The Hessian perspective into the Nature of Convolutional Neural Networks

Sidak Pal Singh [1] Thomas Hofmann [1] Bernhard Schölkopf [2]

Abstract

While Convolutional Neural Networks (CNNs) have long been investigated and applied, as well as theorized, we aim to provide a slightly different perspective into their nature — through the perspective of their Hessian maps. The reason is that the loss Hessian captures the pairwise interaction of parameters and therefore forms a natural ground to probe how the architectural aspects of CNNs get manifested in their structure and properties. We develop a framework relying on Toeplitz representation of CNNs, and then utilize it to reveal the Hessian structure and, in particular, its rank. We prove tight upper bounds (with linear activations), which closely follow the empirical trend of the Hessian rank and in practice also hold for more general settings. Overall, our work generalizes and further establishes the key insight that the Hessian rank grows as the square root of the number of parameters, even in CNNs.

1. Introduction

Nobody would deny that CNNs (Fukushima, 1980; LeCun et al., 1995; Krizhevsky et al., 2017), with their baked-in equivariance to translations, have played a key role in the practical successes of deep learning and computer vision. Yet several aspects of their nature are still unclear. Cutting directly to the chase, for instance, take a look at Figure 1. As the number of channels in the hidden layers increases, the number of parameters grows, as expected, quadratically. But, the rank of the loss Hessian at initialization, measured as precisely as it gets, grows at a much calmer, linear rate. How come?

This question lies at the core of our work. Generally speaking, we would like to investigate the inherent 'nature' of CNNs, i.e., how its architectural characteristics manifest

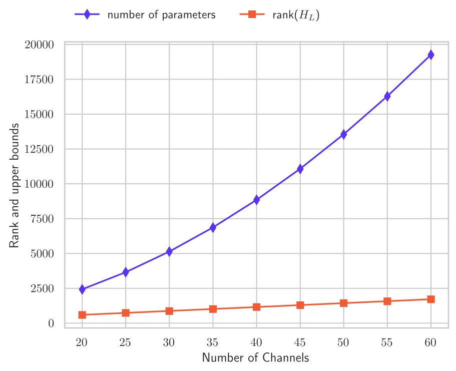


Figure 1: A comparison of the **number of parameters** with the empirically observed **loss Hessian rank** for increasing number of channels m for a 2-hidden layer CNN on CIFAR10. *Can we estimate the precise scaling behaviour of the Hessian rank?*

themselves in terms of a property or a phenomenon at hand — here that of Figure 1, which, among other things, would precisely indicate the effective dimension of the (local) loss landscape (MacKay, 1992b; Gur-Ari et al., 2018).

Of course, when the likes of Transformer (Vaswani et al., 2017; Dosovitskiy et al., 2020), MLPMixer (Tolstikhin et al., 2021) and their kind, have ushered in a new wave of architecture design, especially when equipped with heaps of data, it is tempting to think that studying CNNs might not be as worthwhile an enterprise. Well, that only time and tide can tell — but it seems that, at least for now, the concepts and principles behind CNNs (such as patches, larger strides, weight sharing, downsampling) continue to be carried along while designing newer architectures in the 2020s (Liu et al., 2021; 2022). And, more broadly, if the perspectives and techniques that help us understand the nature of a given architecture are general enough, they may also be of relevance when considering another architecture of interest.

We are inspired by one such account of an intriguing perspective: the extensive redundancy in the parameterization of fully-connected networks, as measured through the Hessian degeneracy (Sagun et al., 2016), and recently outlined rigorously in the work of (Singh et al., 2021). We aim to chart this out in detail for CNNs[1].

[1]ETH Zürich, Switzerland [2]MPI for Intelligent Systems, Tübingen, Germany. Correspondence to: Sidak Pal Singh <sidak.singh@inf.ethz.ch>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]The corresponding code can be found under https://github.com/sidak/hessian_perspective_cnns. Also,

2. Related Work

The Nature of CNNs. To start with, there's the intuitive perspective of enabling a hierarchy of features (LeCun et al., 1995). A more mathematical take is that of (Bruna & Mallat, 2013) where filters are constructed as wavelets and which as a whole provide Lipschitz continuity to deformations. The approximation theory view (Poggio et al., 2015; Mao et al., 2021) reinforces the intuitive benefits of hierarchy and compositionality with explicit constructions. On the optimization side, the implicit bias perspective (Gunasekar et al., 2018) stresses on how gradient descent on CNNs leads to a particular solution. Other notable takes are from the viewpoint of Gaussian processes (Garriga-Alonso et al., 2018), loss landscapes (Nguyen & Hein, 2018; Gu et al., 2020), arithmetic circuits (Cohen & Shashua, 2016) — to list a few. In contrast, we focus on how, *from the initialization itself*, the CNN architecture induces structural properties of the loss Hessian and by extension, of the loss landscape.

Hessian maps and Deep Learning. The Hessian characterizes parameter interactions via second derivative of the loss. Consequently, it has been a central object of study and has been extensively utilized for applications and theory alike, both in the past as well as the present. For instance, generalization (MacKay, 1992a; Keskar et al., 2016; Yang et al., 2019; Singh et al., 2022), optimization (Zhu et al., 1997; Setiono & Hui, 1995; Martens & Grosse, 2015; Cohen et al., 2021), network compression (LeCun et al., 1990; Hassibi & Stork, 1992; Singh & Alistarh, 2020), continual learning (Kirkpatrick et al., 2017), hyperparameter search (LeCun et al., 1992; Schaul et al., 2013), and more.

In recent times, it has been revitalized in significant part due to the 'flatness hypothesis' (Keskar et al., 2016; Hochreiter & Schmidhuber, 1997) and in turn, flatness has become a popular method to probe the extent of generalization as it seems to consistently rank ahead of traditional norm-based complexity measures (Jiang et al., 2019) in multiple scenarios. Given the increasing size of the networks, and the inherent limitation of the Hessian being quadratic in cost, measuring flatness has almost become synonymous with measuring the top eigenvalue of the Hessian (Cohen et al., 2021) or even just the zeroth-order measurement of the loss stability in the parameter space. To a lesser extent, some works still utilize efficient approximations to the Hessian trace (Yao et al., 2020) or log determinant (Jia & Su, 2020). But largely the reliance on the Hessian for neural networks has become a black-box affair.

Understanding of the Neural Network Hessian maps. Lately, significant advances have been made in this direction — a few of the most prominent being: the characterization of its spectra as bulk and outliers (Sagun et al., 2016; Pennington & Bahri, 2017; Ghorbani et al., 2019), the empirically observed significant rank degeneracy (Sagun et al., 2017), and the class/cross-class structure (Papyan, 2020). Despite these advancements, its structure for neural networks primarily gets seen only up to the surface level of the chain rule[2] for a composition of functions, with a few exceptions such as (Wu et al., 2020; Singh et al., 2021).

We take our main inspiration from the latter of these works, namely (Singh et al., 2021), where the authors precisely characterize the Hessian structure for deep fully-connected networks (FCNs) resulting in concise bounds and formulae on the Hessian rank of arbitrary-sized networks, thereby being the premier work to theoretically demonstrate the rank degeneracy of the Hessian. Our aim, hence, is to thoroughly exhibit the structure of the Hessian for deep convolutional networks and explore the distinctive facets that arise therein — as compared to FCNs.

Our contributions are: (1) We develop a framework to analyze CNNs that rests on a Toeplitz representation of convolution[3] and applies for general deep, multi-channel, arbitrary-sized CNNs, while being amenable to matrix analysis. We then utilize this framework to unravel the Hessian structure for CNNs and provide upper bounds on the Hessian rank. Our bounds are exact for the case of 1-hidden layer CNNs, and in the general case, are of the order of the square root of the trivial bounds that are based on parameter count.

(2) Next, we verify our bounds empirically in a host of settings, where we find that our upper bounds remain rather close to the true empirically observed Hessian rank. Moreover, they even hold faithfully outside the confines of the theoretical setting (e.g., choice of the loss and activation functions) used to derive them.

(3) Further, we make a detailed comparison of the key ingredients in CNNs, i.e., local connectivity and weight sharing, in a simplified setting through the perspective of our Hessian results. Here, we also discuss some elements of our proof technique in the hope that it helps provide a better grasp of the results.

We would also like to make a quick remark about the *difference with regards to (Singh et al., 2021)*. While we borrow heavily from their approach, the framework we develop here provides us with the flexibility to handle convolutions, pooling operations, and even fully-connected layers. In particular, our analysis captures their results as a special case when the filter size is equal to the spatial dimension. We

please check out arXiv:2305.09088 for the latest version of our paper.

[2]This is known otherwise as the Gauss-Newton decomposition (Schraudolph, 2002b) and discussed in Eqn. 1.

[3]Convolution as used in practice in deep learning, and not, say circular convolution —despite its relative theoretical ease.

also introduce a novel proof technique relative to (Singh et al., 2021), without which one cannot attain exact bounds on the Hessian rank for the one-hidden layer case.

Overall, we hope that by building on the prior work, we can further push this research direction of understanding the nature of various network architectures through an in-depth, white-box, analysis of the Hessian structure.

3. Setup

Notation. We denote vectors in lowercase bold ($\mathbf{x}$), matrices in uppercase bold ($\mathbf{X}$), and tensors in calligraphic letters ($\mathcal{X}$). We will denote the i-th row and j-th columns of some matrix $\mathbf{A}$ by $\mathbf{A}_{i\bullet}$ and $\mathbf{A}_{\bullet j}$ respectively. We will often use the notation $\mathbf{A}^{(i:j)}$, for $i > j$ to indicate a sequence of matrices from i down to j, i.e., $\mathbf{A}^{(i:j)} = \mathbf{A}^{(i)}\mathbf{A}^{(i-1)}\cdots\mathbf{A}^{(j+1)}\mathbf{A}^{(j)}$. For $i < j$, the same notation[4] would mean the sequence of matrices from i up to j, but *tranposed*. To express the structure of the gradients and the Hessian, we will employ matrix derivatives (Magnus & Neudecker, 2019), wherein we vectorize row-wise (vec_r) the involved matrices and organize the derivative in Jacobian (numerator layout). Concretely, for matrices $\mathbf{X} \in \mathbb{R}^{m\times n}$ and $\mathbf{Y} \in \mathbb{R}^{p\times q}$, we have

$$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}} := \frac{\partial \,\text{vec}_r\, \mathbf{Y}}{\partial\, (\text{vec}_r\, \mathbf{X})^\top} \in \mathbb{R}^{pq\times mn}\,.$$

Setting. Suppose we are given an i.i.d. dataset $S = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$, of size $|S| = n$, drawn from an unknown distribution $p_{\mathbb{X},\mathbb{Y}}$, consisting of inputs $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^d$ and targets $\mathbf{y} \in \mathbb{Y} \subseteq \mathbb{R}^K$. Based on this dataset S, consider we use a neural network to learn the mapping from the inputs to the targets, $\mathrm{F}_{\boldsymbol{\theta}} : \mathbb{X} \mapsto \mathbb{Y}$, parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^p$. To this end, we follow the framework of Empirical Risk Minimization (Vapnik, 1991), and optimize a suitable loss function $\mathrm{L} : \Theta \mapsto \mathbb{R}$. In other words, we solve the following optimization problem,

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\text{argmin}}\; \mathrm{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(\boldsymbol{\theta}; (\mathbf{x}_i, \mathbf{y}_i)\right)\,,$$

say with a first-order method like (stochastic) gradient descent and the choices for ℓ could be mean-squared error (MSE), cross-entropy (CE), etc.

Hessian. We analyze the properties of the Hessian of the loss function, $\mathbf{H}_{\mathrm{L}} = \dfrac{\partial^2 \mathrm{L}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\,\partial\boldsymbol{\theta}^\top}$, with respect to the parameters $\boldsymbol{\theta}$. It is quite well known (Schraudolph, 2002a; Sagun et al., 2017; Singh et al., 2022) that, via the chain rule, the Hessian can be decomposed as a sum of the following two matrices:

$$\mathbf{H}_{\mathrm{L}} = \mathbf{H}_{\mathrm{O}} + \mathbf{H}_{\mathrm{F}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}}\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}_i)\left[\nabla^2_{\mathrm{F}_{\boldsymbol{\theta}}}\ell_i\right]\nabla_{\boldsymbol{\theta}}\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}_i)^\top + \frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{K}[\nabla_{\mathrm{F}_{\boldsymbol{\theta}}}\ell_i]_c\, \nabla^2_{\boldsymbol{\theta}}\mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x}_i) \qquad (1)$$

where, $\nabla_{\boldsymbol{\theta}}\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}_i) \in \mathbb{R}^{p\times K}$ is the Jacobian of the function and $\nabla^2_{\mathrm{F}_{\boldsymbol{\theta}}}\ell_i \in \mathbb{R}^{K\times K}$ is the Hessian of the loss with respect to the network function, at the i-th sample. To facilitate comparison, we refer to these two matrices as the *outer-product Hessian* and the *functional Hessian* following (Singh et al., 2021).

4. Background

Precise bounds on the Hessian rank. Despite the much interest in Hessian over the decades, the upper bounds remained quite trivial (like a factor of the number of samples) and the empirically observed degeneracy (Sagun et al., 2016), because of the need to measure rather small eigenvalues and judge a suitable threshold, remained intractable to establish. Exact upper bounds and formulae have only been made available very recently for fully-connected networks due to (Singh et al., 2021), as a result of subtle choice to have linear activations, together with a novel analysis technique adapted from matrix analysis (Matsaglia & Styan, 1974; Chuai & Tian, 2004). While they find that the presence of non-linearities like ReLU impedes a theoretical analysis, (Singh et al., 2021) thoroughly demonstrate that their bounds hold empirically for ReLU-based FCNs as well. Their empirical analysis is equally rigorous, for they compute exact Hessians (without any approximation) in Float64 precision.

Overall, the surprising finding of (Singh et al., 2021) is that the Hessian rank for FCNs scales[5] linearly in the total number of neurons m, i.e., $\mathcal{O}(m)$; while the number of parameters scale quadratically, $\mathcal{O}(m^2)$. While this concrete bound of $\mathcal{O}(m)$ is shown at initialization, their analysis applies to any point in the loss landscape and thereby also during training — but with the resulting bound in terms of the rank of the individual weight matrices. Further, they highlight that during training, the rank can only decrease (a simple consequence of the functional Hessian $\mathbf{H}_{\mathrm{F}}$ being driven to zero since it is scaled by the gradient of the loss $\nabla_{\mathrm{F}_{\boldsymbol{\theta}}}\ell$). In this sense, their upper bounds hold pointwise throughout the loss landscape and the optimization trajectory.

Why Hessian Rank? The finding about the growth of Hessian rank carries thought-provoking implications on the

[4]When either of i or j are outside the bounds of a particular index set, this notation would devolve to an identity matrix.

[5]More precisely, this should be $\mathcal{O}(q \cdot m)$, however, q denotes the bottleneck dimension in the network — inclusive of input and output dimensions, and hence can be thought of as a constant.

number of effective parameters within a particular architecture. Let us illustrate this by considering the Occam's factor (MacKay, 1992b; Gull, 1989) which, said roughly, describes the extent to which the prior hypothesis space shrinks on observing the data. More formally, this is the ratio of posterior to prior volumes in the parameter space. This is then used to derive the following measure for the effective degrees of freedom, assuming a quadratic approximation to the posterior:

$$\sum_{i=1}^{p} \frac{\lambda_i}{\lambda_i + \epsilon} \,,$$

where, λ_i denotes the i-th eigenvalue of the Hessian, p is the total number of parameters, and ϵ is the weight set upon the prior. In other words, the above measure compares the extent (λ_i) to which a particular direction (along the i-th eigenvector) in the parameter space is determined by the data relative to that determined by the prior ϵ. For small ϵ, which amounts to little or no explicit regularization towards the prior (as is often the case with deep networks in practice), this measure of the degrees of freedom approaches the Hessian rank. In a recent work (Maddox et al., 2020) empirically noted that this measure also explains double descent (Belkin et al., 2019) in neural networks. Thus, it makes it all the more pertinent to explore how rank of the Hessian scales with various architectural parameters in a CNN.

As a sidenote, we also carry out a preliminary study where we sweep over the filter sizes and the number of channels in a CNN, and find that the Hessian rank, *at initialization,* has a higher correlation coefficient (see Figure 15) with generalization error as compared to the raw count of parameters.

5. Toeplitz Framework for CNN Hessians

Preliminaries. In this section, we will lay out the formalism that will lie at the core of our analysis. For brevity, we will develop this in the case of 1D CNNs which are commonly employed for biomedical applications or audio data (Kiranyaz et al., 2021). One can otherwise think of applying our framework to flattened filters and input patches, and such an assumption is also prevalent in the theory literature (Kohn et al., 2021). Besides, throughout our framework we will assume there are no bias parameters, although one can simply consider homogeneous coordinates in the input.

A Gentle Start. Let's say we want to represent the convolution $\mathbf{W} * \mathbf{x}$ of an input $\mathbf{x} \in \mathbb{R}^d$ with m filters of size $k \le d$ that have been organized in the matrix $\mathbf{W} \in \mathbb{R}^{m\times k}$. For now, consider that we have stride 1 and zero padding. We will later touch upon these aspects and see the results for strides > 1 in Section 8. Further, for some vector $\mathbf{z} \in \mathbb{R}^d$, we will use the notation $\mathbf{z}_{j:j+k-1} \in \mathbb{R}^k$ to denote the (shorter) vector formed by considering the indices j to $j+k-1$ (both inclusive) of the original vector. The output of the above convolution can be expressed as the following matrix of shape $m \times (d-k+1)$,

$$\mathbf{W} * \mathbf{x} = \begin{pmatrix} \langle \mathbf{W}_{1\bullet}, \mathbf{x}_{1:k}\rangle & \cdots & \langle \mathbf{W}_{1\bullet}, \mathbf{x}_{d-k+1:d}\rangle \\ \vdots & & \vdots \\ \langle \mathbf{W}_{m\bullet}, \mathbf{x}_{1:k}\rangle & \cdots & \langle \mathbf{W}_{m\bullet}, \mathbf{x}_{d-k+1:d}\rangle \end{pmatrix} . \quad (2)$$

Now, define Toeplitz[6] matrices for each filter, $\{\mathbf{T}^{\mathbf{W}_{i\bullet}}\}_{i=1}^{m}$, with $\mathbf{T}^{\mathbf{W}_{i\bullet}} := \text{toep}(\mathbf{W}_{i\bullet}, d) \in \mathbb{R}^{(d-k+1)\times d}$ such that,

$$\mathbf{T}^{\mathbf{W}_{i\bullet}} = \begin{pmatrix} w_{i1} & \cdots & w_{ik} & 0 & \cdots & 0 \\ 0 & w_{i1} & \cdots & w_{ik} & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_{i1} & \cdots & w_{ik} \end{pmatrix} .$$

Note, the above representation of the Toeplitz matrix also depends on the base dimension (here, d) where the given vector must be 'toeplitzed', i.e., circulated in the above fashion. But we will omit specifying this unless necessary. Let us also denote the matrix formed by stacking the $\mathbf{T}^{\mathbf{W}_{i\bullet}}$ matrices in a row-wise fashion as

$$\mathbf{T}^{\mathbf{W}} := \begin{pmatrix} \mathbf{T}^{\mathbf{W}_{1\bullet}} \\ \vdots \\ \mathbf{T}^{\mathbf{W}_{m\bullet}} \end{pmatrix} \in \mathbb{R}^{m(d-k+1)\times d} .$$

We can now see that the above matrix, $\mathbf{T}^{\mathbf{W}}$, when multiplied by the input $\mathbf{x}$ gives us the output of the convolution operation in Eqn. (2) when vectorized row-wise , i.e.,

$$\text{vec}_r(\mathbf{W} * \mathbf{x}) = \mathbf{T}^{\mathbf{W}}\mathbf{x} .$$

5.1. Toeplitz representation of deep CNNs

Now, let us assume we have L hidden layers, each of which is a convolutional kernel. Hence, the parameters of the l-th layer are denoted by the tensor $\mathcal{W}^{(l)} \in \mathbb{R}^{m_l \times m_{l-1} \times k_l}$, where m_l represent the number of output channels, m_{l-1} the number of input channels, and k_l the kernel size at this layer. As we assume a one-dimensional input, without loss of generality, we can set the number of input channels $m_0 = 1$. In other words, they are already assumed to be flattened when passing the input of dimension $d_0 := d$ into the network. The spatial dimension after being convolved with the l-th layer is denoted by $d_l = d_{l-1} - k_l + 1$ (which is basically the number of hops we can make with the given kernel over its respective input), since we have stride 1

[6]These are matrices $\mathbf{A}$ with $\mathbf{A}_{ij} = \mathbf{a}_{i-j}$ formed via some underlying vector $\mathbf{a}$

and zero padding. Assume, say the ReLU nonlinearity $\sigma(x) := \max(x, 0)$. The network function can then be formally represented as (although it will be actually defined through Eqn. (3) later):

$$\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathcal{W}^{(L+1)} * \sigma(\mathcal{W}^{(L)} * \sigma(\cdots * \sigma(\mathcal{W}^{(1)} * \mathbf{x}))) .$$

As before, we would like to express the above function in terms of a sequence of appropriate Toeplitz matrix products. Unlike the warmup scenario, the convolutional kernels in this general case will be tensors. The key idea is to do a column-wise stacking of individual Toeplitz matrices across the input channels while maintaining the row-wise stacking, as before, across the output channels.

