Using Degeneracy in the Loss Landscape for Mechanistic Interpretability

Lucius Bushnaq [1] **Jake Mendel** [1] **Stefan Heimersheim** [1] **Dan Braun** [1] **Nicholas Goldowsky-Dill** [1] **Kaarel Hänni** [2] **Cindy Wu** [3] **Marius Hobbhahn** [1]

Abstract

Mechanistic Interpretability aims to reverse engineer the algorithms implemented by neural networks by studying their weights and activations. An obstacle to reverse engineering neural networks is that many of the parameters inside a network are not involved in the computation being implemented by the network. These degenerate parameters may obfuscate internal structure. Singular Learning Theory teaches us that neural network parameterizations are biased towards being more degenerate, and parameterizations with more degeneracy are likely to generalize further. We identify 3 ways that network parameters can be degenerate: linear dependence between activations in a layer; linear dependence between gradients passed back to a layer; ReLUs which fire on the same subset of datapoints. We propose that if we can represent a neural network in a way that is invariant to reparameterizations that exploit the degeneracies, then this representation is likely to be more interpretable. We introduce the Interaction Basis, a tractable technique to obtain a representation that is invariant to degeneracies from linear dependence of activations or Jacobians.

1. Introduction

Mechanistic Interpretability aims to understand the algorithms implemented by neural networks (Olah et al., 2017; Elhage et al., 2021; Räuker et al., 2023; Olah et al., 2020; Meng et al., 2023; Geiger et al., 2021; Wang et al., 2022; Conmy et al., 2024). A key challenge in mechanistic interpretability is that neurons tend to fire on many unrelated inputs (Fusi et al., 2016; Nguyen et al., 2016; Olah et al., 2017; Geva et al., 2021; Goh et al., 2021) and any apparent circuits in the model often do not show a single clear functionality and do not have clear boundaries separating them from the rest of the network (Conmy et al., 2023; Chan et al., 2022).

We suggest that a central problem for current methods of reverse engineering networks is that neural networks are *degenerate*: there are many different choices of parameters that implement the same function (Wei et al., 2022; Watanabe, 2009). For example, in a transformer attention head, only the product $W_{OV} = W_O W_V$ of the W_V and W_O matrices influences the network's output; thus, many different choices of W_O and W_V are parameterizations of the same network (Elhage et al., 2021). This degeneracy makes parameters and activations an obfuscated view of a network's computational features, hindering interpretability. While we have workarounds for known architecture-dependent degeneracies such as the W_{OV} case, Singular Learning Theory (SLT, Watanabe, 2009; 2013) suggests that we should expect *additional* degeneracy in trained networks that generalize well.

SLT quantifies the degeneracy of the loss landscape around a solution using the local learning coefficient (LLC) (Lau et al., 2023; Watanabe, 2009; 2013). More degenerate solutions lie in broader 'basins' of the loss landscape, where many alternative parameterizations implement a similar function. Networks with lower LLCs are more degenerate, implement more general algorithms, and generalize better to new data (Watanabe, 2009; 2013). These predictions of SLT are only straightforwardly applicable to the global minimum in the loss landscape; a generalization is required to apply these insights to real networks.

In this paper we make the following contributions. First, in Section 2 we propose changes to SLT to make it useful for interpretability on real networks. Then, in Section 3 we characterize three ways in which neural networks can be degenerate. Finally in Section 4, we propose a practical technique for removing some of these degeneracies in the form of the interaction basis.

Additionally, in Appendix A, we prove a link between some of these degeneracies and sparsity in the interactions between features, and in Appendix B, we develop a technique for searching for modularity based on its relation to degeneracy in the loss landscape.

[1] Apollo Research [2] Cadenza Labs [3] Independent. Correspondence to: Lucius Bushnaq <lucius@apolloresearch.ai>.

2. Modifying Singular Learning Theory for interpretability

If a neural network's parameterisation is degenerate, this means there are many choices of parameters that achieve the same loss. At a global minimum in the loss landscape, more degeneracy in the parametrisation implies that the network lies in a broader basin of the loss. We can quantify how broad the basin is using Singular Learning Theory (SLT, Watanabe 2009; 2013; Wei et al. 2022). Appendix C provides a brief review of overview of key concepts from the SLT literature hat we will make use of.

The most important quantity in SLT for our purposes is the *local learning coefficient* (LLC, denoted by $\hat{\lambda}$). We define a data distribution $x \sim X$ and a family of models with N parameters, parameterised by a vector θ in a parameter space $\Theta \subseteq \mathbb{R}^N$. We also define a population loss function $L(\theta|X)$ which is normalised so that $L(\theta_0|X) = 0$ at global minima $\theta_0 = \arg\min_\theta L(\theta|X)$. Then $\hat{\lambda}$ is defined as (Lau et al., 2023):[1]

$$\hat{\lambda}(\theta^*) = \lim_{\epsilon \to 0} \left[\epsilon \frac{\mathrm{d}}{\mathrm{d}\epsilon} \log \mathrm{V}_{\theta^*}(\epsilon) \right] . \tag{1}$$

where $\mathrm{V}(\epsilon)$ is the *volume* of parameter space with loss less than ϵ in a neighbourhood $\Theta_{\theta^*} \subset \Theta$ of a particular minimum θ^*.:

$$\mathrm{V}_{\theta^*}(\epsilon) = \int_{\{\theta \in \Theta_{\theta^*} :\, L(\theta) < L(\theta^*) + \epsilon\}} \mathrm{d}\theta \tag{2}$$

We would like to use the LLC to quantify the number of degrees of freedom in the parameterisation of a neural network — the number of ways the parameters in a neural network can be changed while still implementing the same function, or at least a highly similar function. However, there are some obstacles to using the LLC for this purpose:

1. The LLC $\hat{\lambda}(\theta^*)$ measures the size of the region of equal loss around a particular local minimum $\theta^* \in \Theta$ in the loss landscape. This loss landscape is defined by a loss function and a dataset of inputs and labels. Unless the network achieves optimal loss on this dataset, points in the region could have equal loss even though they correspond to *different* functions, if these functions achieve the same average performance over the dataset. We do not want our measure of the number of degrees of freedom to include different functions which achieve the same overall loss.

2. The local learning coefficient is only well defined at a local minimum of the loss, but we frequently want to interpret neural networks that have not been trained to convergence and are not at a minimum of the loss on their training distribution.

3. We would like to be able to consider two very similar but not identical functions to be the same function, if they only differ in ways that can be considered noise. This is partially because, after finite training time, a network will not have fully converged on the cleanest version of an algorithm without any noise[2]. However, the formal approach of SLT defines the LLC in the limit of infinite training data. This means that the LLC only contains information about exact degeneracies in the parameterization. Instead, we would prefer to work with a modified LLC which quantifies the number of parameterization choices which correspond to approximately identical functions.

We introduce the *behavioral loss* as a resolution to problems (1) and (2) in Section 2.1, and *finite data SLT* as a resolution to problem (3) in Section 2.2. Together, these concepts will allow us to define the *effective parameter count*, a measure of the number of computationally-relevant parameters in the network. If we achieved our goal of a fully parameterisation-invariant representation of a neural network, its explicit parameter count would equal its effective parameter count.

2.1. Behavioral loss

In this section, we describe how we can define the local learning coefficient of a network to avoid problems 1 and 2 listed above. We want to define a new loss function and corresponding loss landscape for the sake of the SLT formalism (we do not train with this loss) such that all the parameter choices in a region with zero loss correspond to *the same function* on the training dataset: the same map of inputs to outputs. This loss function, which we call the Behavioral Loss, L_B, is defined with respect to an original neural network with an original set of parameters θ^*, and defines how similar the function $\mathbf{f}_\theta$ implemented by a different set of parameters θ is to the original function $\mathbf{f}_{\theta^*}$:

$$L_B(\theta|\theta^*, \mathcal{D}) = \frac{1}{n} \sum_{x \in \mathcal{D}} ||\mathbf{f}_\theta(x) - \mathbf{f}_{\theta^*}(x)||^2 \tag{3}$$

where $\mathcal{D}$ is the training dataset and $||\mathbf{v}||$ denotes the ℓ^2-norm of $\mathbf{v}$[3]. By definition, this loss landscape always has a global minimum at the parameters the model actually uses $\theta = \theta^*$, solving problem 2 above. Additionally, parameter choices

[1] See the original paper for a more rigorous definition of the LLC.

[2] Indeed, sometimes it is possible to remove this noise and improve performance (Nanda et al., 2023)

[3] We arbitrarily chose an MSE loss here, but conceptually we require a loss which is *non-negative* and satisfies *identity of indiscernibles*: $L = 0 \iff \forall x : \mathbf{f}_\theta(x) = \mathbf{f}_{\theta^*}(x)$. For example, when studying an LLM, it may be more suitable to use KL-divergence.

which achieve 0 behavioral loss must have the same input-output behaviour as $\mathbf{f}^*_\theta$ on the entire training dataset, solving problem 1. Note that achieving zero behavioral loss relative to a model with parameters θ^* is a stricter requirement than achieving the same loss as the model with parameters θ^* on the training data. Therefore, the behavioral loss LLC $\hat{\lambda}_B$ will be equal to, or higher than the training loss LLC $\hat{\lambda}$.

2.2. Singular learning theory at finite data

Next we want to resolve the problem that standard SLT formulae concern only the limit of infinite data when the model is certainly at a local minimum of the loss landscape. We would like to think of a neural network trained on a finite amount of data as implementing a core algorithm we are interested in reverse engineering, plus some amount of 'noise' which may vary with the parameterisation and which is not important to interpret. For example, in a modular addition transformer (Nanda et al., 2023), there are parts of the network which can be ablated to *improve* loss: these parts of the network may be present because the model has not fully converged to a minimum yet. In this case, if we have two transformers trained on modular addition which have the same input-output behaviour *after* we have ablated parts to improve performance, then we would like to consider these models as implemtenting the same function 'up to' noise *before* we ablate those parts.

