Supplementary Information for "The inverse variance-flatness relation in Stochastic-Gradient-Descent is critical for finding flat minima"

S1. THE TWO PHASES OF LEARNING: THE INITIAL FAST LEARNING PHASE AND THE SLOW EXPLORATION PHASE

The dynamics of L versus iteration time are shown in Fig. S1, which shows that there are two phases in learning in DNN. There is an initial fast learning phase where the overall loss function decreases quickly and sometimes abruptly followed by an exploration phase when the training error reaches 0 (or nearly 0) and the overall loss L still decreases but much more slowly and gradually. Due to its slow time scale, the exploration phase can be considered as in quasi-steady-state. These two phases exist independent of the hyperparameters and the network architecture as shown in Fig. S1, where the transition region between the two phases are highlighted.

The weights reached in the exploration phase can be considered as solutions of the problem given that the training error vanishes and the test error seems to reach a low steady state value in the exploration phase as shown in Fig. S2.

This transition can also be characterized by the amplitude and persistence of the mini-batch gradients. At a given time t, the SGD gradient is given by:

$$\vec{g}(t) = -\nabla L^{\mu(t)}, \tag{S1}$$

where $\mu(t)$ is the minibatch taken at time (iteration) t.

The amplitude $A_g(t)$ of $\vec{g}$ and the correlation $C_g(t)$ of its direction for two consecutive time are defined as follows :

$$A_g(t) \equiv ||\vec{g}(t)||, \tag{S2}$$

$$C_g(t) \equiv \vec{g}(t) \cdot \vec{g}(t + \Delta t)/[A_g(t)A_g(t + \Delta t)], \tag{S3}$$

where $\Delta t = 1$ is the iteration time step.

$A_g(t)$ and $C_g(t)$ can be used as order parameters of the learning system. As shown Fig. **??**, in the beginning of the learning process, the average $\langle A_g \rangle$ (over a moving window

of size $T_g = 1$ *epoch*) is relatively large and the average $\langle C_g \rangle$ is finite. This corresponds to the initial stage of learning when the gradient directions have some degree of alignment (finite C_g) and the overall error decreases quickly due to the large values of A_g. We call this the directed fast learning phase. As time goes on, the average $\langle A_g \rangle$ becomes smaller but stays at a small finite value while the average $\langle C_g \rangle$ becomes close to zero, which means that the gradient directions for different samples become almost orthogonal. We call this the exploration phase, where the cross-entropy (CE) loss is low and decreases slowly. Due to the finite size of the system, this transition is not infinitely sharp as phase transition in physical systems in thermodynamic limit (infinite system limit). As shown in Fig. **??**, the training error ϵ becomes zero during the transition regime and it stays zero in the exploration phase.

S2. PROPERTIES OF THE FIRST PRINCIPAL COMPONENT MODE

In our study, we used the cross entropy loss function for each sample:

$$d(\vec{Y}_k, \vec{Z}_k) = -\ln\big[\frac{\exp(\vec{Y}_k \cdot \vec{Z}_k)}{\sum_{n=1}^{n_c} \exp(Y_{k,n})}\big] = -\ln\big[\frac{\exp(Y_{k,n(k)})}{\sum_{n=1}^{n_c} \exp(Y_{k,n})}\big], \tag{S4}$$

where n_c is the total number of classes and also the dimension of the network output vector $\vec{Y}_k$ and the correct output vector $\vec{Z}_k$. The correct class for sample k is $n(k)$, and $\vec{Z}_k$ is just a unit vector in the correct ($n(k)$) direction. Let $m(k)$ be the incorrect output component with the largest output value, we can then define $\Delta Y_k \equiv Y_{k,n(k)} - Y_{k,m(k)}$. The cross entropy loss for sample k can then be written as:

$$l_k \equiv d(\vec{Y}_k, \vec{Z}_k) = \ln[1 + a_k \exp(-\Delta Y_k)] \approx a_k \exp(-\Delta Y_k), \tag{S5}$$

where $a_k = \sum_{n \neq n(k)} \exp(Y_{k,n} - Y_{k,m(k)})$ is an order O(1) number given that $Y_{k,m(k)}$ is the largest among all the incorrect outputs ($n \neq n(k)$).

