DEEP CURVATURE SUITE

Anonymous authors
Paper under double-blind review

ABSTRACT

The curvature of the loss, provides rich information on the geometry underlying neural networks, with applications in second order optimisation and Bayesian deep learning. However, accessing curvature information is still a daunting engineering challenge, inaccessible to most practitioners. We hence provide a software package the **Deep Curvature Suite**, which allows easy curvature evaluation for large modern neural networks. Beyond the calculation of a highly accurate moment matched approximation of the Hessian spectrum using Lanczos, our package provides: extensive *loss surface visualisation*, the calculation of the *Hessian variance* and *stochastic second order optimisers*. We further address and disprove many common misconceptions in the literature about the Lanczos algorithm, namely that it learns eigenvalues from the top down. We prove using high dimensional concentration inequalities that for specific matrices a single random vector is sufficient for accurate spectral estimation, informing our spectral visualisation method. We showcase our package practical utility on a series of examples based on realistic modern neural networks such as the VGG-16 and Preactivated ResNets on the CIFAR-10/100 datasets. We further detail 3 specific potential use cases enabled by our software: research in stochastic second order optimisation for deep learning, learning rate scheduling using known optimality formulae for convex surfaces and empirical verification of deep learning theory based on comparing empirical and theoretically implied spectra.

1 INTRODUCTION

The success of deep neural networks trained with gradient based optimisers in speech and object recognition (LeCun et al., 2015), has led to an explosion in easy to use high performance software implementations. Automatic differentiation packages such as TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2017) have become widely adopted. Higher level packages, such as Keras (Chollet, 2015) allow practitioners users to state their model, dataset and optimiser in a few lines of code, effortlessly achieving state of the art performance.

However, software for extracting second order information, representing the curvature of the loss at a point in weight space, has not kept abreast. Researchers aspiring to evaluate curvature information need to implement their own libraries, which are rarely shared or kept up to date. Naive implementations, which rely on full eigendecomposition (cubic cost in the parameter count) are computationally intractable for all but the smallest of models. Hence, researchers typically ignore curvature information or use highly optimistic approximations. Examples in the literature include the diagonal elements of the matrix or of a surrogate matrix,Chaudhari et al. (2016); Dangel et al. (2019), which we show in AppendixE can be very misleading.

2 MOTIVATION

The curvature of the loss informs us about the local conditioning of the problem (i.e the ratio of the largest to smallest eigenvalues $\frac{\lambda_1}{\lambda_P}$). This determines the rate of convergence for first order methods and informs us about the optimal learning and momentum rates (Nesterov, 2013). Hence easily accessible curvature information could allow practitioners to scale their learning rates in an optimal way throughout training, instead of relying on expert scheduling, we investigate this using our software in Section 5.2. Research areas where curvature information features most prominently are analyses of the **Loss Surface** and **Newton** type optimization methods.

2.1 LOSS SURFACES

Recent neural network loss surfaces using full eigendecomposition (Sagun et al., 2016; 2017) have been limited to toy examples with less than five thousand parameters. Hence, loss surface visualisation of deep neural networks have often focused on two dimensional slices of random vectors (Li et al., 2017) or the changes in the loss traversing a set of random vectors drawn from the d-dimensional Gaussian distribution (Izmailov et al., 2018). It is not clear that the loss surfaces of modern expressive neural networks, containing millions or billions of dimensions, can be well captured in this manner. Small experiments have shown neural networks have a large rank degeneracy Sagun et al. (2016) with a small number of large outliers. However high dimensional concentration theorems Vershynin (2018) guarantee that even a large number of randomly sampled vectors are unlikely to encounter such outliers and hence have limited ability to discern the geometry between various solutions. Other works, that try to distinguish between flat and sharp minima have used the diagonal of the Fisher information matrix (Chaudhari et al., 2016), an assumption we will challenge in this paper. Specifically we show in Appendix E, that diagonal approximations do not capture key properties in synthetic and real neural network examples.

From a practical perspective, specific properties of the loss surface are not captured by the aforementioned approaches. Examples include the *flatness* as specified by the trace, Frobenius and spectral norm. These measures have been extensively used to characterise the generalisation of a solution found by SGD (Wu et al., 2018; Izmailov et al., 2018; He et al., 2019; Jastrzkebski et al., 2017; 2018; Keskar et al., 2016). Under a Bayesian and minimum description length argument (Hochreiter and Schmidhuber, 1997) flatter minima should generalise better than sharper minima. The magnitude of these outliers have been linked to poor generalisation performance Keskar et al. (2016) and as a consequences the generalisation benefits of large learning rate SGD Wu et al. (2018); Jastrzkebski et al. (2017). These properties are extremely easy in principle to estimate, at a computational cost of a small multiple of gradient evaluations. However the calculation of these properties are not typically included in standard deep learning frameworks, which limits the ability of researchers to undertake such analysis.