First, we need to introduce a notation about indexing the fibres of a tensor $\mathcal{W}^{(l)} \in \mathbb{R}^{m_l \times m_{l-1} \times k_l}$. Say we need the fibre going in to the plane, across the third mode. Then, the (i, j)-th fibre is denoted by $\mathcal{W}^{(l)}_{(i,j)\bullet} \in \mathbb{R}^{k_l}$, whose associated Toeplitz matrix will be $\mathbf{T}^{\mathcal{W}^{(l)}_{(i,j)\bullet}} := \operatorname{toep}(\mathcal{W}^{(l)}_{(i,j)\bullet}, d_{l-1}) \in \mathbb{R}^{d_l \times d_{l-1}}$. Finally, the Toeplitz matrix associated with the entire l-th convolutional layer $\mathcal{W}^{(l)}$, for which we use the shorthand $\mathbf{T}^{(l)} \in \mathbb{R}^{m_l d_l \times m_{l-1} d_{l-1}}$, can be expressed as:

$$\mathbf{T}^{(l)} := \begin{pmatrix} \mathbf{T}^{\mathcal{W}^{(l)}_{(1,1)\bullet}} & \cdots & \mathbf{T}^{\mathcal{W}^{(l)}_{(1,m_{l-1})\bullet}} \\ \vdots & & \vdots \\ \mathbf{T}^{\mathcal{W}^{(l)}_{(m_l,1)\bullet}} & \cdots & \mathbf{T}^{\mathcal{W}^{(l)}_{(m_l,m_{l-1})\bullet}} \end{pmatrix} .$$

In other words, the Toeplitzed representation $\mathbf{T}^{(l)}$ consists of $m_l \cdot m_{l-1}$ many Toeplitz blocks formed by the vector $\mathcal{W}^{(l)}_{(i,j)\bullet} \in \mathbb{R}^{k_l}$ of size $d_l \times d_{l-1}$. The output (spatial) dimension d_{L+1} is typically 1 (corresponding to $k_{L+1} = d_L$), and the number of output channels for the last layer $m_{l+1} = K$ where K is the number of targets. Now, the network function can be written in the general case as:

$$\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{T}^{(L+1)} \Lambda^{(L)} \mathbf{T}^{(L)} \cdots \Lambda^{(1)} \mathbf{T}^{(1)} \mathbf{x} , \qquad (3)$$

where, $\Lambda^{(i)}$ is an input-dependent diagonal matrix that contains a 1 or 0, based on whether the neuron was activated or not. As the rank analysis of (Singh et al., 2021) requires linear activations, we will follow the same course. However, we can expect that this should still let us contrast the distinctive facets of CNN relative to a FCN. Anyways, later we will elaborate on the case of nonlinearities. Besides, we will refer to the above network function, more concisely as $\mathbf{T}^{(L+1:1)}\mathbf{x}$.

Remark. As evident, the fact that a convolution of a set of vectors can be expressed as a matrix-vector product with a suitable Toeplitz matrix is rather straightforward. Also, a Toeplitz representation for CNN is not new — works from approximation theory incorporate a similar formalism (Zhao et al., 2017; Fang et al., 2020). However, unlike past work (Sedghi et al., 2018; Kohn et al., 2021), we develop our framework in a way that doesn't overlook how convolutions are prominently used, i.e., without circular convolution, with multiple channels, and possibly unequal-sized layers.

5.2. Matrix derivatives of Toeplitz representations

For the gradient and Hessian calculations, we make use of matrix derivatives and the corresponding chain rule. Thus we frequently compute the gradient of the Toeplitz representation $\mathbf{T}^{(l)}$ with respect to the suitably matricized convolutional tensor, $\operatorname{mat} \mathcal{W}^{(l)}$. In order to be consistent with the form of $\mathbf{T}^{(l)}$, we define our matricization of $\mathcal{W}^{(l)}$ as:

$$\operatorname{mat} \mathcal{W}^{(l)} := \begin{pmatrix} \mathcal{W}^{(l)}_{(1,1)\bullet} & \cdots & \mathcal{W}^{(l)}_{(m_l,1)\bullet} \\ \vdots & & \vdots \\ \mathcal{W}^{(l)}_{(1,m_{l-1})\bullet} & \cdots & \mathcal{W}^{(l)}_{(m_l,m_{l-1})\bullet} \end{pmatrix}^{\top} . \qquad (4)$$

where the matricization $\operatorname{mat} \mathcal{W}^{(l)} \in \mathbb{R}^{m_l \times m_{l-1} k_l}$ and we will use the notation $\mathbf{W}^{(l)} := \operatorname{mat} \mathcal{W}^{(l)}$ as a shorthand. Essentially, we have arranged each of the mode-3 fibres as rows in the output channels times input channels format.

The following lemma equips us with the way to carry this out (the proof can be found in Section A.1 of the Appendix).

Lemma 1. *The matrix derivative of* $\mathbf{T}^{(l)}$ *with respect to* $\mathbf{W}^{(l)}$*, is given as follows:*

$$\widetilde{\mathbf{Q}}^{(l)} := \frac{\partial \mathbf{T}^{(l)}}{\partial \mathbf{W}^{(l)}} := \frac{\partial \operatorname{vec}_r \mathbf{T}^{(l)}}{\partial \left(\operatorname{vec}_r \mathbf{W}^{(l)}\right)^{\top}} = \mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)} .$$

The particular structure of $\mathbf{Q}^{(l)}$ is a bit complex, involving various permutation matrices. So, for simplicity, we abstract it out here in the main text.

5.3. CNN Hessian Structure

Like (Singh et al., 2021), for our theoretical analysis, we will consider the case of MSE loss. But the results still hold empirically, say, for CE loss. Let's now have a glance at the kl-th block, $k \leq l$ for both the outer-product and the functional Hessian (the derivations are in Section B). This corresponds to looking at the submatrix of the Hessian corresponding to the k-th and l-th convolutional parameter tensors.[7] Let's start with the outer-product Hessian $\mathbf{H}_{\mathrm{O}}$.

[7]In the case of $\mathbf{H}_{\mathrm{F}}$, the present form is for $k \leq l$. For, $k \geq l$, it's a bit different and detailed in the Appendix.

Proposition 2. *The kl-th block of $\mathbf{H}_\mathrm{O}$ is,*

$$\mathbf{H}_\mathrm{O}^{(kl)} = \widetilde{\mathbf{Q}}^{(k)\top}\Big(\mathbf{T}^{(k+1:L+1)}\mathbf{T}^{(L+1:l+1)} \otimes \mathbf{T}^{(k-1:1)}\boldsymbol{\Sigma}_{\mathbf{xx}}\mathbf{T}^{(1:l-1)}\Big)\widetilde{\mathbf{Q}}^{(l)}\,. \tag{5}$$

A word about $\mathbf{H}_\mathrm{O}$. We should also emphasize that the outer-product Hessian shares exactly the same non-zero spectrum as the Neural Tangent Kernel (Jacot et al., 2018), or roughly up to scaling, in the case of Fisher Information (Amari, 1998), empirical Fisher (Kunstner et al., 2019)).

Moving on to the functional Hessian $\mathbf{H}_\mathrm{F}$, denote $\boldsymbol{\Omega} = \mathbf{E}\left[\boldsymbol{\delta}_{\mathbf{x},\mathbf{y}}\,\mathbf{x}^\top\right] \in \mathbb{R}^{K\times d_0}$ as the (uncentered) covariance of the residual $(\mathbf{y}_i - \mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}_i))$ with the input. Then we have,

Proposition 3. *The kl-th block of $\mathbf{H}_\mathrm{F}$ is:*

$$\mathbf{H}_\mathrm{F}^{(kl)} = \Big(\mathbf{I}_{m_k}\otimes\mathbf{Q}^{(k)\top}\Big)\Big(\mathbf{T}^{(k+1:l-1)}\otimes \mathbf{T}^{(k-1:1)}\boldsymbol{\Omega}^\top\mathbf{T}^{(L+1:l+1)}\Big)\Big(\mathbf{Q}^{(l)}\otimes\mathbf{I}_{m_l}\Big)\,, \tag{6}$$

A word about $\mathbf{H}_\mathrm{F}$. As the outer-product Hessian is positive semi-definite, the functional Hessian is the source of all the negative eigenvalues of the Hessian and is important for optimization as there may be numerous saddles in the landscape (Dauphin et al., 2014). It also has a very peculiar block-hollow structure (i.e., zero diagonal blocks), which leads to the number of negative eigenvalues being approximately half of its rank (c.f., (Singh et al., 2021)).

6. Key results on the CNN Hessian Rank

Finally, we can now present our key results. A quick note about assumptions: for simplicity, assume that the (uncentered) input-covariance $\boldsymbol{\Sigma}_{\mathbf{xx}} = \mathrm{cov}(\mathbf{x})$ has full rank $\mathrm{rk}\left(\boldsymbol{\Sigma}_{\mathbf{xx}}\right) := r = d$. We will analyze the ranks of the outer product and functional Hessian, later combining them to yield a bound on the rank of the loss Hessian. This is without loss of generality for one can always pre-process the input to ensure that this is the case, and results of (Singh et al., 2021) hold for $r \neq d$ with appropriate modifications.

Outer-Product Hessian $\mathbf{H}_\mathrm{O}$. From the structure of the kl-th block in Eqn. (5), it's easy to see to arrive at the following proposition:

Proposition 4. *For a deep linear convolutional network, $\mathbf{H}_\mathrm{O} = \mathbf{Q}_o^\top\mathbf{A}_o\mathbf{B}_o\mathbf{A}_o^\top\mathbf{Q}_o$, where $\mathbf{B}_o = \mathbf{I}_K\otimes\boldsymbol{\Sigma}_{\mathbf{xx}} \in \mathbb{R}^{Kd\times Kd}$,*

$$\mathbf{A}_o^\top = \begin{pmatrix} \mathbf{T}^{(L+1:2)}\otimes\mathbf{I}_d \\ \vdots \\ \mathbf{T}^{(L+1:l+1)}\otimes\mathbf{T}^{(1:l-1)} \\ \cdots \\ \mathbf{I}_K\otimes\mathbf{T}^{(1:L)}\end{pmatrix} \in \mathbb{R}^{Kd\times\widehat{p}}\,,$$

and $\mathbf{Q}_o = \mathrm{diag}\Big(\widetilde{\mathbf{Q}}^{(1)},\cdots,\widetilde{\mathbf{Q}}^{(L+1)}\Big) \in \mathbb{R}^{\widehat{p}\times p}$ where $\mathrm{diag}(\cdot)$ denotes a block-diagonal matrix.

Besides, $p = \sum_{i=1}^{L+1} m_l m_{l-1} k_l$ and $\widehat{p} = \sum_{i=1}^{L+1} m_l m_{l-1} d_l d_{l-1}$. The former denotes the number of parameters in the CNN while the latter is the number of parameters in the 'Toeplitzed' fully-connected network.

Our first key result, the proof of which is located in the Appendix section C.1, can then be described as follows:

Theorem 5. *The rank of the outer-product Hessian is upper bounded as*

$$\begin{aligned}\mathrm{rk}(\mathbf{H}_\mathrm{O}) &\le \min\big(p, d_0\,\mathrm{rk}(\mathbf{T}^{(2:L+1)}) + K\,\mathrm{rk}(\mathbf{T}^{(L:1)}) - \mathrm{rk}(\mathbf{T}^{(2:L+1)})\,\mathrm{rk}(\mathbf{T}^{(L:1)})\big)\\ &= \min\left(p, q\left(d_0 + K - q\right)\right)\,.\end{aligned}$$

Here, $q := \min(d_0, m_1d_1,\cdots,m_Ld_L,K)$.

Assuming no bottleneck layer, we will have that $q = \min(d, K)$, and resulting in $\mathrm{rk}(\mathbf{H}_\mathrm{O}) \le Kd_0$.

Functional Hessian $\mathbf{H}_\mathrm{F}$. Our approach here will be similar to that in the Theorem above. We will try to factor out all the $\mathbf{Q}^{(l)}$ matrices and then analyze the rank of the resulting decomposition. But, this requires more care as the form of the kl-th block is different depending on $k \le l$ or not.

Theorem 6. *For a deep linear convolutional network, the rank of l-th column-block, $\widehat{\mathbf{H}}_\mathrm{F}^{\bullet l}$, of the matrix $\widehat{\mathbf{H}}_\mathrm{F}$, can be upper bounded as*

$$\mathrm{rk}(\widehat{\mathbf{H}}_\mathrm{F}^{\bullet l}) \le \min(\widehat{q}\,m_{l-1}d_{l-1} + \widehat{q}\,m_ld_l - \widehat{q}^{\,2}\,, m_lm_{l-1}k_l)\,,$$

for $l \in [2,\cdots,L]$. When $l=1$, we have

$$\mathrm{rk}(\widehat{\mathbf{H}}_\mathrm{F}^{\bullet 1}) \le \min(\widehat{q}\,m_1d_1 + \widehat{q}\,s - \widehat{q}^{\,2}\,, m_1m_0k_1)\,.$$

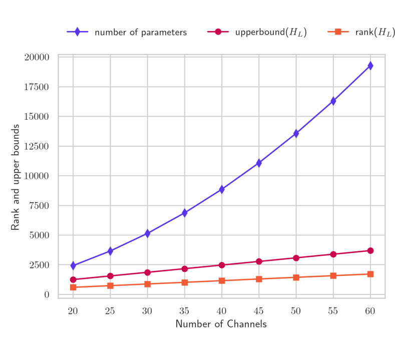
And, when $l = L+1$, we have

$$\mathrm{rk}(\widehat{\mathbf{H}}_\mathrm{F}^{\bullet L+1}) \le \min(\widehat{q}\,m_Ld_L + \widehat{q}\,s - \widehat{q}^{\,2}\,, m_{L+1}m_Lk_{L+1})\,.$$

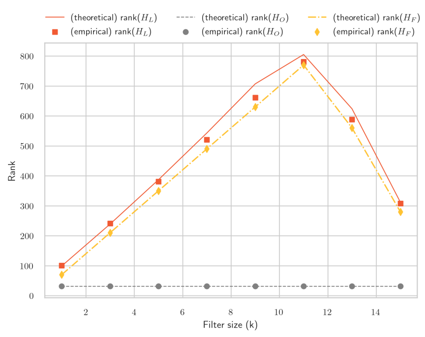
Here, $\widehat{q} := \min(d_0, m_1d_1,\cdots,m_Ld_L,K,s) = \min(q,s)$ and $s := \mathrm{rk}(\boldsymbol{\Omega}) = \mathrm{rk}(\mathbf{E}\left[\boldsymbol{\delta}_{\mathbf{x},\mathbf{y}}\,\mathbf{x}^\top\right])$.

The proof is located in Section C.2 of the Appendix. The upper bound on the rank of $\mathbf{H}_\mathrm{F}$ follows by summing the above result over all the columns, $\mathrm{rk}(\mathbf{H}_\mathrm{F}) \le \sum_{l=1}^{L+1}\mathrm{rk}(\widehat{\mathbf{H}}_\mathrm{F}^{\bullet l})$ by the sub-additivity of rank. A remarkable empirical observation, like in the case of FCNs, is that the block-columns are mutually orthogonal — hence, we don't loose anything by simply summing the ranks of the block columns.

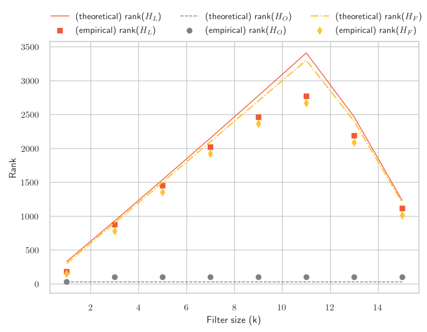
Loss Hessian $\mathbf{H}_\mathrm{L}$. One can then bound the rank of the loss



(a) Linear CNN: Rank vs # Channels



(b) Linear CNN: Rank vs Filter size



(c) ReLU CNN: Rank vs Filter size

Figure 2: Trend of the **upper bound** on the (loss) Hessian rank, compared to the **true rank**, and the **number of parameters**. In other subfigures, we see the rank of the functional Hessian and outer-product Hessian.

Hessian simply as, $\mathrm{rk}(\mathbf{H}_\mathrm{L}) \leq \mathrm{rk}(\mathbf{H}_\mathrm{O}) + \mathrm{rk}(\mathbf{H}_\mathrm{F})$. Also, we can infer that, in the likely case where $q = \min(K, d_0)$, rank of the loss Hessian grows linearly with number of channels, i.e., $\mathcal{O}(m \cdot L \cdot d_0)$, while the number of parameters grow quadratically in the number of channels, i.e., $\mathcal{O}(m^2 \cdot L \cdot d_0)$. Thereby, we confirm that like FCNs, a similar linear trend also holds for CNNs (and hence the Figure 1). Besides, for very large networks, m will be the dominating factor and we can infer that rank will show a square root behaviour relative to the number of parameters. Hence, we generalize the key finding of (Singh et al., 2021) in the fully-connected case to the case of convolutional neural networks.

6.1. Exact results for one-hidden layer case

While the above bounds are much tighter than any existing bounds, we now try to understand how tight our bounds are by looking at the case of a 1-hidden layer:

Theorem 7. *The rank of the outer product Hessian for a one-hidden layer CNN can be bounded as:* $\mathrm{rk}\left(\mathbf{H}_\mathrm{O}\right) \leq \min\big(Kd_0, q(k_1 + K(d_0 - k_1 + 1) - q)\big)$ *, where,* $q = \min(k_1, K(d_0 - k_1 + 1), m_1)$.

Theorem 8. *The rank of the functional Hessian for a one-hidden layer CNN can be bounded as:* $\mathrm{rk}\left(\mathbf{H}_\mathrm{F}\right) \leq 2\min\big(k_1, K(d_0 - k_1 + 1)\big)\, m_1$.

The strategy behind the proofs (section C.3), which is strictly novel relative to (Singh et al., 2021), is to express the CNN as a superposition of functions that act on different input patches and, alongside, utilize the form of Toeplitz derivatives as well as the involved auxiliary permutation matrices. In terms of the results, interestingly, we find that now the filter size k_1 has entered inside the min terms in each of the bounds; thereby further reducing the rank. However, as shown ahead, for larger networks with many channels m, we find that our earlier upper bound still fares decently.

7. Empirical Verification

Verification of upper bounds. We empirically validate our upper bounds in a variety of settings, in particular, with both linear and ReLU activations, MSE and CE loss, as well as on datasets such as CIFAR10, FashionMNIST, MNIST, and a synthetic dataset. However, given the page constraints, we only show a selection of the plots, while the rest can be found in the Appendix D. Following (Singh et al., 2021), to rigorously illustrate the match with our bounds, we compute the exact Hessians, without approximations, and in Float64 precision. These precautions are taken to avoid any imprecision in calculating the rank, since the boundary of non-zero eigenvalues with zero can be otherwise a bit blurry.

Rank vs number of channels. We begin by illustrating the trend of our general upper bounds in Figure 2a for the case of 2-hidden layer CNN on CIFAR10 and is thus the counterpart to the Figure 1 presented before. The upper bound is relatively close to the true rank and similarly shows a linear trend with the number of channels. Similar results for ReLU and CE loss can be found in Appendix D.2.

Rank vs Filter Size. Next, in Figure 2b, we demonstrate the match of our exact bound with the rank as observed empirically. We hold the number of channels as fixed and vary the filter size. First of all, here the markers and the lines indeed coincide for the functional Hessian and outer-product Hessian. On the other hand, the loss Hessian which we bound as the sum of the ranks of $\mathbf{H}_\mathrm{O}$ and $\mathbf{H}_\mathrm{F}$ forms a canopy over all the empirically observed values of the rank across the filter sizes. The upper bound itself is also quite close; it becomes even closer for large values of m, see Section D.3.1).

The case of non-linearities. Further, in a similar comparison across filter sizes, we showcase that our bounds which were derived for linear activations still remain as valid upper bounds and form a canopy as seen in Figure 2c.

Furthermore, in order to quantitatively measure the fidelity of our rank bounds in the case of non-linearities, we report the Hessian reconstruction error in Figure 3 as done in (Singh et al., 2021). To be precise, here we measure the reconstruction error for a four-hidden layer CNN with ReLU non-linearity when only the some of top eigenvectors are used to approximate the Hessian. We can see that using our upper bound (from the linear case) still provides for a rather small reconstruction error. Hence, this provides evidence that our upper bounds we derived in the linear case can serve as a proxy to the numerical rank in the case with non-linearities.

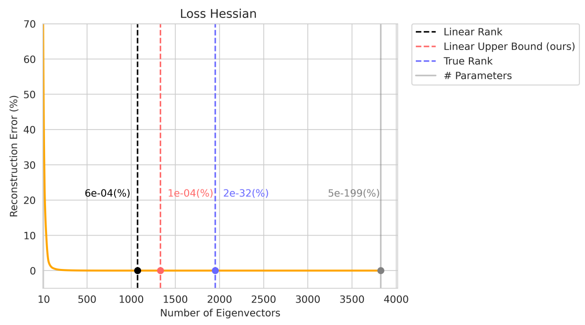


Figure 3: Hessian reconstruction error for four-hidden layer CNN with ReLU non-linearity. 'Linear Rank' corresponds to using the true (empirically measured) rank in the corresponding CNN with linear activation. 'Linear Upper Bound' refers to our upper bound for the corresponding CNN with linear activations. The 'True Rank' refers to the empirically measured rank in this particular CNN with ReLU nonlinearity.