As we described in the main text, the persistent drift in the first PC direction $\vec{p}_1$ can be understood by translating a solution $\vec{w}_0$ found by SGD along $\vec{p}_1$ by θ_1 to a new weight vector $\vec{w} = \vec{w}_0 + \theta_1 \times \vec{p}_1$. We studied how such a change of weights affects ΔY_k for a given sample k. We computed ΔY_k as a function of θ_1 for different samples, and found that the change in ΔY_k depends approximately linearly on θ_1 :

$$\Delta Y_k(\theta_1) \approx \Delta Y_k(0) + \beta_k \theta_1, \tag{S6}$$

where β_k is a sample dependent constant. This linear dependence is shown clearly in Fig. S3A for different samples covering different correct classes.

By using Eq. S6 in the expression for l_k, i.e., Eq. S5, we have:

$$l_k(\theta_1) \propto l_k(0) \times \exp(-\beta_k \theta_1), \tag{S7}$$

which depends exponentially on θ_1 for all samples k. As a result, the overall loss function along the first PC direction:

$$L_1 \equiv \langle l_k(\theta_1) \rangle_k \approx L_0 \exp(-\beta\theta_1 + \beta_2 \theta_1^2), \tag{S8}$$

where β is the average of β_k: $\beta = \langle \beta_k \rangle_k > 0$, and β_2 depends on the variation of β_k among different samples. The expression for $L_1(\theta_1)$ agrees with the one obtained from the MLF ensemble average derived in the main text with the correspondences: $\beta = M_{11}^{(0)} \theta_1^{(0)}$ and $\beta_2 = M_{11}/2$.

The functional form of $\ln(L_1(\theta_1))$ expressed in Eq.S8, i.e., $\ln(L_1(\theta_1)) \approx \ln(L_1(0)) - \beta\theta_1 + \beta_2\theta_1^2$ agrees with our numerical results shown in Fig. S3B, which shows that $\ln(L_1)$ has a finite slope at $\theta_1 = 0$. It is this finite slope that drives the drift motion of θ_1 observed in Fig. 1C in the main text.

Finally, we find that $\vec{p}_1$ is highly aligned with $\vec{w}_0$: $\frac{\vec{p}_1}{|\vec{p}_1|} \cdot \frac{\vec{w}_0}{|\vec{w}_0|} \approx 0.91$, which is demonstrated in Fig. S3C where all the components of $\vec{p}_1$ and $\vec{w}_0$ are plotted against each other. Correspondingly, the coefficient β_k is proportional to $\Delta Y_k(0)$ for different samples k as shown in Fig. S3D. Therefore, moving along $\vec{p}_1$ results roughly in an overall amplification of the solution weight vector $\vec{w}_0$ and the corresponding amplification of the output difference (decision gap) ΔY_k.

S3. THE RELATIONSHIP BETWEEN FLATNESS AND THE HESSIAN MATRIX

As we stated in the main text, our definition of the loss landscape flatness in certain direction i is the size of the region (or the range of θ_i) within which the loss function is smaller than $L_0 \times e$ where e is the natural log base. The motivation for this definition of flatness is that F_i characterizes the size of the region near a minimum (in the PCA direction i) where the loss function is also small (within a factor of e). As defined, F_i contains non-local

properties of loss function landscape as it depends on the loss function landscape in a finite neighborhood of the minimum, whereas Hessian matrix only describes the local curvature of the landscape at the minimum.

The flatness F_i is connected with the Hessian, but not always the same. To see this, we can expand the log of the loss function around its minimum:

$$\ln(L_i) = \ln(L_0) + \frac{\kappa_i}{2}\delta\theta_i^2 + h_i(\delta\theta_i),$$

where κ_i is the eigenvalue of the Hessian matrix and $h_i(\delta\theta_i)$ contains all the higher order terms. If the higher order terms can be neglected in the finite regime $F_i/2 \geq \delta\theta_i \geq -F_i/2$, then the flatness F_i is directly related to the Hessian eigenvalue κ_i: $F_i = \sqrt{\frac{8}{\kappa_i}}$. However, when κ_i is small or zero, the flatness parameter F_i is not determined solely by the local curvature κ_i, it also depends on the higher order terms. For example, take the case when $\ln(L_i) = \ln(L_0) + \delta\theta_i^4/(\epsilon^2 + \delta\theta_i^2)$, with $\epsilon \ll 1$, we have $\kappa_i = 0$, but $F_i \approx 2$.

