Linear CNNs Discover the Statistical Structure of the Dataset Using Only the Most Dominant Frequencies

Hannah Pinson [1]   Joerie Lenaerts [1]   Vincent Ginis [1 2]

Abstract

We here present a stepping stone towards a deeper understanding of convolutional neural networks (CNNs) in the form of a theory of learning in linear CNNs. Through analyzing the gradient descent equations, we discover that the evolution of the network during training is determined by the interplay between the dataset structure and the convolutional network structure. We show that linear CNNs discover the statistical structure of the dataset with non-linear, ordered, stage-like transitions, and that the speed of discovery changes depending on the relationship between the dataset and the convolutional network structure. Moreover, we find that this interplay lies at the heart of what we call the "dominant frequency bias", where linear CNNs arrive at these discoveries using only the dominant frequencies of the different structural parts present in the dataset. We furthermore provide experiments that show how our theory relates to deep, non-linear CNNs used in practice. Our findings shed new light on the inner working of CNNs, and can help explain their shortcut learning and their tendency to rely on texture instead of shape.

In addition to a neural network's pre-defined architecture, the parameters of the network obtain an *implicit* structure during training. For example, it has been shown that weight matrices can exhibit structural patterns, such as clusters and branches (Voss et al., 2021; Casper et al., 2022). On the other hand, the input dataset also has an implicit structure arising from patterns and relationships between the samples. E.g., in a classification task, dogs are more visually similar to cats than to cars. A general theory on how the implicit structure in the network arises and how it depends on the structure of the dataset has yet to be developed. Here we derive such a theory for the specific case of two-layer, linear CNNs, and we provide experiments that show how our insights relate to the evolution of learning in deep, non-linear CNNs. Our approach is inspired by previous work on the learning dynamics in linear fully connected neural networks (FCNN) (Saxe et al., 2014; 2019), but we uncover the role of convolutions and show how they fundamentally alter the internal dynamics of learning. We start by discussing the two involved structures in terms of singular value decompositions (SVD): on the one hand the SVD of the input-output correlation matrix, representing the statistical dataset structure (Sec. 2.1); and the SVD of the convolutional network structure on the other hand (Sec. 2.2). We subsequently consider the equations of gradient descent in the slow learning regime, yielding a set of differential equations (Sec. 2.3). These equations describe the evolution of the implicit network structure, given the statistical dataset structure. Our analysis reveals that the convolutional network structure gives rise to additional factors in those gradient descent equations: these factors represent the interplay between the singular vectors associated to the dataset and those associated to the convolutional network (Sec. 2.4). This interplay changes the speed of discovery of the different parts of the dataset structure, i.e., the singular vectors representing broader to finer distinctions between classes, with respect to the speed of discovery in a FCNN (Sec. 3). Internally, this interplay also leads to a dominant frequency bias: only the dominant frequency components of each singular vector associated to the dataset are used by the CNN (Sec. 4). Experiments with more general datasets confirm the overall dynamics of learning and the existence of the dominant frequency bias (Sec. 5). Finally, we show how our theory relates to deep, non-linear CNNs used in practice (Sec. 6). Our results can be put in the context of the implicit regularisation resulting from training with gradient descent (Du et al., 2019; Arora et al., 2019; Gidel et al., 2019; Advani et al., 2020; Satpathi & Srikant, 2021). In (Saxe et al., 2014; 2019) the authors show that the structure of the final weights of linear FCNNs reflect the dataset structure, at least when starting from random initial conditions and

[1]Data Lab/Applied Physics, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium [2]Harvard John A. Paulson School of Engineering and Applied Sciences, Harvard University, 29 Oxford Street, Cambridge, Massachusetts 02138, USA. Correspondence to: Hannah Pinson <hannah.pinson@vub.be>.

when using gradient descent. They find analytical solutions for the learning dynamics in a two-layer, linear FCNN. These solutions indicate the stage-like discovery of different structural parts of the dataset. In (Gidel et al., 2019), the discrete dynamics are studied and in (Braun et al., 2022; Atanasov et al., 2022), the authors develop the theory for different initialisation regimes. The relationship with a.o. early stopping and learning order constancy for both CNNs and FCNNs is further studied in (Hacohen & Weinshall, 2022) as the principal component bias or PC-bias. Furthermore, similar theories exist for linear and shallow non-linear autoencoders (Bourlard & Kamp, 1988; Pretorius et al., 2018; Refinetti & Goldt, 2022). Finally, it has been shown that linear CNNs trained with gradient descent exhibit an implicit bias in the form of sparsity in the frequency domain (Gunasekar et al., 2018). We here show which mechanism gives rise to this sparsity in the frequency domain, and we find *which* frequencies are developed over time.

1. Prerequisites

Notation The input consists of $n \times n$ images denoted $\boldsymbol{X}^s$ where s is the sample index. We will omit this index when the context is clear. Often, we will need a vectorized (or 'flattened') representation of $\boldsymbol{X}$ and all other $n \times n$ matrices: we turn those matrices in $n^2 \times 1$ vectors through stacking all rows one after the other, and transposing the result. The resulting column vector is denoted with a lower case letter, e.g., $vec(\boldsymbol{X}) = \boldsymbol{x}$. We reverse this operation to show the vectors as 2D images in figures. An index j into the vector which results from vectorizing a 2D matrix, denoted a 'vec-2D' index, runs from 0 to $n^2 - 1$. The corresponding indices in the 2D matrix are given by $l = div(j, n)$ and $m = mod(j, n)$, with div integer division and mod the modulo operator. The row l of a matrix $\boldsymbol{B}$ is denoted $\boldsymbol{B}_{l,:}$; the column m is denoted $\boldsymbol{B}_{:,m}$. $*$ denotes the complex conjugate.

Architecture and assumptions In our theory, we consider a CNN with a single convolutional layer, a flatten layer, and a single fully connected layer. There are only linear activation functions. The convolutional layer consists of a single kernel $\boldsymbol{K}$ with the same dimensions as the input images ($n \times n$). When the kernel reaches the boundaries of the image, it 'wraps' around, such that the convolution is circular. This simplifies the math while still being a good approximation to the zero-padding often used in practice (Gray, 2006). The task is image classification with one-hot encoded labels, we assume a slow learning rate, an MSE loss, and small, random initial conditions.

The singular value decomposition (SVD) The SVD of an $p \times n^2$ ($p < n^2$) real-valued matrix $\boldsymbol{B}$ is given by: $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, where $\boldsymbol{U}$ is an orthogonal $p \times p$ matrix ($\boldsymbol{U}^T = \boldsymbol{U}^{-1}$) containing the left singular vectors as

columns, $\boldsymbol{V}$ is an orthogonal $n^2 \times n^2$ matrix containing the right singular vectors as columns, and $\boldsymbol{S}$ is a $p \times n^2$ rectangular diagonal matrix containing the p real and positive singular values, denoted s_α. The singular vectors and values are by definition sorted from highest to lowest singular value. The modes $\boldsymbol{M}^{(\alpha)}$, $\alpha \in \{0, \ldots, p\}$ form the decomposition of the original matrix: $\boldsymbol{B} = \sum_\alpha \boldsymbol{M}^{(\alpha)}$, with $\boldsymbol{M}^{(\alpha)} = s_\alpha \boldsymbol{U}_{:,\alpha} \boldsymbol{V}_{\alpha,:}^T$.

The vectorized 2D discrete Fourier transform The Fourier transform is another type of decomposition, used to decompose a signal in its frequency components. The resulting amplitudes in the frequency domain, the 'Fourier coefficients', describe the relative importance of each frequency in the original signal. For an $n \times n$ matrix $\boldsymbol{B}$, the conventional 2D discrete Fourier transform (denoted $\mathfrak{F}_{2D}$) is given by $\mathfrak{F}_{2D}(\boldsymbol{B})_{\mu,\nu} = \Sigma_{s,t}^n (\omega_n)^{\mu s} (\omega_n)^{\nu t} B_{s,t}$ with $\omega_n = \exp(-2\pi i/n)$ and $i^2 = -1$. The result is a 2D $n \times n$ complex-valued matrix, whose values tell us the relative importance of the n^2 different spatial frequencies: n frequencies in the vertical direction, combined with n frequencies in the horizontal direction. We will make use of an equivalent transform, but instead of applied to a 2D matrix, applied to the 1D vector that results from vectorizing the $n \times n$ matrix first. This transform is given by multiplication with the $n^2 \times n^2$ matrix $\boldsymbol{Q}$, i.e., $\boldsymbol{Q}\boldsymbol{x} = \frac{1}{n} vec(\mathfrak{F}_{2D}(\boldsymbol{X}))$. $\boldsymbol{Q}$ is illustrated in Fig.1; for its exact definition, see SI. Note that this operation is different from the 1D discrete Fourier transform of a vector: we will call it the vectorized 2D discrete Fourier transform, or vec-2D DFT in short.
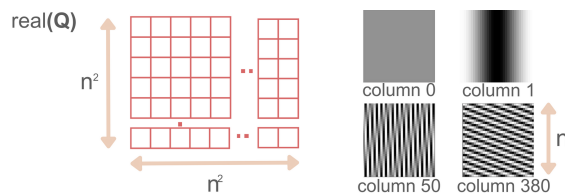

Figure 1. The symmetric matrix $\boldsymbol{Q}$ of wich each row/column is a vectorized Fourier eigenmatrix (4 examples given).

If the original vector is real-valued, the Fourier spectrum of this vector will exhibit symmetries. Let a vec-2D frequency index j correspond to a pair of horizontal and vertical frequency indices (μ, ν), then the symmetric vec-2D frequency index j_{symm} corresponds to the pair $(n - \mu, n - \nu)$, thus $j = (n-1)\mu + \nu$ and $j_{symm} = (n-1)(n-\mu) + (n-\nu)$. The symmetry in the spectrum is then given by the equation $(\boldsymbol{Q}\boldsymbol{x})_{j_{symm}} = (\boldsymbol{Q}\boldsymbol{x})_j^*$.

2. Differential Equations of Gradient Descent

We want to study the relationship between the dataset structure and the network structure during training with gradient descent. To this end, we first discuss the SVDs that cap-
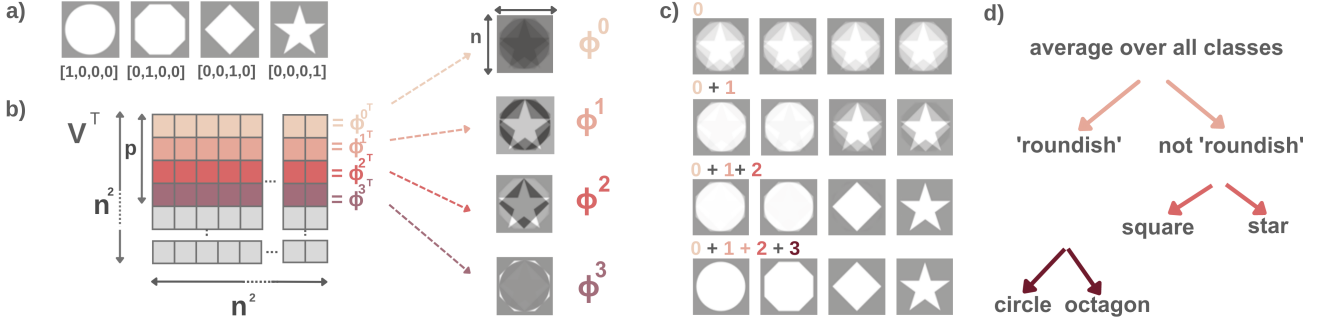
Figure 2. Illustration of the relationship between statistical structure and SVD. **a**) Example dataset consisting of geometric shapes: a circle, an octagon, a square and a star. There is only one sample per class. **b**) Visualisation of $\boldsymbol{V}^T$ and its first p rows, the singular vectors denoted ϕ^{α} (here p=4), as $n \times n$ matrices. The other rows carry no meaning. **c**) Visualisation of the summation of the modes $\boldsymbol{M}^{(\alpha)}$ of $\boldsymbol{\Sigma^{yx}}$, given by the SVD (see prerequisites). Each mode is first reshaped to p $n \times n$ matrices. **d**) Interpretation of the implicit hierarchical structure in $\boldsymbol{\Sigma^{yx}}$, which can be derived from the visualisations in subfigures b) and c).

ture these structures. We will then show how the equations of gradient descent can be reformulated in terms of those SVDs.

2.1. Statistical Structure of the Dataset

The dataset has an implicit structure, e.g., some images contain the same backgrounds, and images of cats are similar to images of dogs. Given a general dataset, it is unclear a priori which patterns might be relevant during the training of a network. For linear CNNs, however, we find that the relevant structure is given by the statistical structure: the structure that can be derived from the correlations in the dataset. In Fig.2, we illustrate the statistical structure of a dataset consisting of geometric shapes. Given ground-truth labels $\boldsymbol{y}$ when there are p classes, we can define the $p \times n^2$ input-output correlation matrix $\boldsymbol{\Sigma^{yx}}$ as:

$$\boldsymbol{\Sigma^{yx}} = \langle \boldsymbol{y}\boldsymbol{x}^T \rangle, \tag{1}$$

where $\boldsymbol{x}$ is an $n^2 \times 1$ vectorized sample, and $\langle \rangle$ denotes the average over all samples. Note that $\boldsymbol{y}\boldsymbol{x}^T$ is an outer product, i.e., $\Sigma_{l,m}^{yx} = \langle y_l x_m \rangle$. Given one-hot encoded labels, each row of $\boldsymbol{\Sigma^{yx}}$ contains the vectorized average image of each class. The structure relevant when training a (linear) neural network is captured through the singular value decompositions (SVD) of this matrix (see (Saxe et al., 2019) and Fig. 2(b-d)). The SVD of $\boldsymbol{\Sigma^{yx}}$ is given by:

$$\boldsymbol{\Sigma^{yx}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T. \tag{2}$$

In this particular case, the first p rows of $\boldsymbol{V}^T$, which we denote ϕ^{α}, $\alpha \in \{0, \dots, p-1\}$, are the principal components of the averaged class images (Pearson, 1901; Jolliffe, 2002). The $p \times p$ matrix $\boldsymbol{U}$ then contains the coefficients to reconstruct the p vectorized, averaged class images as linear combinations of the p right singular vectors ϕ^{α}. By

visualizing the singular vectors ϕ^{α} and the corresponding modes $\boldsymbol{M}^{\alpha}$, we can see the SVD here embodies broader to finer distinctions between the averaged class images (see Fig.2(b-d)). As such, it captures our intuitive understanding of an implicit hierarchical structure between the shapes in the dataset.

We can also consider the singular value decomposition of the input-input correlation matrix $\boldsymbol{\Sigma^{xx}} = \langle \boldsymbol{x}\boldsymbol{x}^T \rangle$. In general, this matrix could have n^2 different singular values and corresponding singular vectors, and its exact shape influences the dynamics of learning as well. However, to focus on the effect of a convolutional architecture only, we consider the case where $\boldsymbol{\Sigma^{xx}}$ is diagonalizable in the basis given by $\boldsymbol{V}$:

$$\boldsymbol{\Sigma^{xx}} = \boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{V^T}, \tag{3}$$

where $\overline{\boldsymbol{\Sigma^{xx}}}$ is a diagonal matrix with only the first p entries on the diagonal non-zero. In our case of classification with one-hot encoded labels, this is the case if all images in a class are the same (see SI). The less images deviate from the average image for their class, the better this assumption holds. Finally, given any set of predictions $\hat{\boldsymbol{y}}$ for a set of input samples $\boldsymbol{x}$, we can also compute the input-predicted output correlation matrix $\boldsymbol{\Sigma^{\hat{y}x}}$ (cfr. Eq. 1):

$$\boldsymbol{\Sigma^{\hat{y}x}} = \langle \hat{\boldsymbol{y}}\boldsymbol{x^T} \rangle, \tag{4}$$

where $\boldsymbol{\Sigma^{\hat{y}x}}$ has dimensions $p \times n^2$. To relate the evolution of this matrix to the structure of the dataset, we study $\boldsymbol{\Sigma^{\hat{y}x}}$ in the SVD basis associated to $\boldsymbol{\Sigma^{yx}}$ (see Eq. 2):

$$\boldsymbol{\Sigma^{\hat{y}x}} = \boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^T. \tag{5}$$

The $p \times n^2$ matrix $\boldsymbol{A}$ is not necessarily rectangular diagonal; if it would be, and if the diagonal elements would be positive real numbers, Eq. 5 would be an SVD with singular values given by the values on the diagonal of $\boldsymbol{A}$. In that case, we could say that $\boldsymbol{\Sigma^{\hat{y}x}}$ has the same structural elements as $\boldsymbol{\Sigma^{yx}}$, only with different strengths.

2.2. Structure of the Network

We also want to study the implicit structure of the convolutional network through an SVD. To this end, we replace the convolution operations with convolution-equivalent weight matrices, such that we can study the SVD of these matrices (Sedghi et al., 2019). A convolutional layer can be seen as a constrained fully connected layer: the sliding of the kernel over the image is actually a repeated application of the same weights (Goodfellow et al., 2016). A convolution-equivalent weight matrix is then a matrix where elements are repeated in a fixed, circulant pattern. Such a matrix is called a doubly block circulant matrix (see Jain 1989; Sedghi et al. 2019), denoted $\boldsymbol{dbc}$ (definition, see SI). We can also define such a matrix through its eigendecomposition: the $n^2 \times n^2$ doubly block circulant matrix of the kernel $\boldsymbol{K}$ is given by

$$\boldsymbol{dbc}(\boldsymbol{K}) = n\,\boldsymbol{Q}^{-1}diag(\boldsymbol{Qk})\boldsymbol{Q} \qquad (6)$$

and it thus acts as a convolution-equivalent weight matrix:

$$vec(\boldsymbol{X} \circledast \boldsymbol{K}) = \boldsymbol{dbc}(\boldsymbol{K})\boldsymbol{x} \qquad (7)$$

where $\circledast$ denotes circular convolution. Here $\boldsymbol{k}$ is the vectorized kernel, and $\boldsymbol{Q}$ is the matrix corresponding to the vec-2D DFT (see prerequisites and Fig. 1). $diag(\boldsymbol{Qk})$ is an $n^2 \times n^2$ diagonal matrix with the n^2 elements of the vec-2D DFT of $\boldsymbol{k}$, thus $\boldsymbol{Qk}$, on the diagonal. Readers familiar with the convolution theorem might recognize it in Eq. 6: to apply a convolution, transform the input to the Fourier domain, multiply with the transformed kernel, and transform the result back to the original domain.

