Jacobian Alignment Explains Grokking and Centroid Alignment Identifies It

Thomas Walker THOMAS.WALKER@RICE.EDU
Rice University

Ahmed Imtiaz Humayun IMTIAZ@RICE.EDU
Rice University

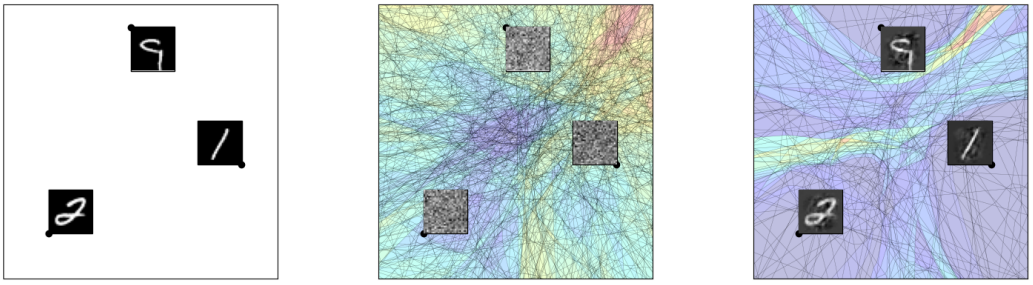
Randall Balestriero RANDALL.BALESTRIERO@GMAIL.COM
Brown University

Richard Baraniuk RICHB@RICE.EDU
Rice University

Abstract

This paper aims to better understand and accelerate the training dynamics of deep networks that lead to delayed generalisation and emergent robustness to input perturbations, known as grokking. Prior work has associated phenomena like delayed generalisation with the transition in a deep network from a linear to a feature learning regime, and emergent robustness with changes to the network's functional geometry, in particular the arrangement of the so-called linear regions in deep networks employing continuous piecewise affine nonlinearities. Here, we explain how grokking is realised in the Jacobian of a deep network and demonstrate that aligning a network's Jacobians with the training data (in the sense of cosine similarity) ensures grokking under a low-rank Jacobian assumption. Our results provide a strong theoretical motivation for the use of Jacobian regularisation in optimizing deep networks, which we show empirically to induce grokking much sooner than more conventional regularizers like weight decay. Moreover, we introduce centroid alignment as a tractable and interpretable simplification of Jacobian alignment that effectively identifies and tracks the stages of deep network training dynamics.

The extended version is GrokAlign: Geometric Characterisation and Acceleration of Grokking.



Data Points | Centroids at Memorisation | Centroids at Generalisation

Figure 1: For deep networks to grok, their Jacobians should align such that the sum of their rows are cosine-similar to the point at which they were computed; we dub this condition *centroid aligned*. We train a ReLU network on the MNIST dataset [20] using Jacobian regularisation. We take three training data points, **left**, and observe the linear regions (using SplineCam [13]) of the deep network along with the centroids [3] of the three data points when it has memorized the training data, **centre**, and when it has generalised, **right**. We colour the linear regions according to the norm of the linear operator acting upon them.

1. Introduction

Deep networks are known to have emergent properties during prolonged training that are essential to understand to facilitate their reliable and effective training. *Delayed generalisation* involves the test accuracy increasing long after train accuracy has increased, in a process initially termed *grokking* [28]. It is a phenomenon that spans multiple deep architectures and domains, including transformers on algorithmic tasks [28] and natural language processing [37], and fully connected networks performing image-classification [23]. Subsequently, the grokking concept has been expanded to include *delayed robustness* [14], which involves prolonged training inducing a robustification of the deep network to input perturbations. Ideally we would accelerate the onset of both generalisation and robustness in deep networks, though in practice there has been an observed tension between them [34, 42].

Despite a high-level understanding of these phenomena, there exists no foundational explanation for why grokking occurs nor a practical and interpretable framework for accelerating a deep network's training dynamics to reach the grokked state more efficiently.

Prior work in this space attributes delayed generalisation to the transition of a deep network from a linear to a feature learning regime [18, 25, 30]. Studies have explored this dynamic from the perspective of the neural tangent kernel [15, 18], the adaptive kernel [30, 31], and mechanistic interpretability [27, 35]. On the one hand, these works have arrived at an array of sufficient conditions for inducing generalisation, including weight-norm at initialisation [23], weight-decay [25, 27], dataset size [35], and output or label scaling [18]. On the other hand, delayed generalisation and robustness has been attributed to the evolution of the functional geometry of the network [14], which is a reference to the arrangement of the so-called linear regions of a continuous piecewise affine network. The Jacobian matrices of a deep network have also been identified as intimately related to their robustness [4, 7, 10, 11, 16, 29]. *In this paper, we prove theoretically and demonstrate empirically that Jacobian norm constraints induces grokking in deep networks. Moreover, we develop a tractable and interpretable approach for monitoring and accelerating grokking in practice based on an efficient summarization of the Jacobian.*

This paper makes three main contributions. First, we demonstrate that deep networks that have optimized their loss function have *aligned* Jacobians at the training data points, in the sense that the rows of the Jacobians at the training points are simply scalar multiples of those points. Deep networks with aligned Jacobians have been empirically demonstrated to be robust [4, 7], and we prove rigorously that they are optimally robust amongst all rank-one Jacobians. Since deep network training dynamics tend to bias the Jacobian towards a low-rank matrix [8, 12, 19, 33, 41], we conclude that *the cause of grokking is the alignment of the network's Jacobian matrices*. Through this theory we motivate the use of Jacobian regularisation to ensure and accelerate grokking.

Second, since working with Jacobian matrices in practice is computationally expensive and current strategies to align the Jacobians are cumbersome [4], we propose to summarize the Jacobian matrices via the sum of their rows. This vector can be efficiently computed through Jacobian vector products [2], and it has a strong geometrical interpretation in the spline theory of deep learning [1], where it corresponds to the centroid of the linear region containing the data point of interest.

Third, we demonstrate that the linear region centroids provide an insightful and more tractable summary of the dynamics of deep network training as compared to Jacobian alignment. Theoretically, the centroids are connected to the neural tangent kernel, and empirically they offer efficiently

computable metrics for monitoring and detecting the emergence of grokking and understanding when additional training could be beneficial.

2. Jacobian Regularisation Explains Grokking

Let $f : \mathbb{R}^d \to \mathbb{R}^C$ be a deep network. Here we focus on the classification setting so that the prediction of the deep network at a point $\mathbf{x}$ is taken to be $\operatorname{argmax}(f(\mathbf{x}))$. In this setting, the deep networks are trained on a data set $\{(\mathbf{x}_p, y_p)\}_{p=1}^m$ – where $\mathbf{x}_p \in \mathbb{R}^d$ and $y_p \in \mathbb{R}$ is its corresponding class – under some loss function $\mathcal{L} = \frac{1}{m}\sum_{p=1}^m \ell\left(f\left(\mathbf{x}_p\right), y_p\right)$. Let $J_{\mathbf{x}}(f)$ be the Jacobian of f at $\mathbf{x}$.

Definition 1 *A deep network is **Jacobian-aligned** at $\mathbf{x} \in \mathbb{R}^d$ if $J_{\mathbf{x}}(f) = \mathbf{c}\mathbf{x}^\top$ for some vector $\mathbf{c} \in \mathbb{R}^d$.*

A deep network is said to have generalised when it learns to extrapolate beyond the training set and perform well on unseen inputs, and it is said to be robust when applying perturbations to inputs does not change the behaviour of the network drastically. The emergence of generalisation is typically formalised under the feature learning regime of training [18, 25, 30], whilst the robustness of deep networks has been connected to its Jacobians [4, 7]. The delayed onset of both these properties is encapsulated in the grokking phenomenon [14, 28].

Let us suppose that f is a continuous piecewise affine deep network – which includes a broad class of architectures, including ReLU feedforward and recurrent networks, convolutional neural networks, and residual networks [1]. Such a deep network has a representation of the form $f(\mathbf{x}) = A_{\omega_{\mathbf{x}}}\mathbf{x} + B_{\omega_{\mathbf{x}}}$ where $A_{\omega_{\mathbf{x}}} \in \mathbb{R}^{C\times d}$ and $B_{\omega_{\mathbf{x}}} \in \mathbb{R}^C$. More specifically, $A_{\omega_{\mathbf{x}}}$ and $B_{\omega_{\mathbf{x}}}$ are the parameters for the affine transformation operating on the linear region $\omega_{\mathbf{x}}$ encompassing $\mathbf{x}$. The *functional geometry* of this deep network is the disjoint union of these linear regions, which is a finite-partition of the input space into a collection of convex polytopes [3]. Note that in this setting $J_{\mathbf{x}}(f) = A_{\omega_{\mathbf{x}}}$.

Theorem 2 *Let $\mathcal{L}$ be the cross-entropy or mean-squared error loss function. Then the continuous piecewise affine deep network f minimising $\mathcal{L}$ under the constraints that $\left\|J_{\mathbf{x}_p}(f)\right\|_F^2 \leq \alpha$ and $B_{\omega_{\mathbf{x}_p}} = \mathbf{0}$ for every $p = 1, \ldots, m$, is Jacobian-aligned.*

Theorem 2 demonstrates that Jacobian-aligned deep networks are optimal in the sense of optimising the training objective. Combined with prior works [4, 7], we also have that Jacobian aligned deep networks are robust.[1] We support this with the following.