Other important areas of loss surface investigation include understanding the effectiveness of batch normalisation(Ioffe and Szegedy, 2015). Recent convergence proofs (Santurkar et al., 2018) bound the maximal eigenvalue of the Hessian with respect to the activations and bounds with respect to the weights on a per layer basis. Bounds on a per layer basis do not imply anything about the bounds of the entire Hessian and furthermore it has been argued that the full spectrum must be calculated to give insights on the alteration of the landscape (Kohler et al., 2018).

2.2 SECOND ORDER OPTIMISATION METHODS

Second order optimisation methods solve the minimisation problem for the loss, $L(\boldsymbol{w}) \in \mathbb{R}$ associated with parameters $\boldsymbol{w} \in \mathbb{R}^{P\times 1}$ and perturbation $\boldsymbol{\delta w} \in \mathbb{R}^{P\times 1}$ to the second order in Taylor expansion,

$$\boldsymbol{\delta w}^* = \text{argmin}_{\boldsymbol{\delta w}} L(\boldsymbol{w} + \boldsymbol{\delta w}) = \text{argmin}_{\boldsymbol{\delta w}} L(\boldsymbol{w} + \boldsymbol{\delta w}) = -\bar{\boldsymbol{H}}^{-1} \nabla L(\boldsymbol{w}). \tag{1}$$

Sometimes, such as in deep neural networks when the Hessian $\boldsymbol{H} = \nabla^2 L(\boldsymbol{w}) \in \mathbb{R}^{P\times P}$ is not positive definite, a positive definite surrogate is used. Note that Equation 1 is not lower bounded unless $\boldsymbol{H}$ is positive semi-definite. Often either a multiple of the identity is added (known as damping) to the Hessian ($\boldsymbol{H} \to \boldsymbol{H} + \gamma \boldsymbol{I}$) Dauphin et al. (2014) or a surrogate positive definite approximation to the Hessian $\bar{\boldsymbol{H}}$, such as the Generalised Gauss-Newton (GGN) (Martens, 2010; Martens and Sutskever, 2012) is employed. To derive the GGN, we express the loss in terms of the activation $L(\boldsymbol{w}) = \sigma(f(\boldsymbol{w}))$ of the output of the final layer $f(\boldsymbol{w})$. Hence the elements of the Hessian can be written as

$$\boldsymbol{H}(\boldsymbol{w})_{ij} = \sum_{k=0}^{d_y} \sum_{l=0}^{d_y} \frac{\partial^2 \sigma(f(\boldsymbol{w}))}{\partial f_l(\boldsymbol{w}) \partial f_k(\boldsymbol{w})} \frac{\partial f_l(\boldsymbol{w})}{\partial w_j} \frac{\partial f_k(\boldsymbol{w})}{\partial w_i} + \sum_{k=0}^{d_y} \frac{\partial \sigma(f(\boldsymbol{w}))}{\partial w_j} \frac{\partial^2 f_k(\boldsymbol{w})}{\partial w_j \partial w_i}. \tag{2}$$

The first term on the LHS of Equation 2 is known as the *Generalised Gauss-Newton* matrix.

Despite the success of second order optimisation methods using the GGN for difficult problems on which SGD is known to stall, such as recurrent neural networks (Martens and Sutskever, 2012), or

auto-encoders (Martens, 2016). Researchers wanting to implement second order methods such as (Vinyals and Povey, 2012; Martens and Sutskever, 2012; Dauphin et al., 2014) face the aforementioned problems of difficult implementation. As a small side note, Bayesian neural networks using the Laplace approximation feature the Hessian inverse multiplied by a vector (Bishop, 2006).

2.3 Contributions

We introduce our package, the **Deep Curvature suite**, a software package that allows analysis and visualisation of deep neural network curvature. The main features and functionalities of our package are:

- **Eigenspectrum analysis of the curvature matrices** Powered by the Lanczos Meurant and Strakoš (2006) techniques implemented in GPyTorch (Gardner et al., 2018) and outlined in Section 3, *with a single random vector* we use the Pearlmutter matrix-vector product trick Pearlmutter (1994) for fast inference of the eigenvalues and eigenvectors of the common curvature matrices of the deep neural networks. In addition to the standard Hessian matrix, we also include the feature for inference of the eigen-information of the Generalised Gauss-Newton matrix, a commonly used positive-definite surrogate to Hessian[1].
- **Advanced Statistics of Networks** In addition to the commonly used statistics to evaluate network training and performance such as the training and testing losses and accuracy, we support computations of more advanced statistics: For example, we support squared mean and variance of gradients and Hessians (and GGN), squared norms of Hessian and GGN, L2 and L-inf norms of the network weights and etc. These statistics are useful and relevant for a wide range of purposes such as the designs of second-order optimisers and network architecture.
- **Visualisations** For all main features above, we include accompanying visualisation tools. In addition, with the eigen-information obtained, we also feature visualisations of the **loss landscape** by studying the sensitivity of the neural network to perturbations of weights.
- **Second Order Optimisation** We include out of the box stochastic lanczos second order optimisers which use the absolute Hessian Dauphin et al. (2014) or the Generalised Gauss Newton Martens (2010) as curvature matrices
- **Batch Normalisation Support** Our code allows for the use of batch normalisation(Ioffe and Szegedy, 2015) in neural networks, which is integrated by default on most modern convolutional neural networks. Since the way in watch batch normalisation is calculated has a big effect on the resulting curvature calculations and can be dependent on the order in which the samples are processed, we offer two models of evaluation. *Train mode* which depends on the order in which the samples are processed and is similar to what the optimiser sees during training and *evaluation mode*, in which the statistics are pre-computed and hence invariant to sample shuffling. We expand on the differences between these methods in Appendix A.

Our software makes calculating and visualising curvature information as simple as calculating the gradient at a saved checkpoint in weight-space. The computational complexity of our approach is $\mathcal{O}(mP)$, where m is the number of Lanczos steps [2] and P is the number of model parameters. This in stark contrast full exact eigendecomposition which has a numerical cost of $\mathcal{O}(P^3)$, which is infeasible for large neural networks.

2.4 Related Work

Recent work making curvature information more available using diagonal approximations, disallows the use of batch normalisation(Dangel et al., 2019). Our software extends seamlessly to support

[1] The computation of the GGN-vector product is similar with the computational cost of two backward passes in the network. Also, GGN uses *forward-mode automatic differentiation* (FMAD) in addition to the commonly employed *backward-mode automatic differentiation* (RMAD). In the current PyTorch framework, the FMAD operation can be achieved using two equivalent RMAD operations.

[2] which corresponds to the number of moments of the underlying Hessian/GGN density which we wish to match

Functionality	Deep Curvature Suite	PyHessian
Hessian Density Estimation	✓	✓
GGN Density Estimation	✓	✗
Loss Surface Visualisation	✓	✓
Gradient Variance	✓	✗
Hessian/GGN Variance	✓	✗
Second Order Optimisers	✓	✗

Table 1: Table of Contributions: comparison between our released software and that of another PyTorch based Hessian tool PyHessian

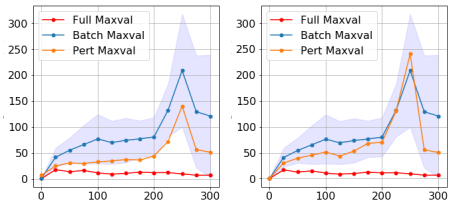


Figure 1: Deviation of the mini-batch Hessian (left)/GGN (right) spectral norm from the full dataset, Figure from Granziol (2020).

batch normalization In Appendix E.5 we explicitly compare against their diagonal MC approximations on synthetic and real neural network examples and find these approximations inadequate for spectral analysis. Independent work has also uses the Lanczos algorithm for Hessian computation Yao et al. (2019); Ghorbani et al. (2019); Papyan (2018); Izmailov et al. (2019). We compare the functionality of our package to that of PyHessian Yao et al. (2018) in Table 1.

One major extra functionality we offer is the calculation of the GGN. For common activatios and loss functions, such as the cross-entropy loss and sigmoid activation, the generalised Gauss-Newton is equivalent to the Fisher information matrix (Pascanu and Bengio, 2013). Hence this approximation to the Hessian is interesting in its own right and many theoretical analyses, consider the generalised Gauss-Newton or Fisher information matrices as opposed to the Hessian Pennington and Worah (2018); Karakida et al. (2019); Papyan (2019). Hence accessing the GGN spectrum could be a major asset to researchers. Another extra functionality we offer is the calculation of the Hessian/GGN variance. This quantity has been used to derive optimal learning rates in Wu et al. (2018) and to predict the effect of minibatching on curvature using a random matrix theory model (Granziol, 2020). We include an example figure where the theoretical prediction is validated using our software in Figure 1.