In short, for our theoretical derivations, we consider the network with predictions $\hat{\boldsymbol{y}}$ given by:

$$\hat{\boldsymbol{y}} = \boldsymbol{W}\,\boldsymbol{dbc}(\boldsymbol{K})\boldsymbol{x}, \qquad (8)$$

with $\boldsymbol{W}$ the $p \times n^2$ weight matrix of the fully connected layer. We can again formalize the notion of implicit structure through considering the singular value decompositions/eigenvalue decompositions of the network's weight matrices $\boldsymbol{W}$ and $\boldsymbol{dbc}(\boldsymbol{K})$. While a general weight matrix $\boldsymbol{W}$ could in principle have any singular-/eigenvalue decomposition, reflecting its unconstrained structure, the eigendecomposition of $\boldsymbol{dbc}(\boldsymbol{K})$ is *fixed* and always given by Eq. 6: the eigenvectors stay the same. Only the eigenvalues, given by $n\boldsymbol{Qk}$, can change. This reflects the fact that of the $n^2 \times n^2$ matrix $\boldsymbol{dbc}(\boldsymbol{K})$, only n^2 values can change independently. These values are given by the $n \times n$ kernel. The different singular values of $\boldsymbol{dbc}(\boldsymbol{K})$ are actually given by $n|(\boldsymbol{Qk})_i|$. For a discussion on the role of the phases, see SI.

2.3. Differential Equations of Gradient Descent

We can now focus on what happens during training with gradient descent. We start from a MSE loss function $L =$

$\frac{1}{2}\sum_{l=0}^{p}(y_l - \hat{y}_l)^2$, and its gradient is used to update the network parameters at every discrete timestep:

$$\Delta\boldsymbol{k} = -\lambda\frac{\partial L}{\partial\boldsymbol{k}^T}, \qquad \Delta\boldsymbol{W} = -\lambda\frac{\partial L}{\partial\boldsymbol{W}^T}, \qquad (9)$$

with λ the learning rate. For the convolutional layer, the gradient of the loss with respect to the kernel is in itself given by a convolution, but with a flipped image (see, e.g., Goodfellow et al. 2016, Ch. 9). The gradients are then given by:

$$\frac{\partial L}{\partial\boldsymbol{k}^T} = dbc(\boldsymbol{X}_{flip})\sum_{l=0}^{p}(y_l - \hat{y}_l)(\boldsymbol{W}^T)_{:,l} \qquad (10)$$

$$\left(\frac{\partial L}{\partial\boldsymbol{W}^T}\right)_{l,:} = (y_l - \hat{y}_l)\boldsymbol{x}^T dbc(\boldsymbol{k})^T. \qquad (11)$$

In the slow learning regime, the weights and kernel change minimally with each update. The updates can then be approximated through an averaged update over samples (Saxe et al. 2019, SI p1.). This allows us to introduce the matrices $\boldsymbol{\Sigma}^{\boldsymbol{yx}}$ (cfr. Eq. 1) and $\boldsymbol{\Sigma}^{\hat{\boldsymbol{y}}\boldsymbol{x}}$ (cfr. Eq. 5) in the equations. Given the slow learning rate, the discrete equations can also be reformulated as differential equations. We can subsequently transform the different variables using the SVD basis of $\boldsymbol{\Sigma}^{\boldsymbol{yx}}$:

$$\overline{\boldsymbol{W}} = \boldsymbol{U}^T\boldsymbol{W}\boldsymbol{R} \qquad (12)$$

$$\overline{\boldsymbol{dbc}(\boldsymbol{K})} = \boldsymbol{R}^{-1}\boldsymbol{dbc}(\boldsymbol{K})\boldsymbol{V} \qquad (13)$$

$$= n\boldsymbol{R}^{-1}\boldsymbol{Q}^{-1}diag(\boldsymbol{Qk})\boldsymbol{QV} \qquad (14)$$

here $\boldsymbol{R}$ is an arbitrary $n^2 \times n^2$ invertible matrix. This matrix reflects the freedom in the actual shape of the weights, as long as their product remains the same. Now we have $\boldsymbol{\Sigma}^{\hat{\boldsymbol{y}}\boldsymbol{x}} = \boldsymbol{W}\,dbc(\boldsymbol{K})\boldsymbol{\Sigma}^{\boldsymbol{xx}} = \boldsymbol{UAV}^T = \boldsymbol{U}\,\overline{\boldsymbol{W}}\,\overline{\boldsymbol{dbc}(\boldsymbol{K})}\,\overline{\boldsymbol{\Sigma}^{\boldsymbol{xx}}}\,\boldsymbol{V}^T$ (see also Eq. 3), and we arrive at the differential equations in the following, somewhat complicated shape (for full derivation, see SI) :

$$\frac{1}{n\lambda}\left(\frac{d\,diag(\boldsymbol{Q}^{-1}\boldsymbol{k})}{dt}\right)_{j,j} = \boldsymbol{Q}_{j,:}^{-1}\boldsymbol{R}^{-*T}\overline{\boldsymbol{W}}^{*T}(\boldsymbol{S} - \boldsymbol{A})(\boldsymbol{QV})_{:,j}^T, \quad (15)$$

$$\frac{1}{n\lambda}\left(\frac{d\overline{\boldsymbol{W}}}{dt}\right) = (\boldsymbol{S} - \boldsymbol{A})(\boldsymbol{QV})^T diag(\boldsymbol{Qk})\boldsymbol{Q}^{-1}\boldsymbol{R}, \qquad (16)$$

where $diag(\boldsymbol{Q}^{-1}\boldsymbol{k})_{j,j} = (\boldsymbol{Q}^{-1}\boldsymbol{k})_j = (\boldsymbol{Q}^{*T}\boldsymbol{k})_j$, and all derivatives of off-diagonal elements of $diag(\boldsymbol{Q}^{-1}\boldsymbol{k})$ are zero. There are a couple of key observation to make about this set of differential equations (Eqs. 15 and 16): firstly, they are coupled and non-linear, making them difficult or impossible to solve analytically but for very specific cases. We can also see that at every step $\boldsymbol{A}$ is brought closer to $\boldsymbol{S}$ (cfr. Eq. 2), that $\boldsymbol{A}$ thus becomes diagonal, and that the updates become zero if these matrices are equal: i.e., the predictions perfectly match the labels. Furthermore, the dynamics of learning are influenced by the vectors $\boldsymbol{QV}_{:,\alpha} = \boldsymbol{Q}\boldsymbol{\phi}^{\boldsymbol{\alpha}}$, the vec-2D Fourier transforms of the right singular vectors $\boldsymbol{\phi}^{\boldsymbol{\alpha}}$.

This turns out to be a crucial point: this embodies the interplay between the dataset structure (characterised by V) and the convolutional network structure (characterised by Q); in the equivalent equations for fully connected networks, no such extra factors are present (see Eqs. 19 and 20 below).

2.4. Comparison with Fully Connected networks

The dynamics of learning during gradient descent in linear FCNNs are discussed in (Saxe et al., 2014; 2019). In the case of a two-layer linear FCNN, we have

$$\Sigma^{\hat{y}x}{}_{FC} = <\hat{y}_{FC}x^T> = W^2W^1\Sigma^{xx} = UA_{FC}V^T, \quad (17)$$

where W^2 and W^1 are the weight matrices for the second and first layer of the FCNN network, respectively. The matrices U and V are the same as before (Eq. 2). However, the evolution of the matrices $\Sigma^{\hat{y}x}{}_{FC}$ and A_{FC} will be different from the evolution of their equivalents in the CNN case. Using transformed weight matrices:

$$\overline{W^1} = R'^{-1}W^1V, \qquad \overline{W^2} = U^TW^2R', \quad (18)$$

with R' an arbitrary invertible matrix, the authors in (Saxe et al., 2014; 2019) then arrive at the system of differential equations:

$$\frac{1}{\lambda}\overline{W^1}t = \overline{W^2}^T(S - A_{FC}) \quad (19)$$

$$\frac{1}{\lambda}\overline{W^2}t = (S - A_{FC})\overline{W^1}^T. \quad (20)$$

which can be compared to the equations we derived for the CNNs (Eqs. 15 and 16 and SI). The authors subsequently show that, starting from small, random initial conditions, $\overline{W}^2$ and $\overline{W}^1$ each quickly become diagonal. Comparing this to the case of the CNN, we see that $\overline{dbc(K)}$ cannot be diagonal (Eq. 13). This indicates that the constrained structure of the convolutional layer in itself cannot fully reflect the dataset structure, while a fully connected layer in an FCNN can.

3. Non-linear Learning Dynamics

We now explore how the interplay between V and Q influences the dynamics of learning. In particular, we study the evolution of the predictions: we can study this evolution through analysing the evolution of the matrix A (see Eq. 5), with $A = \overline{W}\,\overline{dbc(K)}\,\Sigma^{xx}$. We will first derive analytical solutions for this evolution for a particular, illuminating dataset. Later we will show that the characteristics of the found trajectories roughly hold for more general datasets as well. Consider the dataset where the image for each class is given by:

$$X^{(c)}_{l,m} = b^{(c)}\cos(2\pi\frac{\mu l}{n} + 2\pi\frac{\nu m}{n}). \quad (21)$$

Here c is the class index, and for each class we pick a different pair of μ and $\nu \in \{0, \cdots, n-1\}$, indexing the n possible horizontal and vertical frequencies. l and m index the pixels. $b^{(c)}$ is the amplitude of the frequency (one value per class). Examples of this type of input are shown in Fig. 1, since the vectorized input images each correspond to the real part of a column of Q (apart from the amplitude). Since the class vectors are already orthogonal, normalisation yields the p singular vectors $V_{:,\alpha} = \phi^\alpha$. This type of input decouples the differential equations (Eqs. 15 and 16): the intuition behind this is that the structural 'mismatch' between dataset and network partly vanishes, because we pick the vectors ϕ^α to be orthogonal to the real part of the columns of Q. However, Q is complex-valued and the singular vectors are real-valued; therefore an artefact of the mismatch will remain in the form of a factor $\frac{1}{\sqrt{2}}$. In the SI, we first define a set of specific initial conditions such that A starts out rectangular diagonal. Subsequently we show that given the Eqs. 15 and 16, each $a_\alpha = A_{\alpha,\alpha}$ exactly follows a sigmoidal trajectory:

$$a_\alpha(t) = \frac{s_\alpha e^{2n\lambda d_\alpha s_\alpha t}}{e^{2\lambda n d_\alpha s_\alpha t} - 1 + s_\alpha/a_\alpha^{(0)}}, \quad (22)$$

with $d_\alpha = \frac{1}{\sqrt{2}}$ (unless $\mu = 0$ and $\nu = 0$, then $d_\alpha = 1$), and $a_\alpha^{(0)} = a_\alpha(t = 0)$. The time, in number of samples, to grow from an initial value $a_\alpha^{(0)} = \epsilon$ with $\epsilon \ll 1$, to a value $a_\alpha = s_\alpha - \epsilon$, is approximately given by $\frac{d_\alpha n}{s_\alpha \lambda}$ (see Saxe et al. 2019). Under analogous initial conditions and assumptions, the values $a_\alpha(t)$ in a linear FCNN follow a similar sigmoidal trajectory (see Saxe et al. 2019). However, there are no factors d_α and n: the number of samples needed to reach convergence for each value a_α is approximately given by $\frac{1}{s_\alpha \lambda}$ for FCNN.

Given these analytical results, we can draw the following conclusions: first, given the sigmoidal trajectories, the predictions of the network change in such a way that the different structural modes of Σ^{yx} are discovered with rapid, stage-like transitions by both types of networks. This discovery is ordered in time (through the factor $\frac{1}{s_\alpha}$), from highest singular value to lowest singular value, corresponding to the discovery from broader to finer distinctions between the classes. However, the CNN exhibits a different effective learning rate $\lambda_{eff} = nd_\alpha\lambda$ for each mode α w.r.t. the FCNN. The factor n is a speed-up resulting from the convolution with a kernel of dimension n; the factor $d_\alpha <= 1$ reflects a delay stemming from the mismatch between the dataset structure and the constrained network structure. In Fig. 3, we show the results of experiments with a dataset of 'pure cosines' as described by Eq. 21 (for details, see SI). The first class/singular vector ϕ^0 is a constant image, i.e., $\mu = \nu = 0$; the subsequent classes have different randomly selected frequencies with decreasing amplitudes. The FCNN is trained with a higher learning rate $\lambda_{FC} = n\lambda_{CNN}$,

such that the graphs of both networks can be compared on the same plot. $d_0 = 1$, such that the trajectories of a_0 for the CNN and FCNN overlap. $d_\alpha = 1/\sqrt{2} \approx 0.71$ for the other modes, resulting in a shift between the trajectories. The theoretical predictions, made using the same set of initial conditions, exactly match the experimental trajectories.

We derived the analytical solutions using aligned, balanced initial conditions (see SI). These conditions render $\boldsymbol{A}$ rectangular diagonal from the beginning. Several previous studies show that when starting from fully random, small initial conditions, $\boldsymbol{A}$ very quickly becomes diagonal as well (Saxe et al., 2014; 2019; Atanasov et al., 2022; Braun et al., 2022). We discuss this further in the SI. Apart from a small additional delay related to this alignment phase, the trajectories of $\boldsymbol{A}$ when starting from small, random initial conditions are thus described by similar sigmoidal curves.
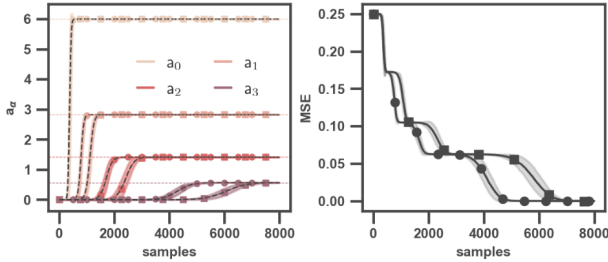

Figure 3. Experimental evolution and theoretical predictions (left) of a_α, and experimental evolution of the MSE loss (right), for linear FCNN (round markers) and linear CNN (square markers), trained on the same dataset. The black, dashed lines indicate the predictions. All plotted lines are averaged over trials, shadowed regions indicate the standard deviations. Dashed horizontal lines left indicate the values s_α.

4. Dominant Frequency Bias

So far, we have studied the evolution of the predictions for two-layer linear CNNs and FCNNs. It turns out that the way these networks arrive at those predictions internally is very different. We will show that during training the kernel of the linear CNN becomes an implicit regularizer: it filters out a small fraction of the frequency components present in the dataset. The whole network arrives at its predictions using only those frequencies. Driving this implicit regularisation is a soft winner-takes-all dynamics (sWTA) (Lazzaro et al., 1988; Fang et al., 1996; Fukai & Tanaka, 1997), where during training different vec-2D Fourier coefficients of the kernel $|\boldsymbol{Qk}|_j$ 'compete' with each other to be part of the final kernel. Whether they 'win' depends on their initial values and the corresponding coefficients of the singular vectors of the dataset $|\boldsymbol{Q\phi}^\alpha|_j$. The word 'soft' denotes that there is not a single winning frequency. This sWTA dynamics—and thus the implicit regularisation—can be derived from the given differential equations.

To see this, we first consider a slightly more general type of dataset: a dataset for which the modes now consist of *sums* of cosines, where each cosine possibly has a different amplitude (see SI). However, we still assume frequencies are not shared between modes, such that modes remain decoupled. Formally, given an index j, $|\boldsymbol{Q\phi}^\alpha|_j$ is non-zero for only one mode α. $\sigma^{(\alpha)}$ then denotes the set of all vec-2D indices j of the frequencies associated to a mode α. If j is the vec-2D index corresponding to the pair of indices (μ, ν), then with j_{symm} we denote the vec-2D index that corresponds to $(n-\mu, n-\nu)$. Since the singular vectors are real-valued, their Fourier spectra exhibit symmetries, and $|\boldsymbol{Q\phi}^\alpha|_j = |\boldsymbol{Q\phi}^\alpha|_{j_{symm}}$. $\sigma^{(\alpha)}_{symm}$ includes all the symmetric indices j_{symm} as well. In the SI, we show that when starting from small, random initial conditions, the development of the vec-2D Fourier coefficients of the kernel $|\boldsymbol{Qk}|_j$ is approximately given by:

$$\frac{1}{2n\lambda} \frac{d|(\boldsymbol{Qk})_j|^2}{dt} = |(\boldsymbol{Qk})_j|^2 |(\boldsymbol{Q\phi}^\alpha)_j|(s_\alpha - a_\alpha), \quad (23)$$

when $j \in \sigma^{(\alpha)}_{symm}$, with

$$a_\alpha = n\overline{\boldsymbol{\Sigma^{xx}}}_{\alpha,\alpha} \Big(2|(\boldsymbol{Qk})_j|^2|(\boldsymbol{Q\phi}^\alpha)_j| $$
$$+ \sum_{j' \in \sigma^{(\alpha)}_{symm} \setminus \{j, j_{symm}\}} |(\boldsymbol{Qk})_{j'}|^2|(\boldsymbol{Q\phi}^\alpha)_{j'}| \Big). \quad (24)$$

While s_α can be considered as the input that drives the system, $-|(\boldsymbol{Qk})_j|^2$ is a self-inhibition term, and $-\sum_{j' \in \sigma^{(\alpha)}_{symm} \setminus \{j, j_{symm}\}} |(\boldsymbol{Qk})_{j'}|^2$ is a term that captures the lateral inhibition coming from the other 2D-vec kernel Fourier coefficients associated to the same mode. A formulation and analysis of sWTA dynamics close to the equations we consider is given in (Fukai & Tanaka, 1997). Note that unlike the common description of WTA dynamics as a system of competing neurons, we here have a system of 'competing' kernel Fourier coefficients. The overall dynamics are as follows: if we initialize the kernel with small initial values, its vec-2D Fourier coefficients $|(\boldsymbol{Qk})_i|$ will also be small. For each input mode α, we have a number of associated frequency indices j, and the corresponding terms $|(\boldsymbol{Qk})_j|$ each contribute to the respective effective singular value a_α (Eq. 24). Initially, a_α is much smaller than s_α. Therefore, the values $|(\boldsymbol{Qk})_j|^2$ initially grow exponentially (Eq. 23 when $a_\alpha \approx 0$). However, they do so with *different* exponents; within the set of frequencies associated to the mode α, the difference lies in the factors $|(\boldsymbol{Q\phi}^\alpha)_j|$. As soon as the values $|(\boldsymbol{Qk})_j|^2$ start growing, the self-inhibition as well as the mutual inhibition between the coefficients associated to the same mode starts to take off. In practice, the coefficients $|(\boldsymbol{Qk})_j|$ with the largest factors $|(\boldsymbol{Q\phi}^\alpha)_j|$ very strongly inhibit the growth of the other coefficients, such that only the former coefficients grow and significantly contribute to the effective singular value a_α. Eventually, a_α
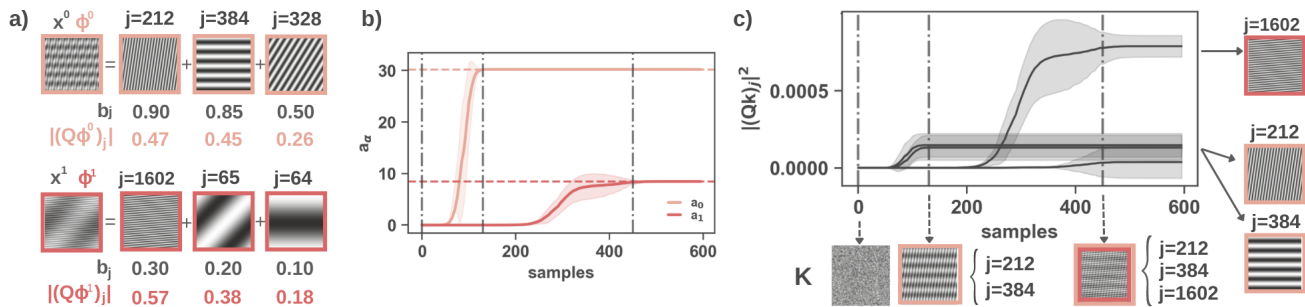
Figure 4. Illustration of the dominant frequency bias. **a)** Illustration of the class images as sums of pure cosines. Since the class vectors $\boldsymbol{x}^c$ are orthogonal, the singular vectors $\boldsymbol{\phi}^\alpha$ are normalised versions of the vectors $\boldsymbol{x}^c$. **b)** Evolution of the effective singular values. **c)** Evolution of the squared Fourier coefficients of the kernel (middle), evolution of the kernel $\boldsymbol{K}$ at selected timepoints (below), frequencies corresponding to highest Fourier coefficients of final kernel (right). Compare the development of the kernel over time with the discovery of the modes, shown in b), and the values of $|\boldsymbol{Q}\boldsymbol{\phi}^\alpha|_j$, shown in a). **b), c)**: solid lines: averages over experimental runs. Shaded regions: standard deviations. Vertical dashed lines: selected timepoints. Horizontal dashed lines: singular values s_α.

saturates to its final value s_α, and all derivatives become zero. Depending on the factors $|(\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_j|$, we thus end up with one or more coefficients $|(\boldsymbol{Q}\boldsymbol{k})_j|$ that have 'won' the competition; this means the final kernel consists of a sum of those frequencies only.