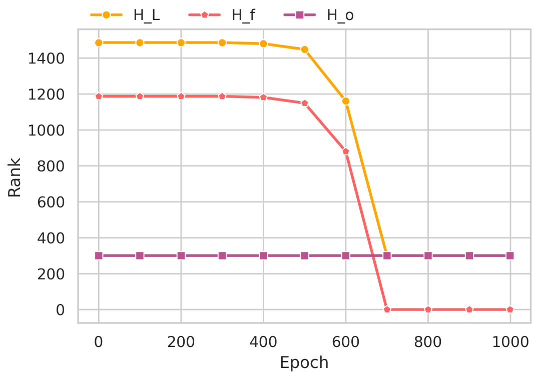


Figure 4: Dynamics of the rank of the Hessian while training a four-hidden layer linear CNN.

The course of training. Like the fully-connected case, as shown in the past work of (Singh et al., 2021), we observe in Figure 4 that CNNs show similar trend in the evolution of Hessian rank during the course of training. In particular, we can see that the rank of the functional Hessian which is learning driven (i.e., contains the residual-input covariance), decreases during training before eventually becoming zero. In contrast, the rank of the outer-product Hessian is unchanged. Thus, the rank of the overall loss Hessian decreases until it becomes the same as the outer-product Hessian, and in turn matches it in rank[8].

8. Locality and Weight-sharing

We would like to do a little deep dive and explore a simplified setting, in order to gather some intuition. We study the two crucial components behind CNN, namely, local connectivity (i.e, the localized patches of the input are convolved with a possibly distinct filter) and weight sharing (where the same filter is applied to each of these patches).

Setting. Concretely, let us begin by considering the case of a one-hidden layer neural network (with linear activations for simplicity). We now try to get rid of the layer indices for the sake of clarity. So the input dimension is d, output dimension is K, filter size is k, number of hidden layer filters is m. Further, we consider non-overlapping filters, like in MLPMixer (Tolstikhin et al., 2021). In other words, the stride is set equal to the filter size k and so the number of patches under consideration will be $t = d/k$, and to avoid unnecessary complexity, we assume d is an integer multiple of k.

Locally Connected Networks (LCNs). As the filters that get applied on the patches are distinct, let us denote them by $\mathbf{W}^{(ii)} \in \mathbb{R}^{m\times k}$ and the output layer matrix as $\mathbf{V} := \begin{pmatrix}\mathbf{V}^{(1)} & \cdots & \mathbf{V}^{(t)}\end{pmatrix}$, with $\mathbf{V}^{(i)} \in \mathbb{R}^{K\times m}$, where the superscript denotes the index of the local patch upon which they get applied. Mathematically we can represent the resulting neural network function as (where $\mathbf{x}^{(i)} \in \mathbb{R}^k$ denotes the i-th chunk of the input):

$$\begin{pmatrix}\mathbf{V}^{(1)} & \cdots & \mathbf{V}^{(t)}\end{pmatrix}\begin{pmatrix}\mathbf{W}^{(11)} & \cdots & \mathbf{0}\\ \vdots & \ddots & \vdots\\ \mathbf{0} & \cdots & \mathbf{W}^{(tt)}\end{pmatrix}\begin{pmatrix}\mathbf{x}^{(1)}\\ \vdots\\ \mathbf{x}^{(t)}\end{pmatrix}$$

We indexed the local weight matrices of the first layer as $\mathbf{W}^{(ii)}$ to reflect that it is (i,i)-th block on the diagonal. After carrying out the block matrix multiplication, the above formulation can also be expressed as:

$$\mathrm{F}^{\text{LCN}}_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{i=1}^{t} \mathbf{V}^{(i)}\,\mathbf{W}^{(ii)}\,\mathbf{x}^{(i)}\,. \tag{7}$$

We can then easily bring to our minds the case of fully-connected networks which will also contain the off-diagonal blocks and would be represented as:

$$\mathrm{F}^{\text{FCN-large}}_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{i,j=1}^{t} \mathbf{V}^{(i)}\,\mathbf{W}^{(ij)}\,\mathbf{x}^{(j)}\,.$$

[8]Besides, we notice in Figure 16a, that the rank of the convolutional layers (when matricized) remains unchanged during training as well.

Clearly, fully-connected networks form a generalization of locally-connected networks. Another point of comparison for LCNs in Eq. (7) with FCNs is that the former may be viewed as a superposition of s distinct *smaller* FCNs acting on disjoin patches of the input.[9]

$$\mathrm{F}^{\mathrm{LCN}}_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{i=1}^{t} \mathrm{F}^{\text{FCN-small}}_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\,;\,\{\mathbf{V}^{(i)},\mathbf{W}^{(ii)}\}\right).$$

In this scenario of LCNs, we get the following bounds on the rank of the outer-product and functional Hessian:

Theorem 9. *For the locally connected network as described in Eqn. (7), the rank of the outer-product and the functional Hessian can be upper bounded as follows:* $\mathrm{rk}(\mathbf{H}_{\mathrm{O}}^{LCN}) \le t \cdot q(k+K-q)$ *, and* $\mathrm{rk}(\mathbf{H}_{\mathrm{F}}{}^{LCN}) \le t\cdot 2m\min(k,K)$*, where* $q=\min(k,K,m)$ *and* $t=d/k$*.*

The proof is located in Section C.4. The neat thing about the above rank expressions is that they are identical to that obtained for the smaller fully-connected network, except where we change the input dimension from $d \to k$ and scale the bounds by a factor of $t = d/k$ (the number of smaller fully-connected that are being superpositioned). In short, even though these weight matrices must 'act' together in the loss, their contributions to the Hessian came out individually (i.e, $\mathbf{H}_{\mathrm{F}}$ and $\mathbf{H}_{\mathrm{O}}$ can be factorized as block diagonals).

Incorporating Weight Sharing (WS). Now all the weight matrices in the first layer are shared, i.e., $\mathbf{W}^{(ii)} = \mathbf{W}$.

$$\begin{pmatrix}\mathbf{V}^{(1)} & \cdots & \mathbf{V}^{(t)}\end{pmatrix}\begin{pmatrix}\mathbf{W} & \cdots & \mathbf{0}\\ \vdots & \ddots & \vdots\\ \mathbf{0} & \cdots & \mathbf{W}\end{pmatrix}\begin{pmatrix}\mathbf{x}^{(1)}\\ \vdots\\ \mathbf{x}^{(t)}\end{pmatrix} \tag{8}$$

Theorem 10. *For the locally connected network with weight sharing defined in Eqn. (8), the rank of the outer-product and the functional Hessian can be bounded as:* $\mathrm{rk}(\mathbf{H}_{\mathrm{O}}^{LCN+WS}) \le q(k+Kt-q)$ *, and* $\mathrm{rk}(\mathbf{H}_{\mathrm{F}}{}^{LCN+WS}) \le 2m\min(k,Kt)$*, where* $q=\min(k,Kt,m)$ *and* $t=d/k$*.*

The proof can be found in Section C.5, but the intuition is that the weight matrix $\mathbf{W}$ is shared across the $\mathbf{V}^{(i)}$, resulting in the intersection of their column space inside the Hessian. Hence, the rank shrinks for both $\mathbf{H}_{\mathrm{O}}$, $\mathbf{H}_{\mathrm{F}}$. Comparing the $\mathrm{rk}(\mathbf{H}_{\mathrm{F}})$ bounds, we see that here t slides inside the minimum, but only in the second term (i.e., K).

Lastly, in Figure 5, we present an illustration of the trend of the rank with increasing filter size for both LCN and the LCN + WS variant. Besides, the interesting scaling behaviour, this figure also validates our theoretical bounds.

[9]This superposition point of view would hold even if we had a non-linearity, and so do our empirical results.

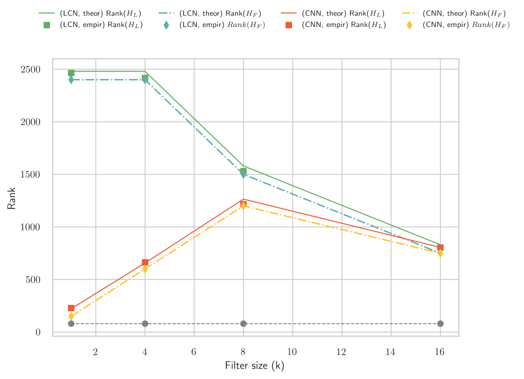


Figure 5: *Hessian rank: LCN vs CNN, (Linear, CIFAR10).*

9. Conclusion

All in all, we have illustrated how the key ingredients of CNNs, such as local connectivity and weight sharing, as well as architectural aspects like filter size, strides, and number of channels get manifested through the Hessian structure and rank. Moreover, we can utilize our Toeplitz representation framework to deliver tight upper bounds in the general case of deep convolutional networks and generalize the recent finding of (Singh et al., 2021) about square root growth of rank relative to parameter count for CNNs as well. Looking ahead, our work raises some very interesting questions: (a) Is the growth of rank as a square root in terms of the number of parameters a universal characteristic of all deep architectures? Including Transformers? Or are there some exceptions to it? (b) Given the uncovered structure of the Hessian in CNNs, there are also questions about understanding which parts of the architecture affect the spectrum more — normalization or pooling layers? (c) On the application side, it would be interesting to see if we can use our results to understand the approximation quality of existing pre-conditioners such as K-FAC or come up with better ones, given the rich properties of Toeplitz matrices.

Acknowledgements

We would like to thank Max Daniels and Gregor Bachmann for their suggestions as well as rest of DALab for useful comments and support. Sidak Pal Singh would also like to acknowledge the financial support from Max Planck ETH Center for Learning Systems and the travel support from ELISE (GA no 951847).

References

Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine learning practice and the bias-variance trade-off, 2019.

Bruna, J. and Mallat, S. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

Chuai, J. and Tian, Y. Rank equalities and inequalities for kronecker products of matrices with applications. *Applied Mathematics and Computation*, 150(1):129–137, 2004. ISSN 0096-3003. doi: https://doi.org/10.1016/S0096-3003(03)00203-0. URL https://www.sciencedirect.com/science/article/pii/S0096300303002030.

Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability, 2021. URL https://arxiv.org/abs/2103.00065.

Cohen, N. and Shashua, A. Inductive bias of deep convolutional networks through pooling geometry. *arXiv preprint arXiv:1605.06743*, 2016.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Fang, Z., Feng, H., Huang, S., and Zhou, D.-X. Theory of deep convolutional neural networks ii: Spherical analysis. *Neural Networks*, 131:154–162, 2020.

Fukushima, K. A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol, Cybern*, 36:193–202, 1980.

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. Deep convolutional networks as shallow gaussian processes. *arXiv preprint arXiv:1808.05587*, 2018.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Gu, Y., Zhang, W., Fang, C., Lee, J. D., and Zhang, T. How to characterize the landscape of overparameterized convolutional neural networks. *Advances in Neural Information Processing Systems*, 33:3797–3807, 2020.

Gull, S. F. *Developments in Maximum Entropy Data Analysis*, pp. 53–71. Springer Netherlands, Dordrecht, 1989. ISBN 978-94-015-7860-8. doi: 10.1007/978-94-015-7860-8_4. URL https://doi.org/10.1007/978-94-015-7860-8_4.

Gunasekar, S., Woodworth, B., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10. IEEE, 2018.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace, 2018.

Hassibi, B. and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, pp. 164–171, 1992.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural computation*, 9(1):1–42, 1997.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Jia, Z. and Su, H. Information-theoretic local minima characterization and regularization. In *International Conference on Machine Learning*, pp. 4773–4783. PMLR, 2020.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., and Inman, D. J. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Kohn, K., Merkh, T., Montúfar, G., and Trager, M. Geometry of linear convolutional networks, 2021. URL https://arxiv.org/abs/2108.01538.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Kunstner, F., Hennig, P., and Balles, L. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

LeCun, Y., Simard, P., and Pearlmutter, B. A. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In *NIPS 1992*, 1992.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992a.

MacKay, D. J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3): 448–472, 05 1992b. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448. URL https://doi.org/10.1162/neco.1992.4.3.448.

Maddox, W. J., Benton, G., and Wilson, A. G. Rethinking parameter counting: Effective dimensionality revisted. *arXiv preprint arXiv:2003.02139*, 2020.

Magnus, J. R. and Neudecker, H. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

Mao, T., Shi, Z., and Zhou, D.-X. Theory of deep convolutional neural networks iii: Approximating radial functions, 2021. URL https://arxiv.org/abs/2107.00896.

Martens, J. and Grosse, R. B. Optimizing neural networks with kronecker-factored approximate curvature. *CoRR*, abs/1503.05671, 2015. URL http://arxiv.org/abs/1503.05671.

Matsaglia, G. and Styan, G. P. H. Equalities and inequalities for ranks of matrices. *Linear and Multilinear Algebra*, 2(3):269–292, 1974. doi: 10.1080/03081087408817070. URL https://doi.org/10.1080/03081087408817070.

Nguyen, Q. and Hein, M. Optimization landscape and expressivity of deep cnns. In *International conference on machine learning*, pp. 3730–3739. PMLR, 2018.

Papyan, V. Traces of class/cross-class structure pervade deep learning spectra. *The Journal of Machine Learning Research*, 21(1):10197–10260, 2020.

Pennington, J. and Bahri, Y. Geometry of neural network loss surfaces via random matrix theory. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2798–2806. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/pennington17a.html.

Poggio, T., Anselmi, F., and Rosasco, L. I-theory on depth vs width: hierarchical function composition. Technical report, Center for Brains, Minds and Machines (CBMM), 2015.

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

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Schaul, T., Zhang, S., and LeCun, Y. No more pesky learning rates, 2013.

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14:1723–1738, 2002a.

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002b.

Sedghi, H., Gupta, V., and Long, P. M. The singular values of convolutional layers, 2018.

Setiono, R. and Hui, L. C. K. Use of a quasi-newton method in a feedforward neural network construction algorithm. *IEEE Transactions on Neural Networks*, 6(1):273–277, 1995.

Singh, R. P. Some generalizations in matrix differentiation with applications in multivariate analysis. 1972.

Singh, S. P. and Alistarh, D. Woodfisher: Efficient second-order approximation for neural network compression, 2020.

Singh, S. P., Bachmann, G., and Hofmann, T. Analytic insights into structure and rank of neural network hessian maps. *Advances in Neural Information Processing Systems*, 34, 2021.

Singh, S. P., Lucchi, A., Hofmann, T., and Schölkopf, B. Phenomenology of double descent in finite-width neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=lTqGXfn9Tv.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

Tracy, D. S. and Dwyer, P. S. Multivariate maxima and minima with matrix derivatives. *Journal of the American Statistical Association*, 64(328):1576–1594, 1969.

Vapnik, V. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, Y., Zhu, X., Wu, C., Wang, A., and Ge, R. Dissecting hessian: Understanding common structure of hessian in neural networks. *arXiv preprint arXiv:2010.04261*, 2020.

Yang, J., Sun, S., and Roy, D. M. Fast-rate pac-bayes generalization bounds via shifted rademacher processes, 2019.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pp. 581–590. IEEE, 2020.

Zhao, L., Liao, S., Wang, Y., Li, Z., Tang, J., and Yuan, B. Theoretical properties for neural networks with weight matrices of low displacement rank. In *international conference on machine learning*, pp. 4082–4090. PMLR, 2017.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.

A. Tools

A.1. Toeplitz Derivative

Lemma 1. *The matrix derivative of $\mathbf{T}^{(l)}$ with respect to $\mathbf{W}^{(l)}$, is given as follows:*

$$\frac{\partial \mathbf{T}^{(l)}}{\partial \mathbf{W}^{(l)}} := \frac{\partial \operatorname{vec}_r \mathbf{T}^{(l)}}{\partial \left(\operatorname{vec}_r \mathbf{W}^{(l)}\right)^\top} = \mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)} \,,$$

which lives in $\mathbb{R}^{m_l\, m_{l-1}\, d_\ell\, d_{l-1} \times m_l\, m_{l-1}\, k_l}$, and the matrix $\mathbf{Q}^{(l)} \in \mathbb{R}^{m_{l-1} d_l d_{l-1} \times m_{l-1} k_l}$ is defined as:

$$\mathbf{Q}^{(l)} := \begin{pmatrix} \mathbf{I}_{m_{l-1}} \otimes \left(\pi_R^0\, \mathbf{I}_{d_{l-1}\times k_l}\right) \\ \vdots \\ \mathbf{I}_{m_{l-1}} \otimes \left(\pi_R^{d_{l-1}-k_l}\, \mathbf{I}_{d_{l-1}\times k_l}\right) \end{pmatrix}$$

where π_R is the permutation matrix performs clockwise rotation of the rows of the matrix it is left-multiplied with, and its superscript the matrix power (so, $\pi_R^0 = \mathbf{I}_{d_{l-1}}$). Also, in the above expression $\mathbf{I}_{d_{l-1}\times k_l}$ is the 'tall' identity matrix, i.e., $[\mathbf{I}_{k_l}, \mathbf{0}]^\top$ (as $k_l \le d_{l-1}$).

Proof. It is quite clear that the non-zero parts in the derivative will arise only when compute the derivative of Toeplitz of one row with respect to the elements of the same row. In other words, only when considering:

$$\frac{\partial \mathbf{T}^{\mathcal{W}^{(l)}_{(i,j)\,\bullet}}}{\partial \mathcal{W}^{(l)}_{(r,s)\,\bullet}} \qquad \text{for } i = r, \quad j = s\,.$$

And rest of the blocks will be zeros. This explains the occurrence of two Kronecker products, with respect to $\mathbf{I}_{m_l}$ and $\mathbf{I}_{m_{l-1}}$.

More concretely, recall that:

$$\mathbf{T}^{(l)} := \begin{pmatrix} \mathbf{T}^{\mathcal{W}^{(l)}_{(1,1)\,\bullet}} & \cdots & \mathbf{T}^{\mathcal{W}^{(l)}_{(1,m_{l-1})\,\bullet}} \\ \vdots & & \vdots \\ \mathbf{T}^{\mathcal{W}^{(l)}_{(m_l,1)\,\bullet}} & \cdots & \mathbf{T}^{\mathcal{W}^{(l)}_{(m_l,m_{l-1})\,\bullet}} \end{pmatrix}, \quad \text{and}$$

$$\begin{aligned} \mathbf{W}^{(l)} := \operatorname{mat} \mathcal{W}^{(l)} &:= \begin{pmatrix} \mathcal{W}^{(l)}_{(1,1)\,\bullet} & \cdots & \mathcal{W}^{(l)}_{(m_l,1)\,\bullet} \\ \vdots & & \vdots \\ \mathcal{W}^{(l)}_{(1,m_{l-1})\,\bullet} & \cdots & \mathcal{W}^{(l)}_{(m_l,m_{l-1})\,\bullet} \end{pmatrix}^\top \\ &= \begin{pmatrix} w_{1,1,1} & \cdots & w_{1,1,k_l} & \cdots & w_{1,m_{l-1},1} & \cdots & w_{1,m_{l-1},k_l} \\ \vdots & & \vdots & & \vdots & & \vdots \\ w_{m_l,1,1} & \cdots & w_{m_l,1,k_l} & \cdots & w_{m_l,m_{l-1},1} & \cdots & w_{m_l,m_{l-1},k_l} \end{pmatrix}. \end{aligned}$$

Let's use the shorthand $\mathbf{T}^{(l)}_{(i,\bullet)} := \left(\mathbf{T}^{\mathcal{W}^{(l)}_{(i,1)\,\bullet}} \;\; \ldots \;\; \mathbf{T}^{\mathcal{W}^{(l)}_{(i,m_{l-1})\,\bullet}}\right)$. Now, the general structure of the required Toeplitz derivative has the following form:

$$\frac{\partial \mathbf{T}^{(l)}}{\partial \mathbf{W}^{(l)}} = \begin{array}{c|ccc} & \operatorname{vec}_r \mathbf{W}^{(l)}_{1\,\bullet} & \cdots & \operatorname{vec}_r \mathbf{W}^{(l)}_{m_l\,\bullet} \\ \hline \operatorname{vec}_r \mathbf{T}^{(l)}_{(1,\bullet)} & \mathbf{Q}^{(l)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{vec}_r \mathbf{T}^{(l)}_{(m_l,\bullet)} & \mathbf{0} & \cdots & \mathbf{Q}^{(l)} \end{array} = \mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)}. \tag{9}$$

The diagonal blocks in the above matrix are identical, for if the same weight is located in the Toeplitz representation with respect to which we differentiate, we would get a 1 else a 0. Besides, it should also be noted that $\mathbf{Q}^{(l)}$ is just a binary matrix (i.e., contains either 0 or 1), with atmost a single 1 per row.

Next, each of the $d_l = d_{l-1} - k_l + 1$ rows of $\mathbf{T}^{(l)}_{(i,\bullet)}$ is a clockwise rotation of its first row. Thus the derivative of each of its row will depend upon the amount of clockwise rotation shift, and to get the derivative of the entire row block $\mathbf{T}^{(l)}_{(i,\bullet)}$ we will just need to stack rowwise each of these obtained derivatives with respect to $\mathbf{W}^{(l)}_{i\,\bullet}$. Since, all the diagonal blocks are identical, without loss of generality, let's assume $i = 1$ and consider the derivative of the *first row* of $\mathbf{T}^{(l)}_{(1,\bullet)}$ with respect to $\mathbf{W}^{(l)}_{1\,\bullet}$.