Theorem 3 *If $A_{\omega_{\mathbf{x}}}$ is a rank-one matrix and $B_{\omega_{\mathbf{x}}} = \mathbf{0}$, then the local mapping on the linear region $\omega_{\mathbf{x}}$ is optimally robust with respect to ℓ_2 perturbations when $A_{\omega_{\mathbf{x}}} = \mathbf{c}\mathbf{x}^\top$, where the maximum entry of $\mathbf{c}$ is at the index of the class of $\mathbf{x}$.*

In practice, the dynamics of deep network training biases toward low rank weight matrices [8, 12, 19, 33, 41], and thus low rank Jacobians (see Figure 9). Hence, from Theorem 3, we determine that delayed robustness ought to necessarily involve the Jacobian alignment of deep networks.

1. Although, in these prior works, the notion of alignment considers only the row of the Jacobian matrix corresponding to the class of the input.

By performing Jacobian regularisation we can enforce the Jacobian norm constraint of Theorem 2 and thus guarantee that optimising the training objective will lead to a grokked network. More specifically, Jacobian regularisation involves appending the average Frobenius norm of the Jacobian matrices at the training data to the loss function with some coefficient, λ_{Jac}. Weight-decay may also be a viable strategy of enforcing the constraint of Theorem 2; however, it is less direct. Indeed, in some cases weight-decay has proven effective for inducing grokking [22, 27, 28, 35], while in other cases it has shown to be insufficient for grokking [18].

3. The Centroid Alignment Perspective

Centroids. Recall that the functional geometry of a continuous piecewise affine deep network refers to the arrangement of its linear regions. That is, the disjoint union of $\{\omega_{\mathbf{x}}\}_{\mathbf{x}\in\mathbb{R}^d/\sim}$ where $\sim$ denotes the equivalence class $\mathbf{x}_1 \sim \mathbf{x}_2$ if and only if $\mathbf{x}_2 \in \omega_{\mathbf{x}_1}$ and vice-versa. Of importance is the fact that this functional geometry can be parametrised with a collection of parameters $\{(\mu_{\mathbf{x}}, \tau_{\mathbf{x}})\}_{\mathbf{x}\in\mathbb{R}^d/\sim} \subseteq \mathbb{R}^d \times \mathbb{R}$, termed the *centroids* and *radii*, according to a power diagram subdivision [3].

Theorem 4 *For a continuous piecewise affine deep network,* $\mu_{\mathbf{x}} = (J_{\mathbf{x}}(f))^\top \mathbf{1}$.

Using Theorem 4, centroids provide a mechanism through which to summarise a Jacobian matrix of any deep network in way that has an elegant geometrical interpretation when the network is continuous piecewise affine. Importantly, the centroid can be computed through a Jacobian vector product, which is much more computationally efficient than computing the Jacobian [2].

Definition 5 *A deep network is* ***centroid-aligned*** *at* $\mathbf{x} \in \mathbb{R}^d$ *if* $\mu_{\mathbf{x}} = c\mathbf{x}$ *for some constant* $c \in \mathbb{R}$.

The geometrical consequences of centroid alignment can be visualised vividly in Figure 1. An aligned centroid can be linked to the region migration phenomenon observed in Humayun et al. [14], which was used as an explanation for delayed robustness. Since a Jacobian-aligned deep network is centroid-aligned (see Proposition 9), it follows that we can consider centroid alignment as an alternative to Jacobian alignment. Although centroid alignment is a weaker property than Jacobian alignment, we will demonstrate that it has explicit connections to feature learning which we will highlight through the neural tangent kernel.

Neural Tangent Kernel. Suppose our network has parameters θ. Then we take the neural tangent kernel [15] between $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ to be $\Theta(\mathbf{x}, \mathbf{x}') = \nabla_\theta f_\theta(\mathbf{x}) (\nabla_\theta f_\theta(\mathbf{x}'))^\top$. The *linear* and *feature* learning regimes of deep network training are characterised by having relatively constant or dynamic neural tangent kernels respectively [5, 26, 38]. The former identifying when the network approximates a linear function, whereas in the latter the network utilises its nonlinearity. [2]

The Dynamics of Centroids. For simplicity we will explore the centroid dynamics of a two-layer network of the form $f_\theta(\mathbf{x}) = W^{(2)}\left(\sigma\left(W^{(1)}\mathbf{x}\right)\right)$, where $W^{(2)} \in \mathbb{R}^{d^{(2)}\times d^{(1)}}$, $W^{(1)} \in \mathbb{R}^{d^{(1)}\times d}$, and σ is a piecewise affine nonlinearity. We will suppose it is being trained using full-batch gradient descent with a learning rate of η. To make the connection to feature learning explicit, we will consider the deep network to have a scalar-output. We provide a treatment of vector-output networks

2. Lazy and rich are also commonly used terms to refer to these different regimes.

in Appendix F where we make an analogous connection between centroids dynamics and feature learning. In this scalar-output setting we suppose the network is being trained with the cross-entropy loss function.

Theorem 6 *In the setting described above, we have* $\partial_t \left(\langle \mathbf{x}, \mu_{\mathbf{x}} \rangle \right) = \frac{\eta}{m} \sum_{p=1}^{m} \Theta \left(\mathbf{x}, \mathbf{x}_p \right) m_{\mathbf{x}_p}$*, where* $m_{\mathbf{x}_p} = y_p - \frac{1}{1+\exp(-f_\theta(\mathbf{x}_p))}$.

Theorem 6 says that the inner-product between some point in the input space, $\mathbf{x}$, and its corresponding centroid, $\mu_{\mathbf{x}}$, is a weighted sum of the neural tangent kernel of the point with the points in the training data. In particular, a changing inner-product involves a dynamic neural tangent kernel, which identifies the feature learning regime of training. More specifically, if the inner-product $\langle \mathbf{x}, \mu_{\mathbf{x}} \rangle$ changes by δ, then the alignment will change by $\frac{\delta}{\|\mathbf{x}\| \|\mu_{\mathbf{x}}\|}$. When we are optimizing a deep network with Jacobian regularisation, we would expect the centroid norm to be low due to Theorem 4, and thus the feature learning regime will be identified by centroid alignment. Consequently, since Jacobian-aligned deep networks are centroid-aligned and centroid alignment is an indicator of feature learning, we have determined that we can use centroid alignment as a metric for effectively monitoring deep network dynamics.

4. Experiments

To compute the centroid alignment of a deep network, we compute the centroid for an input data point using Theorem 4 and then compute the cosine similarity between this and the input. Likewise, we can obtain the centroid inner-product. To perform Jacobian regularisation we utilise the method outlined in Hoffman et al. [11].

Using this we demonstrate that centroid alignment can identify the feature learning regime of deep network training (see Appendix A) and the onset of robustness (see Appendix B). Moreover, we show that Jacobian regularisation can effectively control the training dynamics of deep networks (see Appendix C). For example, we induce delayed robustness, inhibit grokking as well as accelerate grokking. In some cases, we accelerate grokking by up to seven times (see Table 1). Similarly, in Appendix G we demonstrate that this perspective can be effectively applied to transformer models [36] learning modular addition [28] to control the type of solution they learn.

5. Discussion

We have identified that Jacobian alignment is the cause for grokking, by understanding that Jacobian-aligned deep networks optimise the loss function under a Jacobian norm constraint and are optimally robust under the low rank bias of training dynamics. Consequently, we identified Jacobian regularisation as an effective strategy for controlling the dynamics of deep networks. In particular, we showed that we can induce robustness as well as inhibit or accelerate grokking using Jacobian regularisation. Since Jacobian matrices are difficult to interpret and costly to work with in practice, we constructed the centroid alignment perspective as an alternative strategy to monitor the dynamics of deep networks. This perspective is interpretable due to its relationship with the functional geometry of a deep network and is theoretically meaningful due to its connection to the neural tangent kernel. Using this perspective we were able to identify the onset of the generalisation and robustness during deep network training, as well as reason about when prolonging network training would improve these important properties.

References

[1] Randall Balestriero and Richard Baraniuk. A Spline Theory of Deep Learning. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, July 2018.

[2] Randall Balestriero and Richard Baraniuk. Fast Jacobian-Vector Product for Deep Networks, April 2021. arXiv:2104.00219.

[3] Randall Balestriero, Romain Cosentino, B. Aazhang, and Richard Baraniuk. The Geometry of Deep Networks: Power Diagram Subdivision. In *Neural Information Processing Systems*, May 2019.

[4] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. In *International Conference on Learning Representations*, 2020.

[5] Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In *Advances in Neural Information Processing Systems*, 2019.

[6] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[7] Christian Etmann, Sebastian Lunz, Peter Maass, and Carola Schoenlieb. On the Connection Between Adversarial Robustness and Saliency Map Interpretability. In *Proceedings of the 36th International Conference on Machine Learning*, May 2019.

[8] Tomer Galanti, Zachary S. Siegel, Aparna Gupte, and Tomaso A. Poggio. SGD with Weight Decay Secretly Minimizes the Ranks of Your Neural Networks. In *The Second Conference on Parsimony and Learning (Proceedings Track)*, March 2025.

[9] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, Novermber 2020.

[10] S. Gu and Luca Rigazio. Towards Deep Neural Network Architectures Robust to Adversarial Examples. *CoRR*, December 2014.

[11] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust Learning with Jacobian Regularization, August 2019. arXiv:1908.02729.

[12] Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *Transactions on Machine Learning Research*, 2023.

[13] Ahmed Imtiaz Humayun, Randall Balestriero, Guha Balakrishnan, and Richard Baraniuk. SplineCam: Exact Visualization and Characterization of Deep Network Geometry and Decision Boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2023.

[14] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Deep Networks Always Grok and Here is Why. In *High-Dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, June 2024.