In this section, we sketch how to modify SLT so that the LLC becomes a measure of how many different parameterisations implement *nearly* the same function, rather than exactly the same function. In this way, we can numerically vary how much the functions two different parameterisations implement are allowed to differ from each other on the training data.

We start by explaining why SLT takes the limit $\epsilon \to 0$ in the definition of the LLC (equation 1). SLT is a theory of Bayesian learning machines: learning machines which start with some prior over parameters which is nonzero everywhere $\varphi : \Theta \mapsto (0, 1)$, and which learn by performing a Bayesian update on each datapoint they observe. After a dataset $\mathcal{D}_n$ of n datapoints, the posterior distribution over parameters is:

$$p(\theta|\, \mathcal{D}_n) = \frac{e^{-nL(\theta|\, \mathcal{D}_n)}\varphi(\theta)}{p(\mathcal{D}_n)}\,. \qquad (4)$$

where $L(\theta|\, \mathcal{D}_n)$ is the negative log likelihood of the dataset given the model $\mathbf{f}_\theta$, which we identify with the loss function when making a connection between Bayesian learning and SGD (Murphy, 2012), and $p(\mathcal{D}_n)$ is a normalisation factor.

The exponential dependence on n ensures that in the limit $n \to \infty$, a Bayesian learning machine's posterior is only nonzero at points of minimum loss. This means that the asymptotic behaviour of the learning machine depends only on properties of the loss landscape that are asymptotically close to having zero loss. This is the reason that we take $\epsilon \to 0$ in the definition of the learning coefficient.

However, since the parameters θ^* we find after finite steps of SGD correspond to an algorithm plus noise, we want to consider the size of the region of parameter space that achieves a behavioral loss *less than the noise size*. From a bayesian learning perspective, in equation 4, we can see that for large but finite number of data points, most of the posterior concentrates around the regions of low loss, but it does not fully concentrate on the region with exactly minimum loss.

Therefore, we simply refrain from taking the limit as the loss scale ϵ goes to 0 in the definition of the learning coefficient, and consider the learning coefficient at a particular loss scale:

$$\lambda(\epsilon) := \epsilon \frac{\mathrm{d}}{\mathrm{d}\epsilon} \log \mathrm{V}(\epsilon) \qquad (5)$$

To understand how the learning coefficient can vary with epsilon, consider an illustrative example: an extremely simple setup with a single parameter $w \in \mathbb{R}$, and a loss function $L(w) = c^2w^2 + w^4$ with $c \ll 1$. This is a toy model of a scenario where there is a very small quadratic term in the learning coefficient. This term is only 'visible' to the learning coefficient when we zoom in to very small loss values. To see this, we must the calculate how the local volume (equation 2) depends on the loss scale ϵ. For large $\epsilon \gg c^{\frac{1}{4}}$, the quartic term dominates the loss and the region of loss less than ϵ is roughly the interval $[-\epsilon^{\frac{1}{4}}, \epsilon^{\frac{1}{4}}]$. This gives $\mathrm{V}(\epsilon) \approx 2\epsilon^{\frac{1}{4}}$ so the learning coefficient is $\lambda(\epsilon \gg c^{\frac{1}{4}}) = \frac{1}{4}$, the same as if the quadratic term were not present. On the other hand, for small enough $\epsilon \ll c^{\frac{1}{4}}$, the quadratic term becomes visible: $\mathrm{V}(\epsilon) \approx 2\epsilon^{\frac{1}{2}}/c^2$, so $\lambda(\epsilon \ll c^{\frac{1}{4}}) = \frac{1}{2}$.

Determining how to choose an appropriate cutoff ϵ is still an open problem. We suggest that researchers choose the value of the behavioral loss cutoff in the context of the question they would like to answer. For example, if one trains multiple models with different seeds on the same task, then the appropriate loss cutoff may be on the order of the variance between the seeds.

Finally, we are able to quantify the amount of degeneracy in a neural network. We define the *Effective Parameter Count* of a neural network $\mathbf{f}_{\theta^*}$ at noise scale ϵ as two times the local learning coefficient $\lambda_B(\epsilon)$ of the behavioral loss with respect to the network at noise scale epsilon.

$$N_{\text{eff}}(\epsilon) := 2\lambda_B(\epsilon) \qquad (6)$$

We conjecture that a fully parameterisation invariant representation of a neural network which captures all the behaviour up to noise scale ϵ would require $N_{\text{eff}}(\epsilon)$ parameters.

3. Internal structures that contribute to degeneracy

In this section, we will show three ways the internal structure of neural networks can induce degrees of re-parametrization freedom N_{free} in the loss landscape. Since $N_{\text{eff}} = N - N_{\text{free}}$, this is equivalent to showing three ways the internal structures of neural networks determine their effective parameter count. We do not expect that these three sources of re-parametrization freedom offer a complete account of all degeneracy in real networks. They are merely a starting point for relating the degeneracy of networks to their computational structure at all.

For ease of presentation, most of the expressions in this section are only derived for the example case of fully connected networks. They can be generalized to transformers, though we do not show this explicitly here.

In Section 3.1, we show a relationship between the effective parameter count and the *dimensions* of the spaces spanned by the network's activation vectors (Section 3.1.1) and Jacobians (Section 3.1.2) recorded over the training data. In Section 3.2, we show a relationship between the number of *distinct nonlinearities* implemented in a layer of the network on the training data and the effective parameter count.

3.1. Activations and Jacobians

In this section, we show how a network having low dimensional hidden activations or Jacobians leads to re-parametrisation freedom.

We begin by bringing the network's Hessian, which gives the first non-zero term in the Taylor expansion of the loss around an optimum (see Appendix C) into a more convenient form. Each local free direction in the loss landscape corresponds to an eigenvector of the Hessian with zero eigenvalue.[4] Therefore, the rank of the Hessian can be used to obtain a lower bound for the learning coefficient.

Consider the Hessian of a fully connected network, with parameters at the global miniumum $\theta = \theta^*$ and behavioural loss $L_B\left(\theta|\theta^*, \mathcal{D}\right)$. Using the chain rule, the Hessian can be written:

$$\left.\frac{\partial^2 L_B\left(\theta|\theta^*, \mathcal{D}\right)}{\partial\theta^l_{i,j}\partial\theta^{l'}_{i',j'}}\right|_{\theta=\theta^*}$$
$$= \sum_{x\in\mathcal{D}}\sum_{k,k'} \left.\frac{\partial^2 L_B\left(\theta|\theta^*, \mathcal{D}\right)}{\partial f^{l_{\text{final}}}_k \partial f^{l_{\text{final}}}_{k'}}\right|_{\theta=\theta^*} \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial\theta^{l'}_{i',j'}} \frac{\partial f^{l_{\text{final}}}_{k'}(x)}{\partial\theta^{l}_{i,j}}$$

As our loss is MSE [5] and derivatives are 0 at the minimum this equals

$$\frac{1}{2n}\sum_{x\in\mathcal{D}}\sum_{k} \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial\theta^{l'}_{i',j'}} \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial\theta^{l}_{i,j}} \tag{7}$$

for $l = 1, \dots l_{\text{final}}$; $j = 1, \dots, d^l$; $i = 1, \dots, d^{l+1}$. Thus, the Hessian is equal to a Gram matrix of the network's weight gradients $\frac{\partial f^{l_{\text{final}}}_k}{\partial\theta^l_{i,j}}$, and linear dependence of entries of the weight gradients over the training set $\mathcal{D}$ corresponds to zero eigenvalues in the Hessian.

We can apply the chain rule again to rewrite the gradient vector on each datapoint as an outer product of Jacobians and activations:

$$\frac{\partial f^{l_{\text{final}}}_k(x)}{\partial\theta^l_{i,j}} = \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial p^{l+1}_i} f^l_j(x) \tag{8}$$

where the Jacobian is taken with respect to preactivations to layer $l+1$:

$$\mathbf{p}^{l+1}(x) = W^l \mathbf{f}^l(x).^{6} \tag{9}$$

Thus, every degree of linear dependence in the activations f^l_j or Jacobians $\frac{\partial f^{l_{\text{final}}}_k}{\partial p^{l+1}_i}$ in a layer l of the network also causes degrees of linear dependence in the weight gradient $\frac{\partial f^{l_{\text{final}}}_k(x)}{\partial\theta^l_{i,j}}$ of the network, potentially resulting in re-parametrisation freedom for the network. In the next two sections, we explore how linear dependence in the activations and Jacobians respectively impact the effective parameter count.

3.1.1. Activation vectors spanning a low dimensional subspace

Looking at equation 8, each degree of linear dependence of the activations f^l_j in a hidden layer l of width d^l over the training dataset $\mathcal{D}$,

$$\sum_j c_j f^l_j(x) = 0 \; \forall x \in \mathcal{D}\,, \tag{10}$$

corresponds to d^{l+1} linearly dependent entries in the weight gradient $\frac{\partial f^{l_{\text{final}}}_k}{\partial\theta^l_{i,j}}$, d^{l+1} eigenvectors of the Hessian with eigenvalue zero, and d^{l+1} fully independent free directions in the loss landscape than span a fully free d^{l+1} dimensional hyperplane. So the effective parameter count N_{eff} will be lower than the nominal number of parameters in the model N by d^{l+1} for each such degree of linear dependence in the hidden representations.

[4] The reverse does not hold, due to higher order terms in the expansion. See (Watanabe, 2009; 2013).

[5] For different behavioral losses such as KL divergence, the expression would look slightly different, but the story of this section would be largely the same.