The derivative will be zero whenever we try to a differentiate the Toeplitz representation of one input channel with respect to parameters of a different input channel. Hence, the form of this derivative will be $\mathbf{I}_{m_{l-1}} \otimes \mathbf{A}$ for some matrix $\mathbf{A} \in \mathbb{R}^{d_{l-1}\times k_l}$. For the derivative of the first row, $\mathbf{A} = \pi_R^0 \mathbf{I}_{d_{l-1}\times k_l}$, while for the derivative of the j-th row it will be $\mathbf{A} = \pi_R^{j-1}\mathbf{I}_{d_{l-1}\times k_l}$, for $j \in [1, \cdots, d_l]$. Here, by π_R^j we mean taking the j-th matrix power of the following matrix, π_R (which performs clockwise rotation of the rows of matrix it left-multiplies):

$$\pi_R = \begin{pmatrix} 0 & \cdots & & 0 & 1 \\ 1 & 0 & \cdots & & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix},$$

and $\mathbf{I}_{m\times n} \in \mathbb{R}^{m\times n}$, for $m > n$, denotes the tall identity matrix $[\mathbf{I}_{n\times n}, \mathbf{0}_{(m-n)\times n}]^\top$, which is padded by $m - n$ rows of all zeros. This tall identity matrix gets post-multiplied to the clockwise rotation matrix, since we only require the first k_l columns of it.

Therefore, the form of $\mathbf{Q}^{(l)}$ is given as follows:

$$\mathbf{Q}^{(l)} := \begin{pmatrix} \mathbf{I}_{m_{l-1}} \otimes \left(\pi_R^0\, \mathbf{I}_{d_{l-1}\times k_l}\right) \\ \vdots \\ \mathbf{I}_{m_{l-1}} \otimes \left(\pi_R^{d_{l-1}-k_l}\, \mathbf{I}_{d_{l-1}\times k_l}\right) \end{pmatrix}$$

□

Remark. Note that the above matrix is equivalent, upto row permutations, to the following more concisely formulated matrix:

$$\mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)} \equiv \mathbf{I}_{m_l m_{l-1}} \otimes \pi_R^{0\downarrow d_{l-1}-k_l}\, \mathbf{I}_{d_{l-1}\times k_l},$$

where, $\pi_R^{0\downarrow d_{l-1}-k_l} := [\pi_C^0, \pi_C^1, \cdots, \pi_C^{d_{l-1}-k_l}]^\top$ and similar to π_R, $\pi_C := \pi_R^\top$ performs clockwise rotation of the columns of the matrix it right-multiplies.

A.2. Technical Background

A.2.1. Helper Lemmas

Lemma 11. *Let $\mathbf{X} \in \mathbb{R}^{m\times n}$ and $\mathbf{Y} \in \mathbb{R}^{p\times q}$. Then the row-partitioned matrix $\begin{bmatrix} \mathbf{I}_q \otimes \mathbf{X} \\ \mathbf{Y} \otimes \mathbf{I}_n \end{bmatrix}$ has the rank,*

$$\operatorname{rk} \begin{bmatrix} \mathbf{I}_q \otimes \mathbf{X} \\ \mathbf{Y} \otimes \mathbf{I}_n \end{bmatrix} = q\operatorname{rk}(\mathbf{X}) + n\operatorname{rk}(\mathbf{Y}) - \operatorname{rk}(\mathbf{X})\operatorname{rk}(\mathbf{Y})$$

We defer the reader to Chuai & Tian (2004); Singh et al. (2021) for the proof of this lemma.

A.2.2. Auxiliary Matrices

To get tight bounds for convolutional networks, one has to 'play' around with permutation matrices. So, to this end, let us introduce the auxiliary matrices from (Singh, 1972; Tracy & Dwyer, 1969), denoted[10] by $\mathbf{P}_{(m)} \in \mathbb{R}^{mn\times mn}$ and defined as a rearrangement of the identity $\mathbf{I}_{mn}$ by taking every m-th row from the first, then every m-th row from the second etc. Likewise, one can define an rearrangement of the identity by taking every m-th column from the first, and so on, and this will be denoted by $\mathbf{P}^{(m)}$. The following are some of the essential properties of this auxiliary matrix:

- $\mathbf{P}_{(m)} = \mathbf{P}_{(n)}^\top$
- $\mathbf{P}_{(m)}\mathbf{P}_{(n)} = \mathbf{P}_{(n)}\mathbf{P}_{(m)} = \mathbf{I}_{mn}$
- $\mathbf{P}_{(m)} = \mathbf{P}^{(n)}, \mathbf{P}_{(n)} = \mathbf{P}^{(m)}$

In other contexts, this is also known as the commutation matrix (Magnus & Neudecker, 2019). In particular $K_{(m,n)} \in \mathbb{R}^{mn\times mn}$ is a matrix partitioned into $m \times n$ blocks such that ij-th block has ji-th entry 1 and rest all 0. One can check that $\mathbf{P}_{(m)} = \mathbf{K}_{(n,m)}$. For ease of understanding, we will use auxiliary matrices $\mathbf{P}_{(m)}$, but if needed we rely upon the extensive results obtained for it under the name of commutation matrices.

A.2.3. Frequently used properties of Kronecker Products

Often in the analysis, we use several properties of Kronecker products, and we cannot afford to reiterate them at every step of the proof. Hence, if a reader feels a bit uneasy about a particular step, we recommend they consult some of the properties mentioned here. The proofs are readily available online, or check (Magnus & Neudecker, 2019) as a good reference for matrix analysis.

For conformable matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{X}$, we have that:

- $(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = \mathbf{AB}\otimes\mathbf{CD}$
- $\mathrm{rk}(\mathbf{A}\otimes\mathbf{B}) = \mathrm{rk}(\mathbf{A})\cdot\mathrm{rk}(\mathbf{B})$
- $(\mathbf{A}\otimes\mathbf{B})\otimes\mathbf{C} = \mathbf{A}\otimes(\mathbf{B}\otimes\mathbf{C})$
- $(\mathbf{A}\otimes\mathbf{B})^\top = \mathbf{A}^\top\otimes\mathbf{B}^\top$
- $\mathrm{vec}_r(\mathbf{AXB}) = (\mathbf{A}\otimes\mathbf{B}^\top)\,\mathrm{vec}_r(\mathbf{X})$

For column vectors $\mathbf{a}, \mathbf{b}$, we have that: $\mathbf{a}\otimes\mathbf{b} = \mathrm{vec}_r(\mathbf{a}\mathbf{b}^\top)$.

B. CNN Hessian Structure

B.1. Outer-Product Hessian

In particular, since $\nabla^2_{\mathrm{F}_{\boldsymbol\theta}}\,\ell_i = \mathbf{I}_K\ ,\ \forall i\in[n]$, the choice of MSE implies that the outer product Hessian simplifies to:

$$\mathbf{H}_{\mathrm{O}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol\theta}\mathrm{F}_{\boldsymbol\theta}(\mathbf{x}_i)\,\nabla_{\boldsymbol\theta}\mathrm{F}_{\boldsymbol\theta}(\mathbf{x}_i)^\top$$

Proposition 2. *The kl-th block of the outer-product Hessian is given by,*

$$\mathbf{H}_{\mathrm{O}}^{(kl)} = \widetilde{\mathbf{Q}}^{(k)\top}\Big(\mathbf{T}^{(k+1:L+1)}\mathbf{T}^{(L+1:l+1)}\otimes \mathbf{T}^{(k-1:1)}\boldsymbol{\Sigma}_{\mathbf{xx}}\mathbf{T}^{(1:l-1)}\Big)\widetilde{\mathbf{Q}}^{(l)}\,, \tag{10}$$

where, $\widetilde{\mathbf{Q}}^{(l)} = \mathbf{I}_{m_l}\otimes\mathbf{Q}^{(l)}$.

[10]We denote the matrix by $\mathbf{P}$ instead of $\mathbf{I}$ as in the original paper, to avoid confusion with widespread occurrences of the identity matrix in our analysis.

Proof.

$$\frac{\partial \mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{W}^{(l)}} = \left(\mathbf{T}^{(L+1:l+1)} \otimes \mathbf{x}^\top \mathbf{T}^{(1:l-1)}\right) \frac{\partial \mathbf{T}^{(l)}}{\partial \mathbf{W}^{(l)}} \tag{11}$$

$$= \left(\mathbf{T}^{(L+1:l+1)} \otimes \mathbf{x}^\top \mathbf{T}^{(1:l-1)}\right) (\mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)}) \tag{12}$$

Note, we cannot use mixed product property of Kronecker product as the shapes are not conformable. Further the above computation is in the Jacobian format (numerator layout), and to compute the outer-product Hessian we need to transpose in order to obtain it in the gradient format.

$$\nabla_{\mathbf{W}^{(l)}} \mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = (\mathbf{I}_{m_l} \otimes {\mathbf{Q}^{(l)}}^\top) \left(\mathbf{T}^{(l+1:L+1)} \otimes \mathbf{T}^{(l-1:1)}\mathbf{x}\right) \tag{13}$$

Next, we compute the outer-product of the above and average over the input, resulting in the l-th block in the $\mathbf{H}_\mathrm{O}$ diagonal.

$$\begin{aligned}\mathbf{H}_\mathrm{O}^{(ll)} &:= \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{W}^{(l)}} \mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}_i) \cdot \nabla_{\mathbf{W}^{(l)}} \mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}_i)^\top \\ &= (\mathbf{I}_{m_l} \otimes {\mathbf{Q}^{(l)}}^\top)\Big(\mathbf{T}^{(l+1:L+1)}\mathbf{T}^{(L+1:l+1)} \otimes \\ &\qquad \mathbf{T}^{(l-1:1)}\boldsymbol{\Sigma}_{\mathbf{xx}}\mathbf{T}^{(1:l-1)}\Big)(\mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)})\end{aligned}$$

Let's use the shorthand $\widetilde{\mathbf{Q}}^{(l)} = \mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)}$ to make the expressions more compact. Similarly, we can this computation to obtain the kl-th block of the outer-product Hessian,

$$\begin{aligned}\mathbf{H}_\mathrm{O}^{(kl)} = \widetilde{\mathbf{Q}}^{(k)\top}\Big(&\mathbf{T}^{(k+1:L+1)}\mathbf{T}^{(L+1:l+1)} \otimes \\ &\mathbf{T}^{(k-1:1)}\boldsymbol{\Sigma}_{\mathbf{xx}}\mathbf{T}^{(1:l-1)}\Big)\widetilde{\mathbf{Q}}^{(l)}\end{aligned} \tag{14}$$

□

B.2. Functional Hessian

In our functional Hessian calculations, we consider the second derivative wrt the transpose of that matrix. This is not a problem for us as rank is invariant to row or column permutations.

Proposition 3. *For $k \leq l$, the kl-th block of $\mathbf{H}_\mathrm{F}$ is given by:*

$$\begin{aligned}\mathbf{H}_\mathrm{F}^{(kl)} = \left(\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(k)\top}\right)\Big(&\mathbf{T}^{(k+1:l-1)} \otimes \\ &\mathbf{T}^{(k-1:1)}\boldsymbol{\Omega}^\top\mathbf{T}^{(L+1:l+1)}\Big)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right),\end{aligned} \tag{15}$$

While, for $k \geq l$, it is:

$$\begin{aligned}\mathbf{H}_\mathrm{F}^{(kl)} = \left(\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(k)\top}\right)\Big(&\mathbf{T}^{(k+1:L+1)}\boldsymbol{\Omega}\mathbf{T}^{(1:l-1)} \otimes \\ &\mathbf{T}^{(k-1:l+1)}\Big)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right).\end{aligned} \tag{16}$$

Proof. This requires more care. First, let us recall its form:

$$\mathbf{H}_\mathrm{F} = \frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{K}[\nabla_{\mathrm{F}_{\boldsymbol{\theta}}}\ell_i]_c \, \nabla^2_{\boldsymbol{\theta}}\, \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

The gradient of the loss with respect to the function output, in the case of MSE loss, is just the residual. And so we will denote $\nabla_{\mathrm{F}_{\boldsymbol{\theta}}} \ell_i = \hat{\mathbf{y}}_i - \mathbf{y}_i =: \boldsymbol{\delta}_{\mathbf{x}_i, \mathbf{y}_i}$ hereafter.

To begin, we need to compute the Hessian of the network function at the c-th index, i.e., $\nabla^2_{\boldsymbol{\theta}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x})$. We already have the gradient (for all K outputs) with respect to $\mathbf{W}^{(l)}$, the matricized convolutional tensor at layer l, in Eqn. (13). The gradient with respect to only the output at the c-th index is given by:

$$\begin{aligned}\nabla_{\mathbf{W}^{(l)}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x}) &= \widetilde{\mathbf{Q}}^{(l)\top} \left(\mathbf{T}^{(l+1:L+1)} \otimes \mathbf{T}^{(l-1:1)} \mathbf{x} \right) \mathbf{e}_c \\ &= \widetilde{\mathbf{Q}}^{(l)\top} \left(\mathbf{T}^{(l+1:L+1)} \mathbf{e}_c \otimes \mathbf{T}^{(l-1:1)} \mathbf{x} \right) . \end{aligned} \tag{17}$$

Here, $\mathbf{e}_c \in \mathbb{R}^K$ denotes the c-th canonical basis vector. In the second line, we used the mixed-product property of Kronecker products. Now, we need to compute another derivative of the above expression with respect to the other layer matrices. We can clearly see that the block-diagonal terms in the Functional Hessian will be zero, since there is no more $\mathbf{T}^{(l)}$ or $\mathbf{W}^{(l)}$ left in the above expression.

Consider, we take the derivative with respect to layer k. We will have two cases depending upon whether $k < l$ or $k > l$. Before doing that, let's first rewrite the gradient expression and perform the sum over the inputs and targets, as shown below:

$$\nabla_{\mathbf{W}^{(l)}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x}) = \widetilde{\mathbf{Q}}^{(l)\top} \operatorname{vec}_r \left(\mathbf{T}^{(l+1:L+1)} \mathbf{e}_c \mathbf{x}^\top \mathbf{T}^{(1:l-1)} \right)$$

$$= \widetilde{\mathbf{Q}}^{(l)\top} \left(\mathbf{T}^{(l+1:L+1)} \mathbf{e}_c \mathbf{x}^\top \otimes \mathbf{I}_{m_{l-1} d_{l-1}} \right) \operatorname{vec}_r \left(\mathbf{T}^{(1:l-1)} \right) \tag{18}$$

$$= \widetilde{\mathbf{Q}}^{(l)\top} \left(\mathbf{I}_{m_l d_l} \otimes \mathbf{T}^{(l-1:1)} \mathbf{x} \mathbf{e}_c^\top \right) \operatorname{vec}_r \left(\mathbf{T}^{(l+1:L+1)} \right) . \tag{19}$$

Here, we first used the fact the Kronecker product of two vectors $\mathbf{a}$ and $\mathbf{b}$ can be written as vectorization of their outer-product, namely, $\mathbf{a} \otimes \mathbf{b} = \operatorname{vec}_r \left(\mathbf{a}\mathbf{b}^\top \right)$. In the second line, we use the identity

$$\operatorname{vec}_r(\mathbf{A}\mathbf{X}\mathbf{B}) = \left(\mathbf{A} \otimes \mathbf{B}^\top \right) \operatorname{vec}_r(\mathbf{X}) ,$$

for $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\mathbf{B} \in \mathbb{R}^{p \times q}$.

B.2.1. When $k < l$

In this case, we will use the form of the gradient in Eqn. (18) to differentiate with respect to $\mathbf{W}^{(k)}$. As a matter of fact, we will actually perform the derivatives with respect to $\mathbf{W}^{(k)\top}$ to avoid keeping track of the special kind of permutation matrices known as commutation matrices. This will not restrain the analysis since the rank of a matrix is invariant to both row and column permutations.

Let's focus on taking the derivative with respect to $\operatorname{vec}_r \mathbf{T}^{(1:l-1)}$ since that's the only term which will depend on $\mathbf{W}^{(k)}$.

$$\frac{\partial}{\partial \mathbf{W}^{(k)\top}} \operatorname{vec}_r \mathbf{T}^{(1:l-1)} \tag{20}$$

$$= \frac{\partial}{\partial \mathbf{W}^{(k)\top}} \operatorname{vec}_r \mathbf{T}^{(1:k-1)} \mathbf{T}^{(k)\top} \mathbf{T}^{(k+1:l-1)} \tag{21}$$

$$= \left(\mathbf{T}^{(1:k-1)} \otimes \mathbf{T}^{(l-1:k+1)} \right) \frac{\partial}{\partial \mathbf{W}^{(k)\top}} \operatorname{vec}_r \mathbf{T}^{(k)\top} \tag{22}$$

We can exploit Lemma 1 and equivalently write $\mathbf{T}^{(k)} \in \mathbb{R}^{m_k d_k \times m_{k-1} d_{k-1}}$ in terms of $\mathbf{W}^{(k)} \in \mathbb{R}^{m_k \times m_{k-1} k_k}$ and $\mathbf{Q}^{(k)} \in \mathbb{R}^{m_{k-1} d_k d_{k-1} \times m_{k-1} k_k}$, as shown below:

$$\operatorname{vec}_r \mathbf{T}^{(k)} = (\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(k)}) \operatorname{vec}_r \mathbf{W}^{(k)} \tag{23}$$

$$\text{or,} \quad \operatorname{vec}_r \mathbf{T}^{(k)} = \operatorname{vec}_r \mathbf{W}^{(k)} \mathbf{Q}^{(k)\top} \tag{24}$$

Thus we can write $\dfrac{\partial}{\partial \mathbf{W}^{(k)\top}} \operatorname{vec}_r \mathbf{T}^{(k)\top}$ as:

$$\frac{\partial}{\partial \mathbf{W}^{(k)\top}} \operatorname{vec}_r \mathbf{T}^{(k)\top} = \frac{\partial}{\partial \mathbf{W}^{(k)\top}} \operatorname{vec}_r \mathbf{Q}^{(k)} \mathbf{W}^{(k)\top} \tag{25}$$

$$= \left(\mathbf{Q}^{(k)} \otimes \mathbf{I}_{m_k}\right) \frac{\partial \operatorname{vec}_r \mathbf{W}^{(k)\top}}{\partial \mathbf{W}^{(k)\top}} \tag{26}$$

$$= \mathbf{Q}^{(k)} \otimes \mathbf{I}_{m_k} \tag{27}$$

Finally, we can complete the overall expression of derivative of the gradient with respect to layer k:

$$\frac{\partial \nabla_{\mathbf{W}^{(l)}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{W}^{(k)\top}} = \left(\mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)\top}\right) \Big(\mathbf{T}^{(l+1:L+1)} \mathbf{e}_c \mathbf{x}^\top \mathbf{T}^{(1:k-1)} \otimes \mathbf{T}^{(l-1:k+1)}\Big) \left(\mathbf{Q}^{(k)} \otimes \mathbf{I}_{m_k}\right) . \tag{28}$$

Summing the above expression over the inputs and targets, along with scaling by the residuals, yields the (lk)-th block of the functional Hessian:

$$\mathbf{H}_{\mathrm{F}}^{(lk)} = \left(\mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)\top}\right) \Big(\mathbf{T}^{(l+1:L+1)} \boldsymbol{\Omega} \mathbf{T}^{(1:k-1)} \otimes \mathbf{T}^{(l-1:k+1)}\Big) \left(\mathbf{Q}^{(k)} \otimes \mathbf{I}_{m_k}\right) , \tag{29}$$

where, $\boldsymbol{\Omega} := \mathbf{E}\left[\boldsymbol{\delta}_{\mathbf{x},\mathbf{y}}\, \mathbf{x}^\top\right]$ is the (uncentered) residual-input covariance matrix.

Similarly, the (kl)-th block can also be obtained by repeating the same procedure[11], resulting in:

$$\mathbf{H}_{\mathrm{F}}^{(kl)} = \left(\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(k)\top}\right) \Big(\mathbf{T}^{(k+1:l-1)} \otimes \mathbf{T}^{(k-1:1)} \boldsymbol{\Omega}^\top \mathbf{T}^{(L+1:l+1)}\Big) \left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) , \tag{30}$$

B.2.2. WHEN $k > l$

We can repeat a similar calculation to obtain the following expression:

$$\mathbf{H}_{\mathrm{F}}^{(kl)} = \left(\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(k)\top}\right) \Big(\mathbf{T}^{(k+1:L+1)} \boldsymbol{\Omega} \mathbf{T}^{(1:l-1)} \otimes \mathbf{T}^{(k-1:l+1)}\Big) \left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) . \tag{31}$$

By comparing the expressions in Eqns. (30) and (31), we get the hint to factorize out $\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}$ from the columns, and likewise $\mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)\top}$ from the rows, $\forall\, l \in [L+1]$.

Remark. The arrangement of the Toeplitz derivatives on the left and right hand side in the Eqn. (30) above is very similar to that in the case of Eqn. (14) for the outer-product Hessian — except, now on the right hand side the order of Kronecker product between $\mathbf{I}_{m_l}$ and $\mathbf{Q}^{(l)}$ gets changed to reflect the fact that we are taking the derivatives with respect to $\mathbf{W}^{(l)\top}$ here. $\square$

[11]Attention: it is not the same as the transpose of the expression in Eqn. (29), since recall on one axis of the Hessian we are taking derivatives in the order of $\operatorname{vec}_r(\mathbf{W}^{(l)})$ and on the other axis in the order $\operatorname{vec}_r(\mathbf{W}^{(l)\top})$, causing the asymmetry.

C. Omitted Proofs

C.1. Rank of Outer-Product Hessian

Theorem 5. *The rank of the outer-product Hessian is upper bounded as*

$$\begin{aligned}\mathrm{rk}(\mathbf{H}_{\mathrm{O}}) \leq \min\big(p, d_0\,\mathrm{rk}(\mathbf{T}^{(2:L+1)}) + K\,\mathrm{rk}(\mathbf{T}^{(L:1)}) - \\ \mathrm{rk}(\mathbf{T}^{(2:L+1)})\,\mathrm{rk}(\mathbf{T}^{(L:1)})\big) \\ = \min\left(p, q\left(d_0 + K - q\right)\right)\,.\end{aligned}$$

Here, $q := \min(d_0, m_1 d_1, \cdots, m_L d_L, K)$.