While in a linear, fully connected network, the weight matrices take on the inherent structure of the dataset (see Sec. 2.4); the kernel in our linear CNN only picks up the most dominant frequencies associated to each mode, i.e., singular vector associated to the dataset. It thus acts as a filter; not necessarily a low-pass filter, but a filter of the most dominant frequencies. The results of an experiment with two classes, each a sum of pure cosines with different amplitudes, is illustrated in Fig. 4.

For more general datasets, the singular vectors can in general share the same frequency. Formally, $|\boldsymbol{Q}\boldsymbol{\phi}^\alpha|_j$ can be significant for different modes α. In this case, Eq. 23 does not hold and we have to revert back to the more general equation Eq. 15. The experiments discussed in the next section show that in those more general cases a form of the dominant frequency filtering is still present. However, the dominancy of a frequency is now determined through another factor as well: assume two values $|\boldsymbol{Q}\boldsymbol{\phi}^\beta|_j$ and $|\boldsymbol{Q}\boldsymbol{\phi}^\gamma|_j$, with $\beta < \gamma$, are significant. During the discovery of mode β the kernel Fourier coefficient $|(\boldsymbol{Q}\boldsymbol{k})_j|$ develops. Later, at the start of the discovery of the mode γ, the coefficient $|(\boldsymbol{Q}\boldsymbol{k})_j|$ starts from a higher value than coefficients $|(\boldsymbol{Q}\boldsymbol{k})_{j'}|$ that were not developed before, and that still have their small, random initial value. Loosely speaking, this gives the coefficient $|(\boldsymbol{Q}\boldsymbol{k})_j|$ a 'competitive advantage'. This also means that frequencies that are dominant in the first few modes (thus high $|\boldsymbol{Q}\boldsymbol{\phi}^\alpha|_j$ for the first modes α) are the frequencies that are more likely to be important for the final kernel.

5. Experiments with More General Datasets

We now report the results of training a linear CNN from small, random initial conditions on two more general datasets: the dataset of geometric shapes (see Fig. 2) and on the dataset consisting of the first 4 classes of CIFAR-10 ('airplane', 'automobile', 'bird' & 'cat'), which we will call CIFAR-4. We subtract the first mode, i.e. the average over images, from the CIFAR-4 dataset. Therefore only the 3 modes needed to distinguish between 4 classes remain (SI Fig. S.4). CIFAR-4 has multiple samples per class: $\boldsymbol{\Sigma}^{xx}$ is no longer diagonal in the basis given by $\boldsymbol{V}$ (Eq. 3 does not hold), which influences the dynamics through a coupling of the modes. Moreover, we can hold out a separate test set to track the test loss for CIFAR-4. We train the CNN with learning rate λ, and the FCNN with a learning rate $\lambda_{FC} = n\lambda$. The results are shown in Fig. 5; experimental details see SI.

We can conclude that the general insights we derived before are still valid. First of all, in both linear CNNs and linear FCNNs the modes of $\boldsymbol{\Sigma}^{yx}$ are discovered with rapid, successive transitions (Fig. 5 (a) and (b)). However, the CNN exhibits a different effective learning rate for each mode (see Sec. 3). Since the first mode of the geometric shape dataset is essentially the average over the shapes (see Fig. 2), and an average corresponds to the zero-frequency, the CNN does not show an additional delay for this mode ($d_0 \approx 1$). All other mode discoveries have an additional delay with respect to the trajectories for the FCNN. Fig. 5 (c) and (d) show the evolution of the loss. The loss for CIFAR-4 does not go to zero, meaning there are samples that cannot be classified by a linear CNN or FCNN. The FCNN has discovered all the modes after around 18000 samples, and at that point reaches a train and test loss of around 0.21. After that, it starts to overfit. The CNN needs a lot more samples to discover all the modes, but doesn't start to overfit. The latter might
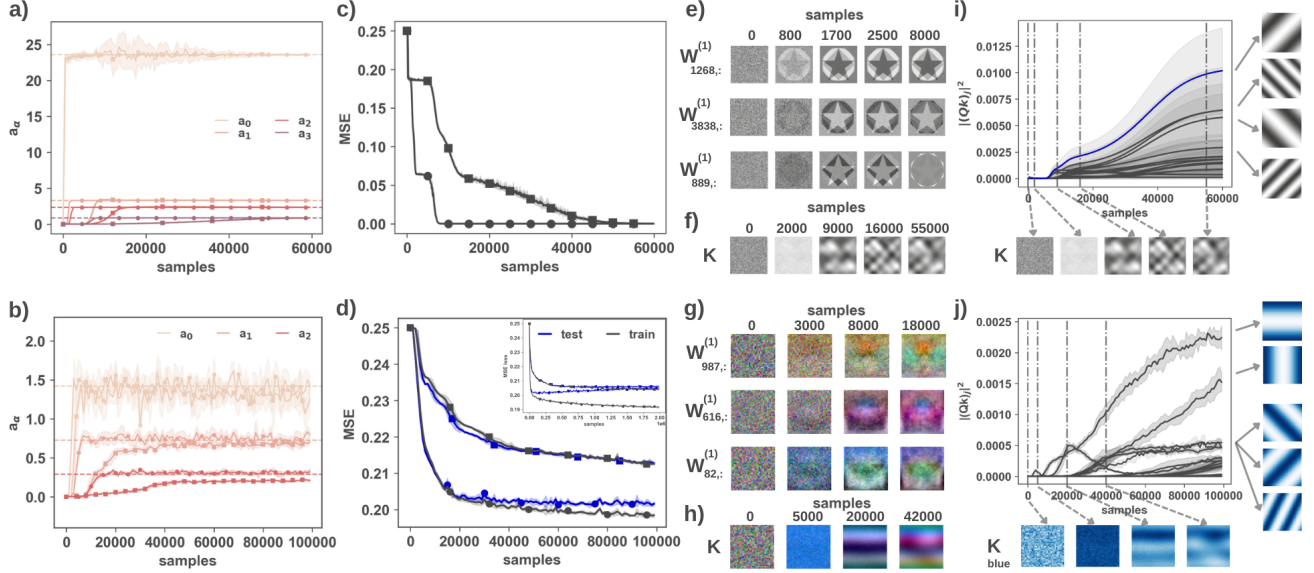
7

Figure 5. Top row: geometric shapes. Bottom row: CIFAR-4. $\boldsymbol{a}, \boldsymbol{b}$) Evolution of the effective singular values for the CNN (square markers) and the FCNN (round markers). $\boldsymbol{c}, \boldsymbol{d}$): MSE loss for the CNN (square markers) and the FCNN (round markers). For CIFAR-4, the test loss is plotted in blue. The inset in d) shows the loss up to 10^6 timesteps. $\boldsymbol{e}, \boldsymbol{g}$) Evolution of the weights connected to 3 randomly selected nodes in the hidden layer of the FCNN, reshaped as $n \times n$ matrices. Shown timepoints (in number of samples) roughly correspond to the moments each mode is fully discovered (compare to subfigures a) and b)). $\boldsymbol{f}, \boldsymbol{h}$) Evolution of the kernel at analogous timepoints for the CNN. $\boldsymbol{i}, \boldsymbol{j}$) Evolution of the coefficients $|(\boldsymbol{Qk})_j|^2$ (middle), frequencies corresponding to highest $|(\boldsymbol{Qk})_j|^2$ of final kernel (right). Only the kernel of the blue channel is shown for CIFAR-4. $\boldsymbol{a} - \boldsymbol{f}$) Solid lines: averages over experimental runs. Shaded regions: standard deviations. Vertical dashed lines: selected timepoints. Horizontal dashed lines: singular values s_α. The kernels and partial weight matrices shown, are randomly selected across experiment runs.

be due to the implicit regularisation given by the dominant frequency bias: Fig. 5 (e) and (g) show that the rows of the first weight matrix in FCNN are mixtures of the discovered singular vectors, as given by Eq. 18. Fig. 5 (f) and (h), on the other hand, show that, at any point during training, the kernel is a sparse mixture of the dominant frequencies of the singular vectors discovered so far. Fig. 5 (i) and (j) show the evolution of the kernel Fourier coefficients $|(\boldsymbol{Qk})_j|$ in detail. In the plot for the geometric shapes dataset, we have highlighted a trajectory in blue. This cascaded trajectory is an example of how a frequency initially becomes dominant through a high value $|\boldsymbol{Q}\phi^\alpha|_j$ during the discovery of mode α, and subsequently can remain dominant due to its higher value at the start of the discovery of a subsequent mode. That the kernel frequencies are really the dominant frequencies is further illustrated in the SI.

6. Experiments with Deep, Non-Linear CNNs

We now consider deep linear and non-linear CNNs trained on CIFAR-10; both have the same overall architecture, but the non-linear network has ReLU activation functions and additional max-pooling layers. The overall network architecture consists of four convolutional layers, each with 16

channels; a flatten layer, and two fully-connected layers. The kernel size is 8×8, while the image size is 32×32: this means these models not only exhibit weight sharing but they also exhibit locality. The additional max-pooling layers will make the non-linear network (partially) translation invariant. The updates now take place in batches, we use multiple channels per layer, and we use zero-padding for the convolutions (details see SI). In fig. 6, we show the results of the experiments. Fig. 6 (a) shows the evolution of the loss for the linear and non-linear model, averaged over trials. Fig. 6 (b) and (c) show the corresponding evolutions of the effective singular values for the linear and non-linear CNN, respectively. Firstly, from fig. 6 (b) and (c) we can see that both networks discover the statistical dataset structure with ordered, stage-like transitions. However, the evolution of the effective singular values for the non-linear network are delayed with respect to the linear network. Secondly, the non-linear CNN exhibits a lower loss, and given that both types of networks eventually exhibit the same, saturated effective singular values, this lower loss cannot be explained from the discovery of the statistical structure of the dataset alone. We hypothesize that while the non-linear model discovers the statistical structure, it at the same time discovers different aspects of the dataset structure as well. These as-
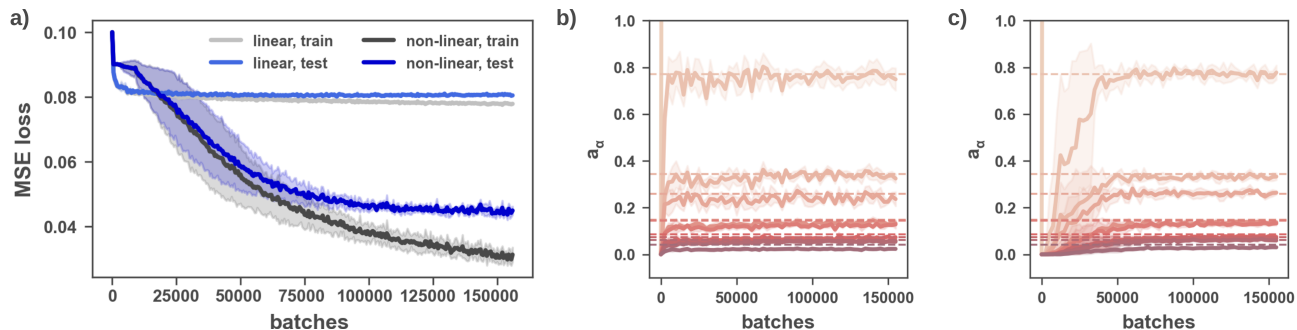
Figure 6. Results of experiments with deep, linear and non-linear CNNs trained on CIFAR-10. a) Evolution of the train and test MSE loss. b) Evolution of the effective singular values for the linear CNN. c) Evolution of the effective singular values for the non-linear CNN. a-c) Solid lines: average over 5 trials from random initial conditions; shaded regions: standard deviations. Horizontal dashed lines: singular values s_α. In fig. b) and c) we limit the y-axis to show more detail of the smaller effective singular values starting from a_1.

pects are likely not captured by an average over samples (Eq. 4), which could explain why they are not captured by the effective singular values a_α (Eq. 5) alone.

7. Discussion

The above results shed new light on the use of convolutions in neural networks. Our theoretical analysis uses a linear CNN and a number of assumptions (listed in Sec. 1). We first discussed how the relevant structure of the dataset is captured by the singular vectors of the input-output correlation matrix, which generally represent broader to finer distinctions between classes. We then showed how the dynamics of learning are influenced by the interplay between the singular vectors associated to the dataset structure and the singular vectors associated to the convolutional network structure. We subsequently showed that linear CNNs discover the statistical dataset structure with rapid, stage-like transitions, and they do so at an effective learning rate that depends on this interplay. Moreover, this interplay yields a dominant frequency bias: internally, only the most dominant frequencies of each singular vector of the dataset are used by the CNN to discern between classes. It is thus both the statistical structure of the dataset, embodied by the corresponding singular vectors, as the convolutional structure of the network—for which the singular vectors are frequencies—that influence the learning. Our experiments show that these conclusions broadly hold when using more general datasets. We moreover show that deep, non-linear CNNs also discover the statistical structure of the dataset in an ordered, stage-like fashion (see also Hacohen & Weinshall 2022). However, they at the same time seem to discover aspects of the dataset structure that are not captured by the statistical structure.

In our results, the statistical structure of the dataset is discovered through *time*. Deeper, non-linear CNNs are known

to process higher-order visual relationships in later layers as well: i.e., there is an additional notion of hierarchy through depth (Krizhevsky et al., 2012; Simonyan et al., 2014; Olah et al., 2017). An extension of our theory could clarify how this hierarchy over depth develops over time, and how this relates to the dominant frequency bias and results on the relationship between learning in linear and non-linear CNNs (Kalimeris et al., 2019; Refinetti et al., 2022). The dominant frequency bias we find is a form of implicit regularization that could help explain why large CNNs sometimes generalize well, even without additional regularization. It is seemingly at odds with the 'spectral bias', which states that lower frequencies are learned first (Xu, 2018; Rahaman et al., 2019; Xu et al., 2019; Basri et al., 2019; 2020; Cao et al., 2021). We claim on the other hand that the bias is towards frequencies that are dominant in the Fourier spectrum of the singular vectors, irrespective of whether these are actually high or low frequencies. One thing to note is that broader distinctions in shape correspond to lower frequencies. We find that broader distinctions are uncovered first, and that therefore lower frequencies are learned first. How the two concepts exactly relate is the topic of future work. Finally, our results could also help explain why CNNs are sensitive to frequency domain perturbations (Jo & Bengio, 2017; Tsuzuku & Sato, 2019), and why they often rely on textures (images with a sparse frequency spectrum) instead of shapes (Baker et al., 2018; Geirhos et al., 2019; Brendel & Bethge, 2019).

References

Advani, M. S., Saxe, A. M., and Sompolinsky, H. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pp. 322–332. PMLR, 2019.

Atanasov, A., Bordelon, B., and Pehlevan, C. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022.

Bach, F. Learning theory from first principles, 2023. URL https://www.di.ens.fr/~fbach/ltfp_book.pdf.

Baker, N., Lu, H., Erlikhman, G., and Kellman, P. J. Deep convolutional networks do not classify based on global object shape. *PLoS computational biology*, 14(12): e1006613, 2018.

Basri, R., Jacobs, D., Kasten, Y., and Kritchman, S. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32, 2019.

Basri, R., Galun, M., Geifman, A., Jacobs, D., Kasten, Y., and Kritchman, S. Frequency bias in neural networks for input of non-uniform density. In *International Conference on Machine Learning*, pp. 685–694. PMLR, 2020.

Bourlard, H. and Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

Braun, L., Dominé, C. C. J., Fitzgerald, J. E., and Saxe, A. M. Exact learning dynamics of deep linear networks with prior knowledge. In *Advances in Neural Information Processing Systems*, 2022.

Brendel, W. and Bethge, M. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Cao, Y., Fang, Z., Wu, Y., Zhou, D. X., and Gu, Q. Towards understanding the spectral bias of deep learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2205–2211, 2021.

Casper, S., Hod, S., Filan, D., Wild, C., Critch, A., and Russell, S. Graphical clusterability and local specialization in deep neural networks. In *ICLR 2022 Workshop on PAIR2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pp. 1675–1685. PMLR, 2019.