[15] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

[16] Daniel Jakubovitz and Raja Giryes. Improving DNN Robustness to Adversarial Attacks Using Jacobian Regularization. In *ECCV*, September 2018.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Technical report, University of Toronto / University of Toronto, Toronto, Ontario, 2009.

[18] Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *The Twelfth International Conference on Learning Representations*, January 2024.

[19] Thien Le and Stefanie Jegelka. Training invariances and the low-rank phenomenon: Beyond linear networks. In *International Conference on Learning Representations*, 2022.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In *Advances in Neural Information Processing Systems*, 2019.

[22] Jaerin Lee, Bong Gyun Kang, Kihoon Kim, and Kyoung Mu Lee. Grokfast: Accelerated Grokking by Amplifying Slow Gradients, June 2024. arXiv:2405.20233.

[23] Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking Beyond Algorithmic Data. In *The Eleventh International Conference on Learning Representations*, September 2022.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[25] Kaifeng Lyu, Jikai Jin, Zhiyuan Li, Simon Shaolei Du, Jason D. Lee, and Wei Hu. Dichotomy of Early and Late Phase Implicit Biases Can Provably Induce Grokking. In *The Twelfth International Conference on Learning Representations*, January 2024.

[26] Edward Moroshko, Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Nathan Srebro, and Daniel Soudry. Implicit bias in deep linear classification: Initialization scale vs training accuracy. *Advances in Neural Information Processing Systems*, 2020.

[27] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, September 2022.

[28] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, January 2022. arXiv:2201.02177.

[29] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[30] Noa Rubin, Inbar Seroussi, and Zohar Ringel. Grokking as a First Order Phase Transition in Two Layer Networks. In *The Twelfth International Conference on Learning Representations*, January 2024.

[31] Inbar Seroussi, Gadi Naveh, and Zohar Ringel. Separation of scales and a thermodynamic description of feature learning in some CNNs. *Nature Communications*, 14(1):908, February 2023.

[32] Zhiquan Tan and Weiran Huang. Understanding Grokking Through A Robustness Viewpoint, February 2024. arXiv:2311.06597.

[33] Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit Regularization Towards Rank Minimization in ReLU Networks. In *Proceedings of The 34th International Conference on Algorithmic Learning Theory*, February 2023.

[34] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[35] Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency, September 2023. arXiv:2309.02390.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, 2017.

[37] George Wang, Matthew Farrugia-Roberts, Jesse Hoogland, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape geometry reveals stagewise development of transformers. In *High-Dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, June 2024.

[38] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, 2020-07-09/2020-07-12.

[39] Zhiwei Xu, Zhiyu Ni, Yixin Wang, and Wei Hu. Let me grok for you: Accelerating grokking via embedding transfer from a weaker model. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In *Proceedings of the 34th International Conference on Meural Information Processing Systems*, 2020.

[41] David Yunis, Kumar Kshitij Patel, Samuel Wheeler, Pedro Henrique Pamplona Savarese, Gal Vardi, Jonathan Frankle, Karen Livescu, Michael Maire, and Matthew Walter. Rank minimization, alignment and weight decay in neural networks. In *High-Dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, 2024.

[42] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. In *Proceedings of the 36th International Conference on Machine Learning*, May 2019.

Appendix A. Centroid Alignment Identifies the Feature Learning Regime

First, we verify Theorem 6 and our conclusions from it in an MNIST [20] two-class classification setting. Figure 2 demonstrates that the inner-product between a point and its linear region's centroid changes in accordance with the neural tangent kernel, meaning that it can identify this important regime of learning.

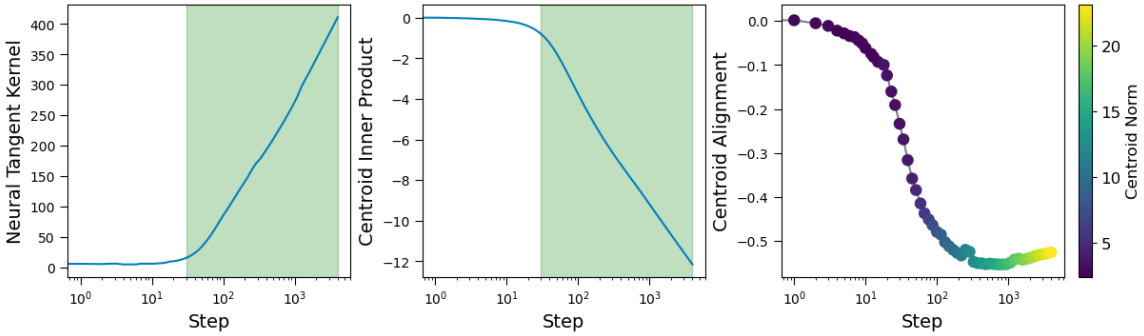


Figure 2: Theorem 6 holds in practice: we observe that a changing inner-product indeed corresponds to a feature learning regime. Here we train a two-layer scalar-output ReLU network using the binary-cross-entropy loss function to distinguish between the zero and one class of the MNIST dataset [20]. We train the model using full-batch gradient descent for 4000 steps at a learning rate of 0.01. At the beginning of training we fix a point from the training set and compute the average value of the neural tangent kernel between itself and the other points from the training set, **left**. We then compute the centroid of the points using Theorem 4 to then obtain the inner-product, **centre**, and its alignment, **right**. We also record the norm of the centroid and encode it in the colours of the markers depicting centroid alignment. In the **left** and **centre** plots we identify the feature learning regime using a green shaded area, this determines where the neural tangent kernel value is changing.

Due to the fact that the norms of the centroids increase during training, eventually the feature learning regime no longer contributes to centroid alignment. This supports the observation that standard training techniques do not maintain a bounded Jacobian norm [40]. Highlighting the necessity of using Jacobian regularisation to ensure the realisation of a Jacobian-aligned deep network in practice.

Without regularisation during training, the initialisation of the network significantly influences its subsequent dynamics. In Appendix D we use alignment to explore this.

Appendix B. Centroid Alignment Identifies Delayed Robustness

Having demonstrated that centroid alignment identifies the feature learning regime, we now determine that it can be used to identify the onset of robustness.

We adopt a set-up similar to that of Xu et al. [39] entailing a scalar-output two-layer fully connected network grokking on XOR cluster data. Note that this network trivially has rank-one Jacobians at every point in the input space. The XOR cluster data contains 40000-dimensional vectors of the form $\mathbf{x} = \left(x_1, x_2, \tilde{\mathbf{x}}^\top\right)^\top \in \mathbb{R}^{40000}$, where $x_1, x_2 \in \{\pm 1\}$ and $\tilde{\mathbf{x}} \in \mathbb{R}^{39998}$. The 400 samples used to train the network are constructed by sampling entries x_1, x_2 uniformly from $\{\pm 1\}$ and entries of $\tilde{x}$ uniformly from $\{\pm\epsilon\}$, here we take $\epsilon = 0.05$. The corresponding label of such a

sample is $x_1 x_2 \in \{\pm 1\}$. A similar sample is generated as a test set. Therefore, by construction our training data has only *signal* in the first two components, whereas the other components contain noise. Hence, generalisation would require recognising the pattern of how the first two components lead to the corresponding label, whilst robustness would require the network to not condition its pattern recognition on the last 39998 components.

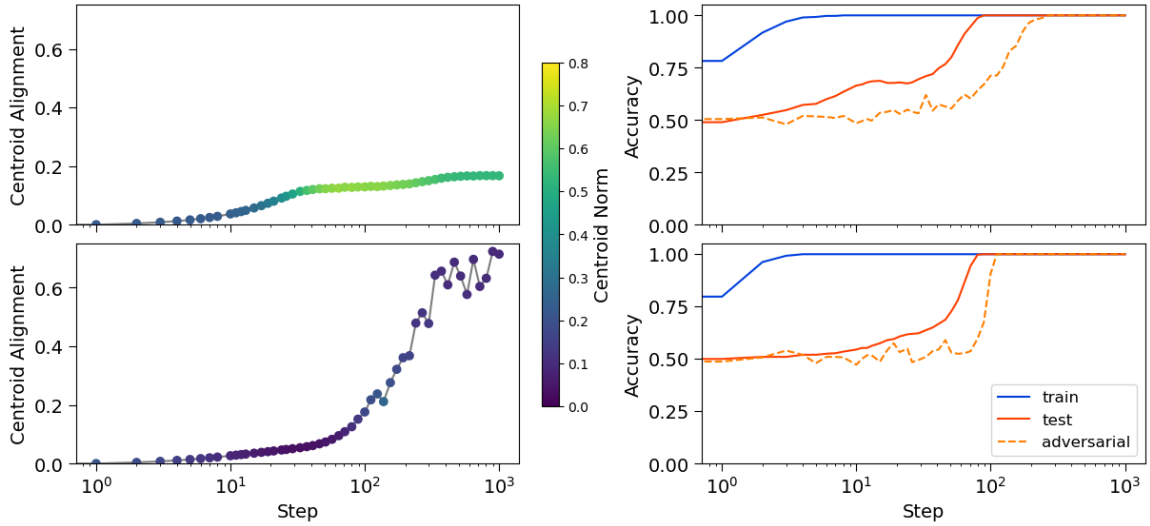


Figure 3: Centroid alignment of a point from the training set identifies the generalisation and robustness of a deep network. We study a two-layer network of width 2048 learning the XOR classification task described in Section B. In the **top** row we train the network using full-batch gradient-descent with a learning rate of 0.1, weight-decay of 0.1 and under the mean-squared error loss function for 1000 steps. We monitor the alignment of a training point to its centroid, as well as the robustness of the network. To measure robustness, we take the entries from the test set and apply Gaussian perturbations of varying standard deviations to the last 39998 components. In the **bottom** row we additionally apply Jacobian regularisation with λ_{Jac} equal to 0.001. The limits for the axes of each column are the same. In the **left** plots we are illustrating the centroid norms as colours for the marker indicating centroid alignment.

