A Deeper Look at the Hessian Eigenspectrum of Deep Neural Networks and its Applications to Regularization

Adepu Ravi Sankar\textsuperscript{1} *, Yash Khasbage\textsuperscript{1} *, Rahul Vigneswaran\textsuperscript{1}, Vineeth N Balasubramanian\textsuperscript{1}

\textsuperscript{1} Dept of Computer Science & Engineering, Indian Institute of Technology Hyderabad, India.

cs14resch11001@iith.ac.in, cs17btech11044@iith.ac.in, rahulvigneswaran@gmail.com, vineethnb@cse.iith.ac.in

Abstract

Loss landscape analysis is extremely useful for a deeper understanding of the generalization ability of deep neural network models. In this work, we propose a layerwise loss landscape analysis where the loss surface at every layer is studied independently and also on how each correlates to the overall loss surface. We study the layerwise loss landscape by studying the eigenspectra of the Hessian at each layer. In particular, our results show that the layerwise Hessian geometry is largely similar to the entire Hessian. We also report an interesting phenomenon where the Hessian eigenspectrum of middle layers of the deep neural network are observed to most similar to the overall Hessian eigenspectrum. We also show that the maximum eigenvalue and the trace of the Hessian (both full network and layerwise) reduce as training of the network progresses. We leverage on these observations to propose a new regularizer based on the trace of the layerwise Hessian. Penalizing the trace of the Hessian at the layer immediately forces Stochastic Gradient Descent to converge to flatter minima, which are shown to have better generalization performance. In particular, we show that such a layerwise regularizer can be leveraged to penalize the middlemost layers alone, which yields promising results. Our empirical studies on well-known deep nets across datasets support the claims of this work.

Introduction

Deep neural networks (DNNs) have been immensely successful in challenging real-world problems in image, video, text, and speech domains. Despite their tremendous success, several questions remain as to why DNNs generalize well despite the very high dimensions of the data involved, non-convexity of the optimization, as well as overparametrization of the models. This has led to explicit efforts over the last few years on trying to understand loss surfaces of DNN models.\textsuperscript{2} Anna, LeCun, and Arous\textsuperscript{2} (2015) specifically pointed out the understanding of loss surfaces as a key open problem in deep learning.

\textsuperscript{2}From a theoretical standpoint, efforts such as Baldi and Hornik\textsuperscript{2} (1988), Anna, LeCun, and Arous\textsuperscript{2} (2015) and Kawaguchi\textsuperscript{2} (2016) have studied the loss landscape of deep linear/non-linear networks under certain assumptions, and characterized its properties. For example, Baldi and Hornik\textsuperscript{2} (1988), Anna, LeCun, and Arous\textsuperscript{2} (2015) and Kawaguchi\textsuperscript{2} (2016) have shown from different perspectives that every local minimum can be a global minimum for DNNs under certain conditions (which do not hold in practice though). Hardt and Ma\textsuperscript{2} (2017) specifically focused on networks with residual connections and showed that arbitrary deep linear residual networks have zero spurious local minima, while Auer, Warmuth, and K\textsuperscript{2} (1996) showed that there are an exponentially high number of equivalent local minima in high dimensions as in DNN models.

From an empirical and analytical perspective, the last few years have seen reasonable efforts in studying the 'flatness' of the minima that DNNs converge to. In a seminal work, Hochreiter\textsuperscript{2} (1997) studied the relation between generalization ability and loss landscape geometry many years ago, and hypothesized that flat minima provide better solutions. More recently, Keskar et al.\textsuperscript{2} (2017) empirically verified that small batch training leads to flat local minima, and hence better generalization. This also led to methods such as Entropy-SGD in Chaudhari et al.\textsuperscript{2} (2017) which aim to bias SGD into flatter minima. On a different note, Dinh et al.\textsuperscript{2} (2017) showed that it is possible for sharp minima to generalize well too, but this is work entirely theoretical with no empirical evidence yet. A popular understanding at this time, however, is that - largely driven by empirical studies - flat minima exhibit better generalization than sharp minima. The connection between the curvature of the minima (flatness or sharpness) and the quality of the obtained solution (trained DNN model) has resulted in efforts that have attempted to study the loss landscape via the eigenspectrum of the Hessian matrix of the loss function. Considering that the explicit computation of Hessian matrix is computationally infeasible, several approximations have been used to this end. In this work, we propose the analysis of the layerwise loss surface of DNN models using their Hessian eigenspectra, as well as the evolution of the eigenspectra over training. To the best of our knowledge, there has been no explicit effort on studying loss surfaces layerwise before. The other notable effort that studied layers recently is Zhang, Bengio, and Singer\textsuperscript{2} (2019), which however had a different objective and provided evidence for the heterogeneity of layers. Studying the layerwise Hessian is also more computationally feasible today than earlier, and was perhaps not-at
Importance of Layerwise Loss Landscape Analysis: Understanding the geometry of loss surfaces of DNN models, and its implications towards understanding generalization properties of DNNs and their relation towards understanding loss landscapes, substantiate the peculiarity of DNN models converging to flat minima; (iii) we present interesting observations of the connection between the middlemost layers and the full network in this context; and (iv) we propose a new regularization method, based on our study and layerwise analysis, that helps improve generalization performance on well-known models and datasets.

Layerwise Loss Landscape Analysis using the Hessian Eigenspectrum

DNN models largely follow a layerwise composition of functions and recent methods to improve performance (e.g., batch normalization) are also introduced at a layer level. It is hence natural to ask how the loss landscape at every layer behaves, especially when compared to the overall loss landscape. We seek to address this question in this section.