To demonstrate their connection and difference, we show the eigenvalues κ_i of the Hessian and $8/F_i^2$ where F_i is defined and determined independently from our analysis of the loss landscape. As we can see from Fig. S4 shown here, the agreement is good for large values of κ_i, however, when κ_i is small or even a small negative value (due to numerical inaccuracy and sampling noise), the flatness remains well defined and finite. Overall, the flatness parameter is a more robust non-local measure of the loss landscape.

S4. STATISTICAL PROPERTIES OF THE MLF ENSEMBLE

In the random landscape theory, the MLF L^μ is approximated by an inverse Gaussian function:

$$\begin{aligned} L^\mu(\vec{\theta}) &\approx L^\mu_{min} \exp[\sum_{i,j=1}^{N} \frac{M^\mu_{ij}}{2}(\theta_i - \theta^\mu_i)(\theta_j - \theta^\mu_j)] \\ &= L^\mu_0 \exp[\sum_i \frac{M^\mu_{ii}}{2}\theta_i(\theta_i - 2\theta^\mu_i) + \sum_{i<j} M^\mu_{ij}(\theta_i\theta_j - \theta_i\theta^\mu_j - \theta_j\theta^\mu_i)], \end{aligned} \tag{S9}$$

where $\vec{\theta^\mu}$ is the location of the minimum for MLF L^μ, $\theta_i = \vec{\theta}\cdot\vec{p_i}$ is the parameter vector projected onto the i-th PC direction, L^μ_{min} is the minimal loss, and $\mathbf{M}^\mu = \{M^\mu_{ij}\}$ is the symmetric Hessian matrix for $\ln(L^\mu)$ at its minimum location $\vec{\theta^\mu}$. For convenience, we define $L^\mu_0 \equiv L^\mu(0) = L^\mu_{min}\exp[\sum_{i,j=1}^{N} M^\mu_{ij}\theta^\mu_i\theta^\mu_j]$ to represent the loss function value for minibatch μ at $\vec{\theta} = 0$, and $L_0 \equiv \langle L^\mu_0\rangle_\mu$ is the minimum loss of the overall loss function L.

To simply the theory, we make the mean-field approximation that M^{μ}_{ij}, M^{μ}_{ii}, and θ^{μ}_{i} are uncorrelated random variables, which is supported by direct simulations. Furthermore, for convenience of analytical treatment, we assume that these random variables follow Normal distributions: $M^{\mu}_{ii} \sim \mathcal{N}(M^{(0)}_{ii}, \sigma^2_{M,i})$ with positive mean $M^{(0)}_{ii} > 0$, $M^{\mu}_{ij} \sim \mathcal{N}(0, \sigma^2_{ij})$ with zero mean for $i \neq j$, $\theta^{\mu}_{i} \sim \mathcal{N}(0, \sigma^2_{\theta,i})$ with zero mean for the diffusive modes ($i \geq 2$), and $\theta^{\mu}_{1} \sim \mathcal{N}(\theta^{(0)}_{1}, \sigma^2_{\theta,1})$ with a finite mean $\theta^{(0)}_{1} (\neq 0)$ for the drift mode ($i = 1$), which allows us to obtain the overall loss function (up to the second order terms in θ_i) by averaging over the distributions of $\mathbf{M}^{\mu}$ and $\vec{\theta}^{\mu}$:

$$\begin{aligned} L \equiv \langle L^{\mu}(\vec{\theta})\rangle_{\mu} &\approx L_0 \big\langle \exp\{\sum_i \frac{M^{\mu}_{ii}}{2}\theta_i(\theta_i - 2\theta^{\mu}_i) + \sum_{i<j} M^{\mu}_{ij}(\theta_i\theta_j - \theta_i\theta^{\mu}_j - \theta_j\theta^{\mu}_i)\}\big\rangle_{\mathbf{M}^{\mu},\vec{\theta}^{\mu}} \\ &\approx L_0 \exp\{-M^{(0)}_{11}\theta^{(0)}_1\theta_1 + \sum_i [\frac{M^{(0)}_{ii}}{2} + \frac{1}{2}\sigma^2_{\theta,i}(M^{(0)}_{ii})^2]\theta^2_i\} \times \prod_i (1-\Gamma_i)^{-1/2} \\ &\approx L_0 \exp(\beta\theta_1 + \sum_i \frac{M_{ii}}{2}\theta^2_i), \end{aligned} \tag{S10}$$