Throughout training we track the centroid alignment of a point in the training set to obtain Figure 3. We observe that as the network memorises, the centroid alignment does not increase significantly. However, during generalisation, the centroid alignment increases. After a slight plateau in centroid alignment, a further increase correlates with the onset of robustness. In this instance, the rank of the Jacobian of the network at the point under consideration is one, and thus from Theorem 3 robustness can only be achieved through alignment.

Critically, in the top row of Figure 3, we observe the indirectness of weight-decay at imposing the Jacobian norm constraint of Theorem 2. Early on in training the norms of the centroids increase and eventually inhibit centroid alignment. Since alignment is an optimum of the training objective, it is only at this stage that under weight-decay the network is incentivised to reduce the norms of the Jacobian resulting in the onset of robustness. Indeed, by applying Jacobian regularisation we directly mitigate this delay and achieve robustness much sooner.

Appendix C. Jacobian Regularisation Effectively Controls Training Dynamics

Thus far we have shown that centroid alignment provides a valuable perspective on the training dynamics of a deep network, as it identifies its points of generalisation, Figure 2, and robustification, Figure 3. We have also shown that standard deep network training, including weight-decay, cannot maintain a Jacobian norm constraint and thus is limited in its ability to Jacobian-align the network.

The most direct approach for enforcing the Jacobian norm constraint is through Jacobian regularisation, and here we will explore how this can used to control a network's training dynamics. More specifically, we use an approximate form of Jacobian regularisation that is efficient to employ in practice, whilst maintaining the same effect [11].

Inducing Delayed Robustness. We train convolutional neural networks on the CIFAR10 dataset [17]. We observe that by using Jacobian regularisation we can induce alignment, as evidenced by the increasing centroid alignment. This capitalises on the diminishing ranks of the Jacobian matrices to increase the robustness of the model. With only weight-decay we do not see the onset of robustness.

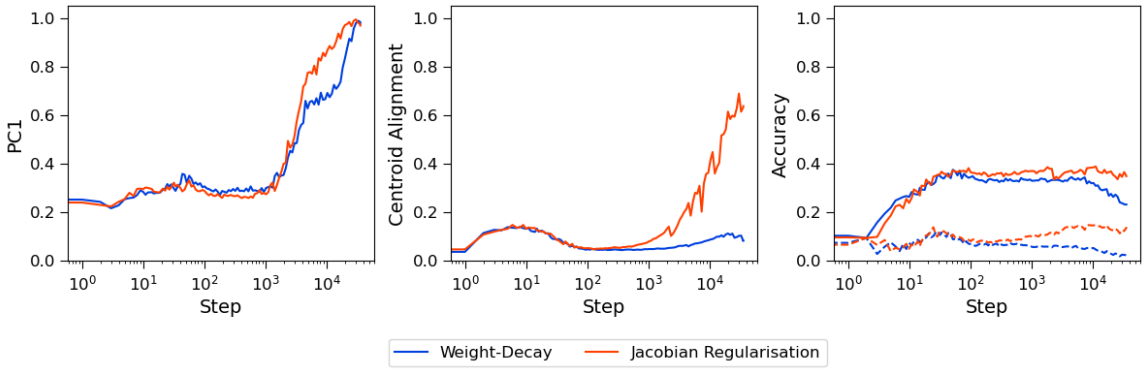


Figure 4: Jacobian regularisation enables us to capitalise on the low rank implicit bias of deep network training to induce robustness. Here we train a convolutional neural network with five convolutional layers and two linear layers, with no bias terms, on a 1024 subset of the CIFAR10 dataset [17] under the mean-squared error loss function. We use the AdamW optimizer [24] at a learning of 0.001, and a batch size of 256 to train the network across 36000 steps. In one instance we apply weight-decay at 0.001, and in another instance we apply Jacobian regularisation with λ_{Jac} equal to 0.001. In the **left** plot we compute the average explained variance of the first principal component of the Jacobians as in Figure 9, and in the **centre** plot we record the average centroid alignment on the training set. In the **right** column we record the test accuracy of the model and the accuracy of the model when ℓ_∞ perturbations of amplitude $\frac{4}{255}$ are applied to the test set using Autoattack [6].

Crucially, we can conclude that prolonging the training is unlikely to improve the properties of the Jacobian regularised model significantly, since the effective rank of the Jacobians is close to one and the centroid alignments are relatively high and have started plateauing.

Inhibiting Delayed Generalisation. Using our reasoning, we would expect that if we were to maintain the Jacobian norms at a high-level, then we ought to prevent alignment and thus generalisation. Therefore, we consider a fully connected network and scale up its weights at initialisation to

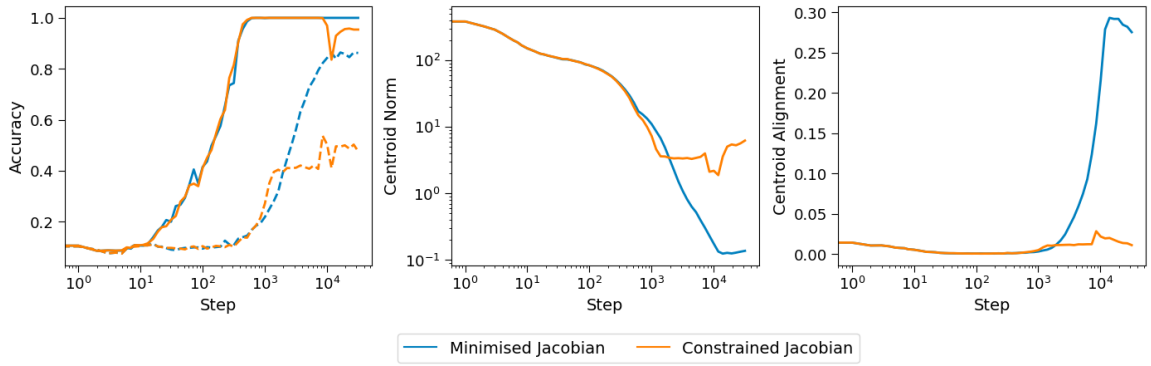


Figure 5: By maintaining the Frobenius norms of the Jacobian at the training data relatively high we can keep the norms of the centroids relatively high which prevents grokking. We take the MNIST grokking set up of Liu et al. [23]. In the minimising case we impose Jacobian regularisation with λ_{Jac} equal to 0.001 during training to minimise the Frobenius norm of the Jacobians at the training data. In the constrained case we apply regularisation to maintain the Frobenius norm of the Jacobian computed at the training data at a relatively high level. More specifically, using a regularisation coefficient of 0.001, we append the difference between five and the average Frobenius norm of the training to the loss function. In the **left** plot we visualise the train and test accuracy with solid and dashed lines respectively. In the **centre** plot we visualise the average norm of the centroids computed at the training data. In the **right** plot we visualise the average centroid alignment, which is just the cosine similarity of the training point with its corresponding centroid.

increase the Jacobian norms, much like Liu et al. [23], and apply Jacobian regularisation in different ways.

We observe in Figure 5 that our prediction is correct, namely minimising the Frobenius norms of the Jacobians leads to generalisation, whilst keeping their value relatively high prevents it. We are able to monitor this through tracking the norms of the centroids, which demonstrates how the centroids provide an effective mechanism to monitor network dynamics.

Accelerating Grokking. Just as we used Jacobian regularisation to inhibit grokking, we can use it to accelerate grokking. For this we consider the standard MNIST grokking set up of Liu et al. [23] which involves applying weight-decay to a deep network initialised with a large initial weight-norm. From our perspective this results in Jacobians with large norm, inhibiting their alignment.

We can quantify the improvement that Jacobian regularisation provides by repeating the experiment over different initialisations and comparing it to other known methods of inducing grokking. For example, we compare it to Grokfast [22], which works to improve the rate of grokking by manipulating the gradients during training to amplify certain signals. Furthermore, we compare it to a method of adversarial training motivated in Tan and Huang [32], which established a connection between robustness and generalisation. The method of adversarial training involves perturbing the inputs during training with noise proportional to the training accuracy of the deep network. In all of our implementations, we will not manipulate the weight-decay of the training procedure, we will keep this parameter constant across all our experiments.

We observe that Jacobian regularisation is extremely effective at inducing the grokked state of the network in this setting, it arrives at the grokked state in 7.56 times fewer steps and 6.31 times

Table 1: Jacobian regularisation significantly speeds up the rate of grokking. In the **top** table we assume the MNIST set up of Liu et al. [23] with the cross-entropy loss function, and in the **bottom** table we assume it with mean-squared error loss function. We repeat the training across ten different random initialisations, where for the cross-entropy loss function we apply Jacobian regularisation with λ_{Jac} equal to 0.001 and with λ_{Jac} equal to 0.0001 for the mean-squared error models. For each run we measure the number of steps and the absolute time, in seconds, taken for the networks to reach 85% test accuracy. In the case of the mean-squared error loss function, we additionally measure the time, in seconds, for the models to go from 20% to 85% test accuracy. We provide the average acceleration (or deceleration) of each method compared to the baseline along with the corresponding standard deviation. All values are to two decimal places.