Fang, Y., Cohen, M. A., and Kincaid, T. G. Dynamics of a winner-take-all neural network. *Neural Networks*, 9(7): 1141–1154, 1996.

Fukai, T. and Tanaka, S. A simple neural network exhibiting selective activation of neuronal ensembles: from winner-take-all to winners-share-all. *Neural computation*, 9(1): 77–97, 1997.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Gidel, G., Bach, F., and Lacoste-Julien, S. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems 32*, pp. 3196–3206, 2019.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Gray, R. M. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006. ISSN 1567-2190.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Gunning, R. and Rossi, H. *Analytic Functions of Several Complex Variables*. Prentice-Hall, 1965.

Hacohen, G. and Weinshall, D. Principal components bias in over-parameterized linear models, and its manifestation in deep neural networks. *Journal of Machine Learning Research*, 23(155):1–46, 2022.

Jain, A. K. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

Jo, J. and Bengio, Y. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.

Jolliffe, I. *Principal Component Analysis*. Springer New York, 2002.

Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., and Zhang, H. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Lazzaro, J., Ryckebusch, S., Mahowald, M. A., and Mead, C. A. Winner-take-all networks of o (n) complexity. *Advances in neural information processing systems*, 1, 1988.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2(11):e7, 2017.

Pearson, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572, 1901.

Pretorius, A., Kroon, S., and Kamper, H. Learning dynamics of linear denoising autoencoders. In *International Conference on Machine Learning*, pp. 4141–4150. PMLR, 2018.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Refinetti, M. and Goldt, S. The dynamics of representation learning in shallow, non-linear autoencoders. In *International Conference on Machine Learning*, pp. 18499–18519. PMLR, 2022.

Refinetti, M., Ingrosso, A., and Goldt, S. Neural networks trained with sgd learn distributions of increasing complexity. *arXiv preprint arXiv:2211.11567*, 2022.

Satpathi, S. and Srikant, R. The dynamics of gradient descent for overparametrized neural networks. In *Learning for Dynamics and Control*, pp. 373–384. PMLR, 2021.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *2nd International Conference on Learning Representations*, 2014.

Saxe, A. M., McClelland, J. L., and Ganguli, S. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

Sedghi, H., Gupta, V., and Long, P. M. The singular values of convolutional layers. In *7th International Conference on Learning Representations*, 2019.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *2nd International Conference on Learning Representations, ICLR 2014 Workshop Track Proceedings*, 2014.

Tsuzuku, Y. and Sato, I. On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 51–60, 2019.

Voss, C., Goh, G., Cammarata, N., Petrov, M., Schubert, L., and Olah, C. Branch specialization. *Distill*, 6(4): e00024–008, 2021.

Xu, Z. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.

Xu, Z., Zhang, Y., and Xiao, Y. Training behavior of deep neural network in frequency domain. In *International Conference on Neural Information Processing*, pp. 264–274. Springer, 2019.

A. Diagonality of Σ^{xx} in the Basis Given by V and Relationship to Class Coherence

We here discuss the diagonality of Σ^{xx} in the basis given by V, and its relationship to class coherence. We start from the assumption that the label vectors are orthonormal, $y_{c_p}^T y_{c_q} = \delta_{c_p c_q}$, where c_p and c_q are class indices. This is the case when the classes are one-hot encoded. Then, a property of the singular value decomposition of a general, real matrix B is that the right singular vectors of this matrix are the eigenvectors of the matrix $B^T B$. We can make use of the orthonormality of the label vectors and general properties of the kronecker product (denoted $\otimes$) to derive:

$$\Sigma^{yx T} \Sigma^{yx} = \langle y \otimes x^T \rangle^T \langle y \otimes x^T \rangle \tag{S.1}$$

$$= \langle x \otimes y^T \rangle \langle y \otimes x^T \rangle \tag{S.2}$$

$$= \frac{C^2}{N^2} \sum_{c_r=0}^{m-1} \sum_{c_q=0}^{m-1} (\langle x \rangle_{c_r} \otimes y_{c_r}^T)(y_{c_q} \otimes \langle x^T \rangle_{c_q}) \tag{S.3}$$

$$= \frac{C^2}{N^2} \sum_{c_r=0}^{m-1} \sum_{c_q=0}^{m-1} y_{c_r}^T y_{c_q} (\langle x \rangle_{c_r} \otimes \langle x^T \rangle_{c_q}) \tag{S.4}$$

$$= \frac{C^2}{N^2} \sum_{c_r=0}^{m-1} \langle x \rangle_{c_r} \otimes \langle x^T \rangle_{c_r} \tag{S.5}$$

where $\langle \rangle_{c_r}$ denotes the average over all samples belonging to the same class c_r, p is the number of classes, N is the total number of samples, and C is the number of samples per class (we assume a balanced dataset). If we define $\Phi^{xx} = \frac{C^2}{N^2} \sum_{c_r=0}^{m-1} \langle x \rangle_{c_r} \otimes \langle x \rangle_{c_r}$, we thus have that $V^T \Phi^{xx} V$ is diagonal: the right singular vectors of Σ^{yx} are the eigenvectors of $\Phi^{xx} = \Sigma^{yx T} \Sigma^{yx}$. The corresponding eigenvalues are equal to the square of the singular values, thus the diagonal values of $S^T S$.

$\Sigma^{xx} = \langle x \otimes x^T \rangle$ and Φ^{xx} (eq. S.5) are different in general, but they are close to equal if samples within a class are very similar (at least in pixel-space):

$$x^{(i)} = \langle x \rangle_{c_r} + \epsilon^{(i)}, \quad \| \epsilon^{(i)} \| << \| \langle x \rangle_{c_r} \| \tag{S.6}$$

where $x^{(i)}$ is the sample with index i belonging to a class with index c_r. We have:

$$\Sigma^{xx} = \langle x \otimes x^T \rangle \tag{S.7}$$

$$= \langle (\langle x \rangle_{c_r} + \epsilon^{(i)}) \otimes (\langle x \rangle_{c_r} + \epsilon^{(i)})^T \rangle \tag{S.8}$$

$$= \langle (\langle x \rangle_{c_r} \otimes \langle x \rangle_{c_r}^T) + (\langle x \rangle_{c_r} \otimes \epsilon^{(i)}) + ... \rangle \tag{S.9}$$

Thus, if $x^{(i)} \approx \langle x \rangle_{c_r}$ (eq. S.6), then:

$$\Sigma^{xx} \approx \langle (\langle x \rangle_{c_r} \otimes \langle x \rangle_{c_r}^T) \rangle \tag{S.10}$$

$$= \frac{C}{N} \sum_{c_p=0}^{m-1} \langle x \rangle_{c_r} \otimes \langle x \rangle_{c_r}^T \tag{S.11}$$

$$= \frac{N}{C} \Phi^{xx} \tag{S.12}$$

$$= p \Phi^{xx}. \tag{S.13}$$

In other words, if samples within a class are very similar (=high class coherence), we can use the same eigen-/right singular vectors to decompose Σ^{xx} and Σ^{yx} into a sum of their respective modes. Thus the higher the class coherence, the more the two matrices represent the same inherent structure. If Σ^{xx} is exactly equal to $p \Phi^{xx}$, the p eigenvalues of Σ^{xx} are given by the p non-zero diagonal elements of $p S^T S$.

B. Definition of Q

We first recall the definition of the *one*-dimensional discrete Fourier transform (DFT) matrix, denoted by the $n \times n$ matrix $\boldsymbol{F}^{(n)}$:

$$(\boldsymbol{F}^{(n)})_{p,q} = (\omega_n)^{pq}, \ \omega_n = \exp(-2\pi i/n), \tag{S.14}$$

where the superscript pq denotes the product of indices p and q applied as an exponent, and $i^2 = 1$. Then the $n^2 \times n^2$ complex-valued matrix $\boldsymbol{Q}$ is given by:

$$\boldsymbol{Q} = \frac{1}{n}\boldsymbol{F}^{(n)} \otimes \boldsymbol{F}^{(n)} \tag{S.15}$$

with $\otimes$ the kronecker product. $\boldsymbol{Q}$ is unitary ($\boldsymbol{Q}^{*T} = \boldsymbol{Q}^{-1}$) and symmetric.

C. Doubly Block Circulant Matrices

C.1. Doubly Block Circulant Matrix Definition

$circ(\boldsymbol{b})$: circulant matrix of the vector $\boldsymbol{b}$ with length q:

$$circ([b_0, b_1, \cdots, b_{q-2}, b_{q-1}]^T) = \begin{bmatrix} b_0 & b_{q-1} & \cdots & b_2 & b_1 \\ b_1 & b_0 & b_{q-1} & \cdots & b_2 \\ b_2 & b_1 & b_0 & b_{q-1} & \cdots \\ \vdots & & & \ddots & \\ b_{q-1} & & \cdots & & b_0 \end{bmatrix} \tag{S.16}$$

then $dbc(\boldsymbol{B})$, the doubly block circulant matrix of the matrix $\boldsymbol{B}$, is given by (Sedghi et al., 2019):

$$dbc(\boldsymbol{B}) = \begin{bmatrix} circ(B_{0,:}) & circ(B_{u-1,:}) & \cdots & circ(B_{1,:}) \\ circ(B_{1,:}) & circ(B_{0,:}) & \cdots & circ(B_{2,:}) \\ \vdots & & \ddots & \\ circ(B_{u-1,:}) & & \cdots & circ(B_{0,:}) \end{bmatrix} \tag{S.17}$$

where u is the number of rows of $\boldsymbol{B}$, and $\boldsymbol{B}_{i,:}$ is the ith row. Note that this is the definition for an actual convolution, see below.

C.2. Convolution and correlation: notes on flipping the kernel or the image

Given a kernel $\boldsymbol{K}$ and an image $\boldsymbol{X}$, the convolution operation is defined as:

$$(\boldsymbol{X} * \boldsymbol{K})_{i,j} = \sum_m \sum_l \boldsymbol{X}_{m,l} K_{i-m,j-l} \tag{S.18}$$

or, equivalently (since convolution is commutative):

$$(\boldsymbol{K} * \boldsymbol{X})_{i,j} = \sum_m \sum_l \boldsymbol{X}_{i-m,j-l} K_{m,l}. \tag{S.19}$$

This implies the kernel (or the image) is applied in a 'flipped' manner, i.e., when we increase m and l, we decrease the corresponding indices. This amounts to flipping the kernel (or image) over the x- and y-axis, or equivalently, rotating it 180 degrees, before applying the convolution.
In practice, however, convolutional layers are usually implemented as correlations (here denoted $\star$):

$$(\boldsymbol{X} \star \boldsymbol{K})_{i,j} = \sum_m \sum_l \boldsymbol{X}_{m,l} K_{i+m,j+l} \tag{S.20}$$

thus without flipping the kernel beforehand. This makes the operation more intuitive (but it's not commutative). Whether the kernel is flipped beforehand or not, the learning algorithm will learn the appropriate values of the kernel in the appropriate

place: the result of learning with correlations instead of convolutions is just a flipped kernel (Goodfellow et al., 2016). Most literature therefore uses the term "convolution" in lieu of "correlation". The actual operation used influences the definition of the doubly block circulant matrices and the gradient descent equations, however, and so it's important to keep track of the details when comparing different sources.

As it turns out, we can define a very similar circulant and doubly block circulant matrix to replace a (circular) correlation (see the definitions used in (Sedghi et al., 2019)):

$$
circ_{corr}([b_0, b_1, \cdots, b_{q-2}, b_{q-1}]^T) = \begin{bmatrix} b_0 & b_1 & \cdots & b_{q-2} & b_{q-1} \\ b_{q-1} & b_0 & b_1 & \cdots & b_{q-2} \\ b_{q-2} & b_{q-1} & b_0 & b_1 & \cdots \\ \vdots & & & \ddots & \\ b_1 & & \cdots & & b_0 \end{bmatrix} \tag{S.21}
$$

and

$$
dbc_{corr}(\boldsymbol{B}) = \begin{bmatrix} circ(B_{0,:}) & circ(B_{1,:}) & \ldots & circ(B_{u-1,:}) \\ circ(B_{u-1,:}) & circ(B_{0,:}) & \ldots & circ(B_{u-2,:}) \\ \vdots & & \ddots & \\ circ(B_{1,:}) & & \ldots & circ(B_{0,:}) \\ . & & & \end{bmatrix} \tag{S.22}
$$

When we compare those definitions to the earlier defined matrices for circular convolutions, we see that the difference lies in a permutation of the indices equal to a reversion and a shift, e.g., for a (column) vector $\boldsymbol{b}$:

$$
rev(\boldsymbol{b}^T) = [b_{q-1}, b_{q-2}, \cdots, b_0] \tag{S.23}
$$

and

$$
shift_{\to 1}(rev(\boldsymbol{b}^T)) = [b_0, b_{q-1}, \cdots, b_1], \tag{S.24}
$$

yielding

$$
circ_{corr}(\boldsymbol{b}) = circ(shift_{\to 1}(rev(\boldsymbol{b}))), \tag{S.25}
$$

and a similar permutation of the rows for the doubly block circulant matrices. Due to the properties of Fourier transforms, we can also flip the kernel through applying the Fourier transform twice:

$$
\boldsymbol{Q}\boldsymbol{Q}\boldsymbol{k} = shift_{\to 1}(rev(\boldsymbol{k}^T)) \tag{S.26}
$$

This allows us to translate all the derivations from implementations with convolutions to implementations with correlations: where

$$
\boldsymbol{dbc}(\boldsymbol{K}) = n\,\boldsymbol{Q}^{-1} diag(\boldsymbol{Q}\boldsymbol{k})\boldsymbol{Q} = n\,\boldsymbol{Q} diag(\boldsymbol{Q}^{-1}\boldsymbol{k})\boldsymbol{Q}^{-1} \tag{S.27}
$$

(since the dbc matrix is real valued, thus $\boldsymbol{dbc}(\boldsymbol{K}) = \boldsymbol{dbc}(\boldsymbol{K})^*$ and $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^*$), we can use eq. S.26 to write:

$$
\boldsymbol{dbc}_{corr}(\boldsymbol{K}) = \boldsymbol{dbc}(\boldsymbol{K}_{flip}) = n\,\boldsymbol{Q} diag(\boldsymbol{Q}^{-1}(\boldsymbol{Q}\boldsymbol{Q}\boldsymbol{k}))\boldsymbol{Q}^{-1} \tag{S.28}
$$

thus

$$
\boldsymbol{dbc}_{corr}(\boldsymbol{K}) = \boldsymbol{dbc}(\boldsymbol{K}_{flip}) = n\,\boldsymbol{Q} diag(\boldsymbol{Q}\boldsymbol{k})\boldsymbol{Q}^{-1}. \tag{S.29}
$$

D. Derivation of the Differential Equations of Gradient Descent

D.1. Definition of the network and learning algorithm

As described in the main paper, we consider a convolutional network where the predictions $\hat{\boldsymbol{y}}$ are given by:

$$
\hat{\boldsymbol{y}} = \boldsymbol{W}\,\boldsymbol{dbc}(\boldsymbol{k})\boldsymbol{x} \tag{S.30}
$$

here $\boldsymbol{x} = vec(\boldsymbol{X})$ is the vectorized input image, $\boldsymbol{k} = vec(\boldsymbol{K})$ is the vectorized kernel, and $\boldsymbol{W}$ is the weight matrix of the fully connected layer.

The loss is given by the MSE loss, i.e.,

$$L = \frac{1}{2} \sum_{l=0}^{p} (y_l - \hat{y}_l)^2. \tag{S.31}$$

The kernel and the weights are updated according to the gradients of this loss:

$$\Delta \boldsymbol{k} = -\lambda \frac{\partial L}{\partial \boldsymbol{k}^T} \tag{S.32}$$

$$\Delta \boldsymbol{W} = -\lambda \frac{\partial L}{\partial \boldsymbol{W}^T} \tag{S.33}$$

where λ is the learning rate.

D.2. Gradient descent for the convolutional layer

For the convolutional layer, the gradient of the loss with respect to the kernel is in itself given by a convolution, but with a flipped image (or, equivalently, an image rotated 180 degrees). We can write this equation using doubly block circulant matrices:

$$\frac{\partial L}{\partial \boldsymbol{k}^T} = vec\left(\boldsymbol{X}_{flip} \circledast \frac{\partial L}{\partial \boldsymbol{H}^T}\right) \tag{S.34}$$

$$= dbc(\boldsymbol{X}_{flip}) \frac{\partial L}{\partial \boldsymbol{h}^T}, \tag{S.35}$$

with

$$\frac{\partial L}{\partial \boldsymbol{h}^T} = -\sum_{l=0}^{p} (y_l - \hat{y}_l)(\boldsymbol{W}^T)_{:,l}. \tag{S.36}$$

Here $\boldsymbol{H}$ denotes the activity of the hidden layer, and $\boldsymbol{h} = vec(\boldsymbol{H})$. $(\boldsymbol{W}^T)_{:,l}$ denotes column l of $\boldsymbol{W}^T$.