which is the same as Eq. 10 in the main text. The constant $\beta = -M^{(0)}_{11}\theta^{(0)}_1$ is related to the finite drift velocity in the first PCA direction ($i = 1$): $v^{(0)}_1 \equiv \langle v_1\rangle_{\mu} \approx -\alpha L_0 \beta$. The flatness $F_i \equiv (8/M_{ii})^{1/2}$ is determined by the statistical properties of the MLF ensemble:

$$M_{ii} = M^{(0)}_{ii} + \sigma^2_{\theta,i}(M^{(0)}_{ii})^2 + \sigma^2_{\theta,i}\sigma^2_{M,i} + \sum_{j\neq i}\sigma^2_{ij}[\sigma^2_{\theta,j} + \delta_{j1}(\theta^{(0)}_1)^2].$$

For each MLF L^{μ}, if we vary $\vec{w} = \vec{w}_0 + \theta_i\vec{p}_i$ along a given PC-direction $\vec{p}_i$, we obtain the MLF profile along the i-th PC direction consistent with those shown in Fig. 3A in the main text obtained from direct simulations:

$$L^{\mu}_i(\theta_i) \propto \exp[\frac{M^{\mu}_{ii}}{2}(\theta_i - \theta^{\mu}_i)^2 - (\sum_{j\neq i} M^{\mu}_{ij}\theta^{\mu}_j)(\theta_i - \theta^{\mu}_i)] \propto \exp[\frac{M^{\mu}_{ii}}{2}(\theta_i - \tilde{\theta}^{\mu}_i)^2], \tag{S11}$$

which has a minimum at $\theta_i = \tilde{\theta}^{\mu}_i$ that is shifted from θ^{μ}_i due to the random off-diagonal Hessian matrix elements:

$$\tilde{\theta}^{\mu}_i = \theta^{\mu}_i + \frac{1}{M^{\mu}_{ii}}\sum_{j\neq i} M^{\mu}_{ij}\theta^{\mu}_j. \tag{S12}$$

It is easy to show that $\tilde{\theta}^{\mu}_i$ has zero mean ($\langle\tilde{\theta}^{\mu}_i\rangle_{\mu} = 0$) and a variance given by:

$$\tilde{\sigma}^2_{\theta,i} \equiv \langle(\tilde{\theta}^{\mu}_i)^2\rangle_{\mu} = \sigma^2_{\theta,i} + \langle(M^{\mu}_{ii})^{-2}\rangle_{\mu}\sum_{j\neq i}\sigma^2_{ij}\sigma^2_{\theta,j} \approx \sigma^2_{\theta,i} + \frac{1}{(M^{(0)}_{ii})^2}\sum_{j\neq i}\sigma^2_{ij}\sigma^2_{\theta,j}, \tag{S13}$$

where we have used the approximation $\langle (M_{ii}^{\mu})^{-2} \rangle_{\mu} \approx \frac{1}{(M_{ii}^{(0)})^2}$.

We have tested the assumptions in our random landscape theory by direct numerical calculation of the distributions of θ_i^{μ}, the diagonal Hessian matrix element M_{ii}^{μ}, the off-diagonal Hessian element M_{ij}^{μ} $(i \neq j)$, and their correlations. As shown in Fig. S5, the distributions for θ_i^{μ}, and the off-diagonal Hessian element M_{ij}^{μ} $(i \neq j)$ can be approximated well by Gaussian distributions with zero mean. The distribution for M_{ii}^{μ} is a little skewed probably because of its non-zero mean value. More importantly, no significant correlation is found between these random variables, which verifies the key assumption made in our paper that leads to the form of the overall loss function given in Eq. 10 in the main text.