The geometry of loss landscapes of DNN models is characterized by the Hessian matrix of the loss function, and our work in this section focuses on analyzing the Hessian eigenspectrum of each layer, in particular by analyzing their individual Gauss-Newton decompositions. We introduce the reader to the Gauss-Newton decomposition of the Hessian matrix, \( \text{Hess}_l = \text{H} + \text{G} \), defined in (Sagun, Bottou, and LeCun 2016) Papyan [2018] where:

\[
\text{H} = \text{Ave}_{i,c} \left\{ \sum_{c'=1}^C \left( \frac{\partial \mathcal{L}(z, y_c)}{\partial z_{c'}} \right) \left( \frac{\partial^2 f_c(x_{i,c}; \theta)}{\partial \theta^2} \right)^T \right\} 
\]

and \( z_{i,c} = f(x_{i,c}; \theta) \), Ave\( i,c \) denotes the average across all observations \( i \) and classes \( c \). It is straightforward to note that the Gauss-Newton decomposition of the layerwise Hessian, \( \text{Hess}_l \), can be given by \( G_l + H_l \) for \( l = 1, \ldots, L \). This follows from the fact that the layerwise Hessian corresponds to training a DNN. The work in (Martens and Grosse 2015) analyzed the Fisher information matrix (as an approximation to Hessian matrix) layerwise and found a computationally efficient way to calculate its inverse by analysing it layerwise. Their work, however, did not study loss surfaces. To the best of our knowledge, this is the first effort that analyzes the loss surfaces of DNN models layerwise.

Preliminaries/Notations

We consider a deep neural network (DNN) with \( L \) layers and model parameters, \( \theta = \bigcup_{l=1}^L \{ \theta_l \} \), where \( \theta_l \) denotes the parameters in a layer \( l \). The training data with a total of \( n \) training examples is provided to the DNN, with \( C \) classes and \( n_c \) \( (c \in \{1, \ldots, C\}) \) samples in each class. A data point \( x_{i,c} \) from the dataset \( \bigcup_{l=1}^L \{ (x_{i,c}, y_i) \}_{i=1}^{n_c} \) is the \( i^{th} \) sample belonging to class \( c \), i.e. \( y_i = c \). Let \( z \) denote the pre-softmax probabilities of a neural network, and \( f(x_{i,c}; \theta) \) denote the pre-activation scores of the model output. The loss of the DNN is given by \( \mathcal{L}(f(x_{i,c}; \theta), y_i) \). We often use \( \text{Hess} \) to denote the Hessian matrix \( \mathcal{L}'' \in \mathbb{R}^{D \times D} \), where \( D \) is the cardinality of \( \theta \). We refer to the layerwise Hessian as \( \text{Hess}_l = \frac{\partial^2 \mathcal{L}}{\partial \theta_l \partial \theta_l} \), where \( i, j \in \{1, \ldots, |\theta_l|\} \) and \( |\theta_l| \) is the number of parameters in layer \( l \). SGD stands for Stochastic Gradient Descent with a suitable minibatch size, and \( Tr(A) \) denotes the trace of matrix \( A \).
blocks around the diagonal in the overall Hessian, and the corresponding Hessian is simply restricted to $\theta_l$, the parameters of that layer. Hence, the layerwise Hessian decomposition into $G_l$ and $H_l$ is given as:

$$G_l = \text{Ave}_{i,c} \left\{ \frac{\partial^2 f(x_i,\theta)}{\partial \theta_l^2} \right\} \quad \text{and} \quad H_l = \text{Ave}_{i,c} \left\{ \sum_{c'=1}^{C} \frac{\partial L(z_i,y_{c'})}{\partial z_i} \left|_{z_i,c} \right. \right\} \left( \frac{\partial^2 f(x_i,\theta)}{\partial \theta_l^2} \right) \left|_{z_i,c} \right. \right\}$$

We study the eigenspectrum of the layerwise Hessian, $Hess_l$, through the eigenspectra of each $G_l$ and $H_l$. Among the tools available to efficiently compute the eigenspectrum, we use the Lanczos method (Lanczos [1950] owing to its better performance in time incurred over competing methods such as KPM (Lin, Saad, and Yang [2016]) in earlier efforts such as (Papyan [2018]). The Lanczos method computes the eigenspectrum of a symmetric matrix by reducing it to a tridiagonal form, $T\in\mathbb{R}^{D\times D}$, and computing the spectrum of $T\in\mathbb{R}^{D\times D}$. The original Lanczos method however requires an inner iterative loop (for $D$ iterations) to re-orthogonalize the obtained vectors in each iteration due to numerical errors. For matrices with very large $D$ such as in DNNs, this orthogonalization step is computationally intensive and necessitates the simultaneous use of multiple high-end GPUs (such as in Ghorbani, Shankar, and Xiao [2019]) for implementation, making it impractical.

To overcome this issue, Lin, Saad, and Yang [2016] proposed an approximation where the Lanczos method (Algorithm details in supplementary section) is used only for $M << D$ iterations, thus obtaining $M$ eigenvalue-eigenvector pairs, and subsequently estimating the eigenspectrum density using a Gaussian convolution on the $M$ outputs of the approximate Lanczos. In particular, this method is based on writing the eigenspectrum of any large matrix as $\phi(t) = \frac{1}{\sqrt{\pi}} \sum_{i=1}^{D} \lambda_i^t$ where $\lambda_i$ is the $i^{th}$ eigenvalue, $D$ is the total number of eigenvalues, and $\phi(t)$ is the frequency of eigenvalue $t$. Instead of computing the entire spectrum, this approximation computes a Gaussian density convolution $\phi_{\sigma}(t) = \frac{1}{\sqrt{\pi}} \sum_{i=1}^{D} y_i[1]^2 g_\sigma(t - \lambda_i)$, where $g_\sigma(t - \lambda_i)$ is a Gaussian centered at $\lambda_i$ with width $\sigma$. For more details of this method, the interested reader is requested to refer to Lin, Saad, and Yang [2016], Papyan [2018].