[6] Throughout this paper, we have added the bias into a new zeroth column of the weights and added a zeroth coordinate to the activation vector, so that $W^l_{i,0} = b^l_i$ and $\forall x : f^l_0(x) = 1$. Now $\sum_{j=0}^d W^l_{ij} f^l_j(x) = \sum_{j=1}^d W^l_{ij} + b^l_i = p^{l+1}_i$

More generally, we can take a PCA of the activation vectors in layer l by diagonalising the Gram matrix of activations

$$G^l := \frac{1}{n}\sum_{x\in\mathcal{D}} \mathbf{f}^l(x)\mathbf{f}^l(x)^T \quad =: {U^l}^T D^l_G U^l \tag{11}$$

If there is linear dependence between the activations on the dataset, some of the singular values (eigenvalues of G^l) will be zero. If we transform into rotated layer coordinates $\tilde{\mathbf{f}}^l(x) = U^l\mathbf{f}^l(x), \tilde{W}^l = W^l {U^l}^T$, then the parameters of the transformed weight matrix in rows which connect to the directions with zero variance can be changed freely without changing the product $\tilde{W}^l\tilde{\mathbf{f}}^l$.

In reality, a gram matrix of activation vectors will never have eigenvalues that are exactly 0. However, if a particular eigenvalue has size $\sqrt{\frac{1}{n}\sum_{x\in\mathcal{D}}\left(\tilde{f}^l_j(x)\right)^2} = O(\epsilon^k)$ for some $\epsilon \ll 1$, changing the transformed parameters inside $\tilde{W}^l$ by $O(\epsilon^{\frac{1}{2}-k})$ only impacts the loss L by $O(\epsilon)$.

This suggests that, under the finite-data SLT picture introduced in Section 2.2, singular values of the set of activation vectors that are less than $\epsilon^{\frac{1}{2}}$ for noise scale ϵ result in a lower effective parameter count, with d^{l+1} effective parameters less for every small singular value. So, if we view the PCA components in a layer l as the 'elementary variables' of that layer, then the fewer elementary variables the network has in total, the lower the effective parameter count will be.

The work presented in this section is related to (Elisenda Grigsby et al., 2023). They studied similar degeneracies in an MLP-only ReLU-only setting. Our work applies to other architectures and activation functions, and considers the behavior on the training distribution rather than arbitrary inputs.

Relationship to weight norm One might be concerned that linear dependencies between the activation vectors on the training dataset might not hold for activation vectors outside the training dataset, such that the entries of the weight matrix that we are treating as free do in fact affect the *off-distribution* outputs of the network.

However, SOTA optimisers often use weight decay or ℓ^2 weight regularisation during training to improve network generalization (Loshchilov and Hutter, 2019). This biases training towards networks with a smaller total ℓ^2-weight norm, $||\theta||_2 = \sum_{l=1}^{l_{\text{final}}} ||W^l||_F$. Since the Frobenius norm $||W^l||_F$ is invariant under orthogonal transformations, this is equivalent to biasing training towards low $||\tilde{W}^l||_F$. Since the entries of $\tilde{W}^l$ which connect to the zero principal components do not affect the output, the training will be biased to push them to 0. This is an example of weight regularisation improving generalisation performance: if, at inference time, an activation vector has variation in a direction not seen during training, a regularised model ignores that component.

3.1.2. Jacobians spanning a low dimensional subspace

We have shown that if the set of activation vectors in some layer have linear dependence over a dataset, then some parameters are free to vary without affecting outputs on that dataset. A similar story can be told when the Jacobians $J^l_{ij} = \frac{\partial f_i^{l_{\text{final}}}(x)}{\partial p_j^{l+1}}$ do not span the full space of the layer. As with the activations, we look for zero eigenvalues in the gram matrix of the Jacobians:

$$K^l := \frac{1}{n}\sum_{x\in\mathcal{D}}\sum_{j} {J^l}^T J^l =: {R^l}^T D^l_P R^l \tag{12}$$

Any zero eigenvalue in this gram matrix leads to d^l zero eigenvalues in the Hessian, analogous to the previous section. We can transform into rotated layer coordinates $\tilde{W}^l = R^l W^l$, $\tilde{J}^l = J^l R^l {R^l}^T$ and the parameters of the transformed weight matrix in columns which connect to the directions with zero variance can be changed freely without changing the product $\tilde{J}^l\tilde{W}^l$. However, unlike with the activation PCA components, the d^l free directions in the Hessian from Jacobians spanning a low-dimensional subspace may not always correspond to d^l full degrees of freedom in the parametrization. This is due to the potential presence of terms above second order in the perturbative expansion around the loss optimum, which can cause the loss to change if the parameters are varied along those directions despite the Hessian being zero (Watanabe, 2009).

Jacobians between hidden layers Note that we can decompose each Jacobian from layer l to layer l_{final} into a product of Jacobians between adjacent layers by the chain rule:

$$\frac{\partial \mathbf{f}^{l_{\text{final}}}(x)}{\partial \mathbf{p}_i^{l+1}} = \frac{\partial \mathbf{f}^{l_{\text{final}}}(x)}{\partial \mathbf{f}^{l_{\text{final}}-1}}\frac{\partial \mathbf{f}^{l_{\text{final}}-1}(x)}{\partial \mathbf{f}^{l_{\text{final}}-2}}\cdots\frac{\partial \mathbf{f}^{l+2}(x)}{\partial \mathbf{f}^{l+1}}\frac{\partial \mathbf{f}^{l+1}(x)}{\partial \mathbf{p}^{l+1}}. \tag{13}$$

Thus, any rank drop in a gram matrix of Jacobians from layer $l+k$ to layer $l+k+1$ necessarily also leads to a rank drop in the gram matrix of the Jacobians from layer l to layer l_{final}, and thus d^l zero eigenvalues in the Hessian.

3.2. Synchronized nonlinearities

In this section, we demonstrate a third example of internal structure that affects the effective parameter count of the model. The two examples we presented in the previous sections might be thought of as showing how the network having fewer relevant variables in its representation in a layer leads to more degeneracy. The example we present in this section shows how the network performing “fewer operations” leads to more degeneracy.

In a dense layer with piecewise linear activation functions (ReLU or LeakyReLU), the effective parameter count is reduced if two neurons have the same set of data points for which they are ‘on’ and ‘off’. We call neurons with this property *synchronized* with each other. For simplicity, in this section, we will consider a dense feedforward network with ReLU nonlinearities at each layer, and the same hidden width d throughout.

We define the neuron firing pattern

$$r_i^l(x) = \frac{f_i^l(x)}{p_i^l(x)} \text{ if } p_i^l(x) \neq 0, \text{ else } r_i^l(x) = 1\,, \tag{14}$$

where $p_i^l(x) = \sum_j W_{i,j}^{l-1} f_j^{l-1}(x)$ is the preactivation of neuron i. We call two neurons i and j synchronized if they always fire simultaneously on the training data, $r_i^l(x) = r_j^l(x) \, \forall x \in \mathcal{D}$.

All synchronized As a pedagogical aid, and to demonstrate a point on how the effective parameter count is invariant to linear layer transitions, we first consider the case of all the neurons in layer $l+1$ being synchronized together in the same firing pattern $r^{l+1}(x)$. Then we can write:

$$\begin{aligned} \mathbf{f}^{l+2}(x) &= \text{ReLU}\left(W^{l+1}\,\text{ReLU}(W^l \mathbf{f}^l(x))\right) \\ &= \text{ReLU}\left(W^{l+1} r^{l+1}(x) W^l \mathbf{f}^l(x)\right)\,, \end{aligned} \tag{15}$$

meaning W^l and W^{l+1} effectively act as a single $d \times d$ dimensional matrix $\tilde{W} = W^{l+1}W^l$. Thus, any setting of the weights W^l and W^{l+1} that yield the same $\tilde{W}$ do not change the network’s outputs on the training data, so long as we avoid changing any of the $r_i^{l+1}(x)$. We can ensure that the $r_i^{l+1}(x)$ do not change as we vary the weights by restricting ourselves to alternate weight matrices

$$W^{l+1} \to W^{l+1}C^{-1},\; W^l \to CW^l \tag{16}$$

with C invertible and $C_{i,j} \geq 0 \, \forall i,j$.

Note that a linear layer (without activation function, i.e. $f_i = p_i$) is just a special case of all neurons being synchronized $\forall i, x : r_i^{l+1}(x) = 1$. When W^l is full rank, the drop in the effective parameter count from full synchronisation is the number of parameters in layer l. So we see that from the perspective of the effective parameter count, linear transitions ‘do not cost anything’ — including the linear transition in the model does not meaningfully increase the effective parameter count compared to skipping the layer entirely. We are simply passing variables to the next layer without computing anything new with them.[7]

Synchronized blocks Now, we consider the general case of arbitrary neuron pairs in a layer being synchronized or approximately synchronized. We can organise neurons into sets $S_a, a = 1, \ldots a_{\text{max}}$, with the same activation patterns $r_{S_a}^{l+1}(x)$ for all neurons in the set. We call these sets synchronized *blocks*. This works because synchronisation is a transitive property, if $r_1^{l+1}(x) = r_2^{l+1}(x)$ and $r_1^{l+1}(x) = r_3^{l+1}(x)$, then $r_1^{l+1}(x) = r_3^{l+1}(x)$.

Each neuron belongs to one block, so $\sum_{a=1}^{a_{\text{max}}} |S_a| = d$ and

$$f_i^{l+2}(x) = \text{ReLU}\left(\sum_j \sum_{a=1}^{a_{\text{max}}} r_{S_a}^{l+1}(x) \sum_{k \in S_a} W_{ik}^{l+1} W_{kj}^l f_j^l(x)\right).$$

We can replace $W^{l+1} \to W^{l+1}C^{-1}$, $W^l \to CW^l$, where the matrix C has a block-diagonal structure with invertible blocks $C_{[a]} \in \mathbb{R}^{|S_a| \times |S_a|}$ and $C_{[a],k',k} > 0$ for all $k, k' \in (1, \ldots, |S_a|)$.