Furthermore, both $M_{ii}^{(0)}$ and $\tilde{\sigma}_{\theta,i}^2$ can be determined numerically for different $i \geq 2$ (different PC directions). As shown in Fig. S6, $\tilde{\sigma}_{\theta,i}$ scales almost linearly with $F_i = (8/M_{ii})^{1/2}$ and $M_{ii}^{(0)}$ scales inversely with F_i, approximately as F_i^{-2}. As a result, we have the diffusion constant $D_i \propto (M_{ii}^{(0)})^2 \tilde{\sigma}_{\theta,i}^2 \propto F_i^{-2}$ as shown in Fig. 3B in the main text.

S5. THE CORRELATION IN SGD VELOCITY FLUCTUATIONS

As we show in the main text, the dynamics of the PCA components can be described by the stochastic equation (for $i \geq 2$):

$$\frac{d\theta_i}{dt} = v_i(t), \tag{S14}$$

where $v_i(t) \approx -\alpha \frac{\partial \delta L^{\mu}(t)}{\partial \theta_i}$ with $\mu(t)$ the minibatch used at time t.

The correlation function of v_i can be defined as:

$$C_i(t) \equiv \langle v_i(t+t') v_i(t') \rangle = D_i c_i(t), \tag{S15}$$

where $c_i(t) = C_i(t)/D_i$ is the normalized correlation function with $D_i = C_i(0)$ and $c_i(0) = 1$. The finite values of $c_i(t)$ for $t \geq \Delta t$ ($\Delta t = 1$ to set the time unit for one iteration step) describes the additional correlation with a finite correlation time beyond those captured by $C_i(0)$, i.e., the equal time velocity correlation or equivalently the velocity variance.

From the normalized velocity-velocity correlation $c_i(t)$, we can define an integrated correlation function:

$$G_i(t) = \int_{-t}^{t} c_i(t') dt', \tag{S16}$$

which is a symmetric function of t (the same is true for $c_i(t)$).

The weight variable $\theta_i(t) = \int_0^t v_i(t')dt'$, and the variance of the weight is given by:

$$\begin{aligned} \sigma_i^2 &\equiv \langle\theta_i^2\rangle - \langle\bar{\theta}_i^2\rangle \\ &= T_w^{-1}\int_0^{T_w}\langle\theta_i^2(t)\rangle dt - T_w^{-2}\langle(\int_0^{T_w}\theta_i(t)dt)^2\rangle, \end{aligned} \tag{S17}$$

where T_w is the PCA window size and $\bar{\theta}_i = T_w^{-1}\int_0^{T_w}\theta_i(t)dt$ is the average weight. Since PCA was done by using the weights shifted by their mean, we have $\bar{\theta}_i = 0$.

By defining $S_i(t) \equiv D_i^{-1}\langle\theta_i^2(t)\rangle$, we have:

$$\begin{aligned} S_i(t) &= D_i^{-1}\int_0^t\int_0^t\langle v_i(t')v_i(t'')\rangle dt'dt'' \\ &= \int_0^t\int_0^t c_i(t'-t'')dt'dt'' \\ &= \int_0^t dt_1\int_{-t_1}^{t_1} c_i(t_2)dt_2, \\ &= \int_0^t G_i(t_1)dt_1, \end{aligned} \tag{S18}$$

where a change of integration variables: $t_1 = t' + t''$, $t_2 = t' - t''$ has been used.

As shown in Fig. S7 below, as time t increases, $G_i(t)$ decreases with time and reaches zero at a time scale τ_i, which defines the correlation time of v_i. Quantitatively, the weight variance within the PCA time window is given by $\sigma_i^2 = D_i T_w^{-1}\int_0^{T_w} S_i(t)dt \equiv D_i\Delta t\tau_i$ and the correlation time scale τ_i is given by:

$$\tau_i \equiv T_w^{-1}\Delta t^{-1}\int_0^{T_w}\Big[\int_0^t\int_0^t c_i(t'-t'')dt'dt''\Big]dt = T_w^{-1}\Delta t^{-1}\int_0^{T_w} S_i(t)dt = T_w^{-1}\Delta t^{-1}\int_0^{T_w}\int_0^t G_i(t_1)dt_1dt, \tag{S19}$$

which can be calculated by determining the functions $G_i(t)$ and $S_i(t)$ from the correlation function $C_i(t)$. Here, we introduce $\Delta t = 1$ explicitly in the definition(expression) for τ_i to make sure it has the dimension of a time scale.