Earlier work that analyzed the entire loss landscape of DNN models using the Hessian eigenspectrum have made interesting observations. In particular, Sagun et al. [2018] as well as Papyan [2018] observed that the Hessian eigenspectrum is divided into a bulk region, and an outlier region. More interestingly, both Sagun et al. [2018] and Papyan [2018] also showed that the number of outliers in the Hessian eigenspectrum is approximately the number of classes, $C$. With this background in context, we study the layerwise loss surface using the Hessian eigenspectrum, as computed using Lanczos method. We conducted studies on many state-of-the-art DNN models including VGG11,13,16 (with and without Batch Normalization), ResNet18 and DenseNet on MNIST, FashionMNIST, and CIFAR10 datasets. Due to space constraints, we report our results only with VGG11-BN on CIFAR-10 in Figure 9 and the remaining results can be found in the Supplementary material. We inferred similar observations for all of these models however, and all our results can be reproduced using our anonymized source code repository shared here.6

Figures 9 show the eigenspectrum of $\{Hess_l\}_{l=1}^L$ and $\{G_l\}_{l=1}^L$ of VGG11 (with Batch Normalization), ResNet18 when trained on the CIFAR10 dataset. The eigenspectrum of $\{G_l\}_{l=1}^L$ is obtained by the decomposition of $\{Hess_l\}_{l=1}^L$ defined in Equations 5 and 6. We do not report the spectrum of $H_l$, since we found the eigenvalues to be mostly $\approx 0$. This was pointed out by Sagun et al. [2018] in their work too, who observed that the spectra of $H \approx 0$ and that of $Hess \approx G$. This is reflected in our results; for e.g., in Figures 9 one can see that the spectra of $\{Hess_l\}_{l=1}^L$ and $\{G_l\}_{l=1}^L$ almost overlap across all layers. Based on our results across models and datasets including Figure 9 we report our inferences below.

Papyan [2018] (cf. Fig 6) reported that the spectrum of $G$ peaks at a particular epoch during training, where the number of outliers in the eigenspectra of $G$ and $Hess$ is $C$, the number of classes. Interestingly, we too find that the number of outliers in the layerwise eigenspectra of $\{G_l\}_{l=1}^L$ and $\{Hess_l\}_{l=1}^L$ are also $C$, across almost all layers. One can notice this on careful observation in Figure 9 (notice the number of peaks in the spectra in each subplot).

More recently, Papyan [2019] showed that $G$ can be written as $G = \frac{1}{\Delta} \Delta^T = \text{Ave}_{i,c} \left\{ \sum_{c'=1}^{C} \delta_{i,c,c'}^T \right\}$, where

https://anonymous.4open.science/r/6eaea11b-879c-43a8-992b-ddb3e95b23c8/
the layerwise understanding on the loss surfaces, especially
match the overall loss surface. These new observations on
scape of the middlemost layers of DNN models consistently
to discriminate between the classes. Further, the loss land-
tances (Villani 2008): between the spectra of
tire network, and computed the normalized Wasserstein dis-
considered the eigenspectrum density of layers, and the en-
layers in these well-known, often-used DNN models whose
FAR10). The last subplot “full” corresponds to
t-SNE embeddings of
\[ \delta \{ \] of different
Figure 3: Plots of normalized Wasserstein distance between
the spectra of \( \text{Hess} \); \( \forall l = 1 \cdots L \) and \( \text{Hess} \) of different
DNN models across datasets.

the influence of middle layers on the loss surface, can poten-
tially be used in regularization methods, use of batch-
normalization, or in other training methods.

Evolution of Hessian Eigenspectra

Continuing the discussion from the previous section, we
now shift our focus towards analyzing the eigenspectra over
the course of DNN training. There have been a few efforts
in the recent past by other researchers (Pratik and Stefano
2018) Ghorbani, Shankar, and Xiao 2019) seeking to un-
derstand gradient descent trajectories. (Pratik and Stefano
2018) showed the relationship between SGD and variational
inference while studying these trajectories; In a related at-
tempt, (Ghorbani, Shankar, and Xiao 2019) studied GD tra-
jectories by studying the Hessian eigenspectra, but their fo-
cus was largely on the tool developed by them to analyze
the Hessian than present any newer observations. The work
closest to ours in this direction is that of (Jastrzebski et al.
2019) where the authors study the trajectories of SGD and
the connection to generalization performance. In particular,
they show the behavior of SGD along sharp directions of the
loss surface, and conclude that a variant of SGD can find
sharp minima with good generalization. We are fundamen-
tally different from this work from multiple perspectives:
(i) Their work follows earlier efforts in (Dinh et al. 2017),
which show that sharp minima can be generalizable too,
while our work follows most earlier efforts such as (Keskar
et al. 2017) which show the importance of flatter minima in
generalization; (ii) Their work relies on experiments on a
simple 4-layered CNN on CIFAR-10, while we use state-
of-the-art DNN models; (iii) We also introduce a tool to ef-
ciently compute the trace of the Hessian and analyze the
spectra, which has not been used before. While the above-
mentioned literature in this field have different perspectives
and our work was conceived independent of these efforts,
the presence of these efforts only support the need for such
analysis in the community.