Regularisation	Number of Steps	Absolute Time (s)
Baseline	-	-
Jacobian Regularisation	$\downarrow \mathbf{7.56} \times (\pm 0.82)$	$\downarrow \mathbf{6.41} \times (\pm 0.69)$
Grokfast	$\uparrow 1.01 \times (\pm 0.04)$	$\uparrow 1.03 \times (\pm 0.04)$
Adversarial Training	$\downarrow 1.32 \times (\pm 0.18)$	$\uparrow 1.92 \times (\pm 0.29)$

Regularisation	Number of Steps	Absolute Time (s)	Grokking Phase Time (s)
Baseline	-	-	-
Jacobian Regularisation	$\downarrow \mathbf{1.69} \times (\pm 0.31)$	$\downarrow \mathbf{1.46} \times (\pm 0.30)$	$\downarrow \mathbf{1.77} \times (\pm 0.53)$
Grokfast	$\downarrow 1.08 \times (\pm 0.17)$	$\downarrow 1.05 \times (\pm 0.23)$	$\downarrow 1.05 \times (\pm 0.27)$
Adversarial Training	$\downarrow 1.23 \times (\pm 0.26)$	$\uparrow 1.92 \times (\pm 0.29)$	$\uparrow 2.02 \times (\pm 0.37)$

faster than the baseline in the case of the cross-entropy loss function. In contrast, Grokfast provides a relatively lower improvement in the mean-squared error case and is ineffective in the cross-entropy case. Furthermore, adversarial training does not improve the rate of grokking over the baseline. In particular, adversarial training does not improve the robustness of the model by way of aligning the Jacobian, unlike Jacobian regularisation (see Appendix E).

Controlling the Behaviour of Deep Networks. The use of Jacobian regularisation was motivated in the setting of classification. Nanda et al. [27] observed that a single layer transformer [36] learning modular addition [28] grokked by learning how to implement an algorithm. An equally viable solution to this problem would be through classification.

Since our new tools extend to transformer models, in Appendix G we explore the application of Jacobian regularisation for controlling their learning dynamics and centroid alignment for monitoring them. We demonstrate that we can bias the network to learn the classification style solution through Jacobian regularisation.

Appendix D. Initialising for Centroid Alignment

Increasing the rate of change of inner-product can be done by increasing the neural tangent kernel, say through scaling the weights or output of the network. However, these will also increase the

norm of the centroid. Moreover, such scaling is known to increase the propensity of the network maintaining a linear learning regime [5], along with increasing the width of the neural network [21] and label rescaling [9]. We explore this trade-off by repeating the experiment of Figure 2, but with various scaling of the weights or output of the network.

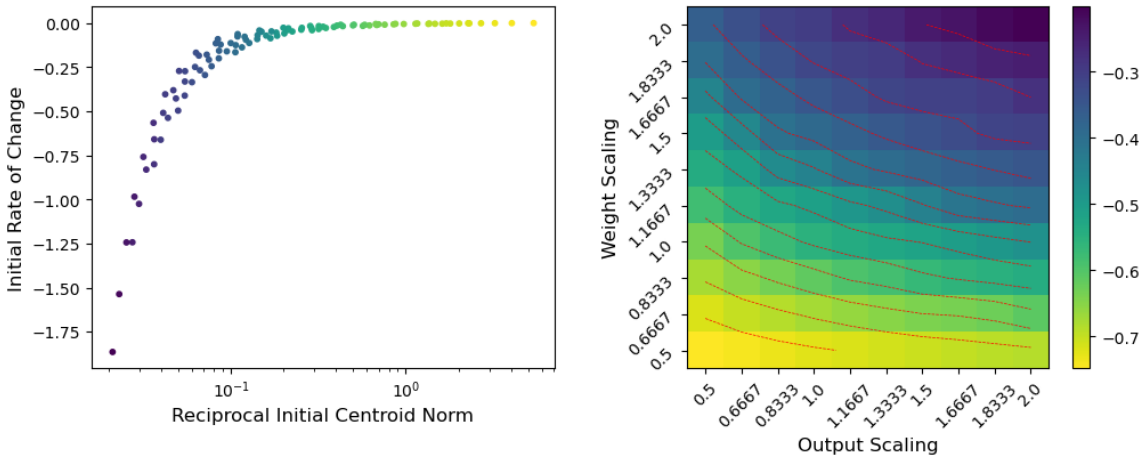


Figure 6: When no regularisation is used, minimising the centroid norm at initialisation is essential for ensuring the alignment of the centroid during training, and output scaling ensures this more effectively than scaling the weights at initialisation. Here we repeat the experiment of Figure 2, but with varying scaling of the weights and output of the network. On the **left** we plot the correlation between the initial rate of change of inner-product, as computed by Theorem 6, and the average reciprocal of the norm of the centroids of the training points. We additionally colour the scatter points according to the maximum alignment of the centroid observed during training. On the **right** we observe how the maximum alignment of the centroid observed during training correlates with the different scaling mechanisms.

In Figure 6, we observe that controlling the norm of the centroid is a more effective strategy for translating the feature learning of the neural network into centroid alignment. However, a priori, knowing how to initialise the deep network for favourable alignment dynamics is challenging, hence, in practice some sort of regularisation is necessary.

Appendix E. Alignment Induced by Adversarial Training

Although both Jacobian regularisation and adversarial training are motivated to induce grokking by improving the model's robustness, the former does this though aligning the functional geometry of the model, whereas the latter does not. We determine this by measuring the cosine similarity between training points and the rows of the Jacobian of the model at those points, Table 2.

Table 2: Here we compare adversarial training to the baseline and Jacobian regularisation grokking set ups of Table 1 in terms of inducing the alignment of the Jacobian at the training data. All values are to three significant figures.

Regularisation	Test Accuracy	Jacobian Row Alignment (max/min)
Baseline	88.9%	−0.254/0.283
Jacobian Regularisation	**91.8%**	**−0.509/0.478**
Adversarial Training	88.0%	−0.253/0.274

Regularisation	Test Accuracy	Jacobian Row Alignment (max/min)
Baseline	86.8%	−0.170/0.272
Jacobian Regularisation	**88.2%**	**−0.638/0.709**
Adversarial Training	87.8%	−0.263/0.376

Appendix F. Centroid Dynamics of Vector-Output Deep Networks

Consider the case of a general vector output, namely $d^{(2)} \geq 2$, with $\mathcal{L}$ being the cross-entropy loss function or the mean-squared error loss function. Namely, we consider

$$\ell\left(f\left(\mathbf{x}_p\right), y_p\right) = -\log\left(\frac{\exp\left(\left[f\left(\mathbf{x}_p\right)\right]_{y_p}\right)}{\sum_{c=1}^{d^{(2)}}\exp\left(\left[f\left(\mathbf{x}_p\right)\right]_c\right)}\right)$$

for the cross-entropy loss function, or

$$\ell\left(f\left(\mathbf{x}_p\right), y_p\right) = \left\|\mathbf{e}_{y_p} - f\left(\mathbf{x}_p\right)\right\|_2^2$$

for the mean-squared error loss function.

Proposition 7 *In the setting described above, we have*

$$\partial_t\left(\langle \mathbf{x}, \mu_{\mathbf{x}}\rangle\right) = \frac{\eta}{m}\sum_{p=1}^{m}\left(\left(\mathbf{m}_{\mathbf{x}_p}^\top W^{(2)} Q_{\mathbf{x}_p} Q_{\mathbf{x}}\left(W^{(2)}\right)^\top \mathbf{1}\right)\langle \mathbf{x}, \mathbf{x}_p\rangle + \mathbf{x}^\top\left(W^{(1)}\right)^\top Q_{\mathbf{x}}\sigma\left(W^{(1)}\mathbf{x}_p\right)\mathbf{m}_{\mathbf{x}_p}^\top \mathbf{1}\right)$$

where

$$\mathbf{m}_{\mathbf{x}_p} = \mathbf{e}_y - \frac{\exp\left(\left[f_\theta\left(\mathbf{x}_p\right)\right]_{y_p}\right)}{\sum_{c=1}^{C}\exp\left(\left[f_\theta\left(\mathbf{x}_p\right)\right]_c\right)}$$

in the case of the cross-entropy loss function and

$$\mathbf{m}_{\mathbf{x}_p} = 2\left(\mathbf{e}_y - f_\theta\left(\mathbf{x}_p\right)\right).$$

Corollary 8 *In the setting of Proposition 7, under the cross entropy loss function, we have*

$$\begin{aligned}\partial_t \left(\langle \mathbf{x}, \mu_{\mathbf{x}} \rangle\right) &= \frac{\eta}{m} \sum_{p=1}^{m} \left(\mathbf{m}_{\mathbf{x}_p}^{\top} W^{(2)} Q_{\mathbf{x}_p} Q_{\mathbf{x}} \left(W^{(2)}\right)^{\top} \mathbf{1} \right) \langle \mathbf{x}, \mathbf{x}_p \rangle \\ &:= \frac{\eta}{m} \sum_{p=1}^{m} \frac{\iota_{\mathbf{x},p}}{\|\mathbf{x}_p\|_2} \langle \mathbf{x}, \mathbf{x}_p \rangle\end{aligned}$$

That is, in the context of the cross-entropy loss function, centroids are aligned in a manner that is proportional to the alignment of their encompassing points with the training data. More specifically, with the intuition that the role of $W^{(2)}$ in f_θ is to be a collection of filters facilitating the classification of each class, the quantity $W^{(2)} Q_{\mathbf{x}_p} \left(W^{(2)} Q_{\boldsymbol{\nu}}\right)^{\top} \mathbf{1}$ can be thought of understanding how each feature of $\mathbf{x}_p$ correlates with the features of the region $\omega_{\boldsymbol{\nu}}$. Since $\mathbf{m}\left[\mathbf{x}_p\right]$ is positive on the correct class and negative for the incorrect classes, the term $\iota_{\mathbf{x}',p}$ is largest when $\omega_{\mathbf{x}'}$ has identified features that correlate with the features of $\mathbf{x}_p$ that indicate the class it belongs too. In such a case, the centroid $\mu_{\mathbf{x}'}$ is moved in the direction of $\mathbf{x}_p$ to further maximize this correlation. Showing how regions $\omega_{\mathbf{x}'}$ are being optimized to capture the features of classes that help it distinguish it from the other classes. Therefore, we can see neural network training more as a process of allocating linear regions to different features that best distinguishes themselves from the other classes. This would suggest that when we observe the centroids of a layer of a neural network aligning with the data it encompasses, the neural network is performing feature extraction. In particular, the centroid of a training point is most incentivised to positively align with itself.

