A. Isotropy of Weight Initialization Implies Conditional Independence

In this section we show that for isotropic initial weight distributions,

$$P(W^0 R) = P(W^0)\;\forall\, R \in O(d)\,, \tag{20}$$

the training activations Z_{train} depend on the training data X_{train} only through the second moment matrix K_{train}. This is summarized in Eq. 6 repeated here for convenience:

$$\mathcal{I}(Z^0_{\text{train}}; X_{\text{train}} \mid K_{\text{train}}) = 0.$$

The argument is as follows, the isotropy of the initial weight distribution implies that the distribution of first layer activations conditioned on the training data is invariant under orthogonal transformations.

$$P(Z^0_{\text{train}}|RX_{\text{train}}) = P(Z^0_{\text{train}}|X_{\text{train}})\;\forall R \in O(d)\,. \tag{21}$$

To derive this we can write the distribution over Z^0_{train} in terms of the distribution over initial weights, $P(Z^0_{\text{train}}|X_{\text{train}}) = \int DW^0 P(W^0)\delta(Z^0_{\text{train}} - W^0 X_{\text{train}})$. Here DW^0 is the uniform measure over the components of W^0, DW^0. We then have

$$\begin{aligned} P(Z^0_{\text{train}}|RX_{\text{train}}) &= \int DW^0 P(W^0)\delta(Z^0_{\text{train}} - W^0 R X_{\text{train}}) \\ &= \int D\tilde{W}^0 P(\tilde{W}^0 R^T)\delta(Z^0_{\text{train}} - \tilde{W}^0 X_{\text{train}}) \\ &= \int D\tilde{W}^0 P(\tilde{W}^0)\delta(Z^0_{\text{train}} - \tilde{W}^0 X_{\text{train}}) \;=\; P(Z^0_{\text{train}}|X_{\text{train}})\,. \end{aligned} \tag{22}$$

Here, δ denotes the Dirac delta function. To arrive at the second line we defined $\tilde{W}^0 := W^0 R$ and used the invariance of the measure DW^0. The third line follows from the $O(d)$ invariance of the initial weight distribution. Now that we have established the rotational invariance of the distribution over first layer activations we can derive Eq. 6.

By the first fundamental theorem of invariant theory (Kraft & Procesi, 1996), the only $O(d)$ invariant functions of n vectors in d dimensions are the n^2 inner products $K_{\text{train}} = X^\top_{\text{train}} X_{\text{train}}$. Thus $P(Z^0_{\text{train}}|X_{\text{train}}) = h(K_{\text{train}})$ for some function h, and $P(Z^0_{\text{train}}|X_{\text{train}}, K_{\text{train}}) = P(Z^0_{\text{train}}|K_{\text{train}})$. Eq. 6 then follows from the definition of conditional mutual information.

$$\begin{aligned} I(Z^0_{\text{train}}; X_{\text{train}} \mid K_{\text{train}}) :&= \mathbb{E}_{K_{\text{train}}}\left[D_{\text{KL}}(P(Z^0_{\text{train}}, X_{\text{train}}|K_{\text{train}})||P(Z^0_{\text{train}}|K_{\text{train}})P(X_{\text{train}}|K_{\text{train}}))\right] \\ &= \mathbb{E}_{K_{\text{train}}}\left[P(X_{\text{train}}|K_{\text{train}})D_{\text{KL}}(P(Z^0_{\text{train}}|X_{\text{train}}, K_{\text{train}})||P(Z^0_{\text{train}}|K_{\text{train}}))\right] \\ &= 0\,. \end{aligned} \tag{23}$$

B. Alternative Proof of Theorem 2.1.1

The proof statement in Eq. 5 can alternatively be directly established by identifying the sources of randomness and mutual information in the update rules for Z^t_{train} and θ^t, and making an inductive argument.

The base case, $t = 0$, is already discussed in the paper body. Assume Eq. 5 holds for $t = i$. By the chain rule,

$$\mathcal{I}(Z^{i+1}_{\text{train}}, \theta^{i+1}; X_{\text{train}} \mid K_{\text{train}}, Y_{\text{train}}) = \mathcal{I}(Z^{i+1}_{\text{train}}; X_{\text{train}} \mid K_{\text{train}}, Y_{\text{train}}) + \mathcal{I}(\theta^{i+1}; X_{\text{train}} \mid K_{\text{train}}, Y_{\text{train}}, Z^{i+1}_{\text{train}}). \tag{24}$$

The update rules for Z_{train} and θ are given in Eqs. 4 and 3, respectively, where $L^i = L(g_{\theta^i}(Z^i_{\text{train}}); Y_{\text{train}})$. When K_{train} and Y_{train} are held fixed, the only sources of randomness in Z^{i+1}_{train} and θ^{i+1} are Z^i_{train} and θ^i. By the initial assumption, these are conditionally independent of X_{train} given K_{train} and Y_{train}, and therefore both terms on the right-hand side of Eq. 24 evaluate to zero.

C. Lower Dimensional Whitening

Here we prove Theorem 3.1.1 describing the loss of information when whitening $d < n$.

Proof. By Theorem 2.2.1 and the causal diagram in Fig. 2(c), we know that the distribution over test set predictions depends on model inputs only through K. Since K is positive semidefinite, it is fully specified by $(n^2+n)/2$ entries. For d < n these entries are not independent. K encodes the inner products between n vectors in d dimensions, These are specified by n magnitudes and $n(d-1) - (d(d-1))/2$ independent angles.