We begin by studying the evolution of the eigenspectra
for the entire Hessian, and subsequently study the layer-
wise Hessians. To this end, we use the maximum eigenvalue,
\( \lambda_{max} \), and the trace of the Hessian, considering it is not triv-
ial to study the entire spectra over all epochs of training.

Figure 2: Layerwise two-dimensional t-SNE embeddings
of \( \delta \) obtained as a decomposition of \( \{ \text{G}\}^L_{l=1} \) in (Papyan
2018) show C clusters in each subplot (VGG11+BN on CI-
FAR10). The last subplot “full” corresponds to \( \delta \) of the en-
tire \( G \) matrix.

To understand further, we studied if there are particular
layers in these well-known, often-used DNN models whose
loss landscape matches the overall loss surface the most. We
considered the eigenspectrum density of layers, and the en-
tire network, and computed the normalized Wasserstein dis-
tance (Villani 2008): between the spectra of \( \{ \text{Hess}\}^L_{l=1} \), and
\( \text{Hess} \) for different DNN models. These results are shown in
Figure 3. It can be clearly observed that the spectra of
\( \text{Hess} \) of middle layers of DNN models are very similar
to spectrum of the overall \( \text{Hess} \). Such an observation has
not been made hitherto, to the best of our knowledge. We
also conducted this study using other measures such as KL-
divergence (results in Supplementary section) which had a similar observation about the middle layers being closest to the overall loss surface.

Summary of Observations: Based on our results, we
propose that the behavior of outliers in the layerwise
Hessian eigenspectra, especially the grouping into C clus-
ters, indicates that every layer of state-of-the-art DNN mod-
els encapsulates discriminative capability, i.e., the capability
to discriminate between the classes. Further, the loss land-
scape of the middlemost layers of DNN models consistently
match the overall loss surface. These new observations on
the layerwise understanding on the loss surfaces, especially
\[ \delta_{l,c,c'} \] is the \( c'-th \) column of a submatrix of \( \Delta \) (please see
Sec 2, Eqns 15-16 of (Papyan2019) for details). It was fur-
ther shown that the t-SNE plots of \( \delta \) (obtained by averag-
ing \( \delta_{l,c,c'} \) over all observations \( i \) and all potential classes \( c' \))
yields \( C \) clusters again. We conducted these studies layer-
wise, and found that the same observation holds for the lay-
erwise \( \{ \text{G}\}^L_{l=1} \) too, as shown in Figure 2 where we plot the
t-SNE embeddings of \( \delta \) obtained by decomposing \( \{ \text{G}\}^L_{l=1} \) matrix for all layers in a DNN.
We note that both these quantities, $\lambda_{\text{max}}$ and trace of Hessian denoted by $Tr(\mathcal{L}'')$, provide understanding of the curvature of the loss surface (higher these values, steeper the curvature). Clearly, the reduction in both these quantities indicates flatter regions of the loss surface. While $\lambda_{\text{max}}$ can be obtained using the methods discussed in Sec. 4, computing the trace of the Hessian of very large matrices (as in DNNs) is computationally non-trivial. Earlier work that attempted such a direction (Jastrzebski et al. 2019) did not take into account the complete spectrum for this reason. We hence introduce the use of tools from randomized numerical linear algebra, in particular the Hutchinson method, to compute the trace of the Hessian without explicit computation of the Hessian (Avron and Toledo 2011). The trace is obtained as:

$$Tr(\mathcal{L}'') = Tr(\mathcal{L}'I) = Tr(\mathcal{L}'' E[vv^T]) = E[Tr(\mathcal{L}'' vv^T)] = E[v^T \mathcal{L}' v]$$

where $v \sim \mathcal{N}(0, I)$. The equality $E[vv^T] = I$ comes from the expectation of quadratic form when $v \sim \mathcal{N}(0, I)$. We can also draw $v$ from the Rademacher distribution where each entry is either $+1$ or $-1$ with probability 0.5. The complete methodology to compute $Tr(\mathcal{L}'')$ using the Hutchinson method is summarized in Algorithm 1.

![Image](image.png)

**Figure 4:** Evolution of $\lambda_{\text{max}}$, trace of Hessian

**Algorithm 1** Hutchinson method to compute Hessian trace

| Input: Model parameters: $\theta$, Loss function $\mathcal{L}$; No of iters: n |
| Output: Trace of $\frac{\partial^2 \mathcal{L}}{\partial \theta^2}$ |
| 1. trace = 0 |
| 2. for $t = 1 \ldots n$ do |
| 3. $v \sim \mathcal{N}(0, I)$; $g_\theta = \frac{\partial \mathcal{L}}{\partial \theta}$; trace = trace + $v^T \frac{\partial(g_\theta^Tv)}{\partial \theta}$; |
| 4. end for |
| 5. return trace/n |

Summary of Observations: Our study of the evolution of Hessian eigenspectra over training presents a few important observations: (i) Our results broadly support earlier work, such as (Keskar et al. 2017), that state-of-the-art DNN models converge to flatter minima (with lower curvatures on the loss surface); (ii) there is an evident connection between the trace of the Hessian and generalization error, as shown by our studies with and without batch normalization and skip connections; and (iii) the same trend holds for the layerwise Hessians too, and importantly, the middlemost layers have $\lambda_{\text{max}}$ closest to that of the entire network across the epochs. This was a consistent observation across our studies, and points to a deeper connection which is yet to be theoretically understood. We hope that this work will raise this pertinent question in the community.