Proof. By the decomposition in Proposition 4, we have that:

$$\mathrm{rk}(\mathbf{H}_{\mathrm{O}}) \leq \min\left(\mathrm{rk}(\mathbf{A}_o), \mathrm{rk}(\mathbf{B}_o), \mathrm{rk}(\mathbf{Q}_o)\right)\,. \tag{32}$$

Now, $\mathrm{rk}(\mathbf{B}_o) = K d_0$, $\mathrm{rk}(\mathbf{Q}_o) \leq \min(\widehat{p}, p)$, and via Theorem 3 of (Singh et al., 2021), we have that $\mathrm{rk}(\mathbf{A}_o) = q(d_0 + K - q)$ which is naturally bounded above by $\widehat{p}$. Hence, we obtain:

$$\mathrm{rk}(\mathbf{H}_{\mathrm{O}}) \leq \min\left(p, q(d_0 + K - q)\right)\,.$$

Here, $q := \min(d_0, m_1 d_1, \cdots, m_L d_L, K)$. □

C.2. Rank of Functional Hessian

Theorem 6. *For a deep linear convolutional network, the rank of* l*-th column-block,* $\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet l}$*, of the matrix* $\widehat{\mathbf{H}}_{\mathrm{F}}$*, can be upper bounded as*

$$\mathrm{rk}(\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet l}) \leq \min(\widehat{q}\, m_{l-1} d_{l-1} + \widehat{q}\, m_l d_l - \widehat{q}^{\,2}\,, m_l m_{l-1} k_l)\,,$$

for $l \in [2, \cdots, L]$. *When* $l = 1$*, we have*

$$\mathrm{rk}(\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet 1}) \leq \min(\widehat{q}\, m_1 d_1 + \widehat{q}\, s - \widehat{q}^{\,2}\,, m_1 m_0 k_1)\,.$$

And, when $l = L + 1$*, we have*

$$\mathrm{rk}(\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet L+1}) \leq \min(\widehat{q}\, m_L d_L + \widehat{q}\, s - \widehat{q}^{\,2}\,, m_{L+1} m_L k_{L+1})\,.$$

Here, $\widehat{q} := \min(d_0, m_1 d_1, \cdots, m_L d_L, K, s) = \min(q, s)$ *and* $s := \mathrm{rk}(\boldsymbol{\Omega}) = \mathrm{rk}(\mathbf{E}\left[\boldsymbol{\delta}_{\mathbf{x},\mathbf{y}}\,\mathbf{x}^\top\right])$.

Proof.

$$\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet l} = \begin{pmatrix} \left(\mathbf{I}_{m_1} \otimes \mathbf{Q}^{(1)\top}\right)\left(\mathbf{T}^{(2:l-1)} \otimes \mathbf{\Omega}^\top \mathbf{T}^{(L+1:l+1)}\right)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) \\ \vdots \\ \left(\mathbf{I}_{m_j} \otimes \mathbf{Q}^{(j)\top}\right)\left(\mathbf{T}^{(j+1:l-1)} \otimes \mathbf{T}^{(j-1:1)}\mathbf{\Omega}^\top \mathbf{T}^{(L+1:l+1)}\right)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) \\ \vdots \\ \left(\mathbf{I}_{m_{l-1}} \otimes \mathbf{Q}^{(l-1)\top}\right)\left(\mathbf{I}_{m_{l-1}} \otimes \mathbf{T}^{(l-2:1)}\mathbf{\Omega}^\top \mathbf{T}^{(L+1:l+1)}\right)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) \\ \mathbf{0} \\ \left(\mathbf{I}_{m_{l+1}} \otimes \mathbf{Q}^{(l+1)\top}\right)\left(\mathbf{T}^{(l+2:L+1)}\mathbf{\Omega}\mathbf{T}^{(1:l-1)} \otimes \mathbf{I}_{m_l}\right)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) \\ \vdots \\ \left(\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(k)\top}\right)\left(\mathbf{T}^{(k+1:L+1)}\mathbf{\Omega}\mathbf{T}^{(1:l-1)} \otimes \mathbf{T}^{(k-1:l+1)}\right)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) \\ \vdots \\ \left(\mathbf{I}_{m_{L+1}} \otimes \mathbf{Q}^{(l+1)\top}\right)\left(\mathbf{\Omega}\mathbf{T}^{(1:l-1)} \otimes \mathbf{T}^{(L-1:l+1)}\right)\left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right) \end{pmatrix}$$

$$= \mathbf{Q}_F \underbrace{\begin{pmatrix} \mathbf{T}^{(2:l-1)} \otimes \mathbf{\Omega}^\top \mathbf{T}^{(L+1:l+1)} \\ \vdots \\ \mathbf{T}^{(j+1:l-1)} \otimes \mathbf{T}^{(j-1:1)}\mathbf{\Omega}^\top \mathbf{T}^{(L+1:l+1)} \\ \vdots \\ \mathbf{I}_{m_{l-1}} \otimes \mathbf{T}^{(l-2:1)}\mathbf{\Omega}^\top \mathbf{T}^{(L+1:l+1)} \\ \mathbf{0} \\ \mathbf{T}^{(l+2:L+1)}\mathbf{\Omega}\mathbf{T}^{(1:l-1)} \otimes \mathbf{I}_{m_l} \\ \vdots \\ \mathbf{T}^{(k+1:L+1)}\mathbf{\Omega}\mathbf{T}^{(1:l-1)} \otimes \mathbf{T}^{(k-1:l+1)} \\ \vdots \\ \mathbf{\Omega}\mathbf{T}^{(1:l-1)} \otimes \mathbf{T}^{(L-1:l+1)} \end{pmatrix}}_{\mathbf{A}_F^{(l)}} \left(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l}\right)$$

In the above, the matrix $\mathbf{Q}_F \in \mathbb{R}^{p \times \widehat{p}}$ is the block diagonal matrix containing on its l-th block the matrix $\mathbf{I}_{m_l} \otimes \mathbf{Q}^{(l)\top}$. The rank of $\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet l}$ can then be upper-bounded as:

$$\begin{aligned} \operatorname{rk}(\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet l}) &\leq \min(\operatorname{rk}(\mathbf{Q}_F), \operatorname{rk}(\mathbf{A}_F^{(l)}), \operatorname{rk}(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l})) \\ &= \min(p, \widehat{p}, \, \widehat{q}\, m_{l-1} d_{l-1} + \widehat{q}\, m_l d_l - \widehat{q}^{\,2}\,, m_l m_{l-1} k_l) \\ &= \min(\widehat{q}\, m_{l-1} d_{l-1} + \widehat{q}\, m_l d_l - \widehat{q}^{\,2}\,, m_l m_{l-1} k_l)\,, \end{aligned}$$

where in the second line, we used the result of the $\operatorname{rk}(\mathbf{A}_F^{(l)})$ from (Singh et al., 2021), while the $\operatorname{rk}(\mathbf{Q}^{(l)} \otimes \mathbf{I}_{m_l})) \leq \min(m_l m_{l-1} k_l, m_l, m_{l-1} d_l d_{l-1}) = m_l m_{l-1} k_l$ as $k_l \leq d_{l-1}$ for valid convolution.

Likewise, we can carry out a similar operation for the first block-columnn $\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet 1}$

$$\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet 1} = \begin{pmatrix} \mathbf{0} \\ \left(\mathbf{I}_{m_2} \otimes \mathbf{Q}^{(2)\top}\right)\left(\mathbf{T}^{(3:L)}\boldsymbol{\Omega} \otimes \mathbf{I}_{m_1}\right)\left(\mathbf{Q}^{(1)} \otimes \mathbf{I}_{m_1}\right) \\ \vdots \\ \left(\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(k)\top}\right)\left(\mathbf{T}^{(k+1:L+1)}\boldsymbol{\Omega} \otimes \mathbf{T}^{(k-1:2)}\right)\left(\mathbf{Q}^{(1)} \otimes \mathbf{I}_{m_1}\right) \\ \vdots \\ \left(\mathbf{I}_{m_{L+1}} \otimes \mathbf{Q}^{(l+1)\top}\right)\left(\boldsymbol{\Omega} \otimes \mathbf{T}^{(L:2)}\right)\left(\mathbf{Q}^{(1)} \otimes \mathbf{I}_{m_1}\right) \end{pmatrix} = \mathbf{Q}_F \underbrace{\begin{pmatrix} \mathbf{0} \\ \mathbf{T}^{(3:L)}\boldsymbol{\Omega} \otimes \mathbf{I}_{m_1} \\ \vdots \\ \mathbf{T}^{(k+1:L)}\boldsymbol{\Omega} \otimes \mathbf{T}^{(k-1:2)} \\ \vdots \\ \boldsymbol{\Omega} \otimes \mathbf{T}^{(L:2)} \end{pmatrix}}_{\mathbf{A}_F^{(1)}} \left(\mathbf{Q}^{(1)} \otimes \mathbf{I}_{m_1}\right)$$

The rank can be then bounded as:

$$\begin{aligned} \mathrm{rk}(\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet 1}) &\leq \min(\mathrm{rk}(\mathbf{Q}_F), \mathrm{rk}(\mathbf{A}_F^{(1)}), \mathrm{rk}(\mathbf{Q}^{(1)} \otimes \mathbf{I}_{m_1})) \\ &= \min(p, \widehat{p},\, \widehat{q}\, m_1 d_1 + \widehat{q}\, s - \widehat{q}^{\,2}\,, m_1 m_0 k_1) \\ &= \min(\widehat{q}\, m_1 d_1 + \widehat{q}\, s - \widehat{q}^{\,2}\,, m_1 m_0 k_1)\,, \end{aligned}$$

Finally, we discuss the case of last column-block:

$$\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet L+1} = \begin{pmatrix} \left(\mathbf{I}_{m_1} \otimes \mathbf{Q}^{(1)\top}\right)\left(\mathbf{T}^{(2:L)} \otimes \boldsymbol{\Omega}^\top\right)\left(\mathbf{Q}^{(L+1)} \otimes \mathbf{I}_{m_{L+1}}\right) \\ \vdots \\ \left(\mathbf{I}_{m_k} \otimes \mathbf{Q}^{(l)\top}\right)\left(\mathbf{T}^{(k+1:L)} \otimes \mathbf{T}^{(k-1:1)}\boldsymbol{\Omega}^\top\right)\left(\mathbf{Q}^{(L+1)} \otimes \mathbf{I}_{m_{L+1}}\right) \\ \vdots \\ \left(\mathbf{I}_{m_L} \otimes \mathbf{Q}^{(l)\top}\right)\left(\mathbf{I}_{m_L} \otimes \mathbf{T}^{(L-1:1)}\boldsymbol{\Omega}^\top\right)\left(\mathbf{Q}^{(L+1)} \otimes \mathbf{I}_{m_{L+1}}\right) \\ \mathbf{0} \end{pmatrix}$$

$$= \mathbf{Q}_F \underbrace{\begin{pmatrix} \mathbf{T}^{(2:L)} \otimes \boldsymbol{\Omega}^\top \\ \vdots \\ \mathbf{T}^{(k+1:L)} \otimes \mathbf{T}^{(k-1:1)}\boldsymbol{\Omega}^\top \\ \vdots \\ \mathbf{I}_{m_L} \otimes \mathbf{T}^{(L-1:1)}\boldsymbol{\Omega}^\top \\ \mathbf{0} \end{pmatrix}}_{\mathbf{A}_F^{(L+1)}} \left(\mathbf{Q}^{(L+1)} \otimes \mathbf{I}_{m_{L+1}}\right)$$

The rank can be then bounded as:

$$\begin{aligned} \mathrm{rk}(\widehat{\mathbf{H}}_{\mathrm{F}}^{\bullet L+1}) &\leq \min(\mathrm{rk}(\mathbf{Q}_F), \mathrm{rk}(\mathbf{A}_F^{(L+1)}), \mathrm{rk}(\mathbf{Q}^{(L+1)} \otimes \mathbf{I}_{m_{L+1}})) \\ &= \min(p, \widehat{p},\, \widehat{q}\, m_L d_L + \widehat{q}\, s - \widehat{q}^{\,2}\,, m_{L+1} m_L k_{L+1}) \\ &= \min(\widehat{q}\, m_L d_L + \widehat{q}\, s - \widehat{q}^{\,2}\,, m_{L+1} m_L k_{L+1})\,, \end{aligned}$$

which completes the proof

□

A corollary is

Corollary 12. *Under the setup of Theorem 6, when $\widehat{q} = q = s$, the rank of $\mathbf{H}_{\mathrm{F}}$ can be further upper bounded as,* $\mathrm{rk}(\mathbf{H}_{\mathrm{F}}) \leq 2\, q\, M - (L-1)\, q^2$ *, where* $\quad M = \sum_{i=1}^{L} m_i\, d_i$.

C.3. Rank results for one-hidden layer CNN case

Theorem 7. *For a one-hidden layer 1D CNN, the rank of the outer product Hessian can be bounded as:* $\operatorname{rank}(\mathbf{H}_{\mathrm{O}}) \leq \min\big(Kd_0, q(k_1 + K(d_0 - k_1 + 1) - q)\big)$ *, where* $q = \min(k_1, K(d_0 - k_1 + 1), m_1)$.

Proof. Consider a one-hidden layer CNN as follows:

$$\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{T}^{(2)}\mathbf{T}^{(1)}\mathbf{x}\,, \quad \text{with} \quad \boldsymbol{\theta} = \{\mathcal{W}^{(2)}, \mathcal{W}^{(1)}\}\,.$$

Now, at the output layer $k_2 = d_1$ so as to project the hidden features to an output of size $K = m_2$ hence Thus, spatially we will have that $d_2 = 1$, and as a result $\mathbf{T}^{(2)} = \mathbf{W}^{(2)}$. Also, the number of input channels is simply assumed to be $m_0 = 1$, else we can flatten the input to obtain this.

Remember that, in this scenario, $\mathbf{T}^{(1)}$ amounts to:

$$\mathbf{T}^{(1)} = \begin{pmatrix} \mathbf{T}^{\mathcal{W}^{(1)}_{(1,1)}\bullet} \\ \vdots \\ \mathbf{T}^{\mathcal{W}^{(1)}_{(m_1,1)}\bullet} \end{pmatrix}.$$

While, as $m_0 = 1$, $\mathbf{W}^{(1)} = \begin{pmatrix} w_{1,1,1} & \cdots & w_{1,1,k_1} \\ w_{m_1,1,1} & \cdots & w_{m_1,1,k_1} \end{pmatrix}$ and we would like to ideally group the elements of $\mathbf{W}^{(1)}$ inside $\mathbf{T}^{(1)}$. We will do this by suitable row permutations using the auxiliary permutation matrices, discussed in the section A.2.2.

Let's rewrite our network function as follows:

$$\begin{aligned} \mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbf{W}^{(2)}\mathbf{T}^{(1)}\mathbf{x} \\ &= \mathbf{W}^{(2)}\,\mathbf{P}^{(d_1)}\mathbf{P}_{(d_1)}\,\mathbf{T}^{(1)}\mathbf{x} \end{aligned}$$

where $\mathbf{P}_{(d_1)}\mathbf{T}^{(1)} = \begin{pmatrix} \widetilde{\mathbf{W}}^{(1)}\pi_C^0 \\ \vdots \\ \widetilde{\mathbf{W}}^{(1)}\pi_C^{d_1-1} \end{pmatrix}$ and $\widetilde{\mathbf{W}}^{(1)} = \begin{pmatrix} \mathbf{W}^{(1)}_{m_1\times k_1} & \mathbf{0}_{m_1\times(d_0-k_1)} \end{pmatrix} = \mathbf{W}^{(1)}\mathbf{I}_{k_1\times d_0}$.

Next, let's denote the matrix $\mathbf{W}^{(2)}\mathbf{P}^{(d_1)} \in \mathbb{R}^{K\times m_1 d_1}$ by $\mathbf{V} = \begin{pmatrix} \mathbf{V}^{(1)} & \cdots & \mathbf{V}^{(d_1)} \end{pmatrix}$. Using these we can represent the network function in the following superposition form:

$$\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{i=1}^{d_1} \mathbf{V}^{(i)}\,\mathbf{W}^{(1)}\,\mathbf{I}_{k_1\times d_0}\,\pi_C^{i-1}\mathbf{x}\,. \tag{33}$$

Having done this reformulation, let's get back to the business of computing the derivatives, which are listed below:

$$\nabla_{\mathbf{V}^{(i)}}\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{I}_K \otimes \mathbf{W}^{(1)}\mathbf{I}_{k_1\times d_0}\,\pi_C^{i-1}\mathbf{x} \tag{34}$$

$$\nabla_{\mathbf{W}^{(1)}}\mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{i=1}^{d_1} {\mathbf{V}^{(i)}}^{\top} \otimes \mathbf{I}_{k_1\times d_0}\,\pi_C^{i-1}\mathbf{x} \tag{35}$$

Now let's collect these individual parameter gradients to form the overall gradient,

$$\nabla_{\boldsymbol{\theta}} \mathrm{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{W}^{(1)} \mathbf{I}_{k_1 \times d_0}\, \pi_C^0 \mathbf{x} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{W}^{(1)} \mathbf{I}_{k_1 \times d_0}\, \pi_C^{d_1-1} \mathbf{x} \\ \sum_{i-1}^{d_1} \mathbf{V}^{(i)\top} \otimes \mathbf{I}_{k_1 \times d_0}\, \pi_C^{i-1} \mathbf{x} \end{pmatrix}$$

$$= \begin{pmatrix} \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{W}^{(1)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{I}_K \otimes \mathbf{W}^{(1)} \end{pmatrix} \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{I}_{k_1 \times d_0}\, \pi_C^0 \mathbf{x} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{I}_{k_1 \times d_0}\, \pi_C^{d_1-1} \mathbf{x} \end{pmatrix} \\ \begin{pmatrix} \mathbf{V}^{(1)\top} \otimes \mathbf{I}_k & \cdots & \mathbf{V}^{(d_1)\top} \otimes \mathbf{I}_k \end{pmatrix} \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{I}_{k_1 \times d_0}\, \pi_C^0 \mathbf{x} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{I}_{k_1 \times d_0}\, \pi_C^{d_1-1} \mathbf{x} \end{pmatrix} \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} \mathbf{I}_{Kd_1} \otimes \mathbf{W}^{(1)} \\ \widetilde{\mathbf{V}} \otimes \mathbf{I}_k \end{pmatrix}}_{\mathbf{A}_o} \underbrace{\begin{pmatrix} \mathbf{I}_K \otimes \mathbf{I}_{k_1 \times d_0}\, \pi_C^0 \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{I}_{k_1 \times d_0}\, \pi_C^{d_1-1} \end{pmatrix}}_{\mathbf{C}} \underbrace{(\mathbf{I}_K \otimes \mathbf{x})}_{\mathbf{B}}$$

where, $\widetilde{\mathbf{V}} = \begin{pmatrix} \mathbf{V}^{(1)\top} & \cdots & \mathbf{V}^{(d_1)\top} \end{pmatrix} \in \mathbb{R}^{m \times Kd_1}$

Finally, we can take the outer product and average them over the dataset to yield the outer-product Hessian:

$$\mathbf{H}_{\mathrm{O}} = \mathbf{A}_o \mathbf{C} \left(\mathbf{I}_K \otimes \boldsymbol{\Sigma}_{\mathbf{xx}} \right) \mathbf{C}^\top \mathbf{A}_o^\top \tag{36}$$

Hence, we have that:

$$\begin{aligned} \mathrm{rk}(\mathbf{H}_{\mathrm{O}}) &\leq \min(\mathrm{rk}(\mathbf{I}_K \otimes \boldsymbol{\Sigma}_{\mathbf{xx}}), \mathrm{rk}(\mathbf{A}_o), \mathrm{rk}(\mathbf{C})) \\ &= \min(Kd_0, q(Kd_1 + k - q)) \end{aligned}$$

where $q = \min(k, Kd_1, m)$ and using the fact that, for $\mathbf{C} \in \mathbb{R}^{Kd_1k_1 \times Kd}$, it has $\mathrm{rk}(\mathbf{C}) \leq \min(Kd_1k_1, Kd) = Kd$. $\square$

Theorem 8. *For a one-hidden layer 1D CNN, the rank of the functional Hessian obeys the following formula:* $\mathrm{rank}\left(\mathbf{H}_{\mathrm{F}}\right) \leq 2 \min \big(k_1, K(d_0 - k_1 + 1)\big)\, m_1$.