Thus, for a model trained on unwhitened data,

$$c = \min\left(nd - \frac{d^2-d}{2}, \frac{n^2+n}{2}\right). \tag{25}$$

Next we consider the case that full data whitening has been performed, such that $\hat{F} = I$. Recall the following identity, where $^+$ indicates the pseudoinverse:

$$\hat{X}^\top = \hat{X}^+ \hat{X} \hat{X}^\top. \tag{26}$$

Using this, we can rewrite $\hat{K}$ as

$$\begin{aligned}\hat{K} &= \hat{X}^\top \hat{X} = \hat{X}^+ \hat{X} \hat{X}^\top \hat{X} = \hat{X}^+ \hat{F} \hat{X} \\ &= \hat{X}^+ \hat{X}.\end{aligned} \tag{27}$$

Next consider a modified dataset

$$\widetilde{X} = Q\hat{X} = [I \cdots], \tag{28}$$

where the matrix $Q \in \mathbb{R}^{d\times d}$ has been chosen such that the first d columns of $\widetilde{X}$ correspond to the identity matrix (ie Q is the inverse of the submatrix formed by the first d columns of $\hat{X}$). Because its first d columns are deterministic, $\widetilde{X}$ can be stored using $(n-d)\,d$ real values. Despite being represented by d^2 fewer values, this compressed dataset can still be used to reconstruct $\hat{K}$,

$$\begin{aligned}\hat{K} &= \hat{X}^+ \hat{X} = \hat{X}^+ Q^{-1} Q \hat{X} = \left(Q\hat{X}\right)^+ Q\hat{X} \\ &= \widetilde{X}^+ \widetilde{X}.\end{aligned} \tag{29}$$

We further observe that when $n > d$, $(n-d)\,d < (n^2+n)/2$. This is enough to establish

$$\hat{c} = (n-d)\,d < c. \tag{30}$$

□

D. Whitening in Linear Models

Linear models are widely used for regression and prediction tasks and provide an instructive laboratory to understand the effects of data whitening. Furthermore, linear models provide additional intuition for why whitening is harmful – whitening puts signal and noise directions in the data second moment matrix, F, on equal footing (see Fig. 1). For data with a good signal to noise ratio, unwhitened models learn high signal directions early during training and only overfit to noise at late times. For models trained on whitened data, the signal and noise directions are fit simultaneously and thus the models overfit immediately.

Consider a linear model with mean squared error loss,

$$f(X) = WX\,, \quad L = \frac{1}{2}||f(X) - Y||^2\,. \tag{31}$$

This loss function is convex. Here we focus on the low dimensional case, $d < n$, where the loss has a unique global optimum $W^\star = F_{\text{train}}^{-1} X_{\text{train}} Y_{\text{train}}$. The model predictions at this global optimum, $f_\star(X) = W^\star X$, are invariant under any whitening transform (3.0.1). As a result, any quality metric (loss, accuracy, etc...) for this global minimum is the same for whitened and unwhitened data.

The story is more interesting, however, during training. Consider a model trained via gradient flow (similar statements can be made for gradient descent or stochastic gradient descent). The dynamics of the weights are given by

$$\frac{dW}{dt} = -\frac{\partial L}{\partial W}\,, \quad W(t) = e^{-tF_{\text{train}}}W(0) + (1 - e^{-tF_{\text{train}}})W^* \,. \tag{32}$$

The evolution in Eq. 32 implies that the information contained in the trained weights $W(t)$ about the training data X is entirely determined by F and $W^\star$. In terms of mutual information, we have

$$\mathcal{I}(W(t); X | F_{\text{train}}, W^\star) = 0\,. \tag{33}$$

As whitening sets $\hat{F}_{\text{train}} = I$, a linear model trained on whitened data does not benefit from the information in F_{train}.

At a more microscopic level, we can decompose Eq. 32 in terms of the eigenvectors, v_i, of F:

$$W(t) = \sum_{i=1}^{d} v_i w_i(t),\ w_i(t) = e^{-t\lambda_i}w_i(0) + (1 - e^{-\lambda_i t})w_i^\star\,. \tag{34}$$

We see that for unwhitened data the eigen-modes with larger eigenvalue converge more quickly towards the global optimum, while the small eigen-directions converge slowly. For centered X, F is the feature covariance and these eigen-directions are exactly the principle components of the data. As a result, training on unwhitened data is biased towards learning the top principal directions at early times. This bias is often beneficial for generalization. Similar simplicity biases have been found empirically in deep linear networks (Saxe et al., 2014) and in deep networks trained via SGD (Rahaman et al., 2018; Ronen et al., 2019) where networks learn low frequency modes before high. In contrast, for whitened data, $\hat{F}_{\text{train}} = I$ and the evolution of the weights takes the form

$$\hat{w}_i(t) = e^{-t}\hat{w}_i(0) + (1 - e^{-t})\hat{w}_i^\star\,. \tag{35}$$

All hierarchy between the principle directions has been removed, thus training fits all directions at a similar rate. For this reason linear models trained on unwhitened data can generalize significantly better at finite times than the same models trained on whitened data. Empirical support for this in a linear image classification task with random features is shown in Fig. 3(a).