We will now rewrite this equation in a form that will turn out to be useful in the next steps:

$$\frac{\partial L}{\partial \boldsymbol{k}^T} = n \sum_{l=0}^{p} \boldsymbol{Q} diag(\boldsymbol{Qx}) \boldsymbol{Q}^{-1} (y_l - \hat{y}_l)(\boldsymbol{W}^T)_{:,l} \tag{S.37}$$

$$\iff \left(\frac{1}{n}\boldsymbol{Q}^{-1}\frac{\partial L}{\partial \boldsymbol{k}^T}\right)_j = \sum_{l=0}^{p} diag(\boldsymbol{Qx})_{j,:} (y_l - \hat{y}_l)(\boldsymbol{Q}^{-1}\boldsymbol{W}^T)_{:,l} \tag{S.38}$$

$$\iff \left(\frac{1}{n}\boldsymbol{Q}^{-1}\frac{\partial L}{\partial \boldsymbol{k}^T}\right)_j = \sum_{l=0}^{p} (\boldsymbol{Qx})_j (y_l - \hat{y}_l)(\boldsymbol{Q}^{-1}\boldsymbol{W}^T)_{j,l} \tag{S.39}$$

$$\iff \left(\frac{1}{n}\boldsymbol{Q}^{-1}\frac{\partial L}{\partial \boldsymbol{k}^T}\right)_j = \sum_{l=0}^{p} \left((\boldsymbol{y} - \hat{\boldsymbol{y}}) \otimes (\boldsymbol{Qx})^T\right)_{l,j} (\boldsymbol{Q}^{-1}\boldsymbol{W}^T)_{j,l} \tag{S.40}$$

$$\iff \frac{1}{n}\left(\boldsymbol{Q}^{-1}\frac{\partial L}{\partial \boldsymbol{k}^T}\right)_j = (\boldsymbol{Q}^{-1}\boldsymbol{W}^T)_{j,:}\left((\boldsymbol{y} - \hat{\boldsymbol{y}}) \otimes \boldsymbol{x}^T\right)\boldsymbol{Q}_{:,j}^T \tag{S.41}$$

D.3. Gradient descent for the fully connected layer

For the fully connected layer with weights $\boldsymbol{W}$, we can write down the gradient in the conventional form:

$$\left(\frac{\partial L}{\partial \boldsymbol{W}^T}\right)_{l,:} = (y_l - \hat{y}_l)\boldsymbol{x}^T dbc(\boldsymbol{k})^T \tag{S.42}$$

15

and, for future use, rewrite this as:

$$\left(\frac{\partial L}{\partial \boldsymbol{W}^T}\right)_{l,:} = (y_l - \hat{y}_l)\boldsymbol{x}^T n \boldsymbol{Q}^T diag(\boldsymbol{Q}\boldsymbol{k})^T \boldsymbol{Q}^{-1T} \tag{S.43}$$

$$\Longleftrightarrow \frac{1}{n}\left(\frac{\partial L}{\partial \boldsymbol{W}^T}\right)_{l,:} = \left((\boldsymbol{y} - \hat{\boldsymbol{y}}) \otimes \boldsymbol{x}^T\right)_{l,:} \boldsymbol{Q} diag(\boldsymbol{Q}\boldsymbol{k})\boldsymbol{Q}^{-1} \tag{S.44}$$

where we used the fact $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{-1T}$ and $\boldsymbol{Q} = \boldsymbol{Q}^T$.

D.4. From discrete gradient descent to differential equations

The equations S.41 and S.44, together with the update rules given by eq. S.32 and S.33 , tell us how to update the kernel and the weight matrix at each discrete time step Δt. We will proceed to make a continuous approximation of these discrete updates. This is a valid approximation as long as the learning rate is small enough. In this slow learning regime, we can assume the weights only change minimally over a number of updates with different samples. The factors $\left((\boldsymbol{y} - \hat{\boldsymbol{y}}) \otimes \boldsymbol{x}^T\right)$ can then be replaced by their average over those samples (see (Saxe et al., 2019)). In the main body, we defined the matrices $\boldsymbol{\Sigma}^{\boldsymbol{xx}} = \langle \boldsymbol{x} \otimes \boldsymbol{x}^T \rangle$ and $\boldsymbol{\Sigma}^{\boldsymbol{yx}} = \langle \boldsymbol{y} \otimes \boldsymbol{x}^T \rangle$, which can be computed from the dataset alone. We also defined $\boldsymbol{\Sigma}^{\hat{\boldsymbol{y}}\boldsymbol{x}} = \langle \hat{\boldsymbol{y}} \otimes \boldsymbol{x}^T \rangle$. These definition allow us to rewrite the difference eqs. S.41, S.44, S.32 and S.33, as:

$$\frac{1}{n\lambda}\left(\frac{d \, diag(\boldsymbol{Q}^{-1}\boldsymbol{k})}{dt}\right)_{j,j} = \boldsymbol{Q}^{-1}_{j,:}\boldsymbol{W}^T(\boldsymbol{\Sigma}^{\boldsymbol{yx}} - \boldsymbol{\Sigma}^{\hat{\boldsymbol{y}}\boldsymbol{x}})\boldsymbol{Q}_{:,j} \tag{S.45}$$

$$\frac{1}{n\lambda}\left(\frac{d\boldsymbol{W}}{dt}\right) = (\boldsymbol{\Sigma}^{\boldsymbol{yx}} - \boldsymbol{\Sigma}^{\hat{\boldsymbol{y}}\boldsymbol{x}})\boldsymbol{Q}diag(\boldsymbol{Q}\boldsymbol{k})\boldsymbol{Q}^{-1} \tag{S.46}$$

where $diag(\boldsymbol{Q}^{-1}\boldsymbol{k})$ is the $n^2 \times n^2$ diagonal matrix with the elements of $\boldsymbol{Q}^{-1}\boldsymbol{k}$ placed on the diagonal. The derivatives of the off-diagonal elements are zero ($\left(\frac{d \, diag(\boldsymbol{Q}^{-1}\boldsymbol{k})}{dt}\right)_{i,j} = 0$ if $i \neq j$).

D.5. Interpreting the gradient descent equations in terms of the dataset structure

We can now finally link the process of gradient descent with the structure present in the dataset. We can describe both the input-input as the input-output structure of the dataset with singular value decompositions. We can use these singular value decompositions to perform a change of variables. Starting from:

$$\boldsymbol{\Sigma}^{\boldsymbol{yx}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{S.47}$$

$$\boldsymbol{\Sigma}^{\hat{\boldsymbol{y}}\boldsymbol{x}} = \boldsymbol{U}\boldsymbol{A}\boldsymbol{V}^T \tag{S.48}$$

$$\boldsymbol{\Sigma}^{\boldsymbol{xx}} = \boldsymbol{V}\overline{\boldsymbol{\Sigma}^{\boldsymbol{xx}}}\boldsymbol{V}^T \tag{S.49}$$

we transform $\boldsymbol{W}$ to $\overline{\boldsymbol{W}}$ and $\boldsymbol{dbc}(\boldsymbol{K})$ to $\overline{\boldsymbol{dbc}(\boldsymbol{K})}$:

$$\boldsymbol{W} = \boldsymbol{U}\overline{\boldsymbol{W}}\boldsymbol{R}^{-1} \tag{S.50}$$

$$\Longleftrightarrow \overline{\boldsymbol{W}} = \boldsymbol{U}^T\boldsymbol{W}\boldsymbol{R} \tag{S.51}$$

where $\boldsymbol{R}$ is an arbitrary invertible matrix, and

$$\boldsymbol{dbc}(\boldsymbol{K}) = n\boldsymbol{Q}^{-1}diag(\boldsymbol{Q}\boldsymbol{k})\boldsymbol{Q} = \boldsymbol{R}\overline{\boldsymbol{dbc}(\boldsymbol{K})}\boldsymbol{V}^T \tag{S.52}$$

$$\Longleftrightarrow \overline{\boldsymbol{dbc}(\boldsymbol{K})} = \boldsymbol{R}^{-1}\boldsymbol{dbc}(\boldsymbol{K})\boldsymbol{V} = n\boldsymbol{R}^{-1}\boldsymbol{Q}^{-1}diag(\boldsymbol{Q}\boldsymbol{k})\boldsymbol{Q}\boldsymbol{V} \tag{S.53}$$

$$\Longleftrightarrow \overline{\boldsymbol{dbc}(\boldsymbol{K})} = n\boldsymbol{R}^{-1}\boldsymbol{Q}diag(\boldsymbol{Q}^{-1}\boldsymbol{k})\boldsymbol{Q}^{-1}\boldsymbol{V} \tag{S.54}$$

where in the last step we used that $\boldsymbol{dbc}(\boldsymbol{K})$ is a real valued matrix, thus $\boldsymbol{dbc}(\boldsymbol{K}) = \boldsymbol{dbc}(\boldsymbol{K})^*$, and that $\boldsymbol{Q}$ and $\boldsymbol{Q}^{-1}(= \boldsymbol{Q}^{*T})$ are unitary and symmetric.

We perform this change of variables such that we can describe the network, given by the product $\boldsymbol{W}\boldsymbol{dbc}(\boldsymbol{K})$, in terms of the singular vectors of the dataset given by $\boldsymbol{U}$ and $\boldsymbol{V}$. With this change of variables, we indeed have (compare to eq.S.47) :

$$\boldsymbol{W}\boldsymbol{dbc}(\boldsymbol{K}) = \boldsymbol{U}\overline{\boldsymbol{W}}\boldsymbol{R}^{-1}\boldsymbol{R}\overline{\boldsymbol{dbc}(\boldsymbol{K})}\boldsymbol{V}^T = \boldsymbol{U}\overline{\boldsymbol{W}}\,\overline{\boldsymbol{dbc}(\boldsymbol{K})}\boldsymbol{V}^T. \tag{S.55}$$

The matrices $\overline{W}$ and $\overline{dbc(K)}$ are thus defined up to an invertible matrix; but their product remains the same. If the $p \times n^2$ product $\overline{W}\,\overline{dbc(K)}$ would be rectangular diagonal with p real, positive entries, we would have a singular value decomposition. Moreover, this singular value decomposition would be the same as that of the input-output relationship in the dataset (eq.S.47), only with different singular values.

We can also use this change of variables to describe the complete input-predicted output relation $\Sigma^{\hat{y}x}$ (eq. 4):

$$\Sigma^{\hat{y}x} = W\,dbc(K)\Sigma^{xx} \tag{S.56}$$

$$= U\overline{W}\,\overline{dbc(K)}\overline{\Sigma^{xx}}V^T \tag{S.57}$$

$$= UAV^T \tag{S.58}$$

with A:

$$A = \overline{W}\,\overline{dbc(K)}\overline{\Sigma^{xx}} = n\overline{W}R^{-1}Q\,diag(Q^{-1}k)Q^{-1}V\overline{\Sigma^{xx}} \tag{S.59}$$

where again, if A would be rectangular diagonal with p real, positive entries, this would be an SVD with the same singular vectors as the input-(ground truth) output relationship of the dataset Σ^{yx}. In that case, we will call the matrix A the matrix of effective singular values. Note that in practice, we can compute A irrespective of the internal details of the network: we only need predictions $\hat{y}$ for samples x to compute $A = U^T\Sigma^{\hat{y}x}V$. The evolution of A during training thus essentially coincides with the evolution of the predictions of the network.

We are now ready to rewrite our system of differential equations of gradient descent using the new variables:

$$\frac{1}{n\lambda}\Big(\frac{d\,diag(Q^{-1}k)}{dt}\Big)_{j,j} = Q_{j,:}^{-1}W^T\big(\Sigma^{yx} - \Sigma^{\hat{y}x}\big)Q_{:,j}^T \tag{S.60}$$

$$\iff \frac{1}{n\lambda}\Big(\frac{d\,diag(Q^{-1}k)}{dt}\Big)_{j,j} = Q_{j,:}^{-1}R^{-*T}\overline{W}^{*T}U^T U\big(S - A\big)V^T Q_{:,j}^T \tag{S.61}$$

$$\frac{1}{n\lambda}U^T\Big(\frac{dW}{dt}\Big)R = U^T\big(\Sigma^{yx} - \Sigma^{\hat{y}x}\big)Q^T diag(Qk)Q^{-1}R \tag{S.62}$$

$$\iff \frac{1}{n\lambda}U^T\Big(\frac{dW}{dt}\Big)R = U^T U\big(S - A\big)V^T Q^T diag(Qk)Q^{-1}R \tag{S.63}$$

We thus arrive at the coupled system of equations:

$$\frac{1}{n\lambda}\Big(\frac{d\,diag(Q^{-1}k)}{dt}\Big)_{j,j} = Q_{j,:}^{-1}R^{-*T}\overline{W}^{*T}\big(S - A\big)(QV)_{:,j}^T \tag{S.64}$$

$$\frac{1}{n\lambda}\Big(\frac{d\overline{W}}{dt}\Big) = \big(S - A\big)(QV)^T diag(Qk)Q^{-1}R \tag{S.65}$$

Which are the equations mentioned in the main paper, Eqs. 15 and 16.

E. Sums of Cosines Datasets

In the theoretical derivations, we will often use a 'sums of cosines dataset'. This dataset consists of a sum of pure 2D cosines, where the frequencies of those cosines are not shared between classes:

$$X_{l,m}^{(c)} = \sum_{j\in\sigma^{(c)}} b_j \cos(2\pi\frac{\mu l}{n} + 2\pi\frac{\nu m}{n} + \delta_j), \tag{S.66}$$

where $\sigma^{(c)}$ is a set, belonging to class c, of vec-2D indices j that map to pairs of horizontal (μ) and vertical (ν) frequency indices. b_j is the corresponding amplitude, and δ_j is the corresponding phase. We here thus consider the case where the sets $\sigma^{(c)}$ are disjoint. In the special case where frequencies are not shared (between modes, see next) we can derive analytical

results.

In this case the right singular vectors $\phi^{(\alpha)}$ are given by:

$$\phi_i^{(\alpha)} = \frac{1}{\sqrt{\sum_{j \in \sigma^{(\alpha)}} b_j^2}} \frac{n}{\sqrt{2}} \sum_{j \in \sigma^{(\alpha)}} b_j \cos(2\pi\frac{\mu l}{n} + 2\pi\frac{\nu m}{n} + (\delta_\phi)_j), \tag{S.67}$$

i.e., again the class vectors normalised and sorted according to singular value: the mode indices α form a permutation of the class indices c based on the singular values. Here $\mu = div(j, n)$, $\nu = mod(j, n)$ are the frequency indices and $l = div(i, n)$, $m = mod(i, n)$ are the pixel indices. There is a subtlety regarding the phase $(\delta_\phi)_j$: the SVD is defined up to a minus sign, i.e., if both the left and right singular vector obtain a minus sign, the total mode remains unaltered. Therefore the phase $(\delta_\phi)_j$ could be the same as the original phase $(\delta_\phi)_j = \delta_j$, or a possible minus sign could be absorbed in the phase as $(\delta_\phi)_j = \delta_j + \pi$. We can furthermore note that the sets σ^α, where each set corresponds to a set $\sigma^{(c)}$, are also disjoint. Formally, this means that given an index j, $|(\boldsymbol{Q}\phi^\alpha)_j|$ is non-zero for only one mode α. The latter property will be crucial in the derivations below. We will also still assume that $\boldsymbol{\Sigma^{xx}} = \boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{V^T}$ with $\overline{\boldsymbol{\Sigma^{xx}}}$ diagonal.

F. Minimal Norm Weights

It is well-known that, when gradient descent is used to train a linear neural network from near-zero initial conditions in the over-parametrized regime, the final network parameters will have minimal norm (for a derivation with MSE loss, see e.g., (Bach, 2023), chapter 11). I.e., among all the possible combinations of network weights that implement the desired input-output map, gradient descent leads to the network weights that have the smallest norm. This is called the implicit regularization of gradient descent. The minimal norm solution for the two-layer linear CNN with a single kernel is given by the following constrained minimization problem:

$$\min_{\boldsymbol{W},\boldsymbol{dbc(K)}} \left(||\boldsymbol{W}||_F^2 + ||\boldsymbol{dbc(K)}||_F^2 \right) \tag{S.68}$$

subject to:

$$\boldsymbol{W}\,\boldsymbol{dbc(K)}\boldsymbol{\Sigma^{xx}} = \boldsymbol{USV^T} \tag{S.69}$$

where $||\boldsymbol{B}||_F$ denotes the Frobenius norm of the $l \times m$ matrix $\boldsymbol{B}$, in essence $||\boldsymbol{B}||_F = \sqrt{\sum_i^l \sum_j^m |B_{ij}|}$.

F.1. Exact Solutions for Sum of Cosines Datasets

We here derive what those final solutions look for a sums of cosines dataset. We first define three new $n^2 \times 1$ vectors, $\boldsymbol{\delta_\phi}$, and $\boldsymbol{\delta_k}$ and $\boldsymbol{\delta_w}$. These vectors contain the phases (or angles) associated to the complex-valued vec-2D DFT of the singular values, of the kernel and of the weights, respectively. The definition of $\boldsymbol{\delta_k}$ is straightforward:

$$\boldsymbol{\delta_k} = angle(\boldsymbol{Qk}). \tag{S.70}$$

The definition of $\boldsymbol{\delta_\phi}$ and $\boldsymbol{\delta_w}$ is more subtle. The $p \times n^2$ matrix $\boldsymbol{W}$ has p rows, and there are also p singular vectors ϕ^α ($\alpha \in \{0, \cdots, p-1\}$) of dimension $n^2 \times 1$. For every index j, there are thus p complex numbers, e.g., $(\boldsymbol{Q}\phi^0)_j$, $(\boldsymbol{Q}\phi^1)_j$, etc. However, when we use a dataset with disjoint sets of frequencies, we have by design a non-zero value $(\boldsymbol{Q}\phi^\alpha)_j$ for only one single mode index α per frequency index j. We select the phase of this value to be the jth element of the vector $\boldsymbol{\delta_\phi}$. We thus have:

$$\begin{cases} (\boldsymbol{\delta_\phi})_j = angle(\boldsymbol{Q}\phi^\alpha)_j, & j \to \alpha, \\ (\boldsymbol{\delta_\phi})_j = 0, & \text{freq. } j \text{ not present in singular vectors.} \end{cases} \tag{S.71}$$

where again $j \to \alpha$ denotes that the frequency j is used in the sum of cosines making up the singular vector ϕ^α. We define $\boldsymbol{\delta_w}$ in a similar way:

$$\begin{cases} (\boldsymbol{\delta_w})_j = angle((\boldsymbol{QW^T})_{j,\alpha}), & j \to \alpha, \\ (\boldsymbol{\delta_w})_j = 0, & \text{freq. } j \text{ not present in singular vectors.} \end{cases} \tag{S.72}$$

Note that because the kernel, the weights and the singular vectors are all real-valued, we have $(\boldsymbol{\delta}_\phi)_{j_{symm}} = -(\boldsymbol{\delta}_\phi)_j$, $(\boldsymbol{\delta}_k)_{j_{symm}} = -(\boldsymbol{\delta}_k)_j$, and $(\boldsymbol{\delta}_w)_{j_{symm}} = -(\boldsymbol{\delta}_w)_j$.