Appendix G. Controlling and Monitoring Transformer Training Dynamics

The computations of centroids is valid without the continuous piecewise affine assumption, it is only their interpretation as characterising a functional geometry that requires the assumption. Therefore, we can examine the alignment of a transformer model [36] being trained on modular addition, and the effect of introducing Jacobian regularisation into the training.

In the top row of Figure 7, we again observe that the alignment of the centroids with the training data changes in accordance with the test accuracy. Although, we find that applying Jacobian regularisation does not accelerate the rate of grokking.

A key aspect that facilitated the transformer in implementing its algorithmic solution was the ability to manipulate the embedding and unembedding matrices [27]. Therefore, if we fix the embedding matrix during training, we a priori bias the the model in learning the *classification* style solution. Under this set up, bottom row of Figure 7, we observe that the transformer groks earlier with Jacobian regularisation than with weight-decay.

We support this by tracking the Gini coefficients of the embedding and unembedding matrices [27] in the case of learning embeddings trained with and without Jacobian regularisation. Clearly, we see that under Jacobian regularisation the Gini coefficients do not increase, indicating that the model is not implementing the identified algorithmic solution, Figure 8.

Appendix H. Supporting Results

Proposition 9 *A Jacobian-aligned deep network is centroid-aligned.*

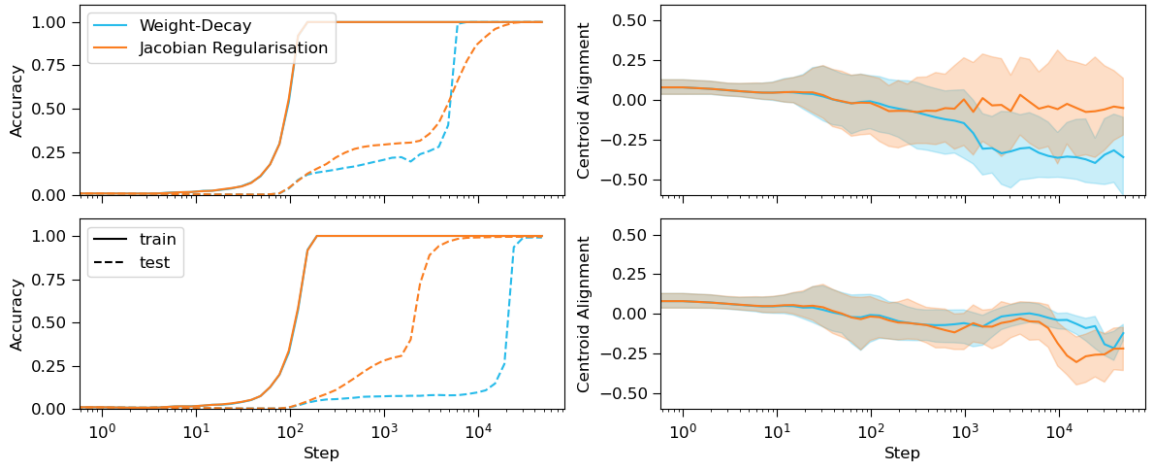


Figure 7: Here we obtain the alignment statistics for a single layer transformer trained on modular addition [28] as in [27] In the **top** row we train the model with learnable embeddings, whilst in the **bottom** row we train the model with fixed embeddings. The shaded region in the **right** columns represents the maximum and minimum values of centroid alignment on the train set with the solid line representing the mean.

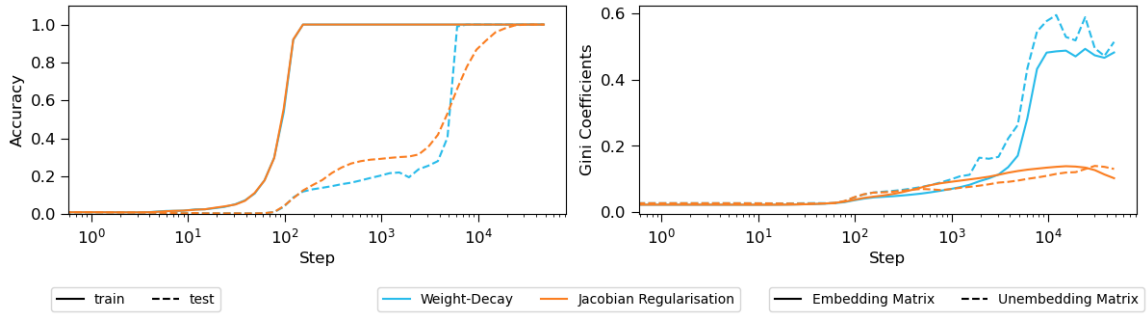


Figure 8: For a single layer transformer trained on modular addition [28] we track the Gini coefficients of the embedding and unembedding matrices for training with and without Jacobian regularisation.

Note that for a continuous piecewise affine network $f = \left(f^{(L)} \circ \cdots \circ f^{(1)}\right)$, each $f^{(l)}$ and subcomponent $f^{(1\leftarrow l)} = \left(f^{(l)} \circ \cdots \circ f^{(1)}\right)$ are also continuous piecewise affine networks. Let $A^{(l)}_{\omega^{(l)}_{\mathbf{x}}}$, $B^{(l)}_{\omega^{(l)}_{\mathbf{x}}}$, $\omega^{(l)}_{\mathbf{x}}$, $\mu^{(\ell)}_{\omega^{(l)}_{\mathbf{x}}}$ and $A^{(1\leftarrow l)}_{\omega^{(1\leftarrow l)}_{\mathbf{x}}}$, $B^{(1\leftarrow l)}_{\omega^{(1\leftarrow l)}_{\mathbf{x}}}$, $\omega^{(1\leftarrow l)}_{\mathbf{x}}$, $\mu^{(1\leftarrow l)}_{\omega^{(1\leftarrow \ell)}_{\mathbf{x}}}$ be analogous notation for the layer and subcomponent networks to that of the continuous piecewise affine networks we introduced in Section 2.

Theorem 10 (Balestriero et al. 3) *The l^{th} layer of a deep network partitions its input space according to a power diagram with centroids*

$$\mu^{(\ell)}_{\omega^{(l)}_{\mathbf{x}}} = \left(A^{(l)}_{\omega_{\mathbf{x}}(l)}\right)^{\top} \mathbf{1}.$$

Lemma 11 *In the setting of Section 3, we have* $\mu_{\mathbf{x}} = \left(W^{(2)} Q_{\mathbf{x}} W^{(1)}\right)^{\top} \mathbf{1}$, *where* $Q_{\mathbf{x}} := \operatorname{diag}\left(\sigma'\left(W^{(1)}\mathbf{x}\right)\right)$.

Lemma 12 *In the setting of Section 3, with* $d^{(2)} = 1$, *the neural tangent kernel of* f_θ *between* $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ *is given by*

$$\Theta\left(\mathbf{x}, \mathbf{x}'\right) = \sigma\left(W^{(1)}\mathbf{x}\right)^{\top} \sigma\left(W^{(1)}\mathbf{x}'\right) + \left(\mathbf{x}^{\top}\mathbf{x}'\right)\left(W^{(2)} Q_{\mathbf{x}} Q_{\mathbf{x}'} \left(W^{(2)}\right)^{\top}\right).$$

Theorem 13 (Balestriero et al. 3) *The continuous piecewise operation of a deep network from the input to the output of the* l^{th} *layer partitions its input space according to a power diagram with centroids*

$$\mu^{(1\leftarrow l)}_{\omega^{(1\leftarrow \ell)}_{\mathbf{x}}} = \left(A^{(l-1)}_{\omega^{(l-1)}_{\mathbf{x}}} \cdots A^{(1)}_{\omega^{(1)}_{\mathbf{x}}}\right)^{\top} \mu^{(l)}_{\omega^{(l)}_{\mathbf{x}}} =: \left(A^{(1\leftarrow l-1)}_{\omega^{(1\leftarrow l-1)}_{\mathbf{x}}}\right)^{\top} \mu^{(l)}_{\omega^{(l)}_{\mathbf{x}}}.$$

Appendix I. Proofs

Theorem 2. Proof

1. In the instance of the cross-entropy loss function,

$$\ell_p := \ell\left(f\left(\mathbf{x}_p\right), y_p\right) = -\log\left(\frac{\exp\left(\left[f\left(\mathbf{x}_p\right)\right]_{y_p}\right)}{\sum_{c=1}^{C}\exp\left(\left[f\left(\mathbf{x}_p\right)\right]_c\right)}\right).$$