D.1. The Global Optimum

At the global optimum, $W^\star = F_{\text{train}}^{-1}X_{\text{train}}Y_{\text{train}}$, the network predictions on test points can be written in a few equivalent ways,

$$f_\star(X_{\text{test}}) = Y_{\text{train}}^T X_{\text{train}}^T F_{\text{train}}^{-1} X_{\text{test}} = Y_{\text{train}}^T K_{\text{train}}^+ K_{\text{train}\times\text{test}} = Y_{\text{train}}^T \hat{K}_{\text{train}\times\text{test}}\,. \tag{36}$$

Here, the $+$ superscript is the pseudoinverse, and $\hat{K}_{\text{train}\times\text{test}}$ is the whitened train-test data-data second moment matrix. These expressions make manifest that the test predictions at the global optimum only depend on the training data through K_{train} and $K_{\text{train}\times\text{test}}$.

Let us briefly consider the case where $n \leq d$ with full data whitening. The global optimum $W^\star$ is still unique up to a pseudoinverse: $W^\star = F_{\text{train}}^+ X_{\text{train}} Y_{\text{train}}$. When full data whitening is performed, we have $\hat{K} = I$ from Eq. 11, and so the mixed second moment matrix $\hat{K}_{\text{train}\times\text{test}}$, which is an off-diagonal block of $\hat{K}$, is a zero matrix. Therefore $f_\star(X_{\text{test}}) = Y_{\text{train}}^T \hat{K}_{\text{train}\times\text{test}} = 0$ for all the test points, and it is particularly simple to see how whitening can destroy generalization.

D.2. High Dimensional Linear Models

The discussion is very similar in the high dimensional case, $d > n$. In this case, there is no longer a unique solution to the optimization problem, but there is a unique optimum within the span of the data.

$$W_\parallel^\star = \left(F_{\text{train}}^{\parallel}\right)^{-1} X_{\text{train}}^{\parallel} Y_{\text{train}}\,, \quad W_\perp^\star = W_\perp(0)\,. \tag{37}$$

Here, we have introduced the notation $\|$ for directions in the span of the training data and $\perp$ for orthogonal directions. Explicitly, if we denote by $V^{\|} = \{v_1, v_2, \dots, v_n\} \in \mathbb{R}^{n\times d}$ the non-null eigenvectors of F_{train} and $V^{\perp} = \{v_{n+1}, v_{n+2}, \dots, v_d\} \in \mathbb{R}^{(d-n)\times d}$ the null eigenvectors, then $X^{\|}_{\text{train}} := V^{\|} X_{\text{train}}$, $W_{\|} := WV^{\|}$, $W_{\perp} := WV^{\perp}$, and $F^{\|}_{\text{train}} := V^{\|} F_{\text{train}} (V^{\|})^T$.

The model predictions at this optimum can be written as

$$f_\star(X_{\text{test}}) = f^0(X_{\text{test}}) - \left(f^0(X_{\text{train}}) - Y_{\text{train}}\right)^T K^{-1}_{\text{train}} K_{\text{train}\times\text{test}} \,. \tag{38}$$

This is the solution found by GD, SGD, and projected Newton's method.

The evolution approaching this optimum can be written (again assuming gradient flow for simplicity) as

$$W_{\|}(t) = e^{-tF^{\|}_{\text{train}}} W_{\|}(0) + (1 - e^{-tF^{\|}_{\text{train}}}) W^*_{\|} \,, \quad W_{\perp}(t) = W_{\perp}(0). \tag{39}$$

In terms of the individual components, $[W_{\|}(t)]_i = e^{-t\lambda_i}[W_{\|}(0)]_i + (1 - e^{-t\lambda_i})[W^*_{\|}]_i$.

As above, the hierarchy in the spectrum allows for the possibility of beneficial early stopping, while whitening the data results in immediate overfitting.

D.3. Supplementary Experiments with Linear Least Squares

In Fig. App.1 we present the same experiment as in Fig. 4(a) at three additional dataset sizes. In all cases, the best test performance achievable by early stopping on whitened data was worse than on unwhitened data.

In Fig. App.2, we study the effect on generalization of using the entire dataset of 60000 CIFAR-10 images to compute the whitening transform regardless of training set size. We call this type of whitening 'distribution whitening' to indicate that we are interested in what happens when the whitening matrix is approaches its ensemble limit.

In Fig. App.3, we compare generalization performance of models trained on whitened versus unwhitened data from two different parameter initializations. Initial distributions with larger variance produce models that generalize worse, but for a fixed initial distribution, models trained on whitened data generally underperform models trained on unwhitened data.

E. Second Order Optimization of Wide Neural Networks

Here we consider second order optimization for wide neural networks. In recent years much progress has been made in understanding the dynamics of wide neural networks (Jacot et al., 2018), in particular it has been realized that wide networks trained via GD, SGD or gradient flow evolve as a linear model with static, nonlinear features given by the derivative of the network map at initialization (Lee et al., 2019).