3 Lanczos, Misconceptions and Spectral Density Approximations

The Lanczos Algorithm Meurant and Strakoš (2006), on which our package is based, (Algorithm 1 in the Appendix) is an iterative algorithm for learning a subset of the eigenvalues/eigenvectors of any Hermitian matrix, requiring only matrix vector products. It is an adaptation of the power iteration method, where the Krylov subspace $\mathscr{K}(\boldsymbol{H}, \boldsymbol{v}) = \text{span}\{\boldsymbol{v}, \boldsymbol{H}\boldsymbol{v}, \boldsymbol{H}^2\boldsymbol{v}, ...\}$, is orthogonalised using Gram-Schmidt, resulting in improved convergence. The Lanczos algorithm, is subject to many misconceptions in the literature, which limits its penetration in the research community. Two of which we state here and explicitly debunk in Appendix B.2

- We can learn the negative and interior eigenvalues by shifting and inverting the matrix sign $\boldsymbol{H} \rightarrow -\boldsymbol{H} + \mu \boldsymbol{I}$
- Lanczos learns the largest m $[\lambda_i, \boldsymbol{u}_i]$ pairs of $\boldsymbol{H} \in \mathbb{R}^{P \times P}$ with high probability (Dauphin et al., 2014)

The key properties of the Lanczos algorithm, which we summarise here (and expand upon in Appendix B) are:

- When seeded with a vector $\boldsymbol{v}$ the Lanczos algorithm gives a discrete spectral density which matches the m moments $\boldsymbol{v}\boldsymbol{H}\boldsymbol{v}, \boldsymbol{v}\boldsymbol{H}^2\boldsymbol{v}, ..., \boldsymbol{v}\boldsymbol{H}^m\boldsymbol{v}$.
- If the vector $\boldsymbol{v}$ is zero mean unit variance (such as a Gaussian or Rademacher Hutchinson (1990)), this in expectation gives an m moment matched approximation to the spectral density $p(\lambda)$ of the underlying matrix $\boldsymbol{H}$.
- For a large dimensional matrix $P \rightarrow$, we expect a single random vector to give an m moment matched approximation of the underlying spectral density. We give the proof for this in Appendix C.This informs our visualisation technique of using a single random vector.
- When seeded with a random vector $\boldsymbol{v}$, Lanczos converges quickly to the end points of the spectral support of $p(\lambda)$, the spectral density of $\boldsymbol{H}$.

4 An Illustrated Example

We give an illustration on an example of using the Deep Curvature package. We train a VGG16 network on CIFAR-100 for 100 epochs. With the checkpoints generated, we may now compute analyse the eigenspectrum of the curvature matrix evaluated at the desired point in training.To evaluate the Hessian/GGN with 100 Lanczos iterations, we run the code given in Table 2.

Network Training

```
train_network(
dir='VGG16/',
dataset='CIFAR100',
data_path='data/',
epochs=100, model='VGG16',
optimizer='SGD',
optimizer_kwargs=
{'lr': 0.03, 'mom': 0.9,
'weight_decay': 5e-4})
```

Eigenspectrum Computation

```
compute_eigenspectrum(
dataset='CIFAR100',
data_path='data/',
model='VGG16',
checkpoint_path='checkpoint.pt',
save_spectrum_path='spectrum-ggn',
save_eigvec=True,
lanczos_iters=100,
curvature_matrix='ggn_lanczos',)
```

Table 2: Training a Neural Network and Calculating the Eigenspectrum using the Deep Curvature Suite package

This function call saves the spectrum results (including eigenvalues, eigenvectors and other related statistics) in the `save_spectrum_path` path string defined. To visualise the spectrum as stem plot we simply run the code given in Table 3 with corresponding example Figure shown.

Eigenspectrum Visualization

```
plot_spectrum('lanczos',
path='spectrum-ggn.npz')
plt.show()
```

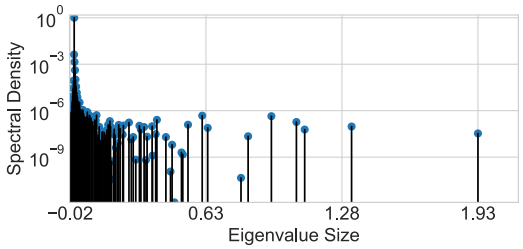


Table 3: Eigenspectrum plotting code and corresponding stem plot

Finally, with the eigenvalues and eigenvectors computed, we might be interested in knowing how sensitive the network is to perturbation along these directions. To achieve this, we first construct a loss landscape by setting the number of query points and maximum perturbation to apply. To achieve that, we call the code given in Table 4. In this example, we set the maximum perturbation to be 1

Loss Surface Visualization

```
build_loss_landscape(
dataset='CIFAR100',
data_path='data/',
model='VGG16',
dist=1., n_points=21,
spectrum_path=hessian',
checkpoint_path='ck.pt',
save_path='scape.npz'
plot_loss_landscape
('landscape-100.npz')
plt.show()))
```