Under the assumptions, the output of the neural network at $\mathbf{x}_p$ is $A_{\omega_{\mathbf{x}_p}}\mathbf{x}_p$. The cross entropy loss of the deep network on $\mathcal{D}$ is

$$\mathcal{L}_{\text{CE}} = \frac{1}{m}\sum_{p=1}^{m}\ell_p$$

where

$$\begin{aligned}\ell_p &= -\log\left(\frac{\exp\left(\left[A_{\omega_{\mathbf{x}_p}}\mathbf{x}_p\right]_{y_p,\cdot}\right)}{\sum_{c=1}^{C}\exp\left(\left[A_{\omega_{\mathbf{x}_p}}\mathbf{x}_p\right]_{c,\cdot}\right)}\right)\\ &= -\left\langle\left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p\right\rangle + \log\left(\sum_{c=1}^{C}\exp\left(\left\langle\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p\right\rangle\right)\right),\end{aligned}$$

which is convex on a convex set. Thus we can consider the sufficient Karush-Kuhn-Tucker conditions with Lagrange multiplier,

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda\left(\sum_{c=1}^{C}\left\|\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}\right\|_2^2 - \alpha\right).$$

In particular, the Karush-Kuhn-Tucker conditions have the form

$$\frac{\partial \mathcal{L}}{\partial \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}} = -\mathbf{1}_{\{y_p=c\}}\mathbf{x}_p + \frac{\mathbf{x}_p \exp\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p \right\rangle\right)}{\sum_{c'=1}^{C} \exp\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c',\cdot}, \mathbf{x}_p \right\rangle\right)} - 2\lambda \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}$$
$$= \mathbf{0}, \tag{1}$$

for $c = 1, \ldots, C$ and

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \alpha - \sum_{c=1}^{C} \left\| \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot} \right\|_2^2 = 0. \tag{2}$$

From (1), we have

$$0 = \sum_{c=1}^{C} \left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \frac{\partial \mathcal{L}}{\partial \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}} \right\rangle$$
$$= -\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p \right\rangle + \sum_{c=1}^{C} \frac{\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p \right\rangle \exp\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p \right\rangle\right)}{\sum_{c'=1}^{C} \exp\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c',\cdot}, \mathbf{x}_p \right\rangle\right)} - 2\lambda \sum_{c=1}^{C} \left\| \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot} \right\|_2^2.$$

Let $\varrho_c = \frac{\exp\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p \right\rangle\right)}{\sum_{c'=1}^{C} \exp\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c',\cdot}, \mathbf{x}_p \right\rangle\right)}$. Then, in conjunction with (2), it follows that

$$\lambda = \frac{1}{2\alpha} \sum_{c=1}^{C} \left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p \right\rangle \varrho_c - \left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p \right\rangle.$$

Using this back in (1) we get,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}} &= -\mathbf{1}_{\{y_p=c\}}\mathbf{x}_p + \mathbf{x}_p\varrho_c + \frac{1}{\alpha}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot} \\
&\quad - \frac{1}{\alpha}\sum_{c'=1}^{C} \varrho_{c'} \left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c',\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot} \\
&= \left(-\mathbf{1}_{\{y_p=c\}} + \varrho_c\right)\mathbf{x}_p + \frac{\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}}{\alpha}\left(1-\varrho_{y_p}\right) \\
&\quad - \frac{1}{\alpha}\sum_{c'\neq y_p} \left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c',\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot} \varrho_{c'} \\
&= \left(-\mathbf{1}_{\{y_p=c\}} + \varrho_c\right)\mathbf{x}_p + \frac{\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}}{\alpha}\left(1-\varrho_{y_p}\right) \\
&\quad - \frac{1}{\alpha}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{i,\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}\left(1-\varrho_{y_p}\right) \\
&= \left(-\mathbf{1}_{\{y_p=c\}} + \varrho_c\right)\mathbf{x}_p + \frac{\left(1-\varrho_{y_p}\right)\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}}{\alpha}\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p \right\rangle - \left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{i,\cdot}, \mathbf{x}_p \right\rangle\right),
\end{aligned}$$

where i is just some incorrect class for $\mathbf{x}_p$, namely $i \neq y_p$. When $c = y_p$ this reduces to

$$\frac{\partial \mathcal{L}}{\partial \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}} = \left(1-\varrho_{y_p}\right)\left(-\mathbf{x}_p + \frac{1}{\alpha}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot} - \left[A_{\omega_{\mathbf{x}_p}}\right]_{i,\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}\right),$$

and when $c \neq y_p$ it reduces to

$$\frac{\partial \mathcal{L}}{\partial \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}} = \left(1-\varrho_{y_p}\right)\left(\frac{1}{C-1}\mathbf{x}_p + \frac{1}{\alpha}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot} - \left[A_{\omega_{\mathbf{x}_p}}\right]_{i,\cdot}, \mathbf{x}_p \right\rangle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}\right)$$

To obtain the optimal A, it suffices to find $A_{\omega_{\mathbf{x}_p}}$ satisfying these conditions. To do so we consider the ansatz

$$\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot} = \begin{cases} a\mathbf{x}_p & c = y_p \\ b\mathbf{x}_p & c \neq y_p. \end{cases}$$

Substituting this into our conditions we obtain the equations

$$\begin{cases} -1 + \frac{1}{\alpha}(a-b)a\left\|\mathbf{x}_p\right\|_2^2 = 0 \\ \frac{1}{C-1} + \frac{1}{\alpha}(a-b)b\left\|\mathbf{x}_p\right\|_2^2 = 0 \\ \left(a^2 + (C-1)b^2\right)\left\|\mathbf{x}_p\right\|_2^2 = \alpha. \end{cases}$$

Solving these systems of equations we arrive at

$$\begin{cases} a = \frac{1}{\left\|\mathbf{x}_p\right\|_2}\sqrt{\frac{\alpha(C-1)}{C}} \\ b = -\frac{1}{\left\|\mathbf{x}_p\right\|_2}\sqrt{\frac{\alpha}{C(C-1)}}. \end{cases}$$

2. In the instance of the mean-squared error,

$$\ell_p := \ell\left(f\left(\mathbf{x}_p\right), y_p\right) = \left\|f\left(\mathbf{x}_p\right) - \mathbf{e}_{y_p}\right\|_2^2,$$

where we use $\mathbf{e}_i \in \mathbb{R}^d$ to denote the i^{th} standard basis vector. Under the assumptions, the output of the neural network at $\mathbf{x}_p$ is $A_{\omega_{\mathbf{x}_p}}\mathbf{x}_p$. The mean squared error loss of the deep network on $\mathcal{D}$ is

$$\mathcal{L}_{\text{MSE}} = \frac{1}{m}\sum_{p=1}^{m}\ell_p$$

where

$$\ell_p = \left\langle A_{\omega_{\mathbf{x}_p}}\mathbf{x}_p - \mathbf{e}_{y_p}, A_{\omega_{\mathbf{x}_p}}\mathbf{x}_p - \mathbf{e}_{y_p}\right\rangle = \sum_{c=1}^{C}\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p\right\rangle - \mathbf{1}_{\{y_p=c\}}\right)^2,$$

which is convex on a convex set. Thus we can consider the sufficient Karush-Kuhn-Tucker conditions with Langrange multiplier,

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} - \lambda\left(\sum_{c=1}^{C}\left\|\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}\right\|_2^2 - \alpha\right).$$

In particular, the Karush-Kuhn-Tucker conditions have the form

$$\frac{\partial\mathcal{L}}{\partial\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}} = 2\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p\right\rangle - \mathbf{1}_{\{y_p=c\}}\right)\mathbf{x}_p - 2\lambda\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}. \tag{3}$$

for $c = 1, \ldots, C$ and

$$\frac{\partial\mathcal{L}}{\partial\lambda} = \sum_{c=1}^{C}\left\|\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}\right\|_2^2 - \alpha = 0. \tag{4}$$

From (3), we have

$$\begin{aligned}
0 &= \sum_{c=1}^{C}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \frac{\partial\mathcal{L}}{\partial\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}}\right\rangle \\
&= -2\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p\right\rangle + 2\sum_{c=1}^{C}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p\right\rangle^2 - 2\lambda\sum_{c=1}^{C}\left\|\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}\right\|_2^2.
\end{aligned}$$

Then, in conjunction with (4), it follows that

$$\lambda = -\frac{1}{\alpha}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p\right\rangle + \frac{1}{\alpha}\sum_{c=1}^{C}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p\right\rangle^2.$$

Using this back in (3) we get,

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}} &= 2\left(\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p\right\rangle - \mathbf{1}_{\{y_p=c\}}\right)\mathbf{x}_p \\
&\quad + \left(\frac{2}{\alpha}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{y_p,\cdot}, \mathbf{x}_p\right\rangle - \frac{2}{\alpha}\sum_{c=1}^{C}\left\langle \left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}, \mathbf{x}_p\right\rangle^2\right)\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot}
\end{aligned}$$

To obtain the optimal A, it suffices to find $A_{\omega_{\mathbf{x}_p}}$ satisfying these conditions. To do so we consider the ansatz

$$\left[A_{\omega_{\mathbf{x}_p}}\right]_{c,\cdot} = \begin{cases} a\mathbf{x}_p & c = y_p \\ b\mathbf{x}_p & c \neq y_p. \end{cases}$$

Substituting this into our conditions it follows that $b = 0$ and $a = \frac{\sqrt{\alpha}}{\|\mathbf{x}_p\|_2}$.