In this section we extend the connection between linear models and wide networks to second order methods. In particular we argue that *wide networks trained via a regularized Newton's method evolve as linear models trained with the same second order optimizer.*

We consider a regularized Newton update step,

$$\theta^{t+1} = \theta^t - \eta \left(\epsilon\mathbf{1} + H\right)^{-1} \frac{\partial L^t}{\partial \theta} \,. \tag{40}$$

This diagonal regularization is a common generalization of Newton's method. One motivation for such an update rule in the case of very wide neural networks is that the Hessian is necessarily rank deficient, and so some form of regularization is needed.

For a linear model, $f_{\text{linear}}(x) = \theta^\top \cdot g(x)$, with fixed non-linear features, $g(x)$, the regularized newton update rule in weight space leads to the function space update.

$$f^{t+1}_{\text{linear}}(x) = f^t_{\text{linear}}(x) - \eta \sum_{x_a, x_b \in X_{\text{train}}} \Theta_{\text{linear}}(x, x_a) \left(\epsilon\mathbf{1} + \Theta_{\text{linear}}\right)^{-1}_{ab} \frac{\partial L_b}{\partial f_{\text{linear}}} \,. \tag{41}$$

Here, Θ_{linear}, is a constant kernel, $\Theta_{\text{linear}}(x, x') = \frac{\partial f}{\partial \theta}^\top \cdot \frac{\partial f}{\partial \theta} = g^\top(x) \cdot g(x')$.

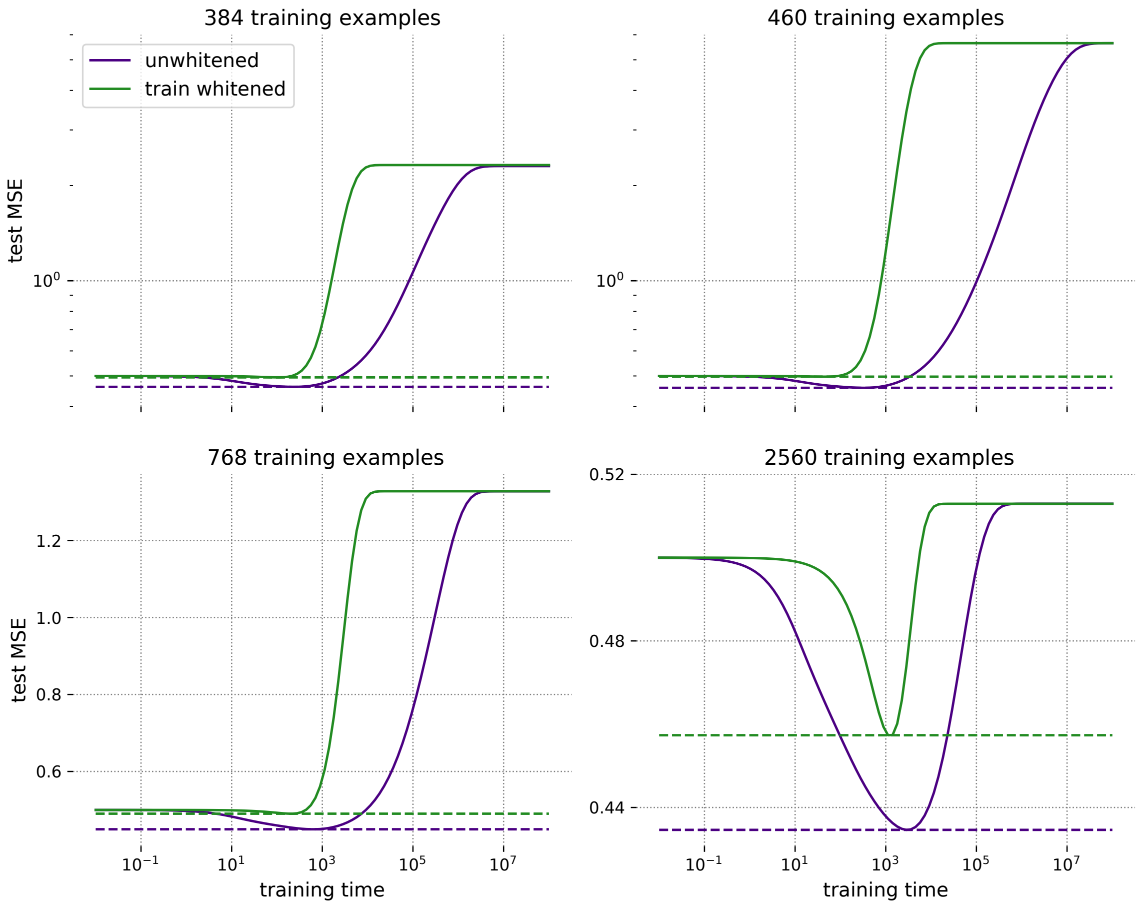

Figure App.1. **Whitening data speeds up training but reduces generalization in linear models.** Here we show representative examples of the evolution of test error with training time in a linear least-squares model where the training set consists of $384, 460, 768, 2560$ examples, as labeled. In all cases, while models trained on train-whitened data (in green) reach their optimal mean squared errors in a smaller number of epochs, they do no better than models trained on unwhitened data (in purple). In the large time limit of training, the two kinds of models are indistinguishable as measured by test error. The y-axis in the top row of plots is in log scale for clarity. In all cases, the input dimensionality of the data was 512.