Proof. We will start atop the network function, eq. (33), and parameter gradients, eq. (34), (35), obtained after carrying out the manipulations with permutation matrices in the proof of Theorem 7. Let's then get the gradient with respect to the parameters at the c-th output index:

$$\begin{aligned} \nabla_{\mathbf{V}^{(i)}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x}) &= \left(\mathbf{I}_K \otimes \mathbf{W}^{(1)} \mathbf{I}_{k_1 \times d_0} \pi_C^{i-1} \mathbf{x} \right) \mathbf{e}_c \\ &= \mathbf{e}_c \otimes \mathbf{W}^{(1)} \mathbf{I}_{k_1 \times d_0} \pi_C^{i-1} \mathbf{x} \\ &= \mathrm{vec}_r \left(\mathbf{e}_c \mathbf{x}^\top \pi_R^{i-1} \mathbf{I}_{d_0 \times k_1} \mathbf{W}^{(1)\top} \right) \end{aligned} \tag{37}$$

And that with respect to $\mathbf{W}^{(1)}$ is:

$$\begin{aligned}\nabla_{\mathbf{W}^{(1)}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x}) &= \sum_{i=1}^{d_1} \left({\mathbf{V}^{(i)}}^\top \otimes \mathbf{I}_{k_1 \times d_0} \pi_C^{i-1} \mathbf{x} \right) \mathbf{e}_c \\ &= \sum_{i=1}^{d_1} {\mathbf{V}^{(i)}}^\top \mathbf{e}_c \otimes \mathbf{I}_{k_1 \times d_0} \pi_C^{i-1} \mathbf{x}^{(i)} \\ &= \sum_{i=1}^{d_1} \mathrm{vec}_r \left({\mathbf{V}^{(i)}}^\top \mathbf{e}_c \mathbf{x}^\top \pi_R^{i-1} \mathbf{I}_{d_0 \times k_1} \right) \end{aligned} \tag{38}$$

Now, we perform the second differentiation with respect to transposed matrices, i.e., $\mathbf{W}^{(1)\top}$ or ${\mathbf{V}^{(i)}}^\top$.

$$\frac{\partial \nabla_{\mathbf{V}^{(i)}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{W}^{(1)\top}} = \mathbf{e}_c \mathbf{x}^\top \pi_R^{i-1} \mathbf{I}_{d_0 \times k_1} \otimes \mathbf{I}_{m_1} \,.$$

$$\frac{\partial \nabla_{\mathbf{W}^{(1)}} \mathrm{F}^c_{\boldsymbol{\theta}}(\mathbf{x})}{\partial {\mathbf{V}^{(i)}}^\top} = \mathbf{I}_{m_1} \otimes \mathbf{x} \mathbf{e}_c^\top \pi_C^{i-1} \mathbf{I}_{k_1 \times d_0} \,.$$

Let's now take the sum of the above matrices over the input as well as, with appropriate scaling by the residual, over the output indices to get the blocks of the functional Hessian. This yields,

$$\mathbf{H}_{\mathrm{F}}^{(\mathbf{V}^{(i)}, \mathbf{W}^{(1)})} = \boldsymbol{\Omega} \pi_R^{i-1} \mathbf{I}_{d_0 \times k_1} \otimes \mathbf{I}_{m_1} \,. \tag{39}$$

$$\mathbf{H}_{\mathrm{F}}^{(\mathbf{W}^{(1)}, \mathbf{V}^{(i)})} = \mathbf{I}_{m_1} \otimes \boldsymbol{\Omega}^\top \pi_C^{i-1} \mathbf{I}_{k_1 \otimes d_0} \,. \tag{40}$$

where, recall $\boldsymbol{\Omega} = \mathbf{E}\left[\boldsymbol{\delta}_{\mathbf{x},\mathbf{y}}\, \mathbf{x}^\top\right] \in \mathbb{R}^{K \times d_0}$ denotes the (uncentered) covariance of the residual with the input.

Since the matrix is zero on the block diagonals (i.e., block hollow), we can analyze the rank of all the row blocks with respect to $\mathbf{W}^{(1)}$ (same as the column block with respect to all the $\mathbf{V}^{(i)}$) or the row blocks with respect to all the $\mathbf{V}^{(i)}$ (same as the column block with respect to the $\mathbf{W}^{(1)}$, i.e., either $\mathbf{H}_{\mathrm{F}}^{(\mathbf{W}^{(1)}, \mathbf{V}^{(i)})}$ or $\mathbf{H}_{\mathrm{F}}^{(\mathbf{V}^{(i)}, \mathbf{W}^{(1)})}$. Focussing on the latter, the column blocks with respect to $\mathbf{W}^{(1)}$, i.e.,

$$\mathbf{H}_{\mathrm{F}}^{(\bullet, \mathbf{W}^{(1)})} := \begin{pmatrix} \mathbf{H}_{\mathrm{F}}^{(\mathbf{V}^{(1)}, \mathbf{W}^{(1)})} \\ \vdots \\ \mathbf{H}_{\mathrm{F}}^{(\mathbf{V}^{(d_1)}, \mathbf{W}^{(1)})} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Omega} \pi_R^{0} \mathbf{I}_{d_0 \times k_1} \otimes \mathbf{I}_{m_1} \\ \vdots \\ \boldsymbol{\Omega} \pi_R^{d_1 - 1} \mathbf{I}_{d_0 \times k_1} \otimes \mathbf{I}_{m_1} \end{pmatrix}$$

The above is equivalent upto row permutations to,

$$\widetilde{\boldsymbol{\Omega}} \otimes \mathbf{I}_{m_1} := \begin{pmatrix} \boldsymbol{\Omega} \pi_R^{0} \mathbf{I}_{d_0 \times k_1} \\ \vdots \\ \boldsymbol{\Omega} \pi_R^{d_1 - 1} \mathbf{I}_{d_0 \times k_1} \end{pmatrix} \otimes \mathbf{I}_{m_1} \tag{41}$$

where, $\widetilde{\boldsymbol{\Omega}} \in \mathbb{R}^{K d_1 \times k_1}$ The rank of the functional Hessian thus amounts to:

$$\begin{aligned} \mathrm{rk}(\mathbf{H}_{\mathrm{F}}) &= 2\, \mathrm{rk}\left(\widetilde{\boldsymbol{\Omega}} \otimes \mathbf{I}_{m_1} \right) \\ &= 2 m_1\, \mathrm{rk}(\widetilde{\boldsymbol{\Omega}}) \\ &\leq 2 \min(K d_1, k_1) m_1 \,. \end{aligned}$$

In the first line, the rank is scaled by 2 to account for both the non-zero blocks present in the functional Hessian. The rest follows from the properties of Kronecker product and rank, thus completing the proof.

□

C.4. Rank results for Locally connected network

Theorem 9. *For the locally connected network as described in Eqn. (7), the rank of the outer-product and the functional Hessian can be upper bounded as follows:* $\mathrm{rk}(\mathbf{H}_{\mathrm{O}}^{LCN}) \leq t \cdot q(k+K-q)$ *, and* $\mathrm{rk}(\mathbf{H}_{\mathrm{F}}{}^{LCN}) \leq t \cdot 2m\min(k,K)$*, where* $q = \min(k,K,m)$ *and* $t = \dfrac{d}{k}$.

Proof. We now proceed to showing the proofs of the rank of outer-product and functional Hessian for the LCN case. Let's recall the Eqn. (7) for the LCN:

$$\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) = \sum_{i=1}^{t} \mathbf{V}^{(i)}\,\mathbf{W}^{(ii)}\,\mathbf{x}^{(i)}\,.$$

Outer-product Hessian. Let's compute the gradient of the function with respect to the matrix $\mathbf{V}^{(i)}$:

$$\nabla_{\mathbf{V}^{(i)}}\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) = \mathbf{I}_K \otimes \mathbf{W}^{(ii)}\mathbf{x}^{(i)} \tag{42}$$

And that with respect to $\mathbf{W}^{(ii)}$ is:

$$\nabla_{\mathbf{W}^{(ii)}}\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) = \mathbf{V}^{(i)}{}^{\top} \otimes \mathbf{x}^{(i)} \tag{43}$$

So the gradient of the function output with respect to all the parameters can be arranged as follows:

$$\nabla_{\boldsymbol{\theta}}\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) = \begin{pmatrix} \nabla_{\mathbf{V}^{(1)}}\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) \\ \nabla_{\mathbf{W}^{(11)}}\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) \\ \vdots \\ \nabla_{\mathbf{V}^{(t)}}\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) \\ \nabla_{\mathbf{W}^{(tt)}}\mathrm{F}_{\boldsymbol{\theta}}^{\mathrm{LCN}}(\mathbf{x}) \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{W}^{(11)}\mathbf{x}^{(1)} \\ \mathbf{V}^{(1)}{}^{\top} \otimes \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{W}^{(tt)}\mathbf{x}^{(t)} \\ \mathbf{V}^{(t)}{}^{\top} \otimes \mathbf{x}^{(t)} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_o^{(1)}\left(\mathbf{I}_K \otimes \mathbf{x}^{(1)}\right) \\ \vdots \\ \mathbf{A}_o^{(t)}\left(\mathbf{I}_K \otimes \mathbf{x}^{(t)}\right) \end{pmatrix},$$

where, $\mathbf{A}_o^{(i)} := \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{W}^{(ii)} \\ \mathbf{V}^{(i)}{}^{\top} \otimes \mathbf{I}_d \end{pmatrix}$. Next, we take the outer-product of the gradient above, and then average over the inputs, resulting in:

$$\mathbf{H}_{\mathrm{O}} = \mathbf{A}_o \underbrace{\begin{pmatrix} \mathbf{I}_K \otimes \boldsymbol{\Sigma}^{(11)} & \cdots & \mathbf{I}_K \otimes \boldsymbol{\Sigma}^{(1t)} \\ \vdots & & \vdots \\ \mathbf{I}_K \otimes \boldsymbol{\Sigma}^{(t1)} & \cdots & \mathbf{I}_K \otimes \boldsymbol{\Sigma}^{(tt)} \end{pmatrix}}_{\mathbf{B}_o} \mathbf{A}_o^{\top} \tag{44}$$

Here, $\mathbf{A}_o = \begin{pmatrix} \mathbf{A}_o^{(1)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{A}_o^{(t)} \end{pmatrix}$, $\mathbf{\Sigma}^{(ij)} := \operatorname{cov}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. Then we have that,

$$\begin{aligned} \operatorname{rk}(\mathbf{A}_o) &= t \operatorname{rk}(\mathbf{A}_o^{(i)}) && (45)\\ &= K \operatorname{rk}(\mathbf{W}^{(ii)}) + k \operatorname{rk}(\mathbf{V}^{(i)}) - \operatorname{rk}(\mathbf{W}^{(ii)}) \operatorname{rk}(\mathbf{V}^{(i)}) && (46)\\ &= t \cdot q(K + k - q) && (47) \end{aligned}$$

where, $q := \min(K, k, m)$.

Now, the matrix $\mathbf{B}_o$ is equivalent, upto row and column permutations, to the matrix $\widetilde{\mathbf{B}}_o = \mathbf{I}_K \otimes \mathbf{\Sigma}_{\mathbf{xx}}$, since

$$\begin{pmatrix} \mathbf{\Sigma}^{(11)} & \cdots & \mathbf{\Sigma}^{(1t)} \\ \vdots & \cdots & \vdots \\ \mathbf{\Sigma}^{(t1)} & \cdots & \mathbf{\Sigma}^{(tt)} \end{pmatrix} \in \mathbb{R}^{d\times d} = \mathbf{\Sigma}_{\mathbf{xx}} = \left(\mathbf{E}\left[x_i x_j\right]\right)_{ij}$$

Therefore, rank of $\mathbf{B}_o$ can be upper bounded as $\operatorname{rk}(\mathbf{B}_o) \leq K \operatorname{rk}(\mathbf{\Sigma}_{\mathbf{xx}}) \leq Kd$. Finally, we can bound the rank of the outer-product Hessian, since the rank of a product of matrices is upper bounded by the minimum of the ranks of individual matrices. Hence,

$$\operatorname{rk}(\mathbf{H}_{\text{O}}) \leq \min\left(\operatorname{rk}(\mathbf{A}_o), \operatorname{rk}(\mathbf{B}_o)\right) = t \cdot q(K + k - q)\,. \qquad (48)$$

Functional Hessian. We need to differentiate again the network output gradients in Eqns. (42), (43). First, let's just obtain the gradient for the output at the c-th index:

$$\begin{aligned} \nabla_{\mathbf{V}^{(i)}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\mathrm{LCN}}(\mathbf{x}) &= \left(\mathbf{I}_K \otimes \mathbf{W}^{(ii)}\mathbf{x}^{(i)}\right) \mathbf{e}_c \\ &= \mathbf{e}_c \otimes \mathbf{W}^{(ii)}\mathbf{x}^{(i)} \\ &= \operatorname{vec}_r\left(\mathbf{e}_c \mathbf{x}^{(i)\top} \mathbf{W}^{(ii)\top}\right) && (49) \end{aligned}$$

And that with respect to $\mathbf{W}^{(ii)}$ is:

$$\begin{aligned} \nabla_{\mathbf{W}^{(ii)}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\mathrm{LCN}}(\mathbf{x}) &= \left(\mathbf{V}^{(i)^\top} \otimes \mathbf{x}^{(i)}\right) \mathbf{e}_c \\ &= \mathbf{V}^{(i)^\top} \mathbf{e}_c \otimes \mathbf{x}^{(i)} \\ &= \operatorname{vec}_r\left(\mathbf{V}^{(i)^\top} \mathbf{e}_c \mathbf{x}^{(i)\top}\right) && (50) \end{aligned}$$

Clearly, if we are to differentiate the above gradients with respect to matrix $\mathbf{V}^{(j)}$ or $\mathbf{W}^{(jj)}$, for $j \neq i$, we will just get a $\mathbf{0}$ matrix. And, even more starkly, differentiating a second time with respect to the same matrix will also lead to that outcome. So, let's focus on the more relevant non-zero cases. A last reminder before we head off to the calculations, we will again consider this second differentiation with respect to transposed matrices, i.e., $\mathbf{W}^{(ii)\top}$ or $\mathbf{V}^{(i)^\top}$.

$$\frac{\partial \nabla_{\mathbf{V}^{(i)}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\mathrm{LCN}}(\mathbf{x})}{\partial \mathbf{W}^{(ii)\top}} = \mathbf{e}_c \mathbf{x}^{(i)\top} \otimes \mathbf{I}_m\,.$$

$$\frac{\partial \nabla_{\mathbf{W}^{(ii)}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\mathrm{LCN}}(\mathbf{x})}{\partial \mathbf{V}^{(i)^\top}} = \mathbf{I}_m \otimes \mathbf{x}^{(i)} \mathbf{e}_c^\top\,.$$

Let's now take the sum of the above matrices over the input as well as, with appropriate scaling by the residual, over the output indices to get the blocks of the functional Hessian. This yields,

$$\mathbf{H}_{\mathrm{F}}^{(\mathbf{V}^{(i)}, \mathbf{W}^{(ii)})} = \mathbf{\Omega}^{(i)} \otimes \mathbf{I}_m\,. \qquad (51)$$

$$\mathbf{H}_{\mathrm{F}}^{(\mathbf{W}^{(ii)},\mathbf{V}^{(i)})} = \mathbf{I}_m \otimes \mathbf{\Omega}^{(i)\top} . \tag{52}$$

where, $\mathbf{\Omega}^{(i)} = \mathbf{E}\left[\boldsymbol{\delta}_{\mathbf{x},\mathbf{y}}\, \mathbf{x}^{(i)\top}\right] \in \mathbb{R}^{K\times k}$ denotes the (uncentered) covariance of the residual with the i-th input chunk. The rank of the functional Hessian thus amounts to:

$$\begin{aligned}\mathrm{rk}(\mathbf{H}_{\mathrm{F}}) &= t \cdot \left(\mathrm{rk}(\mathbf{H}_{\mathrm{F}}^{(\mathbf{V}^{(i)},\mathbf{W}^{(ii)})}) + \mathrm{rk}(\mathbf{H}_{\mathrm{F}}^{(\mathbf{W}^{(i)},\mathbf{V}^{(ii)})})\right)\\ &= t\cdot 2\,\mathrm{rk}(\mathbf{\Omega}^{(i)} \otimes \mathbf{I}_m)\\ &= t\cdot 2m\,\mathrm{rk}(\mathbf{\Omega}^{(i)}) = t\cdot 2m\min(k,K)\,.\end{aligned}$$

□

C.4.1. MISCELLANEOUS

Comparison with the rank of the bigger FCN. Now, the bigger FCN has number of hidden neurons as $M = m.t$, while sharing the same input and output dimensions.

- $\mathrm{rank}\left(\mathbf{H}_{\mathrm{F}}^{\text{FCN-large}}\right) = 2\,\min(d,K)\,m\,t = 2\,\min(d,K)\,M \;=\; \dfrac{\min(d,K)}{\min(k,K)}\,\mathrm{rank}\left(\mathbf{H}_{\mathrm{F}}^{\text{LCN}}\right).$
- $\mathrm{rank}(\mathbf{H}_{\mathrm{O}}^{\text{FCN-large}}) = q(d+K-q)$ where $q = \min(M,d,K)$.
- $\mathrm{rank}\left(\mathbf{H}_{\mathrm{L}}^{\text{FCN-large}}\right) = q(d+K-q) + 2\,\min\left(d,K\right)M \;-(2s-q)q$, where $q=\min(M,d,K)$ and $s=\min(d,K)$.

Fact 13. *For the LCN in Eqn. (7), we have that* $\mathrm{rank}\left(\mathbf{H}_{\mathrm{L}}^{LCN}\right) = \mathrm{rank}\left(\mathbf{H}_{\mathrm{F}}\right) + \mathrm{rank}(\mathbf{H}_{\mathrm{O}}) - t\cdot(2s-q)q.$

C.5. Rank results for LCN with Weight Sharing

Theorem 10. *For the locally connected network with weight sharing defined in Eqn. (8), the rank of the outer-product and the functional Hessian can be bounded as:* $\mathrm{rk}(\mathbf{H}_{\mathrm{O}}^{LCN+WS}) \leq q(k+Kt-q)$ *, and* $\mathrm{rk}({\mathbf{H}_{\mathrm{F}}}^{LCN+WS}) \leq 2m\min(k,Kt)$*, where* $q=\min(k,Kt,m)$ *and* $t=\dfrac{d}{k}$.

Proof. The proof strategy is similar to the case of 1-hidden layer CNN as presented in the proofs of Theorems 7,8. But over here since we are effectively dealing with a CNN that has stride equal to filter size, i.e., k_1, we will now present simplified proofs by not involving the permutation matrices π_R or π_C.

Outer-product Hessian. Let's compute the gradient of the function with respect to the matrix $\mathbf{V}^{(i)}$:

$$\nabla_{\mathbf{V}^{(i)}} \mathrm{F}_{\boldsymbol{\theta}}^{\text{LCN + WS}}(\mathbf{x}) = \mathbf{I}_K \otimes \mathbf{W}\mathbf{x}^{(i)} \tag{53}$$

And that with respect to $\mathbf{W}$ is:

$$\nabla_{\mathbf{W}^{(ii)}} \mathrm{F}_{\boldsymbol{\theta}}^{\text{LCN+WS}}(\mathbf{x}) = \sum_{i=1}^{t} {\mathbf{V}^{(i)}}^{\top} \otimes \mathbf{x}^{(i)} \tag{54}$$

So the gradient of the function output with respect to all the parameters can be arranged as follows:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathrm{F}_{\boldsymbol{\theta}}^{\text{LCN+WS}}(\mathbf{x}) &= \begin{pmatrix} \nabla_{\mathbf{V}^{(1)}} \mathrm{F}_{\boldsymbol{\theta}}^{\text{LCN+WS}}(\mathbf{x}) \\ \vdots \\ \nabla_{\mathbf{V}^{(t)}} \mathrm{F}_{\boldsymbol{\theta}}^{\text{LCN+WS}}(\mathbf{x}) \\ \nabla_{\mathbf{W}} \mathrm{F}_{\boldsymbol{\theta}}^{\text{LCN+WS}}(\mathbf{x}) \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{W}\mathbf{x}^{(1)} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{W}\mathbf{x}^{(t)} \\ \sum_{i=1}^{t} \mathbf{V}^{(i)\top} \otimes \mathbf{x}^{(i)} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{W} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{I}_K \otimes \mathbf{W} \end{pmatrix} \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{x}^{(t)} \end{pmatrix} \\ \begin{pmatrix} \mathbf{V}^{(1)\top} \otimes \mathbf{I}_k & \cdots & \mathbf{V}^{(t)\top} \otimes \mathbf{I}_k \end{pmatrix} \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{x}^{(t)} \end{pmatrix} \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{W} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{I}_K \otimes \mathbf{W} \\ \mathbf{V}^{(1)\top} \otimes \mathbf{I}_k & \cdots & \mathbf{V}^{(t)\top} \otimes \mathbf{I}_k \end{pmatrix} \begin{pmatrix} \mathbf{I}_K \otimes \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{x}^{(t)} \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} \mathbf{I}_{Kt} \otimes \mathbf{W} \\ \mathbf{V}^{\top} \otimes \mathbf{I}_k \end{pmatrix}}_{\mathbf{A}_o} \underbrace{\begin{pmatrix} \mathbf{I}_K \otimes \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{I}_K \otimes \mathbf{x}^{(t)} \end{pmatrix}}_{\mathbf{B}}
\end{aligned}$$

In the last line, we used the associativity of Kronecker product and the fact that $\mathbf{C} \otimes \mathbf{D} = \big(\mathbf{C}_{\bullet 1} \otimes \mathbf{D} \cdots \mathbf{C}_{\bullet n} \otimes \mathbf{D}\big)$.

The outer-product Hessian can the be calculated by taking the outer-product of the above gradient and averaging over the samples in the dataset. This leads to

$$\mathbf{H}_{\mathrm{O}} = \mathbf{A}_o \left(\mathbf{I}_K \otimes \mathbf{B}_o\right) \mathbf{A}_o^{\top},$$

where, $\mathbf{B}_o$ comes out to be the same matrix as in Eqn. (44) before. Now, let's analyze the rank of the resulting matrices:

$$\begin{aligned}
\mathrm{rk}(\mathbf{H}_{\mathrm{O}}) &= \min(\mathrm{rk}(\mathbf{A}_o), K\,\mathrm{rk}(\mathbf{B}_o)) \leq \min(\mathrm{rk}(\mathbf{A}_o), Kd) \\
&= \min(q \cdot (k + Kt - q), Kd) = q \cdot (k + Kt - q),
\end{aligned}$$

with $q = \min(k, Kt, m)$.