We are now ready to state what the final, minimal norm solutions look like. A kernel $\boldsymbol{K}$ and the weight matrix $\boldsymbol{W}$ correspond to a minimal norm solution, if and only if:

$$S_{\alpha,\alpha} = n \sum_{j \in \sigma^\alpha_{symm}} |(\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_j| \, |(\boldsymbol{Q}\boldsymbol{k})_j|^2 \, \overline{\Sigma}_{\alpha,\alpha}, \tag{S.73}$$

with σ^α_{symm} is the original set of indices j in σ^α together with all their corresponding symmetric indices j_{symm}, and

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Omega}\boldsymbol{\Theta}_w\boldsymbol{Q}, \tag{S.74}$$

where $\boldsymbol{\Omega}$ is a $p \times n^2$ matrix defined by:

$$\begin{cases} \boldsymbol{\Omega}_{\alpha,j} = |(\boldsymbol{Q}\boldsymbol{k})_j|, \; \boldsymbol{\Omega}_{\alpha,j_{symm}} = |(\boldsymbol{Q}\boldsymbol{k})_j| & j \to \alpha, \\ \boldsymbol{\Omega}_{\alpha,j} = 0 & else, \end{cases} \tag{S.75}$$

and

$$\boldsymbol{\Theta}_w = diag(e^{i\boldsymbol{\delta}_w}). \tag{S.76}$$

With, for all $j \in \sigma^\alpha_{symm}$:

$$(\boldsymbol{\delta}_k)_j + (\boldsymbol{\delta}_w)_j = -(\boldsymbol{\delta}_\phi)_j. \tag{S.77}$$

From these minimal norm solutions, we can then also conclude (cfr. Eq. 12):

$$\boldsymbol{R} = \boldsymbol{Q}^{-1}, \tag{S.78}$$

$$\overline{\boldsymbol{W}} = \boldsymbol{\Omega}\boldsymbol{\Theta}_{\boldsymbol{w}}. \tag{S.79}$$

Note that the only constraint on the Fourier spectrum of the kernel is given by Eq. S.73; the kernel is not uniquely defined. We show in the main paper that the final coefficients $|(\boldsymbol{Q}k)_j|$ are influenced or determined through a winner-takes-all process that takes place during training with gradient descent. This effect can only be derived from the gradient equations themselves, however.

Proof overview:
We start from $\boldsymbol{dbc}(\boldsymbol{K}) = n\boldsymbol{Q}^{-1}diag(\boldsymbol{Q}k)\boldsymbol{Q}$, with $\boldsymbol{k} = vec(\boldsymbol{K})$ an arbitrary, real-valued kernel, and from $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{P}$ where $\boldsymbol{P}$ is an arbitrary $n \times n$ real-valued matrix, such that $\boldsymbol{W}$ is an arbitrary real-valued weight matrix. We then use the method of Lagrange multipliers on the constrained optimization problem given by Eq. S.68 and Eq. S.69 to find a system of equations involving $\boldsymbol{k}$, $\boldsymbol{P}$, and the given matrices $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{S}$, $\boldsymbol{Q}$ and $\boldsymbol{\Sigma}^{xx}$. This system of equations captures the relationships between these matrices when $\boldsymbol{dbc}(\boldsymbol{K})$ and $\boldsymbol{W}$ form a minimal norm solution. When frequencies are not shared between modes, i.e., when for every frequency index j only one value $\boldsymbol{Q}_{j,:}\boldsymbol{V}_{:,\alpha}$ is significant, the solutions to these equations are as described above.

F.2. Detailed Proof

Let $\boldsymbol{dbc}(\boldsymbol{K}) = n\boldsymbol{Q}^{-1}diag(\boldsymbol{Q}k)\boldsymbol{Q}$, with $\boldsymbol{k} = vec(\boldsymbol{K})$ an arbitrary, real-valued kernel, and let $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{P}$ where $\boldsymbol{P}$ is an arbitrary $n \times n$ real-valued matrix, such that $\boldsymbol{W}$ is an arbitrary real-valued weight matrix. We first reformulate the constrained minimization problem Eq. S.68 using properties of the Frobenius norm. The Frobenius norm can be expressed in function of singular values: $||B||_F = \sqrt{\sum_i^{min(l,m)} \sigma_i(B)^2}$ where $\sigma_i(B)$ are the singular values of B. We can make use of the latter property of the Frobenius norm to explicitly include the constraint on the network structure, given by the eigendecomposition $\boldsymbol{dbc}(\boldsymbol{K}) = n\boldsymbol{Q}^{-1}diag(\boldsymbol{Q}k)\boldsymbol{Q}$. The singular values of $\boldsymbol{dbc}(\boldsymbol{K})$ are the square roots of the eigenvalues of $\boldsymbol{dbc}(\boldsymbol{K})^T\boldsymbol{dbc}(\boldsymbol{K}) = n^2\boldsymbol{Q}^{-1}diag(\boldsymbol{Q}^*k)diag(\boldsymbol{Q}k)\boldsymbol{Q}$. The singular values of $\boldsymbol{dbc}(\boldsymbol{K})$ are thus given by $n|(\boldsymbol{Q}k)_j|$. The Frobenius norm can also be expressed as a trace, $||B||_F = \sqrt{tr(B^{*T}B)}$. Therefore $||\boldsymbol{W}||_F^2$ can be expressed as

$tr\left(\boldsymbol{P}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{P}\right) = tr\left(\boldsymbol{P}^T\boldsymbol{P}\right)$, given that $\boldsymbol{P}$ is real and $\boldsymbol{U}$ is real and orthogonal.

We thus reformulate the constrained minimization problem (Eq. S.68 and S.69) as:

$$\min_{\boldsymbol{P},\boldsymbol{Qk}}\left(tr\left(\boldsymbol{P}^{*T}\boldsymbol{P}\right) + \sum_i^{n^2}|(\boldsymbol{Qk})_i|^2\right), \tag{S.80}$$

subject to

$$n\boldsymbol{P}\boldsymbol{Q}^{-1}diag(\boldsymbol{Qk})\boldsymbol{Q}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}} = \boldsymbol{S}. \tag{S.81}$$

We can make use of Lagrange multipliers to solve the constrained optimization problem given by Eqs. S.80 and S.81. We denote the multipliers by the $p \times n^2$ matrix $\boldsymbol{\Lambda}$, and the Lagrangian $\mathcal{L}$ is then given by:

$$\mathcal{L} = tr\left(\boldsymbol{P}^T\boldsymbol{P}\right) + \sum_j^{n^2}|(\boldsymbol{Qk})_j|^2 + tr\left(\boldsymbol{\Lambda}^T\left(n\boldsymbol{P}\boldsymbol{Q}^{-1}diag(\boldsymbol{Qk})\boldsymbol{Q}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}} - \boldsymbol{S}\right)\right) \tag{S.82}$$

To find the minimal norm solution, we compute the derivatives with respect to the matrix $\boldsymbol{P}$, the values $(Qk)_j$ and the matrix $\boldsymbol{\Lambda}$. These derivatives are zero at the minimum.

$\frac{d\mathcal{L}}{d\boldsymbol{P}}$:

To compute the derivative with respect to the matrix $\boldsymbol{P}$, we make use of the cyclic property of the trace function, and the fact that $\frac{d\,tr(\boldsymbol{BC})}{d\boldsymbol{B}} = \boldsymbol{C}^T$. We obtain:

$$\frac{d\mathcal{L}}{d\boldsymbol{P}} = \frac{d}{d\boldsymbol{P}}\left(tr\left(\boldsymbol{P}\boldsymbol{P}^T\right) + tr\left(\boldsymbol{P}\boldsymbol{Q}^{-1}diag(\boldsymbol{Qk})\boldsymbol{Q}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T\right)\right) \tag{S.83}$$

$$= \boldsymbol{P} + \boldsymbol{\Lambda}\overline{\boldsymbol{\Sigma^{xx}}}^T\boldsymbol{V}^T\boldsymbol{Q}^Tdiag(\boldsymbol{Qk})\boldsymbol{Q}^{-T} \tag{S.84}$$

$\frac{d\mathcal{L}}{d(\boldsymbol{Qk})_j}$:

To calculate this derivative, we will make use of the following properties:

$$\left(diag(\boldsymbol{Qk})\boldsymbol{Q}\right)_{l,m} = \sum_i diag(\boldsymbol{Qk})_{l,i}\boldsymbol{Q}_{i,m} = diag(\boldsymbol{Qk})_{l,l}\boldsymbol{Q}_{l,m} \tag{S.85}$$

yielding

$$(\boldsymbol{Q}^{-1}diag(\boldsymbol{Qk})\boldsymbol{Q})_{i,m} = \sum_l(\boldsymbol{Q}^{-1}_{i,l}\left(diag(\boldsymbol{Qk})\boldsymbol{Q}\right)_{l,m} = \sum_l\boldsymbol{Q}^{-1}_{i,l}diag(\boldsymbol{Qk})_{l,l}\boldsymbol{Q}_{l,m} \tag{S.86}$$

and, for any matrix $\boldsymbol{B}$ and $\boldsymbol{C}$,

$$tr(\boldsymbol{BC}) = \sum_i\boldsymbol{B}_{i,:}\boldsymbol{C}_{:,i} \tag{S.87}$$

since the trace operation is the sum of diagonal elements. Moreover, we can compute the derivative of a function f with respect to a complex number $z = z_{\mathbb{R}} + iz_{\mathbb{I}}$ as $\frac{df}{dz} = \frac{1}{2}(\frac{df}{dz_{\mathbb{R}}} - i\frac{df}{dz_{\mathbb{I}}})$ (see, e.g., (Gunning & Rossi, 1965)). The derivative of

the third term is then given by:

$$\frac{d}{d\,diag(\boldsymbol{Qk})_{j,j}}\Big(tr\big(\boldsymbol{PQ}^{-1}diag(\boldsymbol{Qk})\boldsymbol{QV}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T\big)\Big) \tag{S.88}$$

$$=\frac{d}{d\,diag(\boldsymbol{Qk})_{j,j}}\Big(tr\big(\boldsymbol{P}\sum_l\boldsymbol{Q}_{:,l}^{-1}diag(\boldsymbol{Qk})_{l,l}\boldsymbol{Q}_{l,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T\big)\Big) \tag{S.89}$$

$$=\frac{d}{d\,diag(\boldsymbol{Qk})_{j,j}}\Big(\sum_m\boldsymbol{P}_{m,:}\sum_l\boldsymbol{Q}_{:,l}^{-1}diag(\boldsymbol{Qk})_{l,l}\boldsymbol{Q}_{l,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}_{:,m}^T\Big) \tag{S.90}$$

$$=\frac{1}{2}\big(\sum_m\boldsymbol{P}_{m,:}\boldsymbol{Q}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\boldsymbol{\Lambda}_{:,m}^T-i^2\sum_m\boldsymbol{P}_{m,:}\boldsymbol{Q}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\boldsymbol{\Lambda}_{:,m}^T\big) \tag{S.91}$$

$$=\sum_m\boldsymbol{P}_{m,:}\boldsymbol{Q}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}_{:,m}^T \tag{S.92}$$

$$=tr(\boldsymbol{PQ}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T) \tag{S.93}$$

This yields, for the total derivative $\frac{d\mathcal{L}}{d(\boldsymbol{Qk})_j}$:

$$\frac{d\mathcal{L}}{d(\boldsymbol{Qk})_j}=\frac{1}{2}\big(2(\boldsymbol{Qk})_{\mathbb{R}j}-2i(\boldsymbol{Qk})_{\mathbb{I}j}\big)+tr(\boldsymbol{PQ}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T) \tag{S.94}$$

$$=(\boldsymbol{Qk})_j^*+tr(\boldsymbol{PQ}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T) \tag{S.95}$$

$\frac{d\mathcal{L}}{d\boldsymbol{\Lambda}}$:

Finally, we compute $\frac{d\mathcal{L}}{d\boldsymbol{\Lambda}}$:

$$\frac{d\mathcal{L}}{d\boldsymbol{\Lambda}}=\frac{1}{2}\big(\frac{d\mathcal{L}}{d\boldsymbol{\Lambda}_{\mathbb{R}}}-i\frac{d\mathcal{L}}{d\boldsymbol{\Lambda}_{\mathbb{I}}}\big) \tag{S.96}$$

$$=\frac{1}{2}\Big(\big(n\boldsymbol{PQ}^{-1}diag(\boldsymbol{Qk})\boldsymbol{QV}\overline{\boldsymbol{\Sigma^{xx}}}-\boldsymbol{S}\big)-i^2\big(n\boldsymbol{PQ}^{-1}diag(\boldsymbol{Qk})\boldsymbol{QV}\overline{\boldsymbol{\Sigma^{xx}}}-\boldsymbol{S}\big)\Big) \tag{S.97}$$

$$=n\boldsymbol{PQ}^{-1}diag(\boldsymbol{Qk})\boldsymbol{QV}\overline{\boldsymbol{\Sigma^{xx}}}-\boldsymbol{S} \tag{S.98}$$

The complete system of equations is then given by:

$$\begin{cases}\frac{d\mathcal{L}}{d\boldsymbol{P}}=0\\\frac{d\mathcal{L}}{d(\boldsymbol{Qk})_j}=0\\\frac{d\mathcal{L}}{d\boldsymbol{\Lambda}}=0\end{cases}\iff\begin{cases}\boldsymbol{P}+\boldsymbol{\Lambda}\overline{\boldsymbol{\Sigma^{xx}}}^T\boldsymbol{V}^T\boldsymbol{Q}^Tdiag(\boldsymbol{Qk})\boldsymbol{Q}^{-T}=0\\(\boldsymbol{Qk})_j^*+tr(\boldsymbol{PQ}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T)=0\\n\boldsymbol{PQ}^{-1}diag(\boldsymbol{Qk})\boldsymbol{QV}\overline{\boldsymbol{\Sigma^{xx}}}-\boldsymbol{S}=0\end{cases} \tag{S.99}$$

From the first equation of eqs. S.99 we obtain:

$$\boldsymbol{P}=-\boldsymbol{\Lambda}\overline{\boldsymbol{\Sigma^{xx}}}^T\boldsymbol{V}^T\boldsymbol{Q}^Tdiag(\boldsymbol{Qk})\boldsymbol{Q}^{-T} \tag{S.100}$$

$$\iff\boldsymbol{P}=-\boldsymbol{\Lambda}\overline{\boldsymbol{\Sigma^{xx}}}^T\boldsymbol{V}^T\boldsymbol{Q}^{-T}diag(\boldsymbol{Qk})^*\boldsymbol{Q}^T \tag{S.101}$$

where we used that $\boldsymbol{dbc}(\boldsymbol{K})$ is real-valued. We can plug this in the second equation of the system of eqs. S.99 :

$$(\boldsymbol{Qk})_j^*+tr(\boldsymbol{PQ}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T)=0 \tag{S.102}$$

$$\iff(\boldsymbol{Qk})_j^*+tr(-\boldsymbol{\Lambda}\overline{\boldsymbol{\Sigma^{xx}}}^T\boldsymbol{V}^T\boldsymbol{Q}^{-T}diag(\boldsymbol{Qk})^*\boldsymbol{Q}^T\boldsymbol{Q}_{:,j}^{-1}\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T)=0 \tag{S.103}$$

$$\iff(\boldsymbol{Qk})_j^*+tr(-\boldsymbol{\Lambda}\overline{\boldsymbol{\Sigma^{xx}}}^T\boldsymbol{V}^T\boldsymbol{Q}_{:,j}^{-T}diag(\boldsymbol{Qk})_{j,j}^*\boldsymbol{Q}_{j,:}\boldsymbol{V}\overline{\boldsymbol{\Sigma^{xx}}}\boldsymbol{\Lambda}^T)=0 \tag{S.104}$$

If the frequency indexed by j is not present in any mode, i.e., $(QV)_{j,\alpha} = 0$ for all mode indices α, then we obtain:

$$(Qk)_j^* = (Qk)_j = 0 \tag{S.105}$$

If the frequency indexed by j is present in mode α, then $(QV)_{j,\alpha} \neq 0$ for a single mode α (by assumption). We then obtain:

$$(Qk)_j^* + tr(-\Lambda \overline{\Sigma^{xx}}_{:,\alpha}^T (V^T Q^{-T})_{\alpha,j} diag(Qk)_{j,j}^* (QV)_{j,\alpha} \overline{\Sigma^{xx}}_{\alpha,:} \Lambda^T) = 0 \tag{S.106}$$

and because $\overline{\Sigma^{xx}}$ is diagonal (by assumption):

$$(Qk)_j^* + tr(-\Lambda_{:,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha}^T (V^T Q^{-T})_{\alpha,j} diag(Qk)_{j,j}^* (QV)_{j,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha} \Lambda_{\alpha,:}^T) = 0 \tag{S.107}$$

$$\iff (Qk)_j^* + tr(-\Lambda_{:,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha}^T (QV)_{\alpha,j}^{*T} diag(Qk)_{j,j}^* (QV)_{j,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha} \Lambda_{\alpha,:}^T) = 0 \tag{S.108}$$

$$\iff (Qk)_j^* + tr(-\Lambda_{:,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha}^T diag(Qk)_{j,j}^* |(QV)_{j,\alpha}|^2 \overline{\Sigma^{xx}}_{\alpha,\alpha} \Lambda_{\alpha,:}^T) = 0 \tag{S.109}$$

$$\iff (Qk)_j^* - (Qk)_j^* \overline{\Sigma^{xx}}_{\alpha,\alpha}^2 |(QV)_{j,\alpha}|^2 tr(\Lambda_{:,\alpha} \Lambda_{\alpha,:}^T) = 0 \tag{S.110}$$

$$\iff \Lambda_{\alpha,:}^T \Lambda_{:,\alpha} = \overline{\Sigma^{xx}}_{\alpha,\alpha}^{-2} |(QV)_{j,\alpha}|^{-2} \tag{S.111}$$

This yields:

$$\Lambda_{:,\alpha} = \tilde{R}_{:,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha}^{-1} |(QV)_{j,\alpha}|^{-1} \tag{S.112}$$

with $\tilde{R}$ a $p \times p$ orthogonal matrix.