Just as we do not expect activations and gradients to have exact rank drops, we do not expect *exact* neuron synchronisation to be common in real models. Instead, we can consider two neurons to be approximately synchronized if their activations only meaningfully differ on a few datapoints. Numerically, we can define:

$$\begin{aligned} |r_a^{l+1}|^2 :=& \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{i,i' \in S_a} \\ & \left(r_i^{l+1}(x) p_i^{l+1}(x) - r_i^{l+1}(x) p_{i'}^{l+1}(x)\right)^2 \,. \end{aligned} \tag{17}$$

If $|r_a^{l+1}|^2$ is non-zero but small, choosing different weight matrices as above will only increase the loss by an amount proportional to $O(|r_a^{l+1}|^2)$.

Degeneracy counting For each pair of synchronized neurons $r_i^{l+1}(x), r_{i'}^{l+1}(x)$, we can set a pair of off-diagonal entries $C_{k,k'}, C_{k',k}$ in C to arbitrary positive values when we change the weights to $W^{l+1}C^{-1}$ and CW^l. If W^l is full rank, the rows k and k' are linearly independent, so this synchronized pair will result in two free directions in parameter space. Thus, we have as many free directions in parameter space as we have synchronized neurons. We can also count this as the number of the synchronized neurons in each block squared

$$N^{l+1} = \sum_{a=1}^{a_{\text{max}}} |S_a|^2. \tag{18}$$

We then see that N^{l+1} is highest if all the neuron firing patterns are synchronized, and lowest when all neurons have different firing patterns.

However, W^l is not always full rank. Further, if we want to combine the degrees of freedom from neuron synchronisation with other degrees of freedom from this section, we have to be careful to avoid double-counting. If the activations in layer l lie in low-dimensional subspaces, then some

[7] See (Aoyagi, 2024) for a more complete treatment of effective parameter counts in deep linear networks.

of the d^2 degrees of freedom above may already have been accounted for. If we remove those double-counted degrees of freedom and control for the rank of W^l, each synchronized block only provides *additional* degrees of freedom equal to the dimensionality of the space spanned by the preactivations of block S_a over the dataset $\mathcal{D}$ squared, which we denote

$$s_a^{l+1} := \dim(\text{span}\{p_k^{l+1} | k \in S_a\})\,. \tag{19}$$

So more generally, the additional amount of degeneracy the effective parameter count is lowered by will be

$$N^{l+1} = \sum_a (s_a^{l+1})^2\,. \tag{20}$$

The trivial case of self-synchronisation $i = i'$ is not excluded here in this formula. It corresponds to the generic freedom to vary the diagonal entries of C, $C_{k,k}$ of a ReLU layer: scaling all the weights going into a neuron by $C_{k,k} \in \mathbb{R}^+$ and scaling all the weights out of the neuron by $1/C_{k,k}$ does not change network behavior.

Attention A similar dynamic holds in the attention layers of transformers, with the attention patterns of different attention heads playing the role of the ReLU activation patterns. If two different attention heads h_1, h_2 in the same attention layer have synchronized attention patterns on the training data set, their value matrices $W_V^{h_1}, W_V^{h_2}$ can be changed to add elements in the span of the value vectors of one head to the other head, with the output matrices $W_O^{h_1}, W_O^{h_2}$ that project results back into the residual stream being modified to undo the change. If $W_V^{h_1}, W_V^{h_2}$ are full rank, this results in $2d_{\text{head}}^2$ degrees of freedom in the loss landscape for each synchronized attention head, in addition to the generic d_{head}^2 degrees of freedom per attention head that are present in every transformer model. If $W_V^{h_1}, W_V^{h_2}$ is not full rank, we account for this similarly as we did with the neurons above.

4. The Interaction Basis

In this section, we propose a technique for representing a neural network as an interaction graph that is invariant to reparameterisations that exploit the freedoms in Sections 3.1.1 and 3.1.2. The technique consists of performing a basis transformation in each layer of the network to represent the activations in a different basis that we call the *Interaction Basis*.

This basis transformation removes degeneracies in activations and Jacobians of the layer to make the basis smaller. The basis is also intended to 'disentangle' interactions between adjacent layers as much as possible.

To find a transformation of network weights and activations that is invariant to reparameterisations based on low-rank activations or low-rank Jacobians, we take the network Hessian (equation 7), and use equation 8 to rewrite it as

$$H^{l,l'}_{ij,i'j'}(\theta^*) = \frac{1}{2n}\sum_{x\in\mathcal{D}} f^l_j(x) f^{l'}_{j'}(x) \sum_k \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial p^{l+1}_i}\frac{\partial f^{l_{\text{final}}}_k(x)}{\partial p^{l'+1}_{i'}}\,. \tag{21}$$

Next, we make two *presumptions of independence* (Christiano et al., 2022), assuming that

1. We can take expectations over the activations and Jacobians in each layer independently
2. Different layers are somewhat independent such that the Hessian eigenvectors can be largely *localised* to a particular layer

Both of these assumptions are investigated in (Martens and Grosse, 2020), who test their validity in small networks and use it to derive a cheap approximation to the Hessian and its inverse.

This allows us to approximate the Hessian as

$$\begin{aligned} H^{l,l'}_{ij,i'j'}(\theta^*) \approx \frac{1}{2}\delta_{l,l'} &\left[\frac{1}{n}\sum_{x\in\mathcal{D}} f^l_j(x) f^l_{j'}(x)\right] \\ &\left[\frac{1}{n}\sum_{x\in\mathcal{D}}\sum_k \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial p^{l+1}_i}\frac{\partial f^{l_{\text{final}}}_k(x)}{\partial p^{l+1}_{i'}}\right]. \end{aligned} \tag{22}$$

This effectively turns the Hessian into a product of two matrices, a gram matrix of activations in each layer

$$G^l_{jj'} = \frac{1}{n}\sum_{x\in\mathcal{D}} f^l_j(x) f^l_{j'}(x) \tag{23}$$

and a Gram matrix of Jacobians with respect to the next layer's preactivations

$$K^l_{ii'} = \frac{1}{n}\sum_{x\in\mathcal{D}}\sum_k \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial p^{l+1}_i}\frac{\partial f^{l_{\text{final}}}_k(x)}{\partial p^{l+1}_{i'}}\,. \tag{24}$$

We can then find the eigenvectors of this approximated Hessian by separately diagonalising these two matrices.

We would like to find a basis for f^l that excludes directions connected exclusively to zero eigenvectors of the Hessian. That is, we want to exclude directions in f^l that lie along zero eigenvectors of G^l, and directions that are mapped by the weight matrix W^l to lie along zero eigenvectors of K^l.

To do this, we can backpropagate the Jacobians in equation 24 one step further to include the weight matrices W^l:

$$M^l_{ii'} = \frac{1}{n}\sum_{x\in\mathcal{D}}\sum_k \frac{\partial f^{l_{\text{final}}}_k(x)}{\partial f^l_i}\frac{\partial f^{l_{\text{final}}}_k(x)}{\partial f^l_{i'}} \tag{25}$$

and then search for a basis in f^l that diagonalises M^l and G^l at the same time. This basis will have one basis vector less for each zero eigenvalue of the Gram matrices of the activations and Jacobians, respectively. It will also exclude directions that lie in the null space of W^l.

The matrices G^l, M^l are symmetric, so we can write $G^l = {U^l}^T D_G^l U^l$ and $M^l = {V^l}^T D_M^l V^l$ for diagonal D_G, D_M and orthogonal U^l, V^l.

We can find a basis transformation $\hat{\mathbf{f}} = C^l \mathbf{f}^l$ in which both G^l and M^l are diagonal, in two steps:

1. Apply a *whitening* transformation with respect to G^l: $\tilde{\mathbf{f}}^l = \left({D_G^l}^{1/2}\right)^+ U^l$, where $+$ denotes the Moore-Penrose pseudoinverse. If the activations in layer l do not span the full activation space, then the gram matrix G^l is not full rank, and some diagonal entries of D_G^l are zero. By choosing this pseudoinverse, we effectively eliminate all the degeneracies from low-rank activations. In this basis, $\tilde{G}^l_{ij} = \delta_{ij}$

2. Now that G^l is whitened, we can apply the transformation by V^l which diagonalises M^l without undiagonalisng G^l since the identity matrix is isotropic.[8] At this point both M^l and G^l are diagonal and C^l is defined up to multiplication by a diagonal matrix. We choose to multiply at the end by $\left({D_M^l}^{1/2}\right)^+$ because this eliminates degeneracies from low rank Jacobians.

We call the basis $\hat{\mathbf{f}}^l = \left({D_M^l}^{1/2}\right)^+ V^l \left({D_G^l}^{1/2}\right)^+ U^l \mathbf{f}^l$ the interaction basis. If our two presumptions of independence held exactly, basis vectors in this basis would be aligned with the directions that affect the output most. We conjecture that if we apply the interaction basis transformation to a real neural network, the resulting representation is likely to be more interpretable. The interaction basis is invariant to invertible linear transformations,[9] meaning the basis itself is a largely coordinate-independent object, much like an eigendecomposition (Appendix D).

We made two simplifying assumptions of independence about the Hessian to motivate this basis. While they have been used in other contexts to some success, these are still strong assumptions. Future work might investigate alternative techniques for finding a basis without these assumptions. This might only be possible with an overcomplete basis, connecting the framework of this paper to superposition.

5. Related Work

Explaining generalisation The inductive biases of deep neural networks that leads them to generalise well past their training data has been an object of extensive study (Zhang et al., 2021). Attempts to understand generalisation involve studying simplicity biases (Mingard et al., 2021) and are closely related to attempts to quantify model complexity, for example via VC dimension (Vapnik, 1998), Radamacher complexity (Mohri et al., 2018) and other less widely known methods (Liang et al., 2019; Novak et al., 2018).