Layerwise Hessian Trace Regularization

The proposed layerwise regularization method for DNNs, which we call layerwise Hessian Trace Regularization (or LHTR), is motivated by our empirical observations that state-of-the-art models reduce the trace of layerwise Hessian over training. Importantly, we show that this layerwise regularization approach lends itself to an interesting premise - that regularizing only on the trace of Hessian of the middlemost layers by itself provides strong performance. This observation is in alignment with findings from our earlier sections. To the best of our knowledge, such a layer-specific regularization method has not been studied before.

We modify the DNN training objective as:

$$\theta^* = \arg \min_{\theta} \mathcal{L}(f(x; \theta)), y + \gamma \sum_{l=1}^{L} \text{Tr(Hess}_l)$$ (5)
Figure 5: Evolution of trace of Hessian of state-of-the-art models with and without batch normalization/skip connections on MNIST and CIFAR10. Training accuracy was close to state-of-the-art for all considered models at the end of these iterations.

Figure 6: Evolution of $\lambda_{\text{max}}$ of layerwise Hessian on VGG11+MNIST

where $\text{Tr}(\text{Hess}_l)$ is the trace of the Hessian of the loss function at layer $l$ and $\gamma$ is a regularization hyperparameter. The proposed layerwise regularizer works by penalizing the sum of the trace of layerwise Hessians. We choose a uniform weighting of layers in this work. It is possible to weight Hessians of different layers differently, which is an interesting future direction of work. This penalization at every layer helps encourage SGD to converge to solutions with flatter minima, and hence generalize better (Keskar et al. 2017).

We studied the usefulness of this regularizer on several well-known DNN models across multiple datasets (MNIST, FMNIST, SVHN, CIFAR10, CIFAR100). Cross-entropy loss, $\mathcal{L}_{ce}$, was used to train each of these classification models. We used a momentum of 0.9, learning rate of 1e-2, L2 regularization co-efficient of 1e-3, batch size of 64. We trained our model on a NVidia GeForce GTX 1080 GPU with 12GB GPU memory. We ran 5 trials of each experiment to avoid bias in randomness of initialization. Our anonymized source code (Python) is available for reproducibility.

Table 1 reports the generalization error when proposed HTR is incorporated into the cross-entropy loss objective. It can be clearly noticed that generalization error is better when HTR is incorporated into the objective of the DNN training across the considered datasets and models. All our experiments were conducted in a fair manner by tuning regularization hyperparameters in each of these methods and reporting the best result for each of the considered methods. This improvement in generalization performance happens with almost no change in training accuracy, thus showing promise in reducing the generalization gap. Fig 7 shows the training accuracy with and without HTR. This figure also shows the sum of traces of layerwise Hessians, which is indeed lower at the end of training with the proposed HTR method, suggesting solutions corresponding to relatively flatter minima.

Building further on the observations from the earlier section, where the statistical properties of the loss surface of the middle layers were found to be closest to the loss surface of the overall network, we leveraged the layerwise nature of the proposed HTR to penalize the middlemost layers alone. Such an approach offers computational advantages too, since the number of parameters involved in that term is restricted now to weights only from the middlemost layers. Table 2 reports the generalization error results on different datasets when trained only by penalizing the middlemost layers, against penalizing all layers.

Table 1

| Model + Dataset | $\mathcal{L}_{ce} + $HTR | $\mathcal{L}_{ce}$ |
|-----------------|------------------------|------------------|
| LeNet + MNIST   | 1.19 ± 0.07            | 1.25 ± 0.03      |
| LeNet + FMNIST  | 11.14 ± 0.04           | 11.22 ± 0.11     |
| LeNet + SVHN    | 10.77 ± 0.26           | 10.94 ± 0.06     |
| VGG11 + CIFAR10 | 15.11 ± 0.22           | 18.20 ± 0.45     |
| VGG13 + SVHN    | 7.47 ± 0.01            | 7.73 ± 0.28      |
| ResNet18 + CIFAR10 | 11.95 ± 0.22       | 11.97 ± 0.14     |
| VGG11-BN + CIFAR100 | 44.98 ± 0.05    | 45.25 ± 0.44     |

Table 2

| Model + Dataset | Middle Layers | Full Network |
|-----------------|---------------|--------------|
| LeNet + MNIST   | 1.18 ± 0.07   | 1.19 ± 0.07  |
| LeNet + FMNIST  | 11.10 ± 0.07  | 11.14 ± 0.04 |
| ResNet18 + CIFAR10 | 11.87 ± 0.12  | 11.95 ± 0.22 |
| VGG11-BN + CIFAR100 | 44.81 ± 0.15  | 44.98 ± 0.05 |

The anonymized source code is available at https://anonymous.4open.science/r/6eaea11b-879c-43a8-9926-db6c0f623c8/ for reproducibility.
Table 3: Generalization error, with HTR objective is incorporated into L2 regularizer. (Baseline comparision in Table 1)

| Model                  | Generalization Error ± Standard Deviation |
|------------------------|------------------------------------------|
| LeNet+MNIST            | 1.15 ± 0.02                              |
| LeNet+FMNIST           | 11.11 ± 0.05                             |
| LeNet+SVHN             | 9.96 ± 0.10                              |
| VGG11+CIFAR10          | 15.08 ± 0.16                             |
| ResNet18+CIFAR10       | 11.89 ± 0.13                             |
| VGG11-BN+CIFAR100      | 44.32 ± 0.29                             |

interesting to note that the generalization error that the models in fact perform marginally better in this case, suggesting a deeper connection between the middlemost layer and the overall network’s error surfaces.