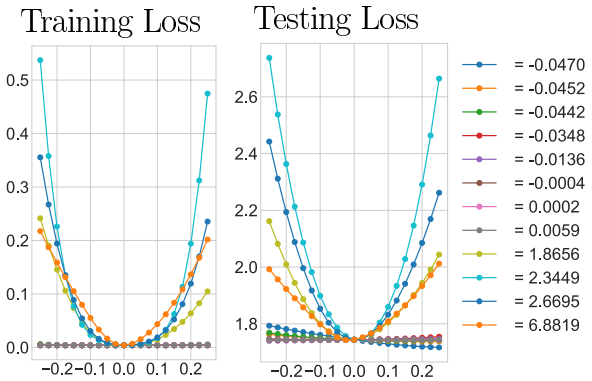


Table 4: Loss surface visualization along the sharpest Hessian eigenvectors with the Deep Curvature Suite

(`dist` argument) and number of query points along each direction to be 21 (`n_points` argument). The corresponding plots in Training and Testing loss are also shown. Although we choose to show loss, the package returns the results for accuracy also, which we show in Appendix D.

5 EXAMPLE RESEARCH APPLICATIONS

In this section we list 3 example scientific applications of our software package. We investigate second order optimisation for deep neural networks using our inbuild stochastic Lanczos optimiser, online learning rate scheduling and verifying predictions made using random matrix theory in theoretical deep learning.

5.1 SIMPLICITY OF SECOND ORDER OPTIMISATION WITH THE DEEP CURVATURE SUITE

To enable researchers to experiment with stochastic second order optimisation algorithms for deep neural networks, we implement a Lanczos based optimiser (which takes the absolute Hessian Dauphin et al. (2014), or Generalised Gauss Newton Martens (2010) as input). The code for running this optimiser is summarised in Table 5. We plot the training error of the VGG-16 network on

SGD Training

```
train_network(
dir='VGG16-CIFAR100/',
dataset='CIFAR100',
data_path='data/',epochs=100,
model='VGG16', optimizer='SGD',
optimizer_kwargs={
'lr': 0.01,
'momentum': 0.9,
'weight_decay': 0
'batch_size': 128})
```

Lanczos 2nd Order Training

```
train_network(
dir='VGG16-CIFAR100/',
dataset='CIFAR100',
data_path='data/',epochs=100,
model='VGG16', optimizer='LancGN',
optimizer_kwargs={
'lr': 1,
'damping': 10,
'weight_decay': 0
'batch_size': 128
'curvature_batch_size': 128})
```

Table 5: Comparison of SGD training and Second Order Optimisation using the Deep Curvature Suite

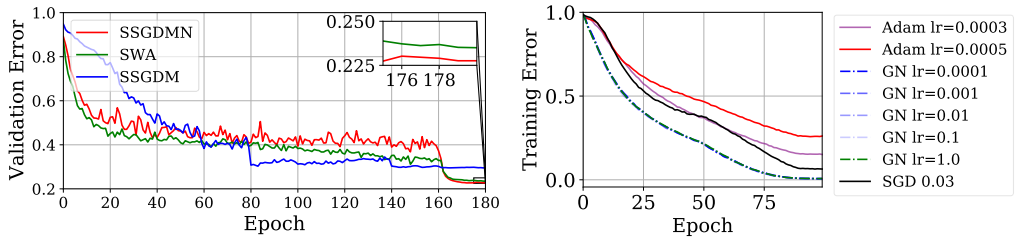
CIFAR-100 dataset, which has over 16 million parameters and hence is out of reach of full inversion methods, in Figure 2b. We keep the ratio of damping constant to learning rate constant, where $\delta = 10\alpha$, for a variety of learning rates in $\{1, 0.1, 0.01, 0.001, 0.0001\}$ with a batch size of 128 for both the gradient and the curvature, all of which post almost identical performance. We also compare against different learning rates of Adam and the best grid searched learning rate of SGD, both of which converge significantly slower per iteration compared to our stochastic Newton methods. We note that in terms of practical optimisation that the cost of running a stochastic second order optimiser is far greater than that of SGD. For this example, where we use 20 Lanczos iterations, the per iteration cost scales as ≈ 40 times that of SGD. Hence we expect these algorithms simply to serve as useful baselines against more computationally efficient approximations. One example future direction could be to invert the matrices less frequently as done in Martens and Grosse (2015), reducing computational cost.