Singular Learning Theory This paper is heavily influenced by Singular Learning Theory (Watanabe, 2009) which uses the local learning coefficient (Lau et al., 2023) to quantify the effective number of parameters in the model via the flatness of minima in the loss landscape. The flatness of minima has been found to predict model generalisation, for example on CIFAR-10 networks (Li et al., 2018). SLT has been used to study the formation of internal structure in neural networks (Chen et al., 2023; Hoogland et al., 2024). Hoogland et al. (2023) propose understanding the internals of neural networks through the geometry of their loss landscapes as a research direction.

Local structure of the loss landscape Other works have investigated the structure of neural network loss landscapes and their degeneracies around solutions found in training. Martens and Grosse (2020) proposed that the Hessian matrix of MLPs can be approximated as being factorisable into independent outer products of activations and gradients, and that its eigenvectors might be approximated as being localised in particular layer of the network. This approximation was later extended to CNNs, RNNs, and transformers (Grosse and Martens, 2016; Martens et al., 2018; Grosse et al., 2023). Wang et al. (2019) use this approximation to compress models by pruning weights along directions with small Hessian eigenvalues. For deep linear networks, an analytical expression for the learning coefficient was derived in (Aoyagi, 2024). Carrol (2021) investigate degeneracies in the loss for ReLU MLP models, while Farrugia-Roberts (2022) do so for one-layer tanh networks. Draxler et al. (2019) find most minima in the loss landscape are connected by a continuous path of minimum loss in CIFAR models.

6. Conclusion

We introduced the idea that the presence of degeneracy in neural networks' parameterizations may be a source of challenges for reverse engineering them. We identified some of the sources of this degeneracy, and suggested a technique (the interaction basis) for removing this degeneracy from the representation of the network. We argued that this representation is likely to have sparser interactions, and we

[8] We need to be careful which coordinate basis we are working in: the entries of V^l in the basis that whitens G^l and in the standard basis are different.

[9] Technically, it is only invariant to up to the uniqueness of the eigenvectors of a certain matrix. But that usually just amounts to freedom under reflections of coordinate axes in practice.

introduced a formula for searching for modules in the new represenation of the network based on a toy model of how modularity affects degeneracy.

7. Impact statement

Our work presents fundamental research in mechanistic interpretability. Interpretability aims to improve our understanding of the inner workings of neural networks, which may help understand issues with AI systems and align future powerful AI systems with human values.

References

Miki Aoyagi. Consideration on the learning efficiency of multiple-layered neural networks with linear units. *Neural Networks*, 172:106132, 04 2024. doi: 10.1016/j.neunet.2024.106132.

Liam Carrol. Phase transitions in neural networks. Master's thesis, School of Computing and Information Systems, The University of Melbourne, October 2021. URL http://therisingsea.org/notes/MSc-Carroll.pdf.

Liam Carroll. Dslt 1. the rlct measures the effective dimension of neural networks, Jun 2023. URL https://www.alignmentforum.org/posts/4eZtmwaqhAgdJQDEg/dslt-1-the-rlct-measures-the-effective-dimension-of-neural.

Lawrence Chan, Adria Garriga-Alonso, Nix Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. Alignment Forum, 2022. URL https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing.

Zhongtian Chen, Edmund Lau, Jake Mendel, Susan Wei, and Daniel Murfet. Dynamical versus bayesian phase transitions in a toy model of superposition. *arXiv preprint arXiv:2310.06301*, 2023.

Paul Christiano, Eric Neyman, and Mark Xu. Formalizing the presumption of independence. *arXiv preprint arXiv:2211.06738*, 2022.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability, 2023.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36, 2024.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape, 2019.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

J. Elisenda Grigsby, Kathryn Lindsey, and David Rolnick. Hidden symmetries of ReLU networks. *arXiv e-prints*, art. arXiv:2306.06179, June 2023. doi: 10.48550/arXiv.2306.06179.

Matthew Farrugia-Roberts. Structural degeneracy in neural networks. Master's thesis, School of Computing and Information Systems, The University of Melbourne, December 2022. URL https://far.in.net/mthesis.

Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66–74, 2016. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2016.01.010. URL https://www.sciencedirect.com/science/article/pii/S0959438816000118. Neurobiology of cognitive behavior.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586, 2021.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer Feed-Forward Layers Are Key-Value Memories, September 2021. URL http://arxiv.org/abs/2012.14913. arXiv:2012.14913 [cs].

Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers, 2016.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, 2023.

Jesse Hoogland. Neural networks generalise because of this one weird trick. https://www.lesswrong.com/posts/fovfuFdpuEwQzJu2w/neural-networks-generalize-because-of-this-one-weird-trick, January 2023.

Jesse Hoogland, Alexander Gietelink Oldenziel, Daniel Murfet, and Stan van Wingerden. Towards developmental interpretability, Jul 2023. URL https://www.alignmentforum.org/posts/TjaeCWvLZtEDAS5Ex/towards-developmental-interpretability.

Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. The developmental landscape of in-context learning, 2024.

Edmund Lau, Daniel Murfet, and Susan Wei. Quantifying degeneracy in singular models via the learning coefficient. *arXiv preprint arXiv:2308.12108*, 2023.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets, 2018.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd international conference on artificial intelligence and statistics*, pages 888–896. PMLR, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2020.

James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyMTkQZAb.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, and Ard A Louis. Is sgd a bayesian sampler? well, almost. *Journal of Machine Learning Research*, 22(79):1–64, 2021.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Daniel Murfet. Singular learning theory iv: the rlct. http://www.therisingsea.org/notes/metauni/slt4.pdf, April 2020. Lecture notes.

Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks, 2016.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks, 2023.

Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis, 2019.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Sumio Watanabe. *Algebraic geometry and statistical learning theory*, volume 25. Cambridge university press, 2009.

Sumio Watanabe. A widely applicable bayesian information criterion. *The Journal of Machine Learning Research*, 14 (1):867–897, 2013.

Susan Wei, Daniel Murfet, Mingming Gong, Hui Li, Jesse Gell-Redman, and Thomas Quella. Deep learning is singular, and that's good. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

A. Interaction sparsity from parameterisation-invariance

In the introduction, we argued that if we can represent a neural network in a parameterisation-invariant way, then this representation is likely to be a good starting point for reverse-engineering the computation in the network. The intuition behind this claim is that in the standard representation, parts of the network which do not affect the outputs act to obfuscate and hide the relevant computational structure — once these are stripped away, computational structure is likely to become easier to see. One way this could manifest is through the new representation having greater *interaction sparsity*.

In this section, we demonstrate that picking the right representation can indeed lead to sparser interactions throughout the network. Specifically, we show that we can find a representation such that, for every drop in the effective parameter count caused by either (a) activation vectors not spanning the activation space (Section 3.1.1) or (b) neuron synchronisation (Section 3.2), there is at least one pair of basis directions in adjacent layers of the network that do not interact.

The role of this section is to provide a first example of a representation of a network which has been made invariant to some reparameterisations, and show that this representation has correspondingly fewer interactions between variables. The algorithm sketch used to find the representation here is not very suitable for selecting sparsely connected bases in practical applications, since it is somewhat cumbersome to extend to non-exact linear dependencies. We introduce a way to choose a basis for the activations spaces that is more suitable for practical applications in Section 4.

Consider a dense feedforward network with ReLU activation functions, with N_{free} degrees of freedom in its parameterization that arise from a combination of

1. The gram matrix of activation vectors in some layers being low rank, see Section 3.1.1.
2. Blocks of neurons being synchronized, see Section 3.2.

We will now show that we can find a representation of the network that

1. exploits the degrees of freedom due to low-dimensional activations to sparsify interactions through a re-parametrisation.
2. exploits the degrees of freedom from neuron synchronisation to sparsify interactions through a coordinate transformation, without losing the sparsity gained in step 1.

Sparsifying using low dimensional activations Here, we show how to exploit the degrees of freedom in the network due to low-dimensional activations in the input layer to sparsify interactions.

Suppose that the gram matrix of activations $\mathbf{f}^{(1)}(x)$ of the input layer, $G^{(1)} = \frac{1}{n}\sum_x f_i^{(1)}(x) f_j^{(1)}(x)$ is not full rank. This means that we can take a set of rank $\left(G^{(1)}\right)$ neurons as a basis for the space. This will be fewer neurons than the width $d^{(1)}$ of the input payer. Writing

$$\forall j \in (\text{rank}\left(G^{(1)}\right) + 1, \ldots, d^{(1)}) : f_j^{(1)} = \sum_{i=1}^{\text{rank}\left(G^{(1)}\right)} (c_j)_i f_i^{(1)}\,, \tag{26}$$

we can replace the weights $W^{(1)}$ with new weights

$$\tilde{W}_{ij}^{(1)} := \begin{cases} W_{ij}^{(1)} + \sum_{k=\text{rank}\left(G^{(1)}\right)+1}^{d^{(1)}} (c_k)_j W_{ik} & 1 \leq j \leq \text{rank}\left(G^{(1)}\right) \\ 0 & \text{rank}\left(G^{(1)}\right) < j \leq d^{(1)} \end{cases} \tag{27}$$

In this way we can disconnect $(d^{(1)} - \text{rank}\left(G^{(1)}\right))$ many neurons from the next layer without changing the activations in layer 2 at all on the training dataset, since $\tilde{W}^{(1)}\mathbf{f}^{(1)} = W^{(1)}\mathbf{f}^{(1)}$. For every degree of linear dependence we may have had in layer 1, we now have $d^{(2)}$ weights set to zero, where $d^{(2)}$ is the width of the second MLP layer. Since two neurons that are connected by a weight of 0 do not interact, this means that we can associate each drop in the effective parameter count caused by linear dependence between activations in layer 1 with a pair of nodes in the interaction graph which do not interact.

Sparsifying using synchronized neurons Now, we show that we can exploit the degrees of freedom in the network from the synchronisation of neurons in the first hidden layer to sparsify interactions without losing any of the sparsity we gained in the previous step.