For a deep neural network, the function space update takes the form

$$\begin{aligned} f^{t+1}(x) =& f^t(x) - \eta \sum_{x_a, x_b \in X_{\text{train}}} \Theta(x, x_a) \left(\epsilon \mathbf{1} + \Theta\right)^{-1}_{ab} \frac{\partial L_b}{\partial f} \\ &+ \frac{\eta^2}{2} \sum_{\mu,\nu=1}^{P} \frac{\partial^2 f}{\partial \theta_\mu \partial \theta_\nu} \Delta\theta^t_\mu \Delta\theta^t_\nu + \cdots . \end{aligned} \tag{42}$$

Here we have indexed the model weights by $\mu = 1 \dots P$, denoted the change in weights by $\Delta\theta^t$ and introduced the neural tangent kernel (NTK), $\Theta(x, x') = \frac{\partial f^\top}{\partial \theta} \cdot \frac{\partial f}{\partial \theta}$.

In general Eqs. 41 and 42 lead to different network evolution due to the non-constancy of the NTK and the higher order terms in the learning rate. For wide neural networks, however, it was realized that the NTK is constant (Jacot et al., 2018) and the higher order terms in η appearing on the second line in vanish at large width (Dyer & Gur-Ari, 2020; Huang & Yau, 2019; Littwin et al., 2020; Andreassen & Dyer; Aitken & Gur-Ari).[2]

[2]These simplifications were originally derived for gradient flow, gradient descent and stochastic gradient descent, but hold equally well

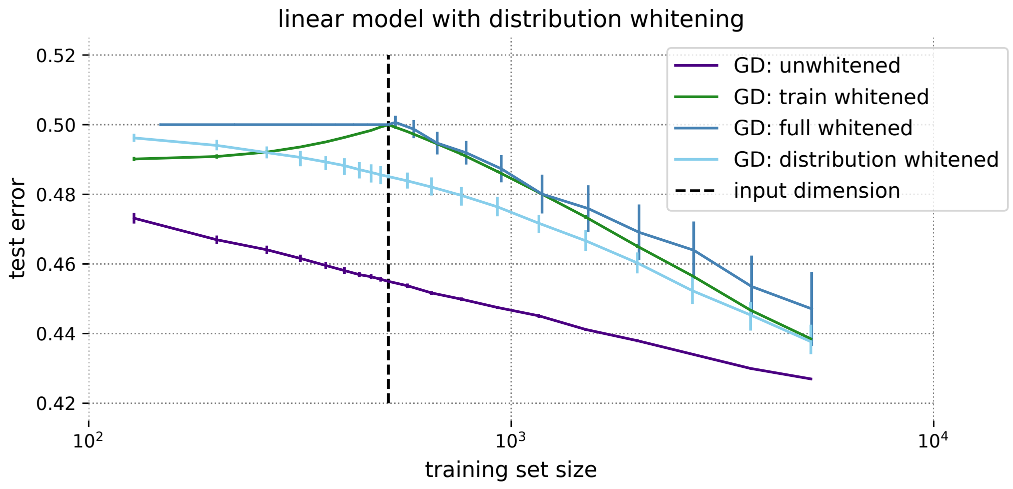

Figure App.2. **Whitening using the entire dataset behaves similarly to conventional whitening, with only a slight improvement in performance.** Whitening using a whitening transform computed on the entire CIFAR-10 dataset of 50000 training and 10000 test images (distribution whitening) improves performance over train- and full whitening, but does not close the performance gap with unwhitened data. With the exception of the 'distribution whitened' line, gradient descent data in this plot is identical to Fig. 3(a).

With these simplifications, the large width limit of Eq. 42 describes the same evolution as a linear model trained with fixed features $g(x) = \frac{\partial f(x)}{\partial \theta}|_{\theta=\theta_0}$ trained via a regularized Newton update.

F. MLP on MNIST

Here we present the equivalent of Fig. 3(b) for an MLP trained on MNIST. Experimental details are given in Appendix G. Similar to the results in Fig. 3(b) on CIFAR-10, in Fig. App.4, we find that models trained on fully whitened data generalize at chance levels (indicated by a test error of 0.9) in the high dimensional regime. Because MNIST is highly rank deficient, this result holds until the size of the dataset exceeds its input rank. Models trained on train-whitened data also exhibit reduced generalization when compared with models trained on unwhitened data, which exhibit steady improvement in generalization with increasing dataset size.

G. Methods

G.1. Whitening Methods

G.1.1. PCA Whitening

Consider a dataset $X \in \mathbb{R}^{d\times n}$. PCA whitening can be viewed as a two-step operation involving rotation of X into the PCA basis, followed by the normalization of all PCA components to unity. We implement this transformation as follows. First, we compute the the singular value decomposition of the unnormalized feature-feature second moment matrix $XX^\top$:

$$XX^\top = U\Sigma V^\top, \tag{43}$$

where the rank of Σ is $\min(n, d)$. The PCA whitening transform is then computed as $M = \Sigma^{-1/2} \cdot V^\top$, where the dot signifies element-wise multiplication between the whitening coefficients, $\Sigma^{-1/2}$, and their corresponding singular vectors. When Σ is rank deficient ($n < d$), we use one of two methods to stabilize the computation of the inverse square root: the addition of noise, or manual rank control. In the former, a small jitter is added to the diagonal elements of Σ before inverting it. This was implemented in the experiments in linear models. In the latter, the last $d - n$ diagonal elements of $\Sigma^{-1/2}$ are explicitly set to unity. This method was implemented in the MLP experiments.

for the regularized Newton updates considered here. This can be seen, for example, by applying Theorem 1 of (Dyer & Gur-Ari, 2020).