5.2 SPECTRAL STOCHASTIC GRADIENT DESCENT WITH THE DEEP CURVATURE SUITE

SGD is incredibly sensitive to the choice of initial learning rate and its scheduling. Whilst adaptive methods are considered more robust to the initial learning rate choice and its schedule, they have been shown to perform poorly on validation or held out test data Wilson et al. (2017). Hence practitioners typically experiment with a variety of schedules, such as the step, linear, exponential, polynomial and cosine annealing. One potential application of our codebase is to use a stochastic estimate of the curvature and to use established optimal learning and momentum rates for the local quadratic approximation at the given point in weightspace Nesterov (2013), given in Equation 3.

$$\alpha_{Polyak} = \frac{2}{\sqrt{\lambda_1} + \sqrt{\lambda_P}}, \rho = \left(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_P}}{\sqrt{\lambda_1} + \sqrt{\lambda_P}}\right)^2, \alpha_{Nesterov} = \sqrt{\frac{\lambda_P}{\lambda_1}}\rho = \frac{\sqrt{\lambda_1} - \sqrt{\lambda_P}}{\sqrt{\lambda_1} + \sqrt{\lambda_P}}. \quad (3)$$

Where λ_1, λ_P refer to the largest and smallest eigenvalues of the Hessian respectively. The learning rates and momenta α, ρ are given for both Polyak and Nesterov momentum respectively. There are several complications to the above method. Firstly the Hessian of neural networks is in general not positive definite and hence $\lambda_P < 0$ (these optimality formulae are strictly for convex methods). Secondly the loss surface at each point in weight space is likely to be different and it is impractical



(a) $\boldsymbol{H}(\boldsymbol{w})$ VGG16BN epoch 300

(b) $\boldsymbol{H}(\boldsymbol{w})$ VGG16BN epoch 300

to recalculate the optimal step size using Equation 3 for every iteration, even if we sub-sample the data. We note from the work of Granziol (2020), that the loss surface when using a mini-batch varies to that of the full surface (it is sharper) and hence the learning rates when training using a mini-batch must be optimal for the minibatch surface and not that of the full dataset. As an example use case for our software we consider whether the method of Izmailov et al. (2018), which combines SGD with Polyak averaging Polyak and Juditsky (1992) at the end of training (which they denote *SWA*), to improve generalisation performance, could be improved. Instead of fine tuning the linear decay learning rate schedule by hand, we instead learn the learning rate using Equation 3 at regular intervals by estimating the curvature using a sub-sampled Lanczos spectral estimate.

We run the preactivated ResNet-110 on the CIFAR-100 dataset for 180 epochs and learn the Polyak and Nesterov learning rates and momenta every 20 epochs using the same mini-batch size as the gradient batch size 128 for 20 Lanczos iterations. Since the smallest Ritz values are very very close to 0, which would result in a momentum $\rho = 1$, we use a heuristic to remove the smallest Ritz values, whereby if the Ritz value of largest spectral mass has more than 50% of the spectral mass, it is removed and the resulting density renormalised, forming the new spectral density of interest. All methods employ Polyak Avearging at epoch 161 as in Izmailov et al. (2018). We plot the results in Figure 2a and show that the Nesterov variant of online learning rate and momentum learning performs well against the baseline (SWA). Interestingly the Polyak variant learns to drop the learning rate at regular intervals, similar to classical training methods, which does not perform as well as keeping the learning rate high and using Polyak averaging at the end of training.

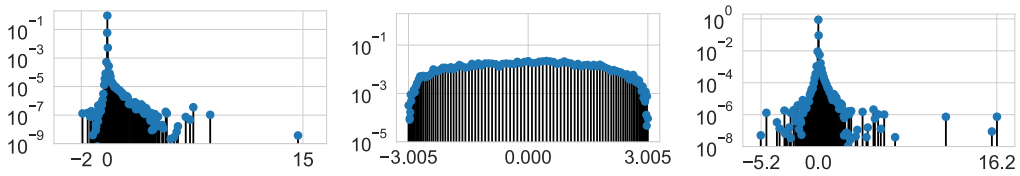
5.3 Verifying Random Matrix Theory in Deep Learning using our Package

The full spectrum of deep neural networks can be used to validate/invalidate novel theoretical contributions. As an example. analysis relating the loss surface to spin-glass models from condensed matter physics and random matrix theory (Choromanska et al., 2015b;a) rely on a number of unrealistic assumptions, such as input independence (i.e. for an input $\boldsymbol{x}$ that the feature x_i is independent of x_j, for images which have spatial correlations, such assumptions are clearly violated). One potential verification of the practical applicability of these results, that does not involve checking that the assumptions are satisfied, would be to visualise the spectra of large real networks and commonly used datasets. This follows because spin glass models have Hessians given by random matrices (Arous and Guionnet, 1997) with known analytical forms (Tao, 2012; Akemann et al., 2011). Hence statistical tests comparing the real world observed spectra and that underlying the specific spin glass model could be undertaken, showing that the results hold despite the assumptions not being met. Two primary random matrix eigenvalue distributions to which the spectra of many classes of random matrices converge (and which feature in theoretical deep learning papers) are the semi-circle and the Marchenko-Pastur densities defined below.