Taking the example of the second layer $\mathbf{f}^{(2)}$, we want to find a new coordinate basis $\hat{\mathbf{f}}^{(2)} = C^{(2)}\mathbf{f}^{(2)}$ in which there is at least one pair of variables $(\hat{f}_i^{(2)}, f_j^{(1)})$ that does not interact for each drop in the effective parameter count caused by neuron synchronisation.

To choose this basis, we start by finding all pairs of neuron firing patterns $r_i^l(x)$ in layer 2 that are synchronized and group them into sets of synchronized blocks. Continuing with the same notation as in Section 3.2, we denote the blocks of synchronized neurons $S_a, a \in (1, \ldots, a_{\text{max}})$, with size $|S_a|$, and we use the notation $M_{[a]}$ to denote the matrix in $\mathbb{R}^{s_a \times s_a}$ with entries given by $M_{ij}\ \forall i, j \in S_a$. Then, we choose the transformation $C^{(2)}$ to be block diagonal

$$C^2 = \begin{pmatrix} C^2_{[1]} & & 0 \\ & \ddots & \\ 0 & & C^2_{[a_{\text{max}}]} \end{pmatrix}, \tag{28}$$

with the blocks given by the inverse[10] of the $|S_a| \times |S_a|$ blocks of $\tilde{W}^{(1)}$:

$$C^{(2)}_{[a]} = \left(\tilde{W}^{(1)}_{[a]}\right)^{-1}, \tag{29}$$

$$\tilde{W}^{(1)}_{[a]} := \begin{pmatrix} W^{(1)}_{\sigma_{a-1}+1,\sigma_{a-1}+1} & \cdots & W^{(1)}_{\sigma_a,\sigma_{a-1}+1} \\ \vdots & \ddots & \vdots \\ W^{(1)}_{\sigma_{a-1}+1,\sigma_a} & \cdots & W^{(1)}_{\sigma_a,\sigma_a} \end{pmatrix} \tag{30}$$

for $\sigma_a = \sum_{b=1}^{a} s_b$.

This coordinate transformation will set one interaction to zero per drop in the effective parameter count caused by neuron synchronisation. To see this, we first consider that $C^{(2)}$ commutes with the nonlinearity applied between layers 1 and 2, that is

$$C^{(2)}\text{ReLU}\left(W^{(1)}\mathbf{f}^{(1)}(x)\right) = \text{ReLU}\left(C^{(2)}W^{(1)}\mathbf{f}^{(1)}(x)\right) \tag{31}$$

for all x. The product $\hat{W}^{(1)} = C^{(2)}\tilde{W}^{(1)}$ will thus have block diagonal entries equal to the identity $\hat{W}^{(1)}_{[a]} = \mathbf{I}_{|S_a|}$. This means $\hat{W}^{(1)}$ will at minimum have an additional $\sum_a \left(s_a^{(2)}\right)^2 - d^{(2)}$ entries that are zero — one non-interacting pair of nodes per degree of non-generic parametrization freedom caused by neuron synchronization, see equations 19, 20. These absent interactions are distinct from those due to the activation vectors in layer 1 not spanning the full activation space we found in the previous step. Thus, the minimum absent interactions add up to be equal or greater to the degrees of freedom in the loss landscape stemming from low dimensional activations in the input layer $f^{(1)}$ or synchronized neurons in the first hidden layer $f^{(2)}$.

Repeat for every layer Now, we can repeat the previous two steps for all layers, moving recursively from the input layer to the output layer. We check if the activation vectors in layer 2 do not span the activation space and pick new weights $\tilde{W}^{(2)}$ accordingly. Then we check if any neurons in layer three are synchronized and transform $\hat{\mathbf{f}}^{(3)} = C^{(3)}\mathbf{f}^{(3)}$ accordingly. We repeat this for every layer in the network.

We thus obtain new weight matrices, and a new basis for the activations of every layer in the network. Treating the new basis vectors in each layer as nodes in a graph, we can build a graph representing the interactions in the network. This graph will have two properties:

1. It has at least one interaction that is zero for every drop in the effective parameter count introduced by neuron synchronisation or activation vectors spanning a low dimensional subspace
2. It is invariant to reparameterisations that exploit these degeneracies.

[10] Technically the pseudoinverse, because $\tilde{W}^{(1)}_{[a]}$ does not need to be invertible.

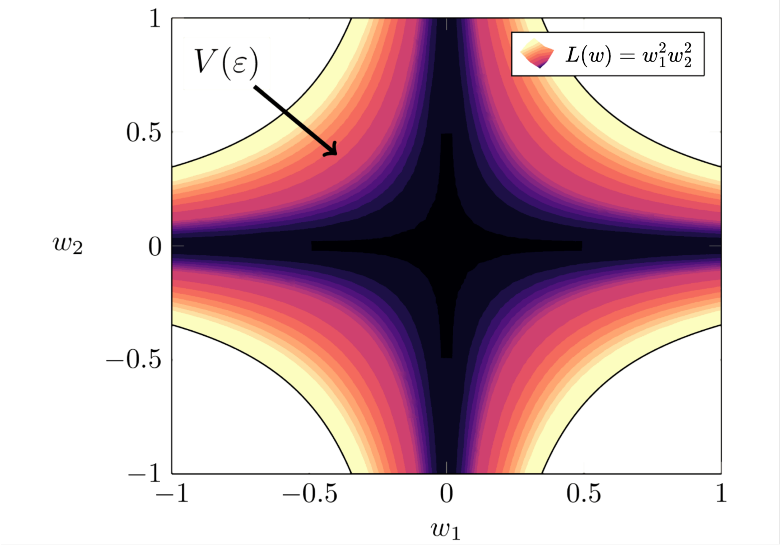

Figure 1: Example of a loss landscape with interacting free directions, from (Carroll, 2023), lightly edited. The loss does not change when changing w_1 alone or w_2 alone, so there are two free directions in the landscape. However, the loss does change when changing both w_1 and w_2 together, so the set of zero loss is cross-shaped rather than spanning the whole plane. Thus, despite there apparently being two free directions, the effective parameter count that characterises the dimensionality of the low loss volume is 1 rather than 0. Non-interacting sets of parameters have no joined terms like this in the loss function, so their free directions always span full subspaces with each other.

B. Modularity may contribute to degeneracy

A core goal of interpretability is breaking up a neural network into smaller parts, such that we can understand the entire network by understanding the individual parts. In this section we propose a particular notion of modularity that could be used to identify these smaller parts. We argue that this notion of modularity is likely to occur in real networks due to its relation to the LLC.

The core claim of this section is that more modular networks are biased towards lower LLC. We argue that if modules in a network interact less (i.e the network is more modular) this yields a higher total degeneracy and thus a lower LLC. Each module has internal degeneracies: if two modules do not interact then the degeneracies in each are independent of each other, so the total amount of degeneracy in the network (from these modules) is at least the sum of the amount of degeneracy within each module. However, if the modules are interacting, then the degeneracies may interact with each other, and the total amount of degeneracy in the network can be less. Therefore, networks which have non- or weakly- interacting modules typically have more degeneracy and thus a lower LLC, which means that neural networks are biased towards solutions which are modular.

The argument in this section does not preclude non-modular networks from having a lower LLC than modular networks in any specific instance. Instead, this section presents an argument that, *all else equal*, modularity is associated with a lower effective parameter count. This argument could fail in practice if more modularity turns out to increase the effective parameter count of models for a different reason, or if real neural networks simply do not have low-loss modular solutions.

In Section B.1 we define interacting and non-interacting degeneracies, and show that the total degeneracy is higher in when individual degeneracies do not interact. In Section B.2 we quantify how modularity affects the LLC by studying a series of increasingly realistic scenarios. First, we consider the case of twomodules which do not interact at all in Section B.2.1. Then we explore how to modify the analysis for modules which have a small number of interacting variables in Section B.2.2. Finally, in Section B.2.3 we extend our analysis to allow for the strength of interactions to vary. We arrive at a modularity metric which can be used to search for modules in a computational graph.

B.1. Interacting and non-interacting degeneracies

If a network's parameterization has a degeneracy, then there is some way the parameters of the network can change without changing the input-output behavior of the network. This change corresponds to a direction that can be traversed through the parameter space along which the behavioral loss stays zero. We call such a direction a *free direction* in the parameter space. It's also possible for a parameterization to have multiple degeneracies and thus multiple free directions.

We call a set of free directions *non-interacting* if traversing along one free direction does not affect whether the other directions remain free. In this case, the set of non-interacting free directions span an entire free subspace of the parameter space. In a parameter space with $\theta = (w_1, w_2, w_3)$ and loss given by $L(w_1, w_2, w_3) = w_1^2$, we are free to pick any value of w_2 and w_3 while remaining at the minimum of the loss provided that $w_1 = 0$. The area of constant loss is a 2-dimensional plane.

The set of free directions is called *interacting* if traversing along one free direction does affect whether other directions remain free. For an extreme example, consider the loss function $L(w_1, w_2) = w_1^2 w_2^2$ (figure 1) at its minimum (0,0). In this case there are two free directions, but when we traverse along one free direction the other direction ceases to be free. The area of constant loss does not span a full subspace (a 2-dimensional plane); here is resembles a cross (see Figure 1) which is a 1-dimensional object.

We can explicitly calculate the number of degrees of freedom (the difference between the effective parameter count (equation 6) and the nominal parameter count) in each of these two loss landscapes. We find that the first landscape has two degrees of freedom but the second has only one. These are two extremes of fully interacting and fully non-interacting free directions. It is also possible to construct intermediate loss landscapes in which the number of degrees of freedom arising from two free directions is a non-integer value between 1 and 2. In general, for a given set of free directions, the lowest the effective parameter count can be is the non-interacting case.