Functional Hessian. We need to differentiate again the network output gradients in Eqns. (53), (54). First, let's just obtain the gradient for the output at the c-th index:

$$\begin{aligned}
\nabla_{\mathbf{V}^{(i)}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\text{LCN+WS}}(\mathbf{x}) &= \left(\mathbf{I}_K \otimes \mathbf{W}\mathbf{x}^{(i)}\right) \mathbf{e}_c \\
&= \mathbf{e}_c \otimes \mathbf{W}\mathbf{x}^{(i)} \\
&= \mathrm{vec}_r \left(\mathbf{e}_c \mathbf{x}^{(i)\top} \mathbf{W}^{\top}\right) \qquad (55)
\end{aligned}$$

And that with respect to $\mathbf{W}$ is:

$$\begin{aligned}\nabla_{\mathbf{W}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\text{LCN+WS}}(\mathbf{x}) &= \sum_{i=1}^{t} \left(\mathbf{V}^{(i)^\top} \otimes \mathbf{x}^{(i)}\right) \mathbf{e}_c \\ &= \sum_{i=1}^{t} \mathbf{V}^{(i)^\top} \mathbf{e}_c \otimes \mathbf{x}^{(i)} \\ &= \sum_{i=1}^{t} \operatorname{vec}_r \left(\mathbf{V}^{(i)^\top} \mathbf{e}_c \mathbf{x}^{(i)\top}\right) \end{aligned} \tag{56}$$

$$\frac{\partial \nabla_{\mathbf{V}^{(i)}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\text{LCN+WS}}(\mathbf{x})}{\partial \mathbf{W}^\top} = \mathbf{e}_c \mathbf{x}^{(i)\top} \otimes \mathbf{I}_m \,.$$

$$\frac{\partial \nabla_{\mathbf{W}} \mathrm{F}_{\boldsymbol{\theta}}^{c,\text{LCN+FS}}(\mathbf{x})}{\partial \mathbf{V}^{(i)^\top}} = \mathbf{I}_m \otimes \mathbf{x}^{(i)} \mathbf{e}_c^\top \,.$$

Let's now take the sum of the above matrices over the input as well as, with appropriate scaling by the residual, over the output indices to get the blocks of the functional Hessian. This yields,

$$\mathbf{H}_{\mathrm{F}}^{(\mathbf{V}^{(i)},\mathbf{W})} = \boldsymbol{\Omega}^{(i)} \otimes \mathbf{I}_m \,. \tag{57}$$

$$\mathbf{H}_{\mathrm{F}}^{(\mathbf{W},\mathbf{V}^{(i)})} = \mathbf{I}_m \otimes \boldsymbol{\Omega}^{(i)\top} \,. \tag{58}$$

where, $\boldsymbol{\Omega}^{(i)} = \mathbf{E}\left[\boldsymbol{\delta}_{\mathbf{x},\mathbf{y}}\, \mathbf{x}^{(i)\top}\right]$ denotes the (uncentered) covariance of the residual with the i-th input chunk.

Hence the rank of the functional Hessian, since it is block hollow, comes out to be:

$$\operatorname{rk}(\mathbf{H}_{\mathrm{F}}) = 2\operatorname{rk}\begin{pmatrix}\boldsymbol{\Omega}^{(1)} \otimes \mathbf{I}_m \\ \vdots \\ \boldsymbol{\Omega}^{(t)} \otimes \mathbf{I}_m\end{pmatrix} = 2\operatorname{rk}(\widetilde{\boldsymbol{\Omega}} \otimes \mathbf{I}_m) = 2m\operatorname{rk}(\widetilde{\boldsymbol{\Omega}}) \le 2m\min(k, Kt)\,.$$

Here in the above line, we can instead consider the rank of $\widetilde{\boldsymbol{\Omega}} = \begin{pmatrix}\boldsymbol{\Omega}^{(1)} \\ \vdots \\ \boldsymbol{\Omega}^{(t)}\end{pmatrix} \in \mathbb{R}^{Kt\times k}$ kroneckered with the identity matrix ($\mathbf{I}_m$) as rank is unaffected by row or column permutations. □

C.6. Miscellaneous

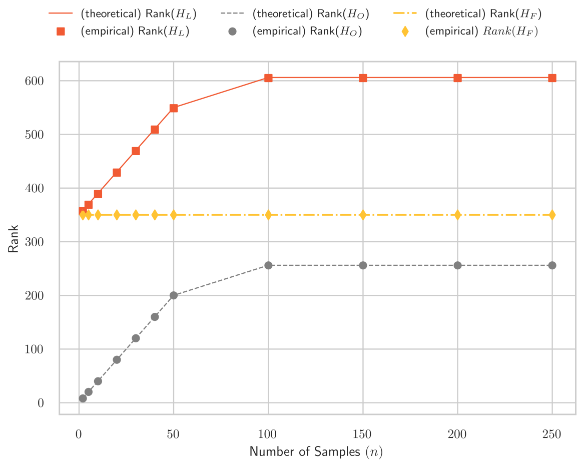
Fact 14. *The rank of the overall loss Hessian has the following formulae*

$$\begin{aligned}\operatorname{rank}\left(\mathbf{H}_{\mathrm{L}}^{\textit{LCN+WS}}\right) = \operatorname{rank}\left(\mathbf{H}_{\mathrm{O}}^{\textit{LCN+WS}}\right) + \\ \operatorname{rank}\left(\mathbf{H}_{\mathrm{F}}^{\textit{LCN+WS}}\right) - (2s-q)q\,,\end{aligned}$$

where $q = \min(k, K, Kt)$ *and* $s = \min(k, Kt)$.

D. Additional Results

D.1. Effect of number of samples n

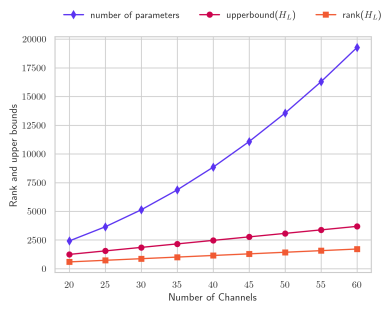


(a) Effect of number of samples

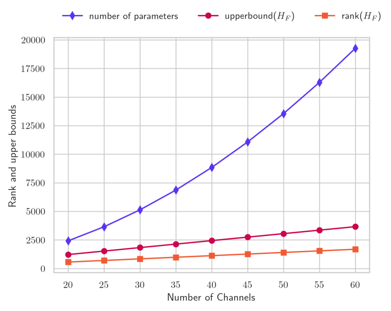
Figure 6: The rank of outer product and loss Hessian increase for a while with increasing number of samples, as until then the covariance matrix has rank n instead of d. Once $n = d$, the ranks remain constant. The rank of the functional Hessian is unaffected throughout.

D.2. Rank vs Number of Channels

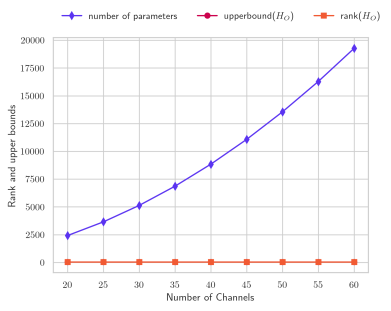
D.2.1. Two-hidden layer CNNs, MSE loss



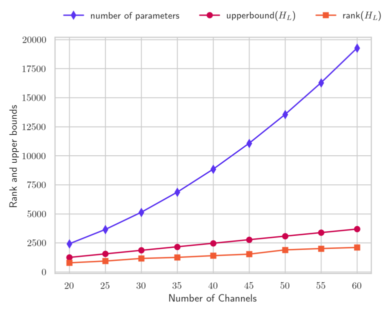
(a) $\mathbf{H}_{\mathrm{L}}$, Linear, CIFAR10



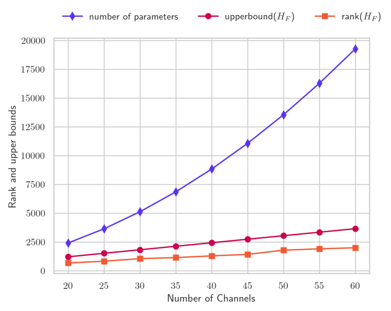
(b) $\mathbf{H}_{\mathrm{F}}$, Linear, CIFAR10



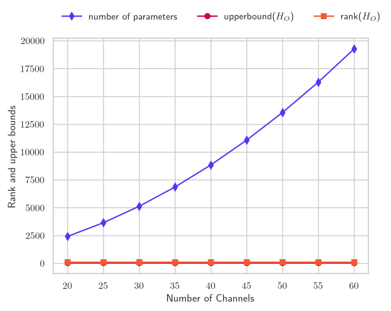
(c) $\mathbf{H}_{\mathrm{O}}$, Linear, CIFAR10



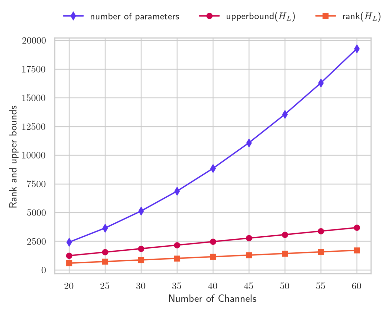
(d) $\mathbf{H}_{\mathrm{L}}$, ReLU, CIFAR10



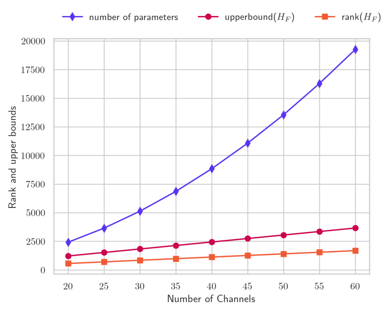
(e) $\mathbf{H}_{\mathrm{F}}$, ReLU, CIFAR10



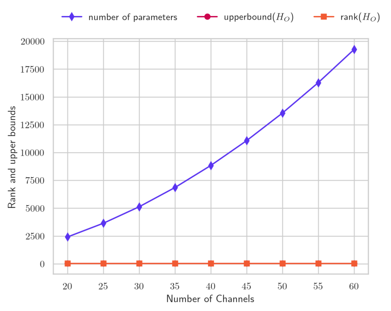
(f) $\mathbf{H}_{\mathrm{O}}$, ReLU, CIFAR10



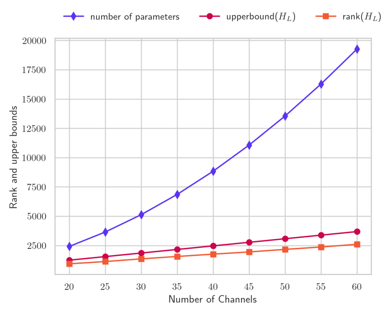
(g) $\mathbf{H}_{\mathrm{L}}$, Linear, Gaussian data



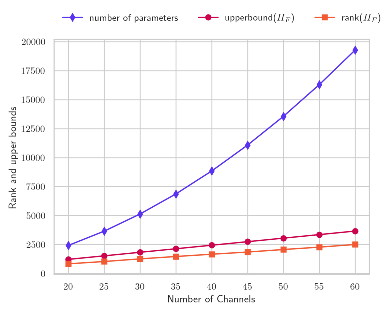
(h) $\mathbf{H}_{\mathrm{F}}$, Linear, Gaussian data



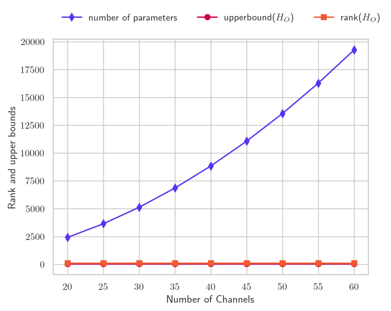
(i) $\mathbf{H}_{\mathrm{O}}$, Linear, Gaussian data



(j) $\mathbf{H}_{\mathrm{L}}$, ReLU, Gaussian data



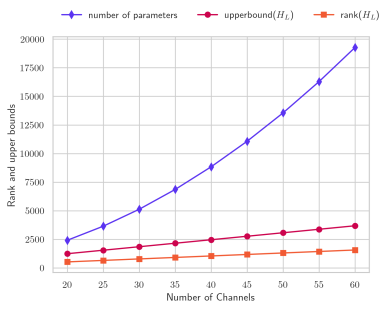
(k) $\mathbf{H}_{\mathrm{F}}$, ReLU, Gaussian data



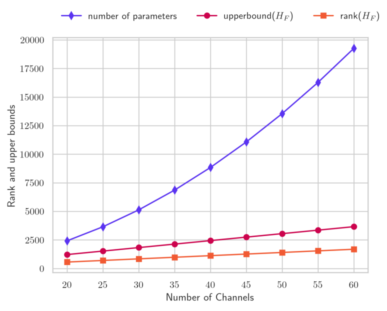
(l) $\mathbf{H}_{\mathrm{O}}$, ReLU, Gaussian data

Figure 7: Rank vs Number of Channels for 2-hidden layer {linear, ReLU}- CNNs on {CIFAR10, random Gaussians} with **Mean Squared Error loss**. The loss, functional, and outer-product Hessian are each shown as separate figures. The bounds for outer-product Hessian exactly coincide with the true values.

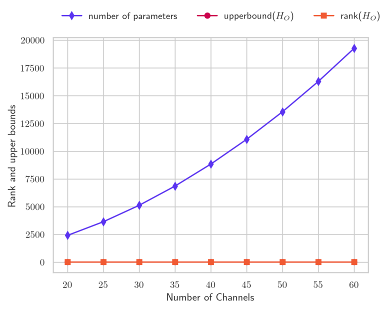
D.2.2. Two-hidden layer CNNs, CE loss



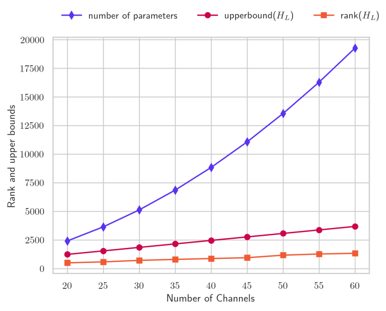
(a) $\mathbf{H}_\mathrm{L}$, Linear, CIFAR10



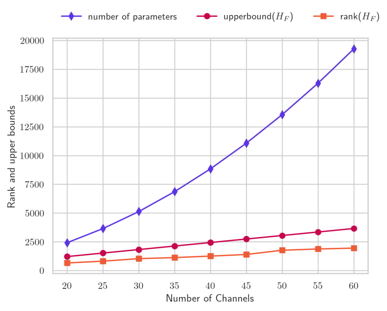
(b) $\mathbf{H}_\mathrm{F}$, Linear, CIFAR10



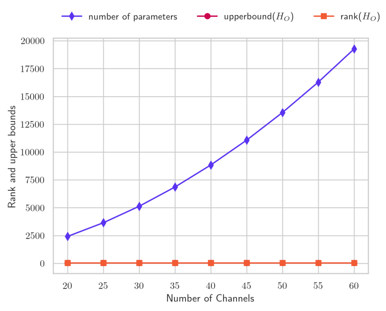
(c) $\mathbf{H}_\mathrm{O}$, Linear, CIFAR10



(d) $\mathbf{H}_\mathrm{L}$, ReLU, CIFAR10



(e) $\mathbf{H}_\mathrm{F}$, ReLU, CIFAR10

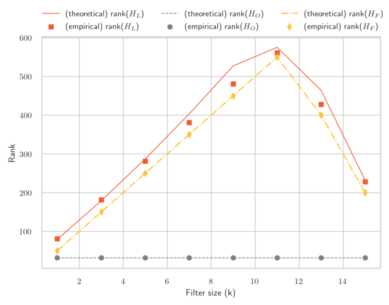


(f) $\mathbf{H}_\mathrm{O}$, ReLU, CIFAR10

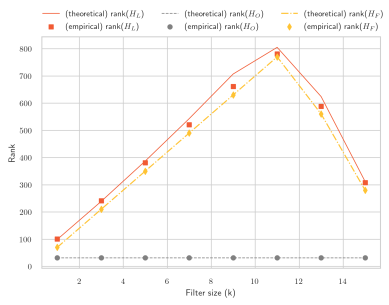
Figure 8: Rank vs Number of Channels for 2-hidden layer {linear, ReLU}- CNNs on CIFAR10 with **Cross-Entropy** loss. The loss, functional, and outer-product Hessian are each shown as separate figures. The bounds for outer-product Hessian exactly coincide with the true values.

D.3. Rank vs Filter Size

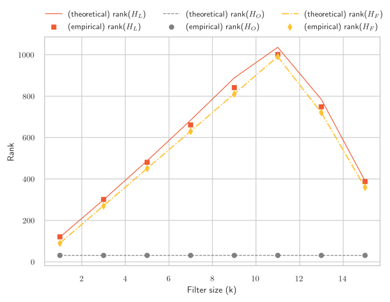
D.3.1. Linear activations, MSE loss



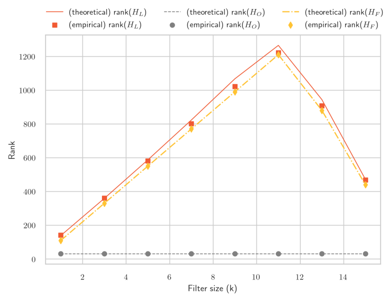
(a) $m = 25$



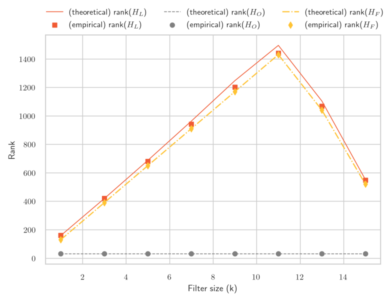
(b) $m = 35$



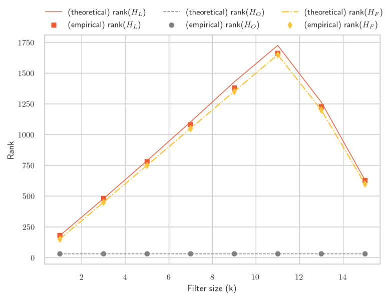
(c) $m = 45$



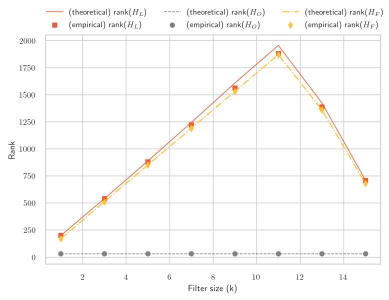
(d) $m = 55$



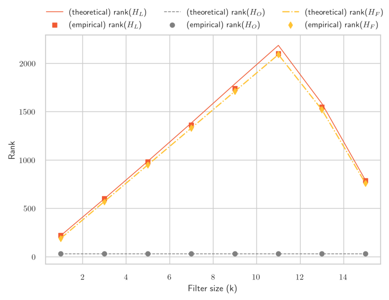
(e) $m = 65$



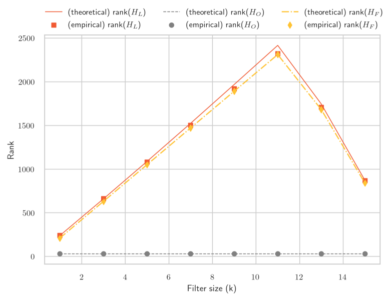
(f) $m = 75$



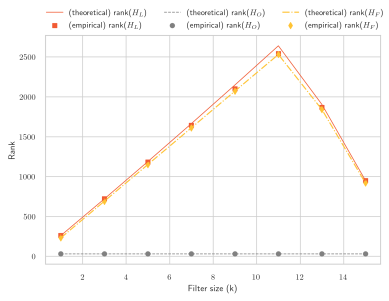
(g) $m = 85$



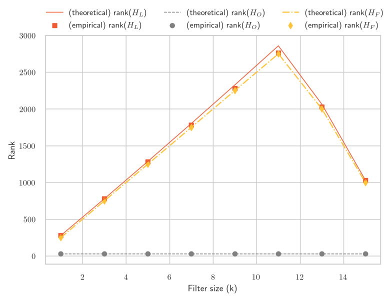
(h) $m = 95$



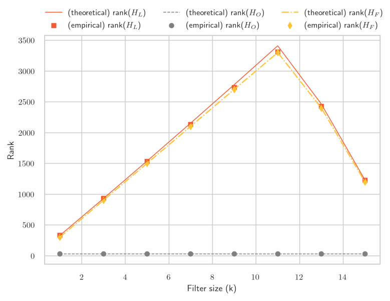
(i) $m = 105$



(j) $m = 115$



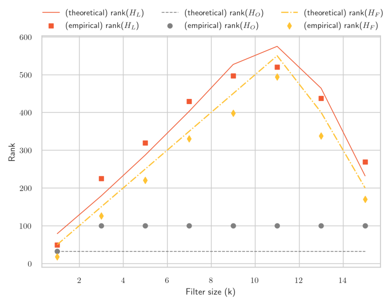
(k) $m = 125$



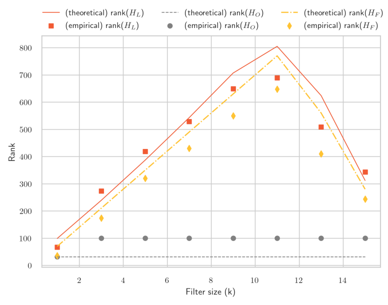
(l) $m = 150$

Figure 9: Rank vs Filter Size for Linear CNN on CIFAR10 with Mean Squared Error loss. The upper bounds are reliable estimator of the true rank values. The rank of the functional and outer-product Hessian, as given by our upper bounds, are exact — the lines and dots coincide perfectly in the plots. The rank of the loss Hessian upper bounded as the sum of the ranks of these two matrices is slightly loose, and this difference becomes negligible for $m \sim 100$.