$$\overbrace{\underbrace{p(\lambda) = \frac{1}{2\pi}\sqrt{4\sigma^2 - \lambda^2}\mathbf{1}_{|\lambda|\leq 2\sigma},}_{\text{Semi-Circle Law}}}^{\text{elements } \boldsymbol{H}_{i,j} \text{ are i.i.d normal}} \quad \overbrace{\underbrace{p(\lambda) = \frac{\sqrt{(\lambda-\lambda_+)(\lambda-\lambda_-)}}{2\pi\lambda q\sigma^2}}_{\text{Marchenko-Pastur}}}^{\text{elements } \boldsymbol{X}_{i,j} \text{ are i.i.d normal } \boldsymbol{H} = \boldsymbol{X}^T\boldsymbol{X}}, \lambda_\pm = \sigma^2(1 \pm \sqrt{q})^2 \tag{4}$$

If there are N samples and P parameters and $N < P$ the Marchenko-Pastur density has a degeneracy fraction $1 - \frac{N}{P}$.

In the example of Choromanska et al. (2015a), the corresponding Hessian spectral denisty is the semi-circle law, shown in Figure 3b. This differs greatly from neural network spectra found us-

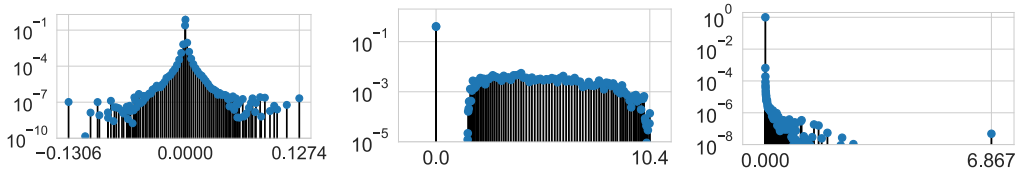


(a) $\boldsymbol{H}(\boldsymbol{w})$ VGG16BN epoch 300 (b) Wigner semi-circle (c) $\boldsymbol{H}(\boldsymbol{w})$ PResNet-110 epoch 300

Figure 3: Neural network Hessian spectra at the end of training for the VGG-16BN and PreResNet-110 on the CIFAR-100 dataset and an example of the semi-circle law

ing our package, such as that of the VGG-16 with BN shown in Figure 3a and the Preactived ResNet-110, for the CIFAR-100 datasets, shown in Figure 3c. Major differences include the rank-degeneracies (we find that over 90% of the spectral mass resides in the peak nearest the origin) and outliers. The proof techniques used in Arous and Guionnet (1997), which underlie the claims in Choromanska et al. (2015a) specifically use the large deviation principle of the Gaussian Orthogonal Ensemble (which converges to the semi-circle law) and hence forbid outliers.

The GGN has been modelled as the Marchenko-Pastur density in the literature (Pennington and Bahri, 2017; Sagun et al., 2017; 2016). The Residual matrix $\mathcal{R}(\boldsymbol{w})$, which is the difference of the Hessian and the GGN, has been modelled by the Wigner ensemble which gives the semi-circle law in Pennington and Bahri (2017). In Figure 4 we plot the GGN and Residual of a VGG-16 network on the CIFAR-100 dataset. The GGN unlike the Marchenko-Pastur has no spectral gap (the density tails off exponentially) and many outliers. Similarly the shape and rank-degeneracy of the Residual matrix varies significantly from the semi-circle and contains outliers. Whilst a single set of negative results does not invalidate an entire framework, it implies that further research is required to extend the claims of Choromanska et al. (2015a); Pennington and Bahri (2017) to practical deep learning.



(a) $\mathcal{R}(\boldsymbol{w})$ VGG-16 Epoch 300 (b) Marchenko-Pastur (c) $\boldsymbol{G}(\boldsymbol{w})$ VGG-16 Epoch 300

Figure 4: The Generalised Gauss Newton and Residual Matrix at various Epochs of Training for the VGG-16 on CIFAR-100 and a sample Marcenko-Pastur density

These results using our package demonstrate that the underlying spectral densities of the matrices in question vary considerably from those given by the theoretical predictions. This implies that the conclusions made may not necassarily hold in real neural networks and requires further investigation.