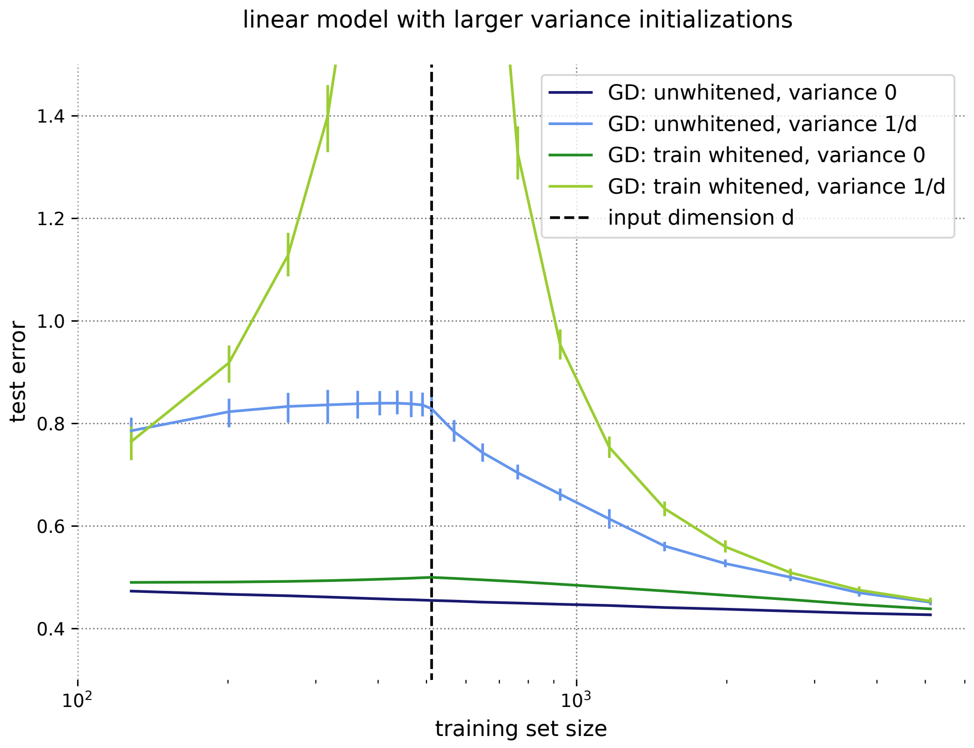

Figure App.3. **The effect of whitening on linear models with non-zero parameter initialization.** Linear models are initialized with parameter variances of 0 or $1/d$. In all cases the test loss is reported for the time during gradient flow training when the model achieves the lowest validation loss. Unwhitened data was scaled to have the same norm accumulated over all samples in the training set as whitened data, for each training set size, to avoid artifacts due to overall input scale. A model output of zero corresponds to a test loss of 0.5. All configurations with loss greater than 0.5 are doing *worse* than an uninformative prediction of 0. At both initialization scales, the model trained on whitened data performs worse than the model trained on unwhitened data for almost all dataset sizes, while for one dataset size they perform similarly. Data for the variance 0 initialization is identical to Fig. 3(a).

G.1.2. ZCA Whitening

ZCA (short for zero-phase components analysis) (Bell & Sejnowski, 1997) can be thought of as PCA whitening followed by a rotation back into the original basis. The ZCA whitening transform is $M = U\Sigma^{-1/2} \cdot V^\top$. ZCA whitening produces images that look like real images, preserving local structure. For this reason, it is used in the CNN experiments.

G.2. Linear Model

Dataset composition. The dataset for this experiment was a modified version of CIFAR-10, where the images were first processed by putting them through an off-the-shelf (untrained) four layer convolutional network with $\tanh$ nonlinearities and collecting the outputs of the penultimate layer. This resulted in a dataset of 512-dimensional images and their associated labels. Both the CIFAR-10 training and test sets were processed in this way. The linear estimator was trained to predict one-hot (ten dimensional) labels.

Training set sizes ranged from 128 to 5120 examples, randomly sampled from the preprocessed CIFAR-10 data. For experiments on unwhitened and train-whitened data, a validation set of 10000 images was split from the CIFAR-10 training set, and test error was measured on the CIFAR-10 test set. For experiments on fully whitened data, validation and test sets of 10 images each were split from the CIFAR-10 training and test sets, respectively.