Continuing further, we note that the proposed HTR is a general idea, and can also be used in conjunction with other regularizers such as L2 weight decay. We experimented further to study how the proposed HTR works when combined with standard L2 regularizer and the results are reported in Table 3. It can be noted that the generalization error has reduced further in comparison to using HTR alone (see Table 1 for the baseline), indicating the effectiveness of HTR when used with other regularizers.

One issue with the proposed regularizer is that computing the Hessian trace or its derivative at every step can be computationally intensive, although this is mitigated to an extent using layerwise Hessians (whose sizes are smaller). However, to address this issue, we propose the use of the HTR term only at periodic intervals over the training process, and not at every iteration. We term this the penalization/update frequency, \( f_r \). We studied further impact of the choice of \( f_r \), and report these results in Figure 8. As evident from the figure, even lower values of the update frequency performed quite well, thus reducing the computational cost of this method. (The results in Table 1 were, in fact, obtained at \( f_r = 50 \).

Analyzing further, it may be a natural question to ask about understanding the difference between regularizing over the trace of the entire Hessian versus the sum of the trace of layerwise Hessians in the proposed method. We note that under the conditions studied in this work for Eqn 5 where we do not weight each layer differently, both are equivalent since the sum of traces of layerwise Hessians is the trace of the overall Hessian. This is the reason for our result in Figure 7 where the trace of the overall Hessian reduces over training with our regularizer. This can also be seen in Figure 6 where the trace of layerwise Hessian reduces, and the overall trace also reduces. There is, however, a considerable computational advantage with layerwise Hessian trace penalization when compared to full Hessian trace penalization. The trace of each layerwise Hessian can be computed in parallel leading to an increase in time efficiency by a factor of \( L \) (number of layers in the neural network). Layerwise penalization is also effective from a memory perspective, since it is a smaller matrix to compute and store. A major reason for the lack of progress in using second-order information in training DNNs is: (i) The Hessian is computationally intensive to obtain, especially for large models with millions of parameters; (ii) Storing the Hessian matrix is also not memory-efficient. The proposed layerwise HTR, which is the first layerwise regularizer to the best of our knowledge, mitigates both these problems. There is now a considerably smaller memory footprint as we consider only a submatrix of the overall Hessian. The method also allows different coefficients for penalizing different layers differently, or even ignoring certain layers, which could be an added advantage and we leave for future work.

Conclusion

The layerwise analysis of loss surfaces of DNN models deserves the attention of the deep learning community. In this work, we analyzed how layerwise landscape loss properties correlate with overall loss landscape by studying properties primarily through the lens of the Hessian eigenspectrum. We analyzed both the spectral density, as well as specific properties such as maximum eigenvalue and trace of the Hessian over the training of these DNN models. Our studies on state-of-the-art models across datasets show that each layer of a DNN also maintains class discriminability, with the middlemost layers having the strongest connection to the overall loss surface. Our study of the evolution of the spectra showed that state-of-the-art DNN models seek flatter minima, and that middlemost layers maintain a relationship with the overall network through the training process. Motivated by this observation, we propose a new layerwise Hessian-based regularizer, Hessian Trace Regularization method that
works promisingly when models and datasets become complex. We believe that the observations presented in this work will help deepen the community’s understanding about DNN models in general.

Acknowledgements
This work has been partly supported by the funding received from DST, Govt of India, through the MATRICS program (MTR/2017/001047), MHRD and the Intel India PhD Fellowship. We also acknowledge IIT-Hyderabad and JICA for provision of GPU servers for the work. We thank the anonymous reviewers for their valuable feedback that improved the presentation of this work.

References
Anna, C.; LeCun, Y.; and Arous, G. B. 2015. Open Problem: The landscape of the loss surfaces of multilayer networks. In COLT.

Auer, P.; Warmuth, M.; and K, M. K. 1996. Exponentially many local minima for single neurons. In NIPS.

Avron, H.; and Toledo, S. 2011. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. Journal of the ACM (JACM) 58(2): 1–34.

Baldi, P.; and Hornik, K. 1988. Neural networks and principal component analysis: learning from examples without local minima. In Neural Networks.

Beaulieu, C. 1993. Numerical data on neocortical neurons in adult rat, with special reference to the GABA population. Brain Research 609(1): 284 – 292. ISSN 0006-8993. URL http://www.sciencedirect.com/science/article/pii/000689939390884P.

Botev, A.; Ritter, H.; and Barber, D. 2017. Practical Gauss-Newton Optimisation for Deep Learning. In ICML.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J.; Sagun, L.; and Zecchina, R. 2017. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. In ICLR ‘17.

Dauphin, Y. N.; Pascanu, R.; Gulcehre, C.; Cho, K.; Ganguli, S.; and Bengio, Y. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In NIPS.

Dinh, L.; Pascanu, R.; Bengio, S.; and Bengio, Y. 2017. Sharp Minima Can Generalize For Deep Nets. In ICML.

Fort, S.; Hu, H.; and Lakshminarayanan, B. 2019. Deep Ensembles: A Loss Landscape Perspective. arXiv preprint arXiv:1912.02757.

Garipov, T.; Izmailov, P.; Podoprikhin, D.; Vetrov, D. P.; and Wilson, A. G. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnn. In NIPS.

Ghorbani; Shankar, K.; and Xiao, Y. 2019. An Investigation into Neural Net Optimization via Hessian Eigenvalue Density. In ICML.

Gur-Ari, G.; Roberts, D. A.; and Dyer, E. 2018. Gradient Descent Happens in a Tiny Subspace.