B.2. Degeneracies in separate modules only interact if the modules are interacting

In this section we quantify the increase in the effective parameter count, and equivalently the LLC, from perfect and near-perfect modularity. We show that a network consisting of non-interacting modules has a low effective parameter count, and that a network with modules which interact through a single variable has only a slightly higher effective paraeter count.

Consider a modular neural network $\mathbf{f}_\theta(x)$ consisting of two parallel modules M_1 and M_2. The modules take in different variables x_1, x_2 from the input $x = (x_1, x_2)$, and the output of the network is the concatenation of the module outputs $\mathbf{f}_\theta(x) = (M_1(x_1), M_2(x_2))$. We assign every activation direction in the network to either M_1 or M_2.

We split the parameter space Θ into 3 subspaces: $\Theta = \Theta_1 \oplus \Theta_2 \oplus \Theta_{1\leftrightarrow 2}$. $\theta_1 \in \Theta_1$ are the parameters *inside* M_1 (i.e.parameters that affect interactions between two activations within M_1), θ_2 is the space of the parameters inside M_2, and $\theta_{1\leftrightarrow 2}$ is the space of parameters which affect interactions between activations of both modules.

B.2.1. Non-interacting case

We start by analyzing a network consisting of two perfectly separated modules; the values of activations in M_1 have no effect on activations in M_2, i.e. $\theta_{1\leftrightarrow 2} = 0$ and the network output is given by

$$\mathbf{f}_\theta(x) = (M_1(\theta_1, x_1), M_2(\theta_2, x_2)). \tag{32}$$

Consider now two free directions in parameter space, where one lies entirely in Θ_1, and the other lies entirely in Θ_2. Since M_1 and M_2 share no variables and do not interact, there is no way for a change to parameters along one free direction to affect the freedom of the other direction. Therefore, *one dimensional degeneracies that are in different disconnected modules must be non-interacting.* By contrast, if M_1 and M_2 were connected, their free directions could interact.

We break up the behavioral loss with respect to this network into three terms:

$$L_B(\theta|\theta^*, \mathcal{D}) = L_1(\theta_1|\theta_1^*, \mathcal{D}) + L_2(\theta_2|\theta_2^*, \mathcal{D}) + L_{1\leftrightarrow 2}(\theta_1, \theta_2, \theta_{1\leftrightarrow 2}|\theta_1^*, \theta_2^*, 0, \mathcal{D}) \tag{33}$$

L_1 and L_2 are the parts of the behavioral loss than involve only θ_1 and θ_2 respectively, and $L_{1\leftrightarrow 2}$ contains all the other parts. So long as we ensure $\theta_{1\leftrightarrow 2} = 0$, we have $L_{1\leftrightarrow 2} = 0$. Then a calculation shows that the overall number of degrees of freedom ($N_{\text{free}} = N - N_{\text{eff}}$) for this behavioral loss, restricted to the subspace in which $\theta_{1\leftrightarrow 2} = 0$, is equal to the sum of the number of degrees of freedom in each module.

There could be additional free directions involving moving $\theta_{1\leftrightarrow 2}$ away from 0. These free directions are not guaranteed not to interact with the free directions in each module, and our argument says nothing about how large additional contributions to the effective parameter count from varying $\theta_{1\leftrightarrow 2}$ may be.

B.2.2. Adding in interactions between modules

Next, we consider the case that there are a small set of activations $v_1, \dots, v_m$ inside M_1 that causally affect the value of some activations inside M_2 (due to not all the parameters in $\theta_{1\leftrightarrow 2}$ being 0). This means that the two modules are now interacting with each other. In that case, the only degeneracies in M_1 which are guaranteed not to interact with the degeneracies in M_2 are those which do not affect the value of any of the v_i.

Picture M_1 as a causal graph, where the nodes are activations and the edges are weights or nonlinearities. The nodes inside M_1 are connected to the 'outside' of M_1 via (a) the input layer, where M_1 takes in inputs, (b) the output layer, where M_1 passes on its outputs, and (c) the 'mediating' nodes v_i where variations affect what happens inside M_2. The free directions inside M_1 that are guaranteed not to interact with free directions outside M_1 are those directions that leave this entire *interaction surface* invariant: the directions which do not change any of the mediating nodes as we traverse along them. Each mediating node that is present is an additional constraint on which free directions are guaranteed to be non-interacting. The more approximately independent nodes that are part of that interaction surface, the fewer free directions in M_1 might be generically expected to satisfy these constraints.

In the previous section, we argued that the degrees of freedom of the network with noninteracting modules, restricted to the subset of parameter space in which $\theta_{1\leftrightarrow 2} = 0$, was equal to the sum of the degrees of freedom in each module. In this section, $\theta^*_{1\leftrightarrow 2} \neq 0$, but modifying the argument to restrict to the subset of parameter space in which $\theta_{1\leftrightarrow 2} = \theta^*_{1\leftrightarrow 2}$ is not sufficient to fix the argument, because the degeneracies interact.

To fix the argument, we introduce the *constrained* loss function for parameters in M_1:

$$L_{1,C}(\theta_1|\theta_1^*, \mathcal{D}, v_1, \dots, v_m) = L_1(\theta_1|\theta_1^*, \mathcal{D}) + \frac{1}{n}\sum_{i=1}^{m}\sum_{x\in\mathcal{D}}\left(v_i(\theta_1^*, x) - v_i(\theta_1, x)\right)^2 \tag{34}$$

This loss function is the same as the part of the behavioral loss that depends only on parameters in M_1, except that it has extra MSE terms added to ensure that the points with very small loss also preserve the values of $v_1, \dots, v_m$ on all datapoints. This means its learning coefficient is higher than for the unconstrained behavioral loss. The key property of the constrained loss landscape is that free directions in are guaranteed to be non interacting with free directions in the loss landscape L_2. Therefore, we are able to say that the total effective parameter count of the network consisting of two interacting modules, when constrained to the subspace $\theta_{1\leftrightarrow 2} = \theta^*_{1\leftrightarrow 2}$, really is twice the sum of the learning coefficient for the loss function L_2, and for the loss function $L_{1,C}$[11].

As before, there could be additional free directions involving moving $\theta_{1\leftrightarrow 2}$ away from $\theta^*_{1\leftrightarrow 2}$, which may interact with the free directions in each module. Since we have not characterized the effect of these free directions on the effective parameter count, we cannot confidently conclude that networks with more separated modules reliably have lower effective parameter counts overall. For example, it may be possible that on most real-world loss landscapes, there are many more non-modular solutions than modular ones, and that typically the place in parameter space with lowest loss and lowest effective parameter count is not modular. However, we are not aware of any compelling reason why non-modular networks have some advantage in terms of having low effective parameter counts, to combat the advantage of modular networks discussed in this section.

B.2.3. Varying the strength of an interaction

In the precious section, we discussed the case that two modules interact via m nodes. However, this model had no notion of how strong an interaction is — every node inside M_1 either is not on the interaction surface, or it is, and all nodes on the interaction surface affects the nodes inside M_2 the same amount. In real networks, the extent to which one activation can affect another is continuous. Therefore, we'd like to be able to answer questions like the following:

> Suppose that we have two networks both consisting of two modules, M_1 and M_2. In the first network, there is a single node inside M_1 that strongly influences M_2, and in the second there are two nodes inside M_1 that both weakly influence M_2. Which of these two networks is likely to have a lower effective parameter count?

In this section we'll attempt to answer this question. To do so, we will make use of the notion of an effective parameter count

[11] For simplicity in this section, we have considered the case in which nodes in M_1 affect nodes in M_2 but the converse is not true. If we wanted interactions to be bidirectional, we could modify the argument of this section by introducing a second constrained loss function $L_{2,C}$.

at a finite loss cutoff ϵ (Section 2.2). We show that the magnitude of the total connections through different independent mediating nodes v_1, v_2 seems to add approximately logarithmically to determine the effective 'size' of the total interaction surface between modules.

As before, we consider two modules M_1 and M_2, connected through a number of mediating variables $v_1, \ldots, v_m$ that are part of M_1 and which M_2 depends on. Let each of these mediating variables connect to M_2 through a single weight, $w_1, \ldots, w_n$[12].

If w_i is sufficiently small relative to the loss cutoff ϵ, the connection between modules via v_i will be so small that it can be considered no connection at all from the perspective of interactions between free directions in different modules. This would be if the loss increases when we traverse along both free directions simultaneously by an amount that is smaller than ϵ.

Quantitatively, if we traverse along a free direction in Θ_1 that changes the value of $v_i(\theta_1|x)$, then for small enough ϵ (and a network with locally smooth-enough activation functions), the resulting change in the MSE loss of the whole network L will be proportional to w_i^2. If $w_i = O\left(\epsilon^{\frac{1}{2}}\right)$, that means the connection is 'effectively zero' relative to the given cutoff ϵ, in the sense that the volume of points with $L(\theta) < \epsilon$ is not substantially impacted by the terms in the loss involving w_i.