Whitening. At each training set size, four copies of the data were made, and three were whitened using the PCA whitening method. For train-whitened data, the whitening transform was computed using only the training examples. For fully whitened data, the twenty validation and test images were included in the computation of the whitening transform. For

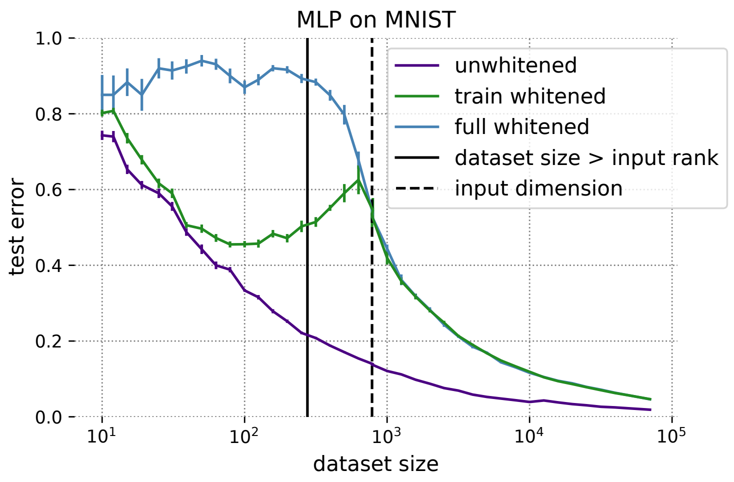

Figure App.4. **Whitening MNIST before training negatively impacts generalization in MLPs.** Models trained on fully whitened data (in blue) are unable to generalize until the size of the dataset exceeds its maximal input rank of 276, indicated by the solid black vertical line. Regardless of how the whitening transform is computed, models trained on whitened data (blue and green) consistently underperform those trained on unwhitened data (in purple).

distribution whitened data (Fig. App.2), the entire CIFAR-10 dataset of 60000 images (train as well as test) was used to compute the whitening transform.

Training and Measurements. We used a mean squared error loss function. Weights were initialized to all-zeros, except for the data in Fig. App.3, for which initial weights were drawn from a Gaussian with variance $1/d$. At each training set size, fifty models (initialized with different random seeds) were trained with full-batch gradient descent, with the optimization path defined by the gradient flow equation. Writing the model parameters as ϕ, this equation is

$$\phi(t) = \phi^* + e^{-tCB}(\phi^* - \phi^{(0)})$$

for preconditioner B, feature-feature correlation matrix C, infinite-time solution θ^*, and initial iterate $\theta^{(0)}$. In the case of gradient descent, the preconditioner B is simply the identity matrix.

In order to generate the plot data for Fig. 3(a), we solved the gradient flow equation for the parameters ϕ that achieved the lowest validation error, and calculated the test error achieved by those parameters. Mean test errors and their inner 80th percentiles across the twenty different initializations and across whitening states were computed and plotted. To make the plots in Fig. 4(a) and App.1, we tracked test performance over the course of training on unwhitened and train-whitened data.

On train-whitened datasets, we also implemented Newton's Method. This was done by putting the preconditioner B in the gradient flow equation equal to the inverse Hessian, i.e. $\left(XX^\top\right)^{-1}$. The preconditioner was computed once using the whole training set, and remained constant over the course of training. For experiments on whitened data, the data was whitened before computing the preconditioner.

G.3. Multilayer Perceptron

G.3.1. On MNIST

Architecture. We used a $784 \times 512 \times 512 \times 10$ fully connected network with a rectified linear nonlinearity in the hidden layers and a softmax function at the output layer. Initial weights were sampled from a normal distribution with variance 10^{-4}.

Dataset composition. The term "dataset size" here refers to the total size of the dataset, i.e. it counts the training as well as test examples. We did not use validation sets in the MLP experiments. Datasets of varying sizes were randomly sampled from the MNIST training and test sets. Dataset sizes were chosen to tile the available range (0 to 70000) evenly in $\log$ space. The smallest dataset size was 10 and the two largest were 50118 and 70000. For all but the largest size, the ratio of training to test examples was $8 : 2$. The largest dataset size corresponded to the full MNIST dataset, with its training set of 60000 images and test set of 10000 images.

The only data preprocessing step (apart from whitening) that we performed was to normalize all pixel values to lie in the range $[0, 1]$.

Whitening. At each dataset size, three copies of the dataset were made and two were whitened. Of these, one was train-whitened and the other fully whitened. PCA whitening was performed. The same whitening transform was always applied to both the training and test sets.

Training and Measurements. We used sum of squares loss function. Initial weights were drawn from a Gaussian with mean zero and variance 10^{-4}. Training was performed with SGD using a constant learning rate and batch size, though these were both modulated according to dataset size. Between a minimum of 2 and a maximum of 200, batch size was chosen to be a hundredth of the number of training examples. We chose a learning rate of 0.1 if the number of training examples was ≤ 50, 0.001 if the number of training examples was ≥ 10000, and 0.01 otherwise. Models were trained to 0.999 training accuracy, at which point the test accuracy was measured, along with the number of training epochs, and the accuracy on the full MNIST test set of 10000 images. This procedure was repeated twenty times, using twenty different random seeds, for each dataset. Means and standard errors across random seeds were calculated and are plotted in Fig. App.4.

For example, at the smallest dataset size of 10, the workflow was as follows. Eight training images were drawn from the MNIST training and two as test images were drawn from the MNIST test set. From this dataset, two more datasets were constructed by whitening the images. In one case the whitening transform was computed using only the eight training examples, and in another by using all ten images. Three copies of the MLP were initialized and trained on the eight training examples of each of the three datasets to a training accuracy of 0.999. Once this training accuracy was achieved, the test accuracy of each model on the two test examples, and on the full MNIST test set, was recorded, along with the number of training epochs. This entire procedure was repeated twenty times.

