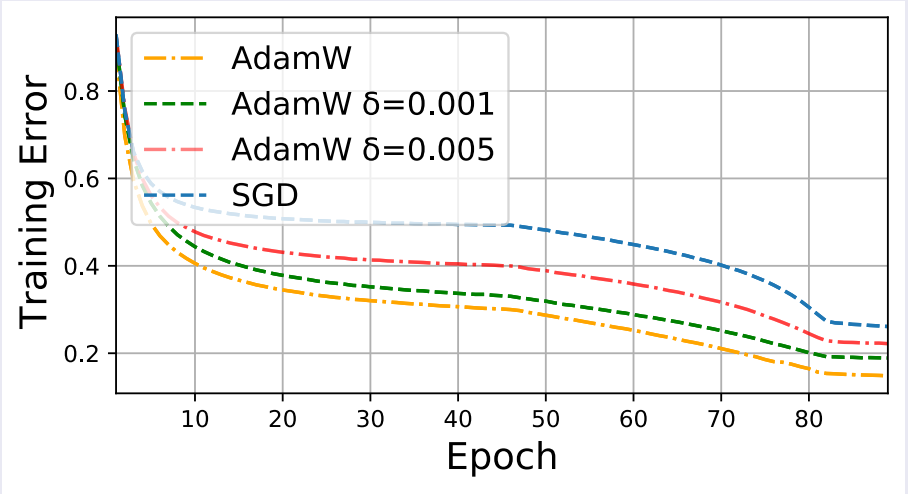


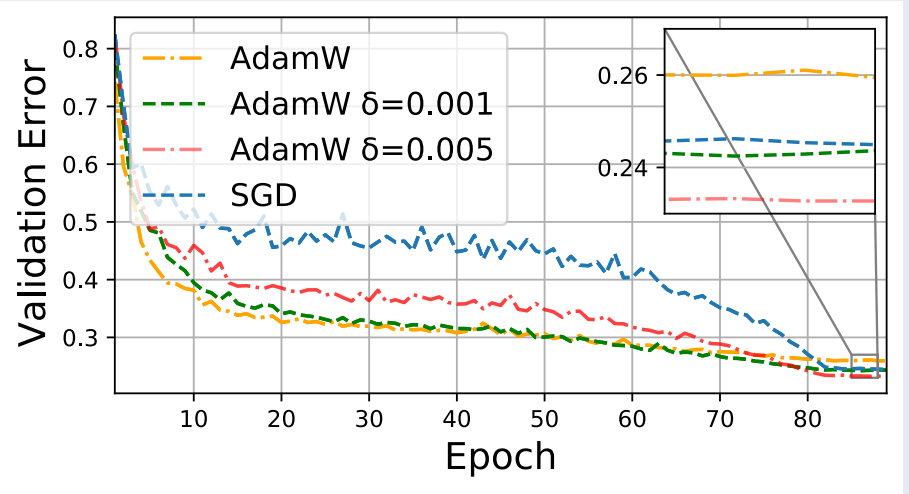
Figure: Training/test error of LanczosOPT/Gradient Descent (LOPT/GD) optimisers for logistic regression on the MNIST dataset with fixed learning rate $\alpha = 0.01$ across different damping values, δ. LOPT[η] denotes a modification to the LOPT algorithm that perturbs a subset of update directions by a factor of η.

RMT approach to Minibatching

Key Concept: Fluctuations Matrix



(a) ResNet-50 Training Error



(b) ResNet-50 Testing Error

Figure: (a-b) The influence of δ on the generalisation gap. Train/Val curves for ResNet-50 on ImageNet. The generalisation gap is completely closed with an appropriate choice of δ.

Results Teaser

Our Algorithms Gadam & GadamX

Data/Model	SGD	Adam-D	Adam
C100/VGG16	65.3 ± 0.6	65.5 ± 0.7	61.9 ± 0.4
ImgNet/Res50	75.7 ± 0.1	76.6 ± 0.1	-

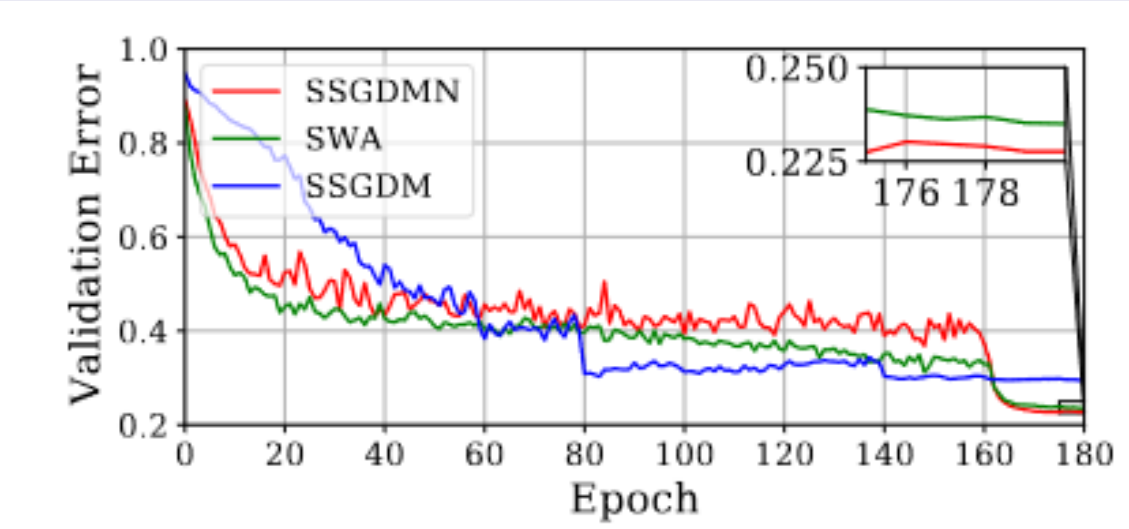
Table: Statistical Significance of Results

RMT approach to Minibatching

Open Questions, Further Practical Implications

Can we learn the learning rate on the fly? no hand-tuning..?

Stochastic Gradient Descent with Nesterov Momentum, Parameters estimated every 20 epochs on CIFAR-100 PreResNet-110



Does this hold for NLP (non-image) tasks? (Not investigated)

Can we relate the loss surface or elements of the loss surface to choices of the architecture, dataset, loss function?

RMT approach to Minibatching

Open Questions, Further Practical Implications

e.g see Early Convolutions Help Transformers See Better - switching from AdamW to SGD drops Vistion Transformer accuracy by 10%,

can this be related to the loss surface?

Can we define what a good architecture is from the surface?

Good initialisation?

ESRGAN, removes batch norm (known to reduce the Lipshitz constant) and changes the intialisation scheme..

recent work considering classification on the modified square loss $k * (f_C(x) - M)^2 + \sum_i f_i(x)^2$ has been shown to be competitive with cross entropy in terms of training speed and final score, why?