Hardt, M.; and Ma, T. 2017. Identity Matters in Deep Learning. In ICLR 2017.

Hochreiter, Schmidhuber, a. J. 1997. Flat Minima. Neural Computation .

Jastrzbeiski, S.; Kenton, Z.; Ballas, N.; Fischer, A.; Bengio, Y.; and Storkey, A. 2019. On the Relation Between the Sharpest Directions of DNN Loss and the SGD Step Length. In ICLR.

Kawaguchi, K. 2016. Deep Learning without Poor Local Minima. In NIPS.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In ICLR 2017.

Lanczos, C. 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Natl. Bur. Stand. B.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the Loss Landscape of Neural Nets. In NIPS.

Lin, L.; Saad, Y.; and Yang, C. 2016. Approximating spectral densities of large matrices. SIAM Review.

Martens, J.; and Grosse, R. 2015. Optimizing Neural Networks with Kronecker-Factored Approximate Curvature. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML’15, 2408–2417. JMLR.org.

Papyan, V. 2018. The Full Spectrum of Deep Net Hessians At Scale: Dynamics with Sample Size. CoRR abs/1811.07062.

Papyan, V. 2019. Measurements of Three-Level Hierarchical Structure in the Outliers in the Spectrum of Deepnet Hessians. In ICML.

Pratik, C.; and Stefano, S. 2018. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In ICLR.

Sagun, L.; Bottou, L.; and LeCun, Y. 2016. Eigenvalues of the hessian in deep learning: Singularity and beyond. arXiv preprint arXiv:1611.07476.

Sagun, L.; Evci, U.; Güney, V. U.; Dauphin, Y. N.; and Bottou, L. 2018. Empirical Analysis of the Hessian of Over-Parametrized Neural Networks. In ICLR, Workshop Track Proceedings.

Villani, C. 2008. Optimal transport – Old and new, xxii+973. doi:10.1007/978-3-540-71050-9.

Zhang, C.; Bengio, S.; and Singer, Y. 2019. Are all layers created equal? arXiv preprint arXiv:1902.01996.
Supplementary Material

In this section, we include more details on how to estimate the eigenspectrum of Hessian matrix using the Lanczos method, and more experimental results on the eigenspectra of state-of-the-art networks such as ResNet and DenseNet, which could not be included in the main paper due to space constraints.

Eigenspectrum Estimation Algorithm:

Algorithm 2 presents the Lanczos method which estimates the eigenspectrum of a given matrix. It should be noted that the Lanczos method requires only Hessian-vector product (lines 5,7 in Algorithm) which can be easily approximated for a deep neural network.

Algorithm 2: Eigenspectrum Estimation of Hessian using Lanczos method.

**Input:** Hessian $A$ with spectrum in range [-1, 1], Num of iterations $M$, Num of samples $K$, $\kappa$  

**Output:** (Approx) Eigenspectrum density of $A$

```plaintext
1: for $i = 1, \ldots, M$ do
2:   \text{if } i == 1 then
3:     sample $v \sim \mathcal{N}(0, I)$;
4:     $v = \gamma ||v||_2$;
5:     $v_{next} = Av$;
6:   else
7:     $v_{next} = Av - \beta_m - 1 v_{prev}$;
8: end if
9:   $\alpha_m = v_{next}^Tv$;
10:  $v_{next} = v_{next} - \alpha_m v$;
11:  $\beta_m = ||v_{next}||_2$;
12:  $v_{next} = v_{next}/\beta_m$;
13:  $v_{prev} = v$;
14:  $v = v_{prev}$;
15: end for
16: $T_M = \begin{bmatrix} \alpha_1 & \beta_1 \\ \beta_1 & \alpha_2 \\ \vdots & \vdots \\ \beta_{M-1} & \alpha_M \end{bmatrix}$;
17: $\{\lambda_i\}_{i=1}^{M}$, $\{y_i\}_{i=1}^{M} = \text{eig}(T_M)$;
18: $\{t_k\}_{k=1}^{K} = \text{linspace}(-1, 1, K)$;
19: for $k = 1 \ldots K$ do
20:   $\sigma = (M-1)^{1/2} \log(K)$;
21:   $\omega_k = \sum_{i=1}^{M} y_i^{1/2} g_\sigma(t - \lambda_i)$;
22: end for
23: \text{return } $\{\omega_k\}_{k=1}^{K}$
```

More Results:

We ran the layerwise eigenspectrum analysis further and present the results on ResNet-18 and DenseNet in Figs 9, 10 when trained on the CIFAR10 dataset. Also, the layerwise t-SNE plots obtained upon clustering $G$ when a DenseNet is trained on CIFAR-10 is shown in Fig 11.

Continuing our analysis further on studying the distance between the spectra of $\{\text{Hess}_l\}_{l=1}^{L}$ and $\{G_l\}_{l=1}^{L}$, we calculated the Jensen–Shannon divergence between the layerwise Hessian and full network Hessian (instead of Wasserstein distance used in the main paper). The results are shown in Fig 12. The results corroborate the inferences made in the main paper. Even with a different distance metric, it can be observed that the spectra of Hess$_l$ of middle layers of DNN models are very similar to spectrum of the overall Hess. Such an observation has not been made hitherto, to the best of our knowledge.

In continuation to Fig 4 in the main paper, we present results on the evolution of the layerwise Hessian eigenspectra using $\lambda_{max}$ (we presented the evolution of the full Hessian’s $\lambda_{max}$ in Fig 4). Fig 13 shows how the $\lambda_{max}$ of each layerwise Hessian of ResNet-18 when trained on CIFAR-10 evolves as learning progresses. It can be observed that $\lambda_{max}$ of each layerwise Hessian goes down over training and also the middle layers have values of $\lambda_{max}$ closest to that of the entire network across the epochs.

Ablation Studies and Analysis:

We conducted further experiments to study the performance of the proposed method more closely. As a first step, we studied the effect of the weighting co-efficient, $\gamma$ (Eqn 5), of the proposed HTR method. We varied $\gamma$ from $1e^{-3}$ to $1e^{-1}$ and studied the performance of HTR. Figure 14 shows the results, which shows that a weighting co-efficient around $1e^{-2}$ performed the best in this experiment. Lower values
Figure 10: Eigenspectra of \( \{\text{Hess}_i\}_{i=1}^L \) and \( \{\text{G}_i\}_{i=1}^L \) on DenseNet trained on CIFAR-10 (Best viewed in color).

Figure 11: Layerwise two-dimensional t-SNE embeddings of \( \delta_c \), obtained as a decomposition of \( \{\text{G}_i\}_{i=1}^L \) in (Papyan 2018) show \( C \) clusters in each subplot (DenseNet on CIFAR-10). The last subplot “full” corresponds to \( \delta_c \) of the entire \( \text{G} \) matrix.
Figure 12: Plots of normalized Jensen–Shannon divergence between the spectra of $\text{Hess}_l; \forall l = 1 \cdots L$ and $\text{Hess}$ of different DNN models across datasets.

gave worse performance, showing the usefulness of the HTR term.

We also performed a wall-clock time comparison of training with and without HTR on various networks, and observed that the trace penalization takes approximately twice the time to finish across the different experiments. Table 4 reports these results. We can further accelerate the speed of HTR by penalizing the Hessian trace of specific layers in specific epochs of training, based on our understanding of the training process, or adaptively choose the weight coefficients of the layers of the Hessian with a sparsity constraint to ensure that only few layers are considered for regularization in each iteration. This will make computations cheaper, and we leave this interesting direction for future work.

Figure 13: Evolution of $\lambda_{\text{max}}$ of ResNet-18 when trained on CIFAR-10

**Connection between Hessian Trace and Eigenspectrum:**

When we penalize only the Hessian trace during training, there might exist situations when $\lambda_{\text{min}}$ of the Hessian can become negative (and potentially large in magnitude) making learning unstable. The Hessian trace can be near zero under three conditions: (i) All eigenvalues are near zero and positive; (ii) Eigen spectrum is symmetric about zero, or (iii) Eigen spectrum is asymmetric about zero, but with an equal density on the positive and negative side. The above conditions, when combined with the fact that $\lambda_{\text{max}}$ also goes to near zero (which our results in Sec 4 show) largely eliminates condition (iii). When all eigenvalues are positive and near zero, trace penalization leads to a flat minima, and is useful as has been shown in earlier work. When eigen distribution is symmetric around zero, trace penalization is still well-behaved as long as $\lambda_{\text{max}}$ is reduced - due to the symmetry assumption, the magnitude of $\lambda_{\text{min}}$, even if negative, is also close to zero. When there is an equal amount of energy on both positive and negative sides of the eigen spectrum, the reduction of $\lambda_{\text{max}}$ as learning progresses ensures that the magnitude of $\lambda_{\text{min}}$ must be close to zero. Thus, our empirical findings suggest that as trace and $\lambda_{\text{max}}$ of the Hessian go to zero over training, a flat minimum ensues - thus motivating the HTR method.

Figure 14: Ablation study on regularization parameter $\gamma$ in HTR, LeNet+FMNIST (Best viewed in color, solid lines denote train accuracy, dashed lines denote test accuracy)

| Model + Dataset       | freq | ratio      |
|-----------------------|------|------------|
| LeNet+MNIST           | 25   | 1.07 ± 0.13|
|                       | 50   | 1.14 ± 0.11|
|                       | 100  | 1.14 ± 0.15|
|                       | 200  | 1.00 ± 0.09|
| VGG11+CIFAR10         | 25   | 3.85 ± 0.06|
|                       | 50   | 2.40 ± 0.01|
|                       | 100  | 1.71 ± 0.01|
|                       | 200  | 1.42 ± 0.03|
| VGG11-bn+CIFAR10      | 25   | 4.93 ± 0.03|
|                       | 50   | 2.98 ± 0.02|
|                       | 100  | 1.99 ± 0.01|
|                       | 200  | 1.49 ± 0.01|
| ResNet18+CIFAR10      | 25   | 17.19 ± 0.06|
|                       | 50   | 9.09 ± 0.01|
|                       | 100  | 5.07 ± 0.05|
|                       | 200  | 3.00 ± 0.01|
| VGG11-bn+CIFAR100     | 25   | 3.01 ± 0.02|
|                       | 50   | 2.00 ± 0.01|
|                       | 100  | 1.50 ± 0.01|
|                       | 200  | 1.25 ± 0.00|

Table 4: Ratios of wall clock times per epoch between proposed HTR regularization and Ordinary (SGD) training
Connection to Neuroscience:

The observation of middle layers being similar to the overall network has been made in studies conducted in neuroscience too. The work by Beaulieu [1993] studied how the neuronal density of parts of the brain, viz. occipital, parietal, and frontal, varies in a layerwise manner, and made similar conclusions. It was found that the neuron density in the middle layers of the brain is equal to the overall average neuronal density of the brain. This correlation between our observation of the middle layer Hessian spectrum closely matching the overall spectrum, and the middle layer neuronal density approximately equal to the average overall density is an interesting connection between the two fields.