Now we consider larger connections $w_i = \epsilon^{k_i}$ with $k_i \in (0, \frac{1}{2})$. We can model this situation by taking the size of w_i into account in the constrained loss (equation 34). We define the weighted constrained loss by a sum over mean squared errors for preserving each mediating variable, weighted by the size of the variable:

$$L_{1,C}(\theta_1|\theta_1^*, \theta_{1\leftrightarrow 2}^*, \mathcal{D}, v_1, \ldots, v_m) = L_1(\theta_1|\theta_1^*, \mathcal{D}) + \frac{1}{n}\sum_{i=1}^{m} \epsilon^{2k_i} \sum_{x\in\mathcal{D}} \left(v_i(\theta_1^*, x) - v_i(\theta_1, x)\right)^2 \tag{35}$$

where we've made $L_{1,C}$ depend on $\theta_{1\leftrightarrow 2}^*$ here because w_i are parameters in $\theta_{1\leftrightarrow 2}^*$. We are interested then in how much smaller the learning coefficient for loss landscape L_1 is than the learning coefficient on landscape $L_{1,C}$, as a function of loss cutoff ϵ. This depends heavily on the details of the model. If the constraints are completely independent, we could perhaps model the presence of each constraint as destroying some number γ_i of degrees of freedom compared to the model in which the constraints were not present (and the modules were fully non-interacting).

$$N_{\text{eff, C}} = N_{\text{eff}} + \sum_{i=1}^{m} \gamma_i \,.$$

Now, we seek an expression for γ_i in terms of w_i. Since we require $L_B(\theta) < \epsilon$, and each term in L_B is positive, we also have that each constraint MSE must be smaller than ϵ. Rearranging, we find that

$$\frac{1}{n}\sum_{x\in\mathcal{D}} \left(v_i(\theta_1^*, x) - v_i(\theta_1, x)\right)^2 = \epsilon^{1-2k_i} = \tilde{\epsilon}\,. \tag{36}$$

Therefore, the weights ϵ^{2k_i} of each constraint effectively correspond to measuring the volume of points satisfying that constraint at a larger loss cutoff $\tilde{\epsilon}_i = \epsilon^{1-2k_i}$. Now, we make an assumption that if all the weights were 1, then each constraint would be responsible for removing a similar number $\tilde{\gamma}$ of degrees of freedom from the network. In other words, each constraint would restrict the volume of parameter space that achieves loss less than ϵ by the same amount. Then, we can rescale this region by the factor ϵ^{1-2k_i} and we find that:

$$\gamma_i = (1 - 2k_i)\,\tilde{\gamma} = \left(1 - 2\frac{\log w_i}{\log \epsilon}\right)\tilde{\gamma}\,, \tag{37}$$

Therefore, the size of the logarithm of the weight w_i relative to the logarithm of the cutoff ϵ becomes a prefactor reducing the number of degrees of freedom removed by constraint i. If $w_i = 1$, then $\gamma_i = \tilde{\gamma}$, and if $w_i \leq \epsilon^{\frac{1}{2}}$, then $\gamma_i = 0$[13].

With this in mind, let us return to the question introduced at the start of this section. We will call the network with two weak interactions between modules network A, with two mediating nodes $v_{A,1}, v_{A,2}$ and mediating weights $w_{A,1} = w_{A,2}$. Likewise, we denote the network with one strong interaction between modules by network B, with one mediating node $v_{B,1}$

[12] We could also consider w_i to be the sum of weights connecting node v_i to M_2.

[13] For $w_i < \epsilon^{\frac{1}{2}}$, this is effectively zero from the resolution available at loss cutoff ϵ.

and one mediating weight w_B. How large must w_B be compared to $w_{A,1}$ and $w_{A,2}$ for the interactions between modules in network B to effectively remove the same number of degrees of freedom as the interactions between modules in network A? Using equation 37, we find that

$$\log\left(\frac{w_B}{\epsilon^{\frac{1}{2}}}\right) = \log\left(\frac{w_{A,1}}{\epsilon^{\frac{1}{2}}}\right) + \log\left(\frac{w_{A,2}}{\epsilon^{\frac{1}{2}}}\right). \tag{38}$$

So, the analysis in this section implies that connections through different mediating nodes should be considered to add together logarithmically for the purpose of estimating the number of interaction terms between degrees of freedom that live in different modules. In practice, the constraints different mediating variables impose on the loss 35 are likely rarely completely independent, so this should be seen as a rough approximation to be used as a starting guess for the relevant scale of the problem.

If circuits in neural networks correspond to modules, the analysis in this section implies that we could identify circuits in networks by searching for a partition of the interaction graph of the network into modules which minimises the sum of logs of cutoff-normalised interaction strengths between modules.

C. Background: the local learning coefficient

The most important quantity in SLT is the *learning coefficient* λ. We define a data distribution $x \sim X$ and a family of models with N parameters, parameterised by a vector θ in a parameter space $\Theta \subseteq \mathbb{R}^N$. We also define a population loss function $L(\theta|X)$ which is normalised so that $L(\theta_0|X) = 0$ at the global minimum $\theta_0 = \arg\min_\theta L(\theta|X)$. Then λ is defined as (Watanabe, 2009):[14]

$$\lambda := \lim_{\epsilon\to 0}\left[\epsilon\frac{\mathrm{d}}{\mathrm{d}\epsilon}\log \mathrm{V}(\epsilon)\right], \tag{39}$$

where $\mathrm{V}(\epsilon)$ is the *volume* of the region of parameter space Θ with loss less than ϵ:

$$\mathrm{V}(\epsilon) := \int_{\{\theta\in\Theta:\, L(\theta)<\epsilon\}} \mathrm{d}\theta \tag{40}$$

The learning coefficient quantifies the way the volume of points with low loss changes as we 'zoom in' to lower and lower loss.

Since the loss landscape can have many different solutions with minimum loss, this definition does not necessarily single out a region corresponding to a single solution. Therefore (Lau et al., 2023) introduce the *local* learning coefficient (LLC, denoted by $\hat{\lambda}$) as a way to use the machinery of SLT to study the loss landscape geometry in the neighbourhood of a particular local minimum at θ^* by restricting the volume in the definition of the learning coefficient to a neighbourhood of that minimum $\Theta_{\theta^*} \subset \Theta$ satisfying $\theta^* = \arg\min_{\theta\in\Theta_{\theta^*}} L(\theta|X)$. Then we define the local volume:

$$\mathrm{V}_{\theta^*}(\epsilon) = \int_{\{\theta\in\Theta_{\theta^*}:\, L(\theta)<L(\theta^*)+\epsilon\}} \mathrm{d}\theta \tag{41}$$

and the local learning coefficient:

$$\hat{\lambda}(\theta^*) = \lim_{\epsilon\to 0}\left[\epsilon\frac{\mathrm{d}}{\mathrm{d}\epsilon}\log \mathrm{V}_{\theta^*}(\epsilon)\right]. \tag{42}$$

The LLC is a measure of basin broadness, and SLT predicts that networks are biased towards points in the loss landscape with lower learning coefficient.

To see why the LLC can be thought of as counting the degeneracy in the network, consider a network with N parameters, with N_{free} degrees of freedom in the parameterisation (such that N_{free} of the parameters can be freely varied together or independently, without affecting the loss). Then, we can approximate the loss by a Taylor series around the minimum:

$$L(\theta|X) = L(\theta^*) + \frac{1}{2}(\theta-\theta^*)^T H(\theta^*)(\theta-\theta^*) + O(||\theta-\theta^*||^3) \tag{43}$$

[14] See (Watanabe, 2009) for a more rigorous definition of the learning coefficient.

where $H(\theta^*)$ is the Hessian at the mininum. Consider the case that all functionally relevant parameters all contribute a quadratic term to the loss to leading order, and degrees of freedom correspond to parameters which the loss does not depend on at all. In this case, (Murfet, 2020) explicitly calculate the LLC, showing that it equals $\frac{1}{2}(N - N_{\text{free}})$ — i.e. the LLC counts the number of functionally relevant parameters in the model.

There is a sense that in such a model, the nominal parameter count is misleading, and if there are N_{free} degrees of freedom then there are effectively only $N - N_{\text{free}}$ actual parameters in the model. Indeed, this is the right perspective to take for selecting a model class to fit data with. (Watanabe, 2013) demonstrates that for models with parameter-function maps that are not one-to-one, the Bayesian Information Criterion (Schwarz, 1978), which predicts which model fit to given data generalizes best (Hoogland, 2023), should be modified: the parameter count of the model N should be replaced with 2λ.

In this simple example, the LLC is equal to half the rank of the Hessian at the minimum, and one might wonder if these two quantities are always related in this way. It turns out that they are only the same when the loss landscape can be written locally as a sum of quadratic terms, but this isn't always true. For example, the loss landscape could be locally quartic in some directions, or the set of points with loss equal to 0 may form complicated self intersecting shapes like a cross. In these cases, it is the LLC, not the rank of the Hessian, that measures how much freedom there is to change parameters and how much we expect a particular model to generalise.

D. Invariance of the Interaction Basis under linear transformations

The Interaction Basis is largely a coordinate-independent object, in the sense that it is invariant under linear transformations. If we apply a transformation $\mathbf{f}^l \to \mathbf{f}^l_R = R\mathbf{f}^l, W^l \to W^l_R = W^l R^{-1}$ to the activation space, the final interaction basis is unchanged ($\hat{\mathbf{f}}^l_R = \hat{\mathbf{f}}^l$) for any $R \in \text{GL}_{d^l}(\mathbb{R})$ up to trivial axis reflections, unless M^l has repeated eigenvalues.

To show this, first note that in the whitened basis $\tilde{\mathbf{f}}^l = \left({D^l_G}^{1/2}\right)^+ U^l \mathbf{f}^l$, G^l is by definition always transformed to the identity matrix

$$\tilde{G}^l = \left({D^l_G}^{1/2}\right)^+ U^l G^l \left(\left({D^l_G}^{1/2}\right)^+ U^l\right)^T = \mathbf{I}\,. \tag{44}$$

So if we whiten after applying the transformation R, $\tilde{\mathbf{f}}^l_R$ can only differ from $\tilde{\mathbf{f}}^l$ by an orthogonal transformation. Call this orthogonal matrix Q_R. In the whitened basis, M^l_R will then be:

$$M^l_R = Q_R M^l Q^T_R\,. \tag{45}$$

So M^l_R and M^l only differ by an orthogonal transformation. The interaction basis will be the eigenbasis of M^l_R and M^l, respectively. So long as a real matrix does not have degenerate eigenvalues, its eigendecomposition is basis invariant if a convention for the eigenvector normalisation is chosen, up to reflections. So if M^l does not have multiple identical eigenvalues, the interaction basis we end up in is the same up to reflections whether we transformed with R first or not. If M^l does have identical eigenvalues, the basis will still be identical up to orthogonal transforms in the eigenspaces of M^l.