We now propose to decompose P as $P = \Omega \Theta_w Q$. We can use the obtained equations to determine the shape of Ω and Θ_w corresponding to a minimal norm solution. From the first equation:

$$P = -\Lambda \overline{\Sigma^{xx}}^T V^T Q^{-T} diag(Qk)^* Q^T \tag{S.113}$$

$$\iff \Omega \Theta_w = -\Lambda \overline{\Sigma^{xx}}^T V^T Q^{-T} diag(Qk)^* \tag{S.114}$$

$$\iff (\Omega \Theta_w)_{:,j} = -\Lambda_{:,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha}^T (QV)_{\alpha,j}^{*T} diag(Qk)_{j,j}^* \tag{S.115}$$

$$\iff (\Omega \Theta_w)_{:,j} = -\tilde{R}_{:,\alpha} \overline{\Sigma^{xx}}_{\alpha,\alpha}^{-1} |(QV)_{j,\alpha}|^{-1} \overline{\Sigma^{xx}}_{\alpha,\alpha}^T (QV)_{\alpha,j}^{*T} diag(Qk)_{j,j}^* \tag{S.116}$$

$$\iff (\Omega \Theta_w)_{:,j} = -\tilde{R}_{:,\alpha} |(QV)_{j,\alpha}|^{-1} e^{-i(\delta_\phi)_j} |(QV)_{\alpha,j}| e^{-i(\delta_k)_j} |(Qk)_j| \tag{S.117}$$

$$\iff \Omega_{:,j} (\Theta_w)_{j,j} = -\tilde{R}_{:,\alpha} e^{-i(\delta_\phi)_j} e^{-i(\delta_k)_j} |(Qk)_j| \tag{S.118}$$

Thus we obtain, for the real-valued matrix Ω:

$$\Omega_{:,j} = -\tilde{R}_{:,\alpha} |(Qk)_j| \tag{S.119}$$

and for the diagonal matrix with associated phases:

$$(\Theta_w)_{j,j} = e^{i(\delta_w)_j} = e^{i\left(-(\delta_\phi)_j - (\delta_k)_j\right)}, \tag{S.120}$$

or also

$$(\delta_w)_j = -(\delta_\phi)_j - (\delta_k)_j. \tag{S.121}$$

Finally, we can use the third equation of the system of eqs. S.99, involving the $p \times n^2$ rectangular diagonal matrix S with

real, positive values on the diagonal of $\boldsymbol{S}_{0:p,0:p}$, and zero elsewhere. Starting with the elements of $\boldsymbol{S}_{0:p,0:p}$:

$$\boldsymbol{S}_{\beta,\alpha} = n\boldsymbol{\Omega}_{\beta,:}\boldsymbol{\Theta}_w diag(\boldsymbol{Qk})\boldsymbol{QV}\overline{\boldsymbol{\Sigma^{xx}}}_{:,\alpha} \tag{S.122}$$

$$= \sum_j n\boldsymbol{\Omega}_{\beta,j}\boldsymbol{\Theta}_{wj,j} diag(\boldsymbol{Qk})_{j,j}(\boldsymbol{QV})_{j,\alpha}\overline{\boldsymbol{\Sigma^{xx}}}_{\alpha,\alpha} \tag{S.123}$$

$$= \sum_j n\boldsymbol{\Omega}_{\beta,j}\boldsymbol{\Theta}_{wj,j}\boldsymbol{\Theta}_{kj,j}\boldsymbol{\Theta}_{\phi j,j}|diag(\boldsymbol{Qk})_{j,j}||(\boldsymbol{QV})_{j,\alpha}|\overline{\boldsymbol{\Sigma^{xx}}}_{\alpha,\alpha} \tag{S.124}$$

$$= \sum_j n\boldsymbol{\Omega}_{\beta,j}|diag(\boldsymbol{Qk})_{j,j}||(\boldsymbol{QV})_{j,\alpha}|\overline{\boldsymbol{\Sigma^{xx}}}_{\alpha,\alpha} \tag{S.125}$$

$$= \sum_j -n\tilde{\boldsymbol{R}}_{\beta,\alpha}|(\boldsymbol{Qk})_j|^2|(\boldsymbol{QV})_{j,\alpha}|\overline{\boldsymbol{\Sigma^{xx}}}_{\alpha,\alpha} \tag{S.126}$$

If $\beta \neq \alpha$, then $\boldsymbol{S}_{\beta,\alpha} = 0$. Therefore $\tilde{\boldsymbol{R}}_{\beta,\alpha} = 0$ if $\beta \neq \alpha$. Given that $\tilde{\boldsymbol{R}}$ is orthogonal, and $\boldsymbol{S}_{\alpha,\alpha}$ is positive and real, we have $\tilde{\boldsymbol{R}}_{\alpha,\alpha} = -1$. The other elements of $\boldsymbol{S}$, $\boldsymbol{S}_{\alpha,i}$ with $i = p$, are indeed zero, since $\overline{\boldsymbol{\Sigma^{xx}}}_{\alpha,i} = 0$. This completes the proof.

For completeness, we also show that $\boldsymbol{P}$ is a real-valued matrix:

$$\boldsymbol{P}_{l,m} = (\boldsymbol{\Omega}\boldsymbol{\Theta}_w\boldsymbol{Q})_{l,m} \tag{S.127}$$

$$= \sum_j \boldsymbol{\Omega}_{l,j} diag(e^{i\boldsymbol{\delta}_w})_{j,j}\boldsymbol{Q}_{j,m} \tag{S.128}$$

$$= \sum_{j\in\sigma^\alpha} \left(\boldsymbol{\Omega}_{l,j} diag(e^{i\boldsymbol{\delta}_w})_{j,j}\boldsymbol{Q}_{j,m} + \boldsymbol{\Omega}_{l,j_{symm}} diag(e^{i\boldsymbol{\delta}_w})_{j_{symm},j_{symm}}\boldsymbol{Q}_{j_{symm},m}\right) \tag{S.129}$$

Given the vec-2D indices and their corresponding pair of horizontal and vertical indices, $j \to (\mu,\nu)$, $m \to (a,b)$, we find:

$$n\boldsymbol{Q}_{j,m} = (\boldsymbol{F} \otimes \boldsymbol{F})_{(\mu-1)n+\nu,(a-1)n+b} \tag{S.130}$$

$$= F_{\mu,a}F_{\nu,b} \tag{S.131}$$

$$= e^{-\frac{2\pi i}{n}(\mu a + \nu b)} \tag{S.132}$$

and for $j_{symm} \to (n - \mu, n - \nu)$:

$$n\boldsymbol{Q}_{j_{symm},m} = F_{n-\mu,a}F_{n-\nu,b} \tag{S.133}$$

$$= e^{-\frac{2\pi i}{n}((n-\mu)a+(n-\nu)b)} \tag{S.134}$$

$$= e^{-2\pi i\frac{n}{n}a}e^{-2\pi i\frac{n}{n}b}e^{\frac{+2\pi i}{n}(\mu a+\nu b)} \tag{S.135}$$

$$= e^{+\frac{2\pi i}{n}(\mu a+\nu b)} \tag{S.136}$$

$$= n\boldsymbol{Q}^*_{j,m} \tag{S.137}$$

The vector $\boldsymbol{\delta}_w$ has the property that $(\boldsymbol{\delta}_w)_j = -(\boldsymbol{\delta}_w)_{j_{symm}}$. This yields $diag(e^{i\boldsymbol{\delta}_w})_{j_{symm},j_{symm}} = diag(e^{i\boldsymbol{\delta}_w})^*_{j,j}$. Since $\boldsymbol{\Omega}$ is real-valued, each term in the summation in Eq.S.129 is real, and therefore $\boldsymbol{P}$ is real.

G. Silent Alignment, Balanced Weights and WTA Dynamics

In the previous section, we derived the shape of the final weights and kernel for datasets consisting of sums of cosines. These were the minimal norm solutions—the kind of network parameters the linear CNN converges to when starting from small, random initial conditions. In the next section, we derive analytic solutions to the evolution of the predictions of the network, for specific datasets and assuming *aligned* and *balanced* initial conditions. With *aligned* initial conditions, we mean initial conditions that render the matrix $\boldsymbol{A}$ rectangular diagonal, with positive real, entries from the start. The alignment can be

interpreted as follows: if A is such a rectangular diagonal matrix, the network parameters are aligned with the statistical dataset structure, in the sense that $\Sigma^{\hat{y}x}$ and Σ^{yx} have the same singular vectors. With *balanced* we mean that the (Fourier transformed) weight matrix consists of (Fourier transformed) elements of the kernel; implying some balancedness between the two types of layers. We borrowed this term from the theory of fully connected networks (Saxe et al., 2019), but the shapes of the matrix involved for CNN are more complicated, which makes the concept more subtle.

The assumption of aligned and balanced initial conditions turns out to be a good approximation to small, random initial conditions and a sums of cosines dataset: when starting from small, random initial conditions, we see that the network parameters quickly become aligned and balanced. This also happens in a stage-like fashion, and the alignment associated to a mode occurs before that mode is learned and the loss appreciably decreases. Therefore the dynamics of learning can be approximately described by assuming aligned and balanced conditions from the start. This phenomenon has been rigorously analysed for the case of fully connected networks (Atanasov et al., 2022; Braun et al., 2022). It has also been shown that the error of this approximation decreases with initialization scale, e.g., it decreases as the standard deviation σ used in a gaussian initializer decreases.
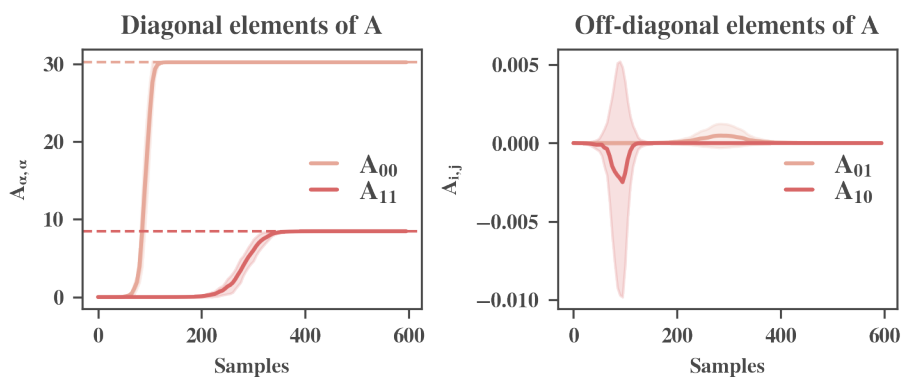

Figure S.1. Evolution of the matrix $A_{0:2,0:2}$ during training: diagonal elements (left) and off-diagonal elements (right). Solid lines: average over trials. Shaded regions: standard deviation over trials. Horizontal, dashed lines: singular values s_α.
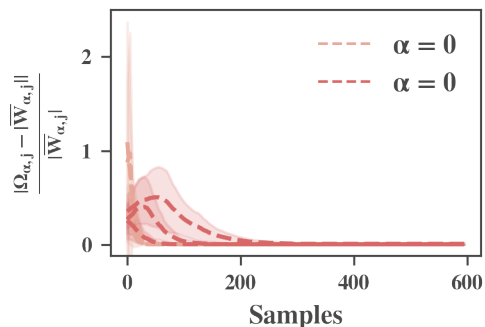

Figure S.2. Plot of the evolution of the different fractions $\frac{||\overline{W}_{\alpha,j}|-\Omega_{\alpha,j}|}{|\overline{W}_{\alpha,j}|}$, a measure of how quickly the network parameters become balanced.

In Fig. S.1, we have plotted, for the sums of cosines dataset, both the evolution of the two diagonal and the two off-diagonal elements of $A_{0:2,0:2}$, averaged over 10 trials using small, random, initial conditions ($\sigma = 0.00001$). As discussed in the main paper, the diagonal values of A follow sigmoidal, stage-like trajectories: after some time, they suddenly appear to grow very fast, eventually saturating on their corresponding singular value. I.e., $A_{0,0}$ saturates at the value s_0 and $A_{1,1}$ saturates at the value s_1. The off-diagonal elements, on the other hand, remain relatively small (compare the y-axis between the left and right plot) and exhibit transient peaks, before becoming ≈ 0.

Fig.S.2 shows how quickly the magnitudes of the elements $\overline{W}_{\alpha,j}$ become equal to $\Omega_{\alpha,j} = |(Qk)_j|$, i.e., how quickly the

solution becomes balanced. We plotted the evolution of the fraction $\frac{||\overline{W}_{\alpha,j}|-\Omega_{\alpha,j}|}{|\overline{W}_{\alpha,j}|}$ during training, for all the different frequencies (indexed by j) involved in the two modes $\alpha = 0$ and $\alpha = 1$ of the sums of cosines dataset. $\overline{W}$ is computed from W using $R = Q^{-1}$. We see that the balancedness also happens in a stage-like or cascaded fashion: very quickly, the elements corresponding to the first mode become balanced; in a next phase and after the first mode has been learned, the elements corresponding to the second mode also become balanced.

For the sums of cosines dataset, we thus quickly arrive at the following shape for the fully connected weight matrix W:

$$W(t) = U\Omega(t)\Theta_w(t)Q \tag{S.138}$$

where $\Omega(t)$ is a $p \times n^2$ matrix defined by:

$$\begin{cases} \Omega_{\alpha,j} = |(Qk(t))_j|, \ \Omega_{\alpha,j_{symm}} = |(Qk(t))_j| & j \to \alpha \\ \Omega_{\alpha,j} = 0 & else \end{cases} \tag{S.139}$$

and

$$\Theta_w(t) = diag(e^{i\delta_w(t)}) \tag{S.140}$$

With, for all $j \in \sigma^\alpha_{symm}$:

$$(\delta_k)_j(t) + (\delta_w)_j(t) = -(\delta_\phi)_j(t) \tag{S.141}$$

.

Naturally, the product $\overline{W\,dbc(K)}\Sigma^{xx} = A$ then becomes:

$$\begin{cases} A_{\alpha,\alpha}(t) = n \sum_{j\in\sigma^\alpha_{symm}} |(Q\phi^\alpha)_j| \, |(Qk(t))_j|^2 \, \overline{\Sigma}_{\alpha,\alpha} & \alpha \in \{0,\cdots,p-1\} \\ A_{\alpha,\beta}(t) = 0 & else \end{cases} \tag{S.142}$$

And we also have (cfr. Eq. 12):

$$R = Q^{-1} \tag{S.143}$$
$$\overline{W}(t) = \Omega(t)\Theta_w(t) \tag{S.144}$$
$$W = U\overline{W}(t)Q \tag{S.145}$$

as soon as the parameters become aligned and balanced. Given that this happens fast, the dynamics can be approximated by assuming this shape for W from the start.

We can plug this in the differential equation given by Eq. 15 to obtain:

$$\frac{1}{n\lambda}\Big(\frac{d\,diag(\boldsymbol{Q}^{-1}\boldsymbol{k})}{dt}\Big)_{j,j} = \boldsymbol{Q}_{j,:}^{-1}\boldsymbol{R}^{-*T}\overline{\boldsymbol{W}}^{*T}(\boldsymbol{S}-\boldsymbol{A})(\boldsymbol{QV})_{:,j}^{T} \tag{S.146}$$

$$\Longleftrightarrow \frac{1}{n\lambda}\Big(\frac{d\,diag(\boldsymbol{Q}\boldsymbol{k})}{dt}\Big)_{j,j} = \boldsymbol{Q}_{j,:}\boldsymbol{R}^{-*T}\overline{\boldsymbol{W}}^{*T}(\boldsymbol{S}-\boldsymbol{A})(\boldsymbol{Q}^{-1}\boldsymbol{V})_{:,j}^{T} \tag{S.147}$$

$$\Longleftrightarrow \frac{1}{n\lambda}\Big(\frac{d\,diag(\boldsymbol{Q}\boldsymbol{k})}{dt}\Big)_{j,j} = \boldsymbol{Q}_{j,:}\boldsymbol{Q}^{-1}(\boldsymbol{\Omega}\boldsymbol{\Theta_w})^{*T}(\boldsymbol{S}-\boldsymbol{A})(\boldsymbol{Q}^{-1}\boldsymbol{V})_{:,j}^{T} \tag{S.148}$$

$$\Longleftrightarrow \frac{1}{n\lambda}\Big(\frac{d\,diag(\boldsymbol{Q}\boldsymbol{k})}{dt}\Big)_{j,j} = \boldsymbol{Q}_{j,:}\boldsymbol{Q}^{-1}(\boldsymbol{\Omega}\boldsymbol{\Theta_w})^{*T}(\boldsymbol{S}-\boldsymbol{A})(\boldsymbol{Q}^{-1}\boldsymbol{V})_{:,j}^{T} \tag{S.149}$$

$$\Longleftrightarrow \frac{1}{n\lambda}\Big(\frac{d\,diag(\boldsymbol{Q}\boldsymbol{k})}{dt}\Big)_{j,j} = \boldsymbol{I}_{j,:}\boldsymbol{\Theta_w}^{*}(\boldsymbol{\Omega})^{T}(\boldsymbol{S}-\boldsymbol{A})(\boldsymbol{Q}^{-1}\boldsymbol{V})_{:,j}^{T} \tag{S.150}$$

$$\Longleftrightarrow \frac{1}{n\lambda}\Big(\frac{d\,diag(\boldsymbol{Q}\boldsymbol{k})}{dt}\Big)_{j,j} = \boldsymbol{\Theta_w}_{j,j}^{*}(\boldsymbol{\Omega})_{j,\alpha}^{T}(\boldsymbol{S}-\boldsymbol{A})_{\alpha,\alpha}(\boldsymbol{Q}^{-1}\boldsymbol{V})_{\alpha,j}^{T} \tag{S.151}$$

$$\Longleftrightarrow \frac{1}{n\lambda}\frac{d\big(|(\boldsymbol{Q}\boldsymbol{k})_j|e^{i(\boldsymbol{\delta_k})_j}\big)}{dt} = e^{-i(\boldsymbol{\delta_w})_j}|(\boldsymbol{Q}\boldsymbol{k})_j|(\boldsymbol{S}-\boldsymbol{A})_{\alpha,\alpha}e^{-i(\boldsymbol{\delta_\phi})_j}|(\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_j| \tag{S.152}$$

$$\Longleftrightarrow \frac{1}{n\lambda}\frac{d\big(|(\boldsymbol{Q}\boldsymbol{k})_j|e^{i(\boldsymbol{\delta_k})_j}\big)}{dt}|(\boldsymbol{Q}\boldsymbol{k})_j|e^{-i(\boldsymbol{\delta_k})_j} = e^{-i((\boldsymbol{\delta_w})_j+(\boldsymbol{\delta_k})_j+(\boldsymbol{\delta_\phi})_j)}|(\boldsymbol{Q}\boldsymbol{k})_j|^2(\boldsymbol{S}-\boldsymbol{A})_{\alpha,\alpha}|(\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_j| \tag{S.153}$$

$$\Longleftrightarrow \frac{1}{2n\lambda}\Big(\frac{d\,|(\boldsymbol{Q}\boldsymbol{k})_j|^2}{dt}\Big)_{j,j} = |(\boldsymbol{Q}\boldsymbol{k})_j|^2(\boldsymbol{S}-\boldsymbol{A})_{\alpha,\alpha}|(\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_j|. \tag{S.154}$$

This is Eq. 23 of the main paper, describing the winner-takes-all dynamics.

H. Analytical Solutions to the Dynamics of Learning for Pure Cosines Datasets

We derive exact, analytical solutions to the evolution of the effective singular values given the following input:

$$X_{l,m}^{(c)} = b^{(c)}\cos(2\pi\frac{\mu l}{n} + 2\pi\frac{\nu m}{n}) \tag{S.155}$$

as discussed in the main paper. We thus use a *single* cosine per class image, and we set phase $\delta_j = 0$ (cfr. Eq. S.66). The choice of the phase does not influence the conclusions, but makes the discussion in the main paper less complicated.