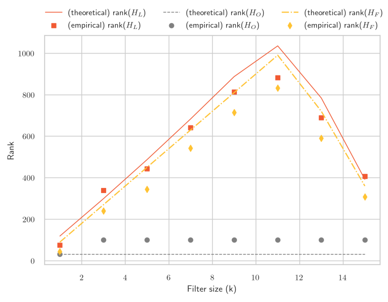
D.3.2. ReLU activations, MSE loss



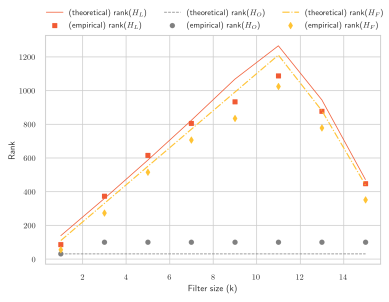
(a) $m = 25$



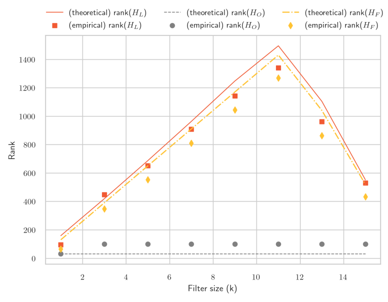
(b) $m = 35$



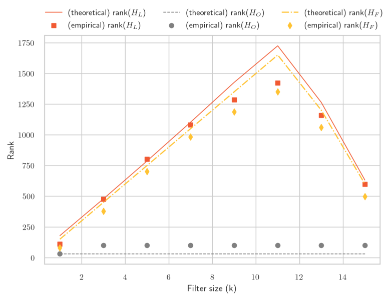
(c) $m = 45$



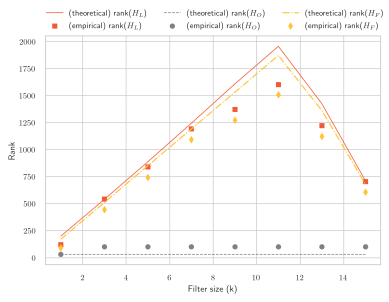
(d) $m = 55$



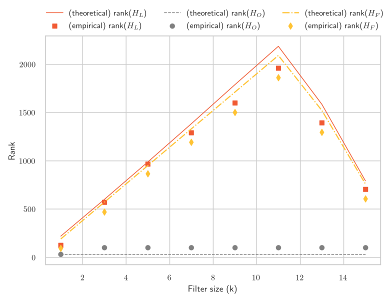
(e) $m = 65$



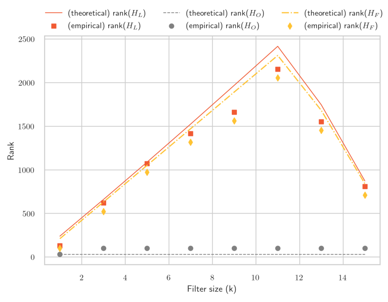
(f) $m = 75$



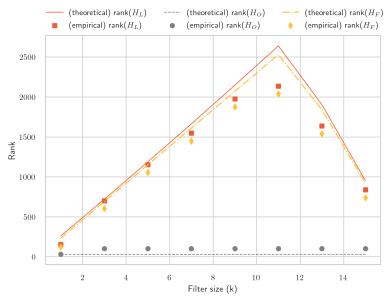
(g) $m = 85$



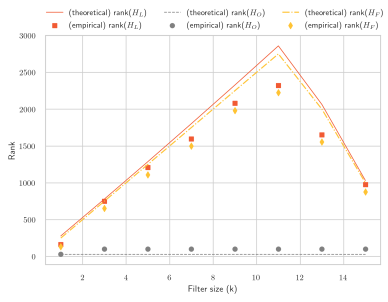
(h) $m = 95$



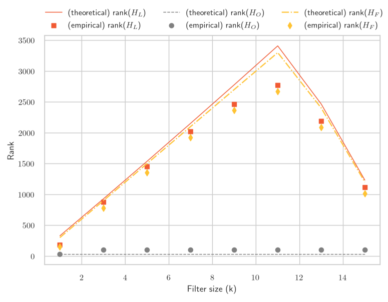
(i) $m = 105$



(j) $m = 115$



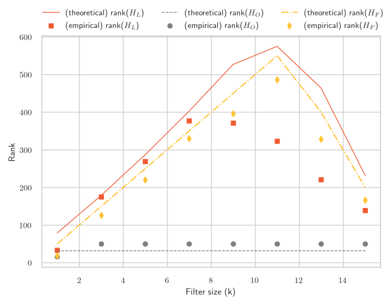
(k) $m = 125$



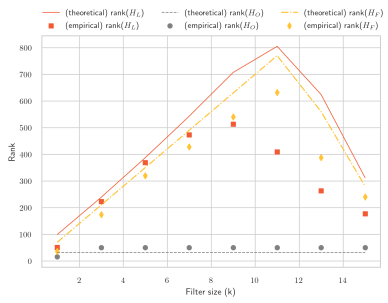
(l) $m = 150$

Figure 10: Rank vs Filter Size for ReLU CNN on CIFAR10 with Mean Squared Error loss. The upper bounds are reliable estimator of the true rank values. The rank of the functional and outer-product Hessian, as given by our upper bounds, are exact — the lines and dots coincide perfectly in the plots. The rank of the loss Hessian upper bounded as the sum of the ranks of these two matrices is slightly loose, and this difference becomes negligible for $m \sim 100$.

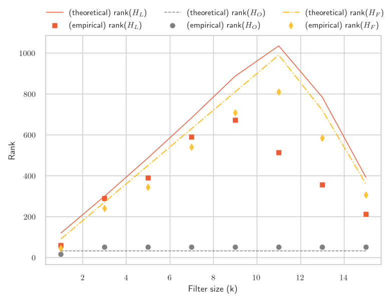
D.3.3. ReLU activations, CE loss



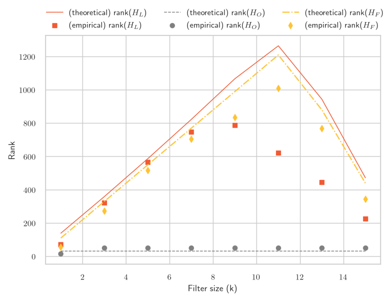
(a) $m = 25$



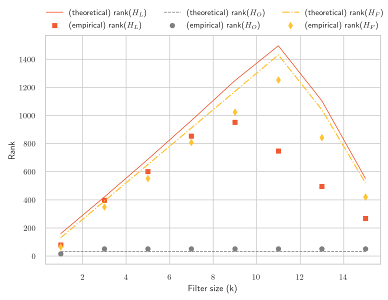
(b) $m = 35$



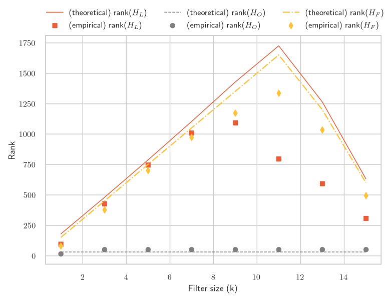
(c) $m = 45$



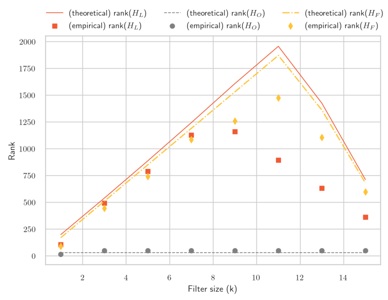
(d) $m = 55$



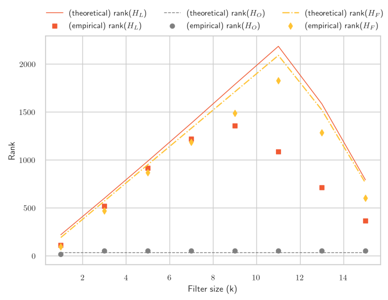
(e) $m = 65$



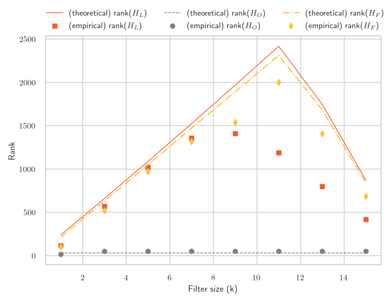
(f) $m = 75$



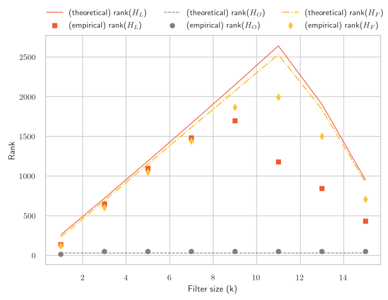
(g) $m = 85$



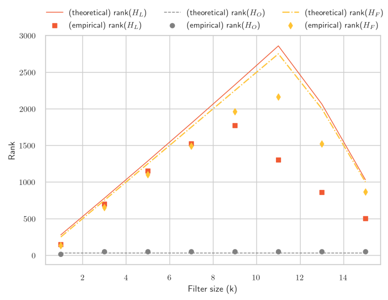
(h) $m = 95$



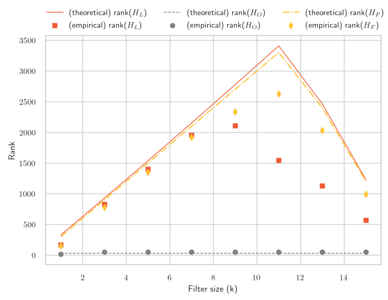
(i) $m = 105$



(j) $m = 115$



(k) $m = 125$

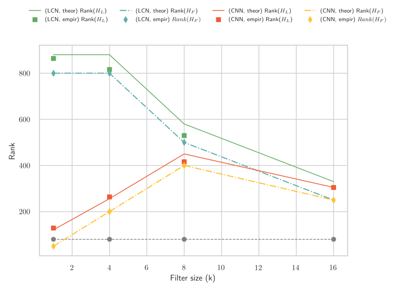


(l) $m = 150$

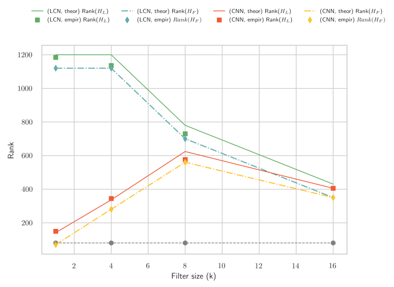
Figure 11: Rank vs Filter Size for ReLU CNN on CIFAR10 with Cross Entropy loss. The upper bounds are reliable estimator of the true rank values. The rank of the functional and outer-product Hessian, as given by our upper bounds, are exact — the lines and dots coincide perfectly in the plots. The rank of the loss Hessian upper bounded as the sum of the ranks of these two matrices is slightly loose, and this difference becomes negligible for $m \sim 100$.

D.4. LCN vs LCN + WS

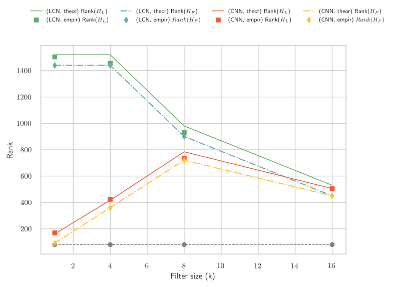
This comparison is done based on Linear activations with MSE loss as a verification of our theory and the dataset used is CIFAR10.



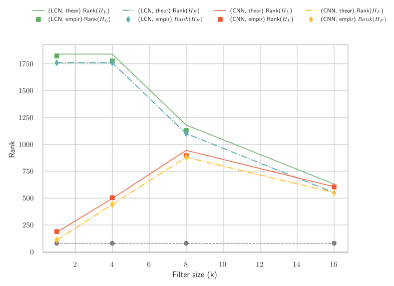
(a) $m = 25$



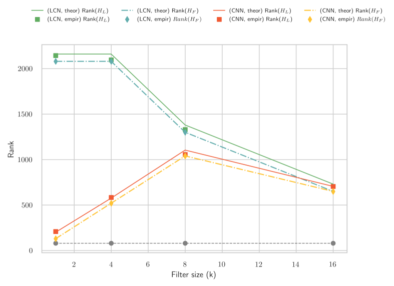
(b) $m = 35$



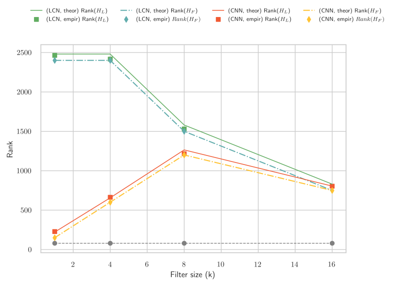
(c) $m = 45$



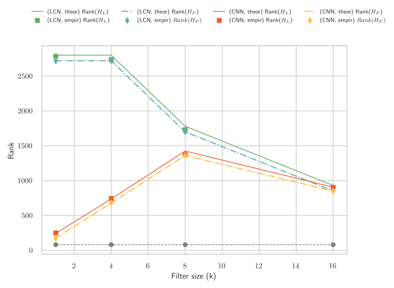
(d) $m = 55$



(e) $m = 65$



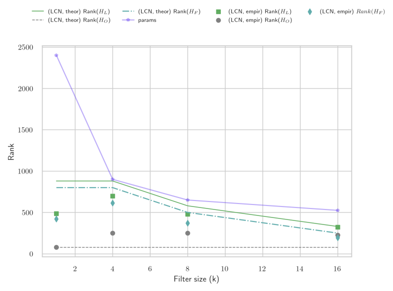
(f) $m = 75$



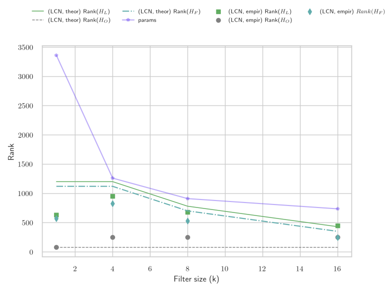
(g) $m = 85$

Figure 12: Rank vs Filter Size for Linear LCN and LCN + WS on CIFAR10 with Mean Squared Error loss.

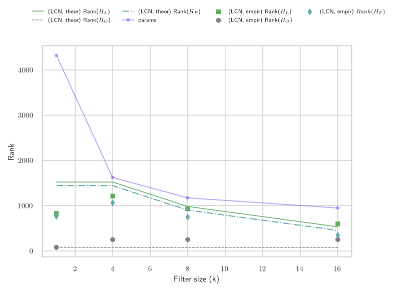
D.5. ReLU-based LCNs



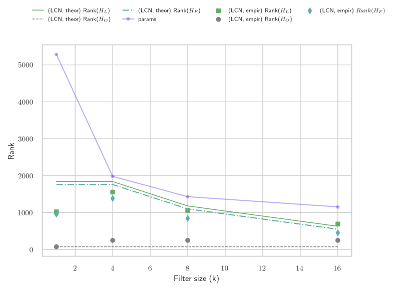
(a) $m = 25$



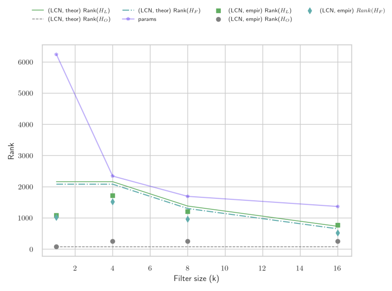
(b) $m = 35$



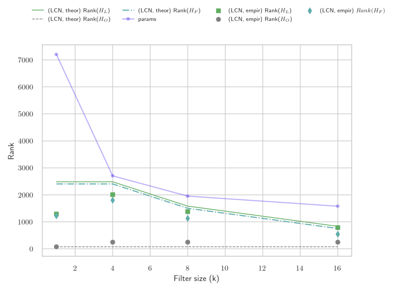
(c) $m = 45$



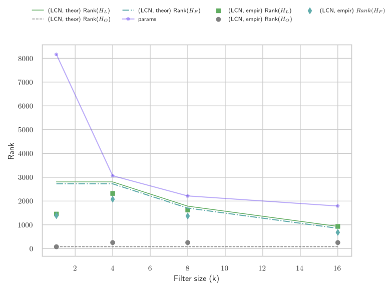
(d) $m = 55$



(e) $m = 65$



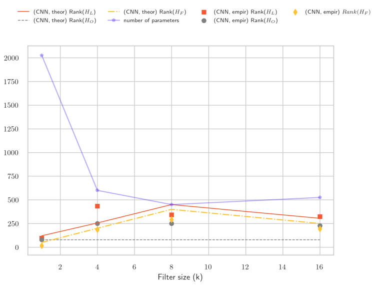
(f) $m = 75$

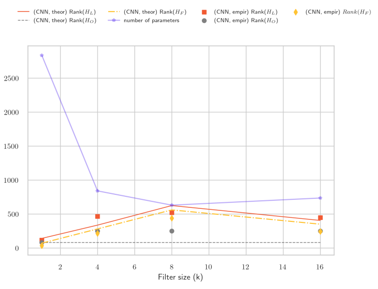


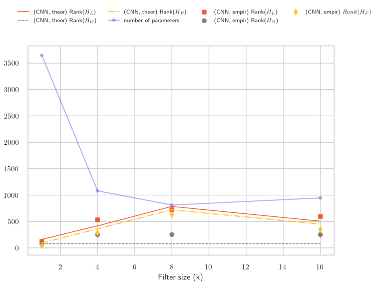
(g) $m = 85$

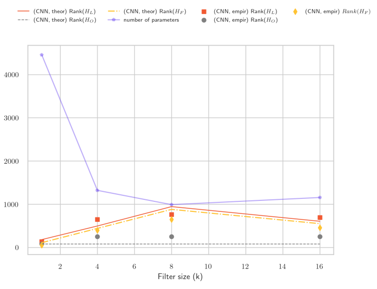
Figure 13: Rank vs Filter Size for ReLU LCNs on CIFAR10 with Mean Squared Error loss.

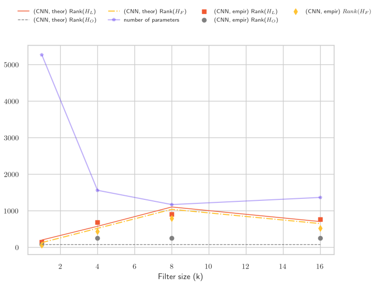
D.6. ReLU-based LCNs + Weight Sharing

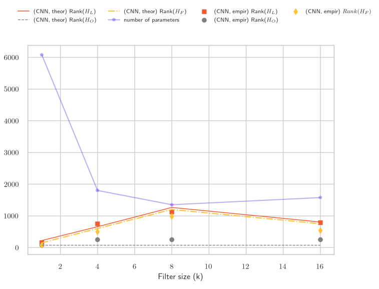

(a) $m = 25$


(b) $m = 35$


(c) $m = 45$


(d) $m = 55$


(e) $m = 65$


(f) $m = 75$

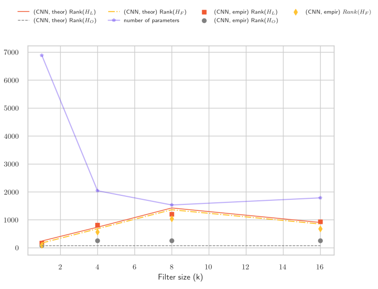

(g) $m = 85$

Figure 14: Rank vs Filter Size for ReLU-based LCN + WS on CIFAR10 with Mean Squared Error loss.

D.7. Correlation of Rank with Generalization Performance

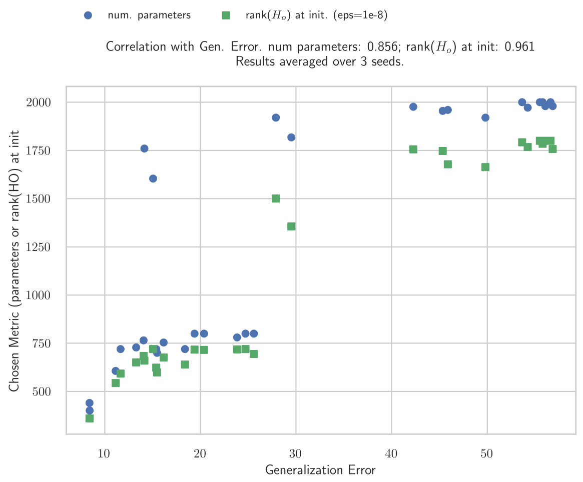
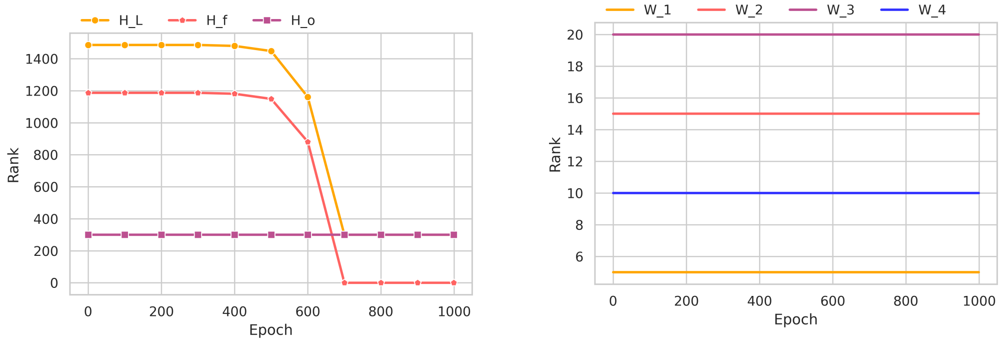


Figure 15: Correlation of Hessian rank at initialization with generalization error for a sweep over filter sizes and number of channels in a one-hidden layer CNN.

D.8. Dynamics of rank of the Hessian and convolutional layers during training



(a) Evolution of rank of convolutional filters

Figure 16: Dynamics of the rank of the Hessian while training a four-hidden layer linear CNN.