As shown in Fig. S8, we find that τ_i decreases with F_i roughly as a power law:

$$\tau_i \sim F_i^{-\psi_\tau},$$

with an exponent $\psi_\tau \approx 1.8$. Taken this new result together with the dependence of D_i on F_i (Fig. S8, which is also shown as Fig. 3B in the main text), we can explain the steeper inverse dependence of weight variance on the landscape flatness.

S6. GENERALITY OF THE INVERSE VARIANCE-FLATNESS RELATION

As shown in the main text (Fig. 2D), the inverse relation between the SGD variance and the flatness of the loss function landscape is the same for different choices of the hyper-parameters, batch size B and learning rate α. Since all the results reported in the main text are from a simple network with 2 hidden layers applied to the MNIST dataset, we have tested the robustness of this inverse variance-flatness relation in different networks, for different algorithms and datasets. We first tested the variance-flatness relation in a DNN with a larger number of hidden layers, e.g., 4 hidden layer network as shown in Fig. S9A. We found the same inverse variance-flatness relation for weights in all different layers as shown in Fig. S9B. We have also tested the inverse variance-flatness relationship in other DNN architectures such as LeNet (illustrated in Fig. S9C), different learning algorithms such as ADAM, and different dataset such as CIFAR10. As shown in Fig. S9D, the inverse variance-flatness relation holds true in all these cases we studied.

We have tested the dependence of D_i and τ_i on F_i for more complex networks such as network with multiple-intermediate-layers and a CNN network for the CIFAR10 datasets. As shown in Figs. S10&S11, the inverse dependence of τ_i and D_i on F_i seem general as they also hold true for all the cases we studied.

S7. PROPERTIES OF THE CNN FILTERS IN DIFFERENT PCA DIRECTIONS

We have investigated properties of CNN filters in different PCA directions by using a CNN network described in the Method section in the main text on the MNIST dataset. In this network, the first convolution layer has 16 filters (labeled by $k \in [1, 16]$), and each filter has 9 (3×3) weights (labeled by $l \in [1, 9]$), so the weights in this layer can be arranged in a 16×9 weight matrix $w(l, k)$.

We did PCA for the weight fluctuations around a solution $w^*(k, l)$. Each principle component (PC) direction i in the weight space can be characterized by a PCA weight matrix $w_i(k, l)$ (note that the PCA weight matrices are orthogonal to each other: $\sum_{k=1}^{9} \sum_{l=1}^{16} w_i(k, l) w_j(k, l) = \delta_{ij}$). We first checked the alignment of the solution w^* with the PCA weight matrices by computing the projected magnitude p_i defined as: $p_i \equiv ||w^* \cdot w_i||^2 = [\sum_{k,l} w^*(k, l) w_i(k, l)]^2$. As shown in Fig. S12 (upper panel), there is

no clear evidence of whether w^* is aligned more with the flat directions (larger i) or the sharp directions (smaller i). We have also studied the PCA weight matrices $w_i(k,l)$ themselves. As shown in Fig. S12 (lower panels), the weight matrices in the sharper directions (smaller i) seem to have elements that are more evenly distributed while elements in the PCA weight matrices in the flatter directions with higher values of i (e.g., $i = 130$, 144) are only sparsely distributed. We do not fully understand the origin and possible implications of this observation. It seems to suggest that the PCA weights along sharp directions tend to capture certain global characteristics of the picture while those in the flatter directions depend more on local features. Further studies are needed to verify and understand these observations.

S8. ROBUSTNESS OF THE LDC ALGORITHM

In the landscape-dependent constraints (LDC) algorithm proposed in our paper, we can determine the flatness F_i efficiently from the variance σ_i by using the inverse variance-flatness relationship discovered in this study, i.e., $F_i^{-2} \propto \sigma_i^{4/\psi}$. However, although the inverse variance-flatness relation is robust, the exact value of the power law exponent or the power-law form of the inverse dependence itself is not universal for different layers in the network and for different network architectures. Fortunately, the general results of the algorithm do not seem to depend sensitively on the exponent as we show below (see Fig. S13) for three different choices of $\psi = 3, 4, 5$ (note $\psi = 4$ is what we used in the main text).