In this case the right singular vectors $\phi^\alpha = \phi^{(\alpha)*}$ are given by:

$$\phi_i^\alpha = \frac{1}{d_\alpha n}\cos(2\pi\frac{\mu l}{n} + 2\pi\frac{\nu m}{n} + (\delta_\phi)_j), \tag{S.156}$$

with $(\delta_\phi)_j = 0$ or $(\delta_\phi)_j = \pi$, introducing a possible minus sign (see Eq. S.67 and discussion following the equation). Here $\mu = div(j,n)$, $\nu = mod(j,n)$ are the frequency indices and $l = div(i,n)$, $m = mod(i,n)$ are the pixel indices. For ease of notation, we have introduced d_α:

$$\begin{cases} d_\alpha = 1 & j = 0, j \to \alpha \\ d_\alpha = \frac{1}{\sqrt{2}} & j \neq 0, j \to \alpha \end{cases} \tag{S.157}$$

where with $j \to \alpha$ we indicate that j is the vec-2D index that indexes the horizontal (index μ) and vertical (index ν) frequencies that are used to construct the singular vector indexed by α. We then find:

$$\begin{cases} (\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_j = d_\alpha e^{i(\delta_\phi)_j} & j \to \alpha \\ (\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_{j_{symm}} = d_\alpha e^{i(\delta_\phi)_{j_{symm}}} = d_\alpha e^{-i(\delta_\phi)_j} & j \to \alpha \\ (\boldsymbol{Q}\boldsymbol{\phi}^\alpha)_j = 0 & else \end{cases} \tag{S.158}$$

where we denote with j_{symm} the frequency index that corresponds to the pair of frequency indices $(n-\nu, n-\mu)$.

We initialize the kernel with small, random weights, and set the initial weights of the fully connected layer according to Eq. S.138. This makes the network balanced and aligned. We get (see Eq. S.142):

$$\begin{cases} A_{\alpha,\alpha}(t) = 2nd_\alpha |(\boldsymbol{Q}\boldsymbol{k}(t))_j|^2 \, \overline{\Sigma}_{\alpha,\alpha} & j \to \alpha, j \neq 0, \alpha \in \{0, \cdots, p-1\} \\ A_{\alpha,\alpha}(t) = nd_\alpha |(\boldsymbol{Q}\boldsymbol{k}(t))_j|^2 \, \overline{\Sigma}_{\alpha,\alpha} & j \to \alpha, j = 0, \alpha \in \{0, \cdots, p-1\} \\ A_{\alpha,\beta}(t) = 0 & \text{else} \end{cases} \tag{S.159}$$

Which can be succinctly written as:

$$\boldsymbol{A}_{\alpha,\alpha} = \frac{n}{d_\alpha} |(\boldsymbol{Q}\boldsymbol{k})_j|^2 \overline{\boldsymbol{\Sigma}^{xx}}_{\alpha,\alpha} \tag{S.160}$$

and $\boldsymbol{A}_{\alpha,\beta} = 0$ if $\alpha \neq \beta$.

These initial conditions decouple the general system of equations (eqs. 15 and 16); we can use Eq. S.154 for the specific pairs of modes and frequencies (α, j) used in the single cosines dataset; all other derivatives are zero. From this equation and the fact that we only have one frequency j per mode α, we can derive the full evolution of the effective singular values $A_{\alpha,\alpha}$:

$$\frac{d\boldsymbol{A}_{\alpha,\alpha}}{dt} = \frac{n}{d_\alpha} \frac{d|(\boldsymbol{Q}\boldsymbol{k})_j|^2}{dt} \overline{\boldsymbol{\Sigma}^{xx}}_{\alpha,\alpha} \tag{S.161}$$

$$= 2n\lambda \frac{n}{d_\alpha} \overline{\boldsymbol{\Sigma}^{xx}}_{\alpha,\alpha} |(\boldsymbol{Q}\boldsymbol{k})_j|^2 (\boldsymbol{S} - \boldsymbol{A})_{\alpha,\alpha} d_\alpha \tag{S.162}$$

$$= 2nd_\alpha \lambda \boldsymbol{A}_{\alpha,\alpha} (\boldsymbol{S}_{\alpha,\alpha} - \boldsymbol{A}_{\alpha,\alpha}) \tag{S.163}$$

This is a non-linear, separable differential equation that we can solve for time t (see also (Saxe et al., 2019)). To simplify notation, we will use $a_\alpha = A_{\alpha,\alpha}$ and $s_\alpha = S_{\alpha,\alpha}$. Using

$$t = \frac{1}{2nd_\alpha\lambda} \int_{a_\alpha^{(0)}}^{a_\alpha^{(f)}} \frac{da_\alpha}{a_\alpha(s_\alpha - a_\alpha)} = \frac{1}{2nd_\alpha\lambda s_\alpha} ln\Big(\frac{a_\alpha^{(f)}(s_\alpha - a_\alpha^{(0)})}{a_\alpha^{(0)}(s_\alpha - a_\alpha^{(f)})}\Big) \tag{S.164}$$

where t is the training time (in number of samples) required to arrive at value $a_\alpha^{(f)}$ at time t from initial value $a_\alpha^{(0)}$. The evolution during training of the effective singular value, $a_\alpha(t)$, is then given by:

$$a_\alpha(t) = \frac{s_\alpha e^{2\lambda nd_\alpha s_\alpha t}}{e^{2\lambda d_\alpha ns_\alpha t} - 1 + s_\alpha/a_\alpha^{(0)}}. \tag{S.165}$$

I. Details of experiments

All experiments are implemented with tensorflow/keras. A factor 1/p is used in the implementation of the MSE loss, with p the number of classes. All experiments can be completed in a couple of hours on a basic CPU.

When only one image per class is used, this means in practice that for each update, a sample/class is randomly selected. For the CIFAR-4 dataset, we used the more conventional setup with a number of epochs, wherein the samples of the train set are given as input in a random order.

We use CNNs with a single convolutional layer, a flatten layer, and a single fully connected layer. There are only linear activation functions. The convolutional layer consists of a single kernel $\boldsymbol{K}$ with the same dimensions as the input images $(n \times n)$ for grayscale images, and 3 such kernels for colour images. Images are circularly padded beforehand: we extend the images with n-1 rows/columns of the pixel values of the opposite side. When the kernel of $n\times$ is applied, the padding is then in practice equivalent to wrapping the kernel around the image when it slides over the original image boundary. The result of this convolution is again an $n \times n$ matrix.

FCNNs consist of two fully connected layers; the hidden one with n^2 nodes, the output one with p nodes. The first weight matrix therefore has dimensions $n^2 \times n$, the second $p \times n^2$. The choice of nodes in the hidden layer corresponds to the architecture of the CNN, which performs a convolution from an $n \times n$ to an $n \times n$ image, and therefore has an associated doubly block circulant matrix of size $n^2 \times n^2$. For the CIFAR-4 dataset, we use $3n^2$ nodes in the hidden layer. Images are flattened to $n^2 \times 1$ or $3n^2 \times 1$ vectors beforehand.
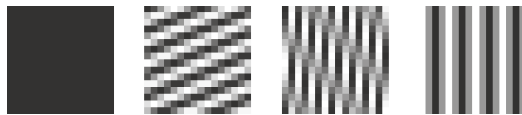
Figure S.3. Cosines used in the 'pure cosines' experiment.

I.1. Pure cosines experiment

Image dimension: 16×16; horizontal and vertical freq. pairs for each class: (0,0), (5,2), (1,7), (0,4); amplitudes for each class: 1.5, 1, 0.5, 0.2.

- $\lambda_{CNN} = 1/2000$, $\lambda_{FCNN} = 16/2000$

- init CNN: random normal with $\mu = 0$ and $\sigma = 0.00001$ for the kernel $\boldsymbol{K}$, weights $\boldsymbol{W}$ balanced to $\boldsymbol{K}$ using $\boldsymbol{R} = \boldsymbol{Q}$ (eq. S.157). This makes $\boldsymbol{A}_{init}$ rectangular diagonal.

- init FCNN: using $\boldsymbol{A}_{init}$ from CNN, setting the first p diagonal elements of $\overline{\boldsymbol{W}}^1$ and $\overline{\boldsymbol{W}}^2$ to the value $\sqrt{\frac{\boldsymbol{A}_{(init)\alpha,\alpha}}{\Sigma^{xx}_\alpha}}$, other elements to zero.

- number of updates: 8000

- number of repeated experiments: 10

I.2. Dominant frequency bias experiment (sums of cosines)

Sums of cosines dataset as shown in fig. 4, image dimensions 64×64.

- $\lambda_{CNN} = 1/10000$, $\lambda_{FCNN} = 64/10000$

- init CNN: random normal with $\mu = 0$ and $\sigma = 0.00001$ for the kernel and weights.

- init FCNN: random normal with $\mu = 0$ and $\sigma = 0.00001$ for all weights.

- number of updates: 600

- number of repeated experiments: 10

I.3. Geometric shapes experiment

Dataset as shown in fig. 2. Image dimensions 64×64.

- $\lambda_{CNN} = 1/20000$, $\lambda_{FCNN} = 64/20000$

- init CNN: random normal with $\mu = 0$ and $\sigma = 0.00001$ for the kernel and weights.

- init FCNN: random normal with $\mu = 0$ and $\sigma = 0.00001$ for all weights.

- number of updates: 60000

- number of repeated experiments: 10

I.4. CIFAR-4 experiment

First four classes of the CIFAR-10 dataset. Train set consists of the 5000 samples for each class in the CIFAR-10 train set (thus 20000 samples in total) with mean subtracted, test set consist of the 1000 samples for each class in the CIFAR-10 test set (4000 samples in total) with mean subtracted. Image dimension 32×32.

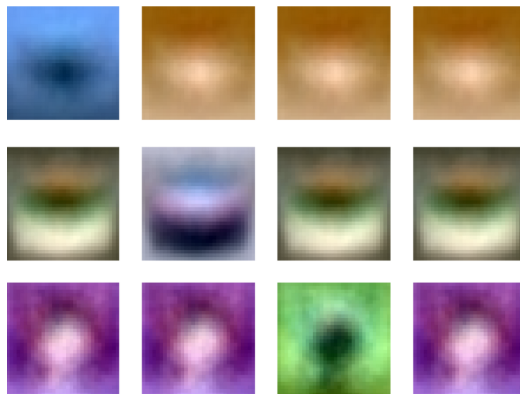- $\lambda_{CNN} = 1/10000$, $\lambda_{FCNN} = 32/10000$

Figure S.4. Modes of the CIFAR-4 dataset with removed first mode (i.e., subtracted mean over samples). Three modes remain to distinguish between the four classes. The top row is a visualisation of the first mode, the second row of the second mode, etc. $1 \times 3n^2$ vectors are visualised as $n \times n$ color images. Each image is normalised to values between 0 and 1. In this way, we can see that the first mode distinguishes central objects in a blue background, which is especially relevant for the first class 'airplane'; the second mode discovers the class 'automobile' based on a reddish color and a vague frontal view of a car, and the third mode distinguishes 'bird' based in a central object in a predominantly green background.

- init CNN: random normal with $\mu = 0$ and $\sigma = 0.0001$ for the kernel and weights.

- init FCNN: random normal with $\mu = 0$ and $\sigma = 0.0001$ for all weights.

- number of updates: 100000

- number of repeated experiments: 10

J. Comparison of the 2D-vec Fourier Spectra of the Kernel and the Singular Vectors

In Fig. S.5 (next page), we visualize that the frequencies that make up the kernel are the dominant frequencies of the singular vectors. The data shown are from the experiment with the geometric shapes dataset (cfr. Fig. 5).

Each row of the figure corresonds to a timepoint during training with the CNN:

- row 0, timepoint = 2000 samples, mode 0 discovered;

- row 1, timepoint = 9000 samples, mode 1 discovered;

- row 2, timepoint = 16000 samples, mode 2 discovered;

- row 3, timepoint = 55000 samples, mode 3 discovered

(cfr. Fig. 5 (a)). At each row we plot the 2D-vec Fourier spectra of the singular vectors, i.e., the values $|(\boldsymbol{Q}\phi^{\alpha})_j|$, that correspond to the modes discovered at that point. These spectra are unaltered during training, and are thus repeated across rows. The kernel, and therefore the values $|(\boldsymbol{Q}\boldsymbol{k})_j|$, do change during training. We consider the average of the values $|(\boldsymbol{Q}\boldsymbol{k})_j|$ over the number of experiment runs. We overlay these averaged values $|(\boldsymbol{Q}\boldsymbol{k})_j|$ for the given timepoints at the corresponding values $|(\boldsymbol{Q}\phi^{\alpha})_j|$. This overlay is indicated with black crosses. The size of these crosses codes for the magnitude of the values $|(\boldsymbol{Q}\boldsymbol{k})_j|$. In this way, we can first note that the Fourier spectrum of k is much sparser than the different Fourier spectra of the singular vectors (in the spectrum of each singular vectors, the number of crosses is much smaller than the total number of other markers). Moreover, large values $|(\boldsymbol{Q}\boldsymbol{k})_j|$ (large crosses) mainly correspond to large values of $|(\boldsymbol{Q}\phi^{\alpha})_j|$. Thus the kernel is, at any point during training, a sparse mixture of the dominant frequencies of the singular vectors discovered so far.

K. Details of Experiments with Deep Networks

The deep networks have the following architecture:

- a convolutional layer, 16 channels, zero padding

- a convolutional layer, 16 channels, zero padding

- (non-linear network) a max-pooling layer with size 2×2

- a convolutional layer, 16 channels, zero padding

- a convolutional layer, 16 channels, zero padding

- (non-linear network) a max-pooling layer with size 2×2

- flatten layer

- fully connected layer with 16 nodes

- fully connected layer to 10 output nodes

For the non-linear network, we apply the non-linear ReLU activation functions on all convolutional layers and the first fully connected layer. The last layer has no activation function (=linear activation).

We use the MSE loss and the full dataset, i.e., all 50000 samples in the training set. The batch size is set to 8 and there are thus 6240 batches per epoch. We train the networks for 25 epochs, yielding a total of 156000 batches. Every 500 batches we compute the test and train loss on a subset of 5000 train and test samples, respectively. At these points, we also compute $\mathbf{\Sigma}^{\hat{y}x}$ (eq. 4) and, from this result, we compute the matrix $\mathbf{A}$ (eq. 5).

The network is initialized with the Glorot normal initializer. This is a truncated normal distribution with mean 0 and standard deviation $\sqrt{2/(\text{fan in} + \text{fan out})}$. Here *fan in* is the number of input units in the weight matrix and *fan out* is the number of output units in the weight matrix. We scaled the values drawn from this initializer with a factor 0.1 to ensure that the weights would be small enough to yield a rich training regime.
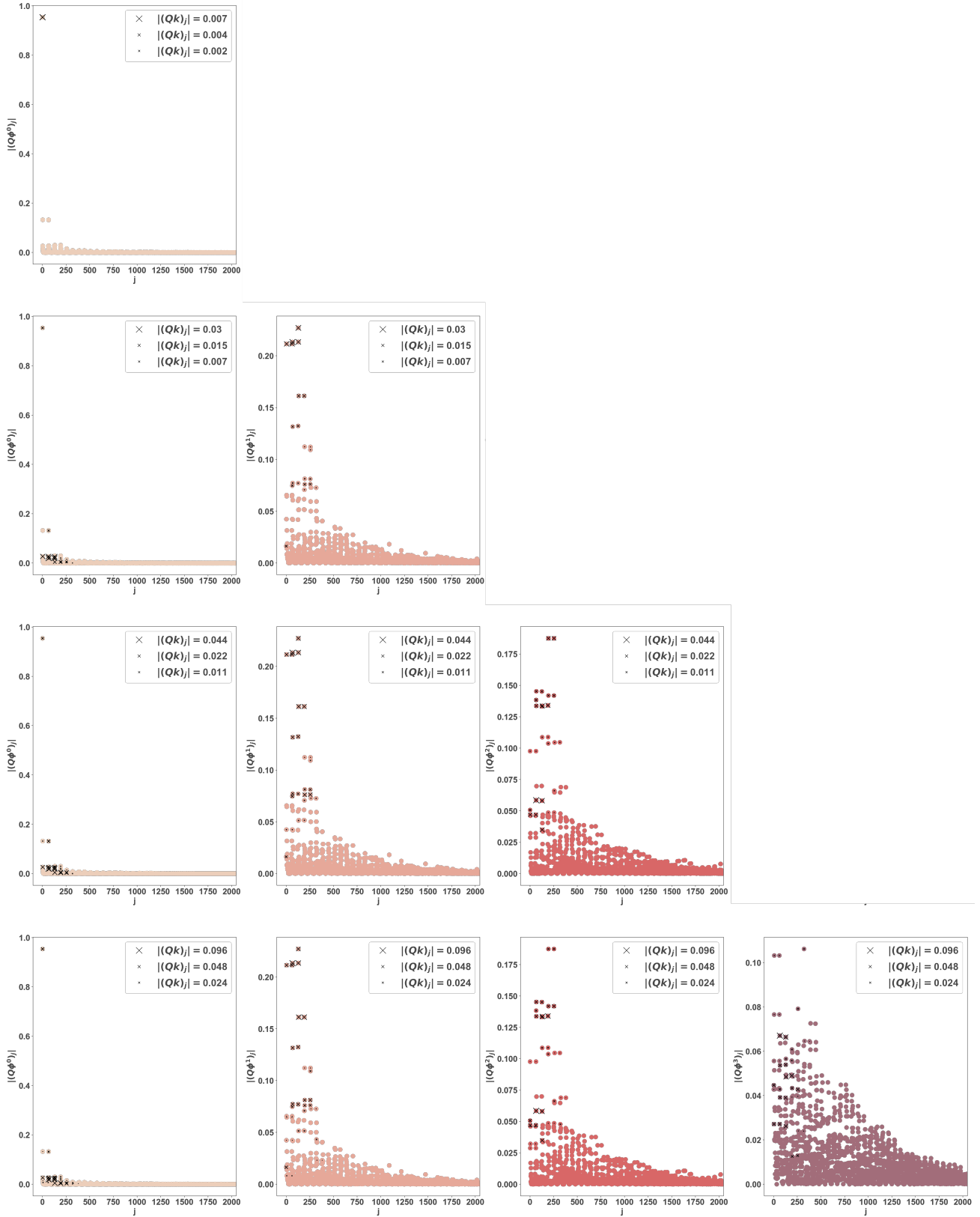
Figure S.5. Comparison of the 2D-vec Fourier Spectra of the Kernel and the Singular Vectors at Selected Timepoints.