■

Theorem 3. **Proof** Without loss of generality, we can assume $A_{\omega_{\mathbf{x}}}$ to be of the form $\mathbf{c}\mathbf{v}^\top$ for some $\mathbf{v} \in \mathbb{R}^d$. In particular, we assume that the vector $\mathbf{v}$ is of the same norm as $\mathbf{x}$. Then, locally in $\omega_{\mathbf{x}}$, we have

$$f_\theta(\mathbf{x} + \boldsymbol{\epsilon}) = \mathbf{c}\mathbf{v}^\top (\mathbf{x} + \boldsymbol{\epsilon}).$$

Therefore, $\mathbf{x}$ will only be misclassified by the neural network when $\mathbf{v}^\top (\mathbf{x} + \boldsymbol{\epsilon}) < 0$. From the Cauchy-Schwartz inequality we have that

$$-\|\mathbf{v}\|_2 \|\boldsymbol{\epsilon}\|_2 \leq \mathbf{v}^\top \boldsymbol{\epsilon} < -\mathbf{v}^\top \mathbf{x}.$$

Hence,

$$\|\boldsymbol{\epsilon}\|_2 > \frac{\mathbf{v}^\top \mathbf{x}}{\|\mathbf{v}\|_2},$$

the right-hand side of which is maximized when $\mathbf{v}$ is $\mathbf{x}$. ■

Theorem 4. **Proof** Using Theorem 10 and Theorem 13 it follows that

$$\begin{aligned} \mu^{(1\leftarrow l)}_{\omega_{\mathbf{x}}^{(1\leftarrow l)}} &= \left(A^{(l-1)}_{\omega_{\mathbf{x}}^{(l-1)}} \cdots A^{(1)}_{\omega_{\mathbf{x}}^{(1)}}\right)^\top \mu^{(l)}_{\omega_{\mathbf{x}}^{(l)}} \\ &= \left(A^{(l-1)}_{\omega_{\mathbf{x}}^{(l-1)}} \cdots A^{(1)}_{\omega_{\mathbf{x}}^{(1)}}\right)^\top \left(A^{(l)}_{\omega_{\mathbf{x}}^{(l)}}\right)^\top \mathbf{1} \\ &= \left(A^{(l)}_{\omega_{\mathbf{x}}^{(l)}} \cdots A^{(1)}_{\omega_{\mathbf{x}}^{(1)}}\right)^\top \mathbf{1} \\ &= \left(A^{(1\leftarrow l)}_{\omega_{\mathbf{x}}^{(1\leftarrow l)}}\right)^\top \mathbf{1}. \end{aligned}$$

Extending this to the L^{th} yields the desired result. ■

Proposition 9. **Proof** Using Theorem 4, the centroid of an aligned Jacobian is $\mu_{\mathbf{x}} = \mathbf{x}\mathbf{c}^\top \mathbf{1} = c\mathbf{x}$ where $c = \mathbf{c}^\top \mathbf{1}$. ■

Lemma 11. **Proof** This follows immediately from the application of Theorem 4. ■

Lemma 12. **Proof** Observe that in this setting we have

$$\Theta\left(\mathbf{x}, \mathbf{x}'\right) = \left\langle \nabla_{W^{(2)}} f_\theta(\mathbf{x}), \nabla_{W^{(2)}} f_\theta\left(\mathbf{x}'\right)\right\rangle + \left\langle \nabla_{W^{(1)}} f_\theta(\mathbf{x}), \nabla_{W^{(1)}} f_\theta\left(\mathbf{x}'\right)\right\rangle.$$

Therefore, noting that

$$\nabla_{W^{(2)}} f_\theta(\mathbf{x}) = \sigma\left(W^{(1)}\mathbf{x}\right)$$

and

$$\nabla_{W^{(1)}} f_\theta(\mathbf{x}) = W^{(2)} Q[\mathbf{x}]\mathbf{x}^\top,$$

the result follows. ■

Theorem 6. **Proof** In a similar way to Proposition 7, one can show that

$$\begin{aligned}\partial_t \mu_{\mathbf{x}} = \frac{\eta}{m}\sum_{p=1}^{m}\Bigg(&\left(m\left[\mathbf{x}_p\right]^\top W^{(2)} Q\left[\mathbf{x}_p\right] Q[\mathbf{x}]\left(W^{(2)}\right)^\top \mathbf{1}\right)\mathbf{x}_p \\ &+ \left(W^{(1)}\right)^\top Q[\mathbf{x}]\sigma\left(W^{(1)}\mathbf{x}_p\right) m\left[\mathbf{x}_p\right]^\top \mathbf{1}\Bigg).\end{aligned}$$

Using Lemma 12 this simplifies to

$$\partial_t\left(\left\langle \mathbf{x}', \mu_{\boldsymbol{\nu}}\right\rangle\right) = \frac{\eta}{m}\sum_{p=1}^{m}\Theta\left(\mathbf{x}', \mathbf{x}_p\right) m\left[\mathbf{x}_p\right].$$

■

Proposition 7. **Proof** Form Lemma 11, observe that

$$\partial_t \mu_{\mathbf{x}} = \left(\partial_t\left(W^{(2)}\right) Q[\mathbf{x}] W^{(1)} + W^{(2)} Q[\mathbf{x}]\partial_t\left(W^{(1)}\right)\right)^\top \mathbf{1},$$

where

$$\partial_t\left(W^{(i)}\right) = -\eta \nabla_{W^{(i)}}\mathcal{L}$$

for $i = 1, 2$. One can show that

$$\nabla_{W^{(1)}}\mathcal{L} = -\frac{1}{m}\sum_{p=1}^{m}\left(W^{(2)} Q\left[\mathbf{x}_p\right]\right)^\top \mathbf{m}\left[\mathbf{x}_p\right]\mathbf{x}_p^\top$$

and

$$\nabla_{W^{(2)}}\mathcal{L} = -\frac{1}{m}\sum_{p=1}^{m}\mathbf{m}\left[\mathbf{x}_p\right]\sigma\left(W^{(1)}\mathbf{x}_p\right)^\top.$$

Therefore,

$$\begin{aligned}\partial_t \mu_{\mathbf{x}} = \frac{\eta}{m}\sum_{p=1}^{m}\Bigg(&\left(\mathbf{m}\left[\mathbf{x}_p\right]^\top W^{(2)} Q\left[\mathbf{x}_p\right] Q[\mathbf{x}]\left(W^{(2)}\right)^\top \mathbf{1}\right)\mathbf{x}_p \\ &+ \left(W^{(1)}\right)^\top Q[\mathbf{x}]\sigma\left(W^{(1)}\mathbf{x}_p\right)\mathbf{m}\left[\mathbf{x}_p\right]^\top \mathbf{1}\Bigg).\end{aligned}$$

In particular,

$$\mathbf{m}\left[\mathbf{x}_p\right]^\top \mathbf{1} = 1 - \frac{\sum_{c=1}^{C} \exp\left(\left[f_\theta\left(\mathbf{x}_p\right)\right]_c\right)}{\sum_{c'=1}^{C} \exp\left(\left[f_\theta\left(\mathbf{x}_p\right)\right]_{c'}\right)} = 0,$$

meaning

$$\partial_t \mu_{\mathbf{x}} = \frac{\eta}{m} \sum_{p=1}^{m} \left(\mathbf{m}\left[\mathbf{x}_p\right]^\top W^{(2)} Q\left[\mathbf{x}_p\right] Q[\mathbf{x}] \left(W^{(2)}\right)^\top \mathbf{1} \right) \mathbf{x}_p.$$

Therefore, the result follows since $\partial_t \langle \mathbf{x}, \mu_{\mathbf{x}} \rangle = \langle \mathbf{x}, \partial_t \left(\mu_{\mathbf{x}}\right) \rangle$. ■

Appendix J. Compute Resources

Our experiments were computed on a range of NVIDIA GPUs including GTX TITAN Xs, RTX 2080Tis, and RTX 8000s. Below we indicate roughly how long each of our main experiments took to run.

Table 3: The computational resources utilised to perform the experiments of this work.

Experiment	Hardware	Time
Figure 9	GTX TITAN X	4 hours
Figure 2	GTX TITAN X	Less than 1 hour
Figure 3	GTX TITAN X	Less than 1 hour
Figure 4	GTX 1080 Ti	6 hours
Figure 5	GTX TITAN X	4 hours
Figure 6	GTX TITAN X	1 day
Table 1	GTX 1080 Ti, RTX 8000	5 days

Appendix K. Supporting Figures

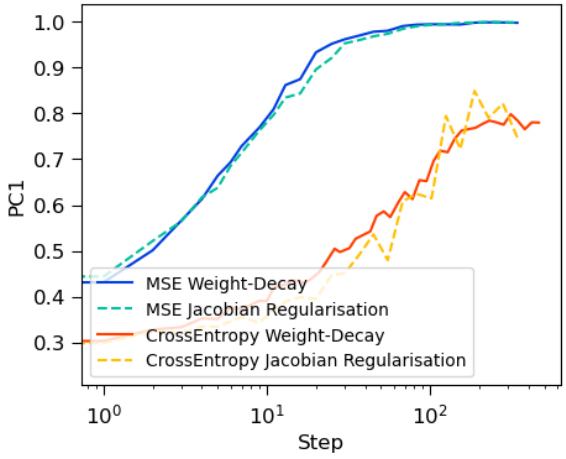


Figure 9: Under weight-decay and Jacobian regularisation, the effective rank of the Jacobian matrices evaluated at the training data tends towards rank one. Here we trained ReLU networks on the MNIST classification task [20] under the mean-squared error and cross-entropy loss functions using the AdamW optimizer [24]. Throughout training, we recorded the average explained variance of the first principal component of the Jacobians evaluated at the training data (PC1), namely $\frac{\sigma_1^2}{\sum_{i=1}^r \sigma_i^2}$, where σ are the singular values of the Jacobian. When this normalized value equals one, the Jacobian matrix is rank one.