S9. APPLICATION OF THE LDC ALGORITHM IN MORE COMPLEX CASES

To demonstrate the landscape-dependent constraints (LDC) algorithm in preventing catastrophic forgetting, we have applied it to problems that are harder than those shown in the main text: 1) learning two groups of 5 digits: $(0, 1, 2, 3, 4)$ and $(5, 6, 7, 8, 9)$ sequentially without forgetting; 2) sequentially learning all the animals and all man-made objects in CIFAR10. The results shown in Fig. S14 are better than those by using EWC. In general, the results from LDC are better than those from EWC.

The detailed findings are also interesting. For example, for the 5 digit learning task in MNIST, the minimum number of constraints $N_c^* = 600$ which is larger than $N_c^* = 200$

reported in the main text for a simpler case of learning two digits. This means that for the same architecture, the network needs more components to store the information when tasks become harder. For the CIFAR dataset, we used a small CNN (described in the Methods section) for convenience of analysis. In this small CNN, the best average accuracy is around 70% on CIFAR10, which is lower than the current state-of-the-art performance. However, it is not our goal to improve the performance of a single task. Here, we showed that by using LDC the average accuracy is around 60% after learning two tasks in sequence, which demonstrates the usefulness of LDC because without it the previous task would be totally forgotten with an accuracy $\sim 25\%$ close to random choice.

S10. THE FLATNESS-DETECTING NOISE (FDN) IMPROVES BATCH LEARNING

To test our "active temperature" hypothesis for the SGD based learning algorithm, we developed a new batch learning algorithm where a "flatness detecting" noise is introduced to the deterministic gradient descent (GD) learning dynamics. In particular, we have added an anisotropic noise term whose strength depends explicitly on the flatness of the landscape. We describe this "flatness detecting noise" (FDN) method in the following.

First, we define the gradient matrix $\frac{\partial L^\mu}{\partial w_i} = Q_{i\mu}$ and its correlation matrix F:

$$F = \frac{1}{N_m} Q Q^T, \tag{S20}$$

where N_m is the number of minibatches used in computing F.

When the minibatch size $B = 1$ and $N_m = M$ the total number of training samples , F is exactly the Empirical Fisher Information, which is the same as the Hessian matrix at a local minimum. Let $\vec{\theta_i}$ and e_i denote the eigenvector and eigenvalue of F respectively, the dynamics of the weights in FDN is given by:

$$\frac{d\vec{w}}{dt} = -\alpha(\frac{\partial L}{\partial \vec{w}} + \sum_{i=1}^{N_c} \sqrt{e_i}\vec{\theta}_i\tilde{\eta}), \tag{S21}$$

where $\tilde{\eta} \sim \mathcal{N}(0, \sigma_n)$ is a Gaussian white noise with the variance σ_n that characterize the overall noise strength. Since e_i depends on the curvature of the loss landscape in the direction $\vec{\theta_i}$, a smaller noise is added to a flatter direction and a larger noise is added to a steeper direction in the FDN method (Eq. S21).

We compare the performance of the original gradient descent (GD) algorithm, the gradient descent method with the flatness detecting noise (Eq. S21), which we call GD$^{\eta*}$, GD with a constant noise, which we call GD^{η}, and SGD. We use a small fully connected network (input layer: 784 neurons, hidden layer 1: 35 neurons, hidden layer 2: 35 neurons, output layer: 10 neurons) and train the network on MNIST data set. In order to speed up the training process, we use $B = 50$,$N_m = 2M/B = 2400$ and only add noise on the first three hundred components ($N_c = 300$) for each layer. The F matrix is updated for every 50 epochs.

As shown in Fig. S15, for GD, the loss decreases smoothly and it has a higher test error (3.27%) at the end. The loss in SGD fluctuates as it decreases and it seems to go through a "phase transition" before it reaches a steady state, where even though the training loss is similar to that reached by GD but the test error (2.54%) is lower. In the GD$^{\eta*}$ algorithm where we add the "flatness detecting noise" to GD, similar phase transition occurs that allows GD$^{\eta*}$ to reach a low test error(2.41%) basin. It is important to note that *only* the flatness-detecting noise, i.e., anisotropic noise whose strength in a given direction depends inversely on the flatness of the loss landscape in that direction, works. When we added an isotropic constant noise to GD, i.e., GD$^{\eta}$, the performance is not improved.