6 Conclusion

We introduce the **Deep Curvature** suite in **PyTorch** framework, based on the Lanczos algorithm implemented in GPyTorch (Gardner et al., 2018), that allows deep learning practitioners to learn spectral density information as well as eigenvalue/eigenvector pairs of the curvature matrices at specific points in weight space. Our package also allows the user to evaluate the Generalised Gauss Newton spectral density, the variance of both the Hessian and the Generalised Gauss Newton and includes two stochastic Lanczos based optimisers. Together with the software, we also include a succinct summary of the linear algebra, iterative method theory including proofs of convergence and misconceptions and stochastic trace estimation that form the theoretical underpinnings of our work. We also give 3 example scientific applications of our work, namely second order optimisation on large scale neural networks, learning rate scheduling using spectral information and evaluation of theoretical results which make predictions on spectral denisities of the Hessian of the neural networks which they are modelling.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

Gernot Akemann, Jinho Baik, and Philippe Di Francesco. *The Oxford handbook of random matrix theory*. Oxford University Press, 2011.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2007.

Anonymous. Towards understanding the true loss surface of deep neural networks using random matrix theory and iterative spectral methods. In *Submitted to International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gza2NtwH. under review.

G Ben Arous and Alice Guionnet. Large deviations for wigner's law and voiculescu's non-commutative entropy. *Probability theory and related fields*, 108(4):517–542, 1997.

Zhaojun Bai, Gark Fahey, and Gene Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1-2):71–89, 1996.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016.

François Chollet. Keras. https://github.com/fchollet/keras, 2015.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015a.

Anna Choromanska, Yann LeCun, and Gérard Ben Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In *Conference on Learning Theory*, pages 1756–1760, 2015b.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. *arXiv preprint arXiv:1912.10985*, 2019.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via Hessian eigenvalue density. *arXiv preprint arXiv:1901.10159*, 2019.

Gene H Golub and Gérard Meurant. Matrices, moments and quadrature. *Pitman Research Notes in Mathematics Series*, pages 105–105, 1994.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2012.

Diego Granziol. Curvature is key: Sub-sampled loss surfaces and the implications for large batch training. *arXiv preprint arXiv:2006.09092*, 2020.

Diego Granziol, Binxin Ru, Stefan Zohren, Xiaowen Dong, Michael Osborne, and Stephen Roberts. Meme: An accurate maximum entropy method for efficient approximations in large-scale machine learning. *Entropy*, 21(6):551, 2019.

Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *arXiv preprint arXiv:1902.00744*, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *Uncertainty in Artificial Intelligence (UAI)*, 2018.

Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for bayesian deep learning. *Uncertainty in Artificial Intelligence (UAI)*, 2019.

Stanislaw Jastrzkebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in SGD. *arXiv preprint arXiv:1711.04623*, 2017.

Stanislaw Jastrzkebski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of DNN loss and the SGD step length. 2018.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1032–1041. PMLR, 2019.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Ming Zhou, Klaus Neymeyr, and Thomas Hofmann. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. *arXiv preprint arXiv:1805.10694*, 2018.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM Review*, 58(1):34–65, 2016.

Vladimir Alexandrovich Marchenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 114(4):507–536, 1967.

James Martens. Deep learning via Hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

James Martens. *Second-order optimization for neural networks*. PhD thesis, University of Toronto, 2016. URL http://www.cs.toronto.edu/~jmartens/docs/thesis_phd_martens.pdf.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

James Martens and Ilya Sutskever. Training deep and recurrent networks with Hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012.

Gérard Meurant and Zdeněk Strakoš. The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, 15:471–542, 2006.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Vardan Papyan. The full spectrum of deep net Hessians at scale: Dynamics with sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *arXiv preprint arXiv:1901.08244*, 2019.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

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. 2017.

Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1):147–160, 1994.

Jeffrey Pennington and Yasaman Bahri. Geometry of neural network loss surfaces via random matrix theory. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2798–2806. JMLR. org, 2017.

Jeffrey Pennington and Pratik Worah. The spectrum of the fisher information matrix of a single-hidden-layer neural network. In *Advances in Neural Information Processing Systems*, pages 5410–5419, 2018.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Levent Sagun, Léon Bottou, and Yann LeCun. Eigenvalues of the Hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the Hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

Shashanka Ubaru and Yousef Saad. Applications of trace estimation techniques.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.

Oriol Vinyals and Daniel Povey. Krylov subspace descent for deep learning. In *Artificial Intelligence and Statistics*, pages 1261–1268, 2012.

Eugene P Wigner. Characteristic vectors of bordered matrices with infinite dimensions i. In *The Collected Works of Eugene Paul Wigner*, pages 524–540. Springer, 1993.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

Lei Wu, Chao Ma, and E Weinan. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems*, pages 8279–8288, 2018.

Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, pages 4949–4959, 2018.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael Mahoney. Pyhessian: Neural networks through the lens of the hessian. *arXiv preprint arXiv:1912.07145*, 2019.