Computation of the input rank of MNIST data. MNIST images are encoded by 784 pixel values. However, the input rank of MNIST is much smaller than this. To estimate the maximal input rank of MNIST, at each dataset size (call it n) we constructed twenty samples of n images from MNIST. For each sample, we computed the 784×784 feature-feature second moment matrix F and its singular value decomposition, and counted the number of singular values larger than some cutoff. The cutoff was 10^{-5} times the largest singular value of F for that sample. We averaged the resulting number, call it r, over the twenty samples. If $r = n$, we increased n and repeated the procedure, until we arrived at the smallest value of n where $r > n$, which was 276. This is what we call the maximal input rank of MNIST, and is indicated by the solid black line in the plot in Appendix F.

G.3.2. On CIFAR-10

The procedure for the CIFAR-10 experiments was almost identical to the MNIST experiments described above. The differences are given here.

The classifier was of shape $3072 \times 2000 \times 2000 \times 10$ - slightly larger in the hidden layers and of necessity in the input layer.

The learning rate schedule was as follows: 0.01 if the number of training examples was ≤ 504, then dropped to 0.005 until the number of training examples exceeded 2008, then dropped to 0.001 until the number of training examples exceeded 10071, and 0.0005 thereafter.

The CIFAR-10 dataset is full rank in the sense that the input rank of any subset of the data is equal to the dimensionality, 3072, of the images.

G.3.3. Fig. 3(b), App.4 plot details

In Figs. 3(b) and App.4, for models trained on unwhitened data and train-whitened data, we have plotted test error evaluated on the full CIFAR-10 and MNIST test sets of 10000 images. For models trained on fully whitened data, we have plotted the errors on the test examples that were included in the computation of the whitening transform.

G.4. Convolutional Networks

G.4.1. WRN

Architecture. We use the ubiquitous Wide ResNet 28-10 architecture from (Zagoruyko & Komodakis, 2016). This architecture starts with a convolutional embedding layer that applies a 3×3 convolution with 16 channels. This is followed by three "groups", each with four residual blocks. Each residual block features two instances of a batch normalization layer, a convolution, and a ReLU activation. The three block groups feature convolutions of 160 channels, 320 channels, and 640 channels, respectively. Between each group, a convolution with stride 2 is used to downsample the spatial dimensions. Finally, global-average pooling is applied before a fully connected readout layer.

Dataset composition. We constructed thirteen datasets from subsets of CIFAR-10. The thirteen training sets ranged in size from 10 to 40960, and consisted of between 2^0 and 2^{12} examples per class. In addition, we constructed a validation set of 5000 images taken from the CIFAR-10 training set which we used for early stopping. Finally, we use the standard CIFAR-10 test set to report performance.

Whitening. We performed ZCA whitening using only the training examples to compute the whitening transform.

Training and Measurements. We used a cross entropy loss and the Xavier weight initialization. We performed full-batch gradient descent training with a learning rate of 10^{-3} until the training error was less than 10^{-3}. We use TPUv2 accelerators for these experiments and assign one image class to each chip. Care must be taken to aggregate batch normalization statistics across devices during training. After training, the test accuracy at the training step corresponding to the highest validation accuracy was reported. At each dataset size, this procedure was repeated twice, using two different random seeds. Means and standard errors across seeds were calculated and are plotted in Fig. 3(c).

G.4.2. CNN

Architecture. The network consisted of a single ResNet-50 convolutional block followed by a flattening operation and two fully connected layers of sizes 100 and 200, successively. Each dense layer had a $tanh$ nonlinearity and was followed by a layer norm operation.

Dataset composition. Training sets were of eleven sizes: 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, and 10240 examples. A validation set of 10000 images was split from the CIFAR-10 training set.

Training and Measurements. We used a cross entropy loss. Initial weights were drawn from a Gaussian with mean zero and variance 10^{-4}. Training was accomplished once with the Gauss-Newton method (see (Botev et al., 2017) for details), once with full batch gradient descent, and once with a regularized Gauss-Newton method. With a regularizer $\lambda \in [0, 1]$, the usual preconditioning matrix B of the Gauss-Newton update was modified as $\left((1-\lambda)B + \lambda I\right)^{-1}$. This method interpolates between pure Gauss-Newton ($\lambda = 0$) and gradient descent ($\lambda = 1$). In the Gauss-Newton experiments, we used conjugate gradients to solve for update directions; the sum of residuals of the conjugate gradients solution was required to be at most 10^{-5}.

For the gradient descent and unregularized Gauss-Newton experiments, at each training set size, ten CNNs were trained beginning with seven different initial learning rates: $2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}, 1, 4$, and 16. After the initial learning rate, backtracking line search was used to choose subsequent step sizes. Models were trained until they achieved 100% training accuracy. The model with the initial learning rate that achieved the best validation performance of the seven was then selected. Its test performance on the CIFAR-10 test set was evaluated at the training step corresponding to its best validation performance. The entire procedure was repeated for five random seeds. In Fig. 3(d), we have plotted average test and validation losses over the random seeds as functions of dataset size and training algorithm. In Fig. 4(c), we have plotted an example of the validation and training performance trajectories over the course of training for a training set of size 10240.

For the regularized Gauss-Newton experiment, the only difference is that we trained one CNN at each initial learning rate per random seed, and then selected the model with the best validation performance. In Fig. 5, we have plotted average metrics over the five random seeds. Errorbars and shaded regions indicate twice the standard error in the mean.