These results based on the GD$^{\eta*}$ algorithm with engineered flatness-detecting noise strongly support the conclusion in our study that the anisotropic landscape-dependent noise in SGD is responsible for driving the system to the flat minima where there is better generalization.

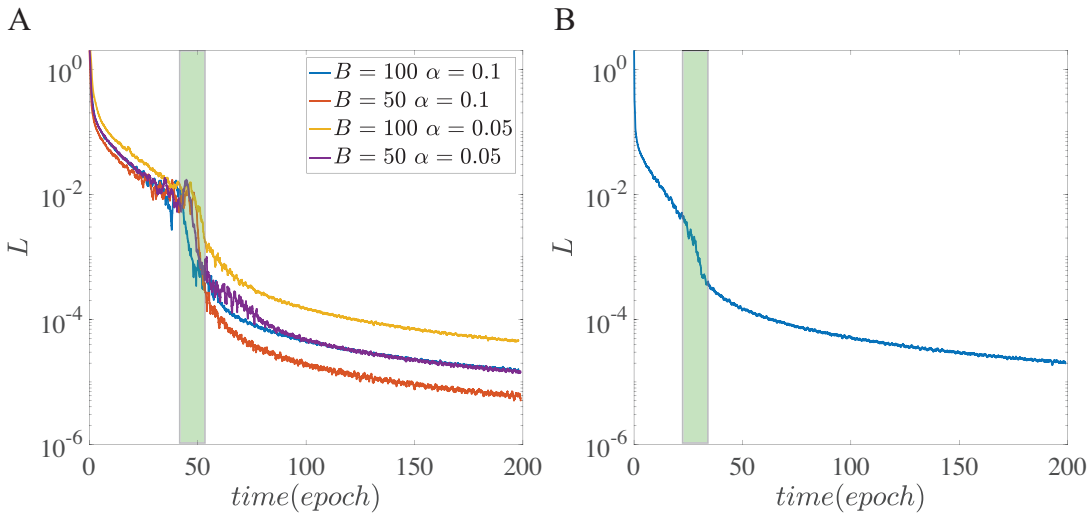

FIG. S1: Two different phases in learning. (A) shows the phase transition in a multi-layer fully-connected neural network. Here we use a network with four hidden layers and each hidden layer has 50 units. The experiment is done on the MNIST data sets. We can see that there is an obvious fast drop of cross-entropy loss around 50 epochs after which the system enters the exploration phase where the loss is low and changes only slowly. This phase transition holds true for different combination of learning rate and batch size. (B) shows the phase transition in a LeNet, which has two convolution layers with size $1 \times 3 \times 3 \times 16$ and $16 \times 5 \times 5 \times 32$, and one fully-connected neural network with size 1586×10 (see Fig. S9C). We can see that there is also a drop of cross-entropy loss around 30 epochs

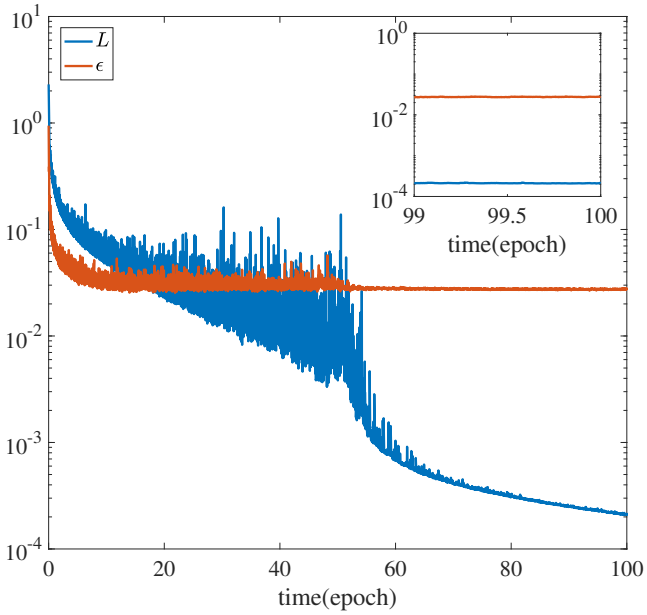

FIG. S2: The cross-entropy (CE) loss and testing error versus training time. The inset shows that the test error reaches a steady-state low value in the exploration phase while the CE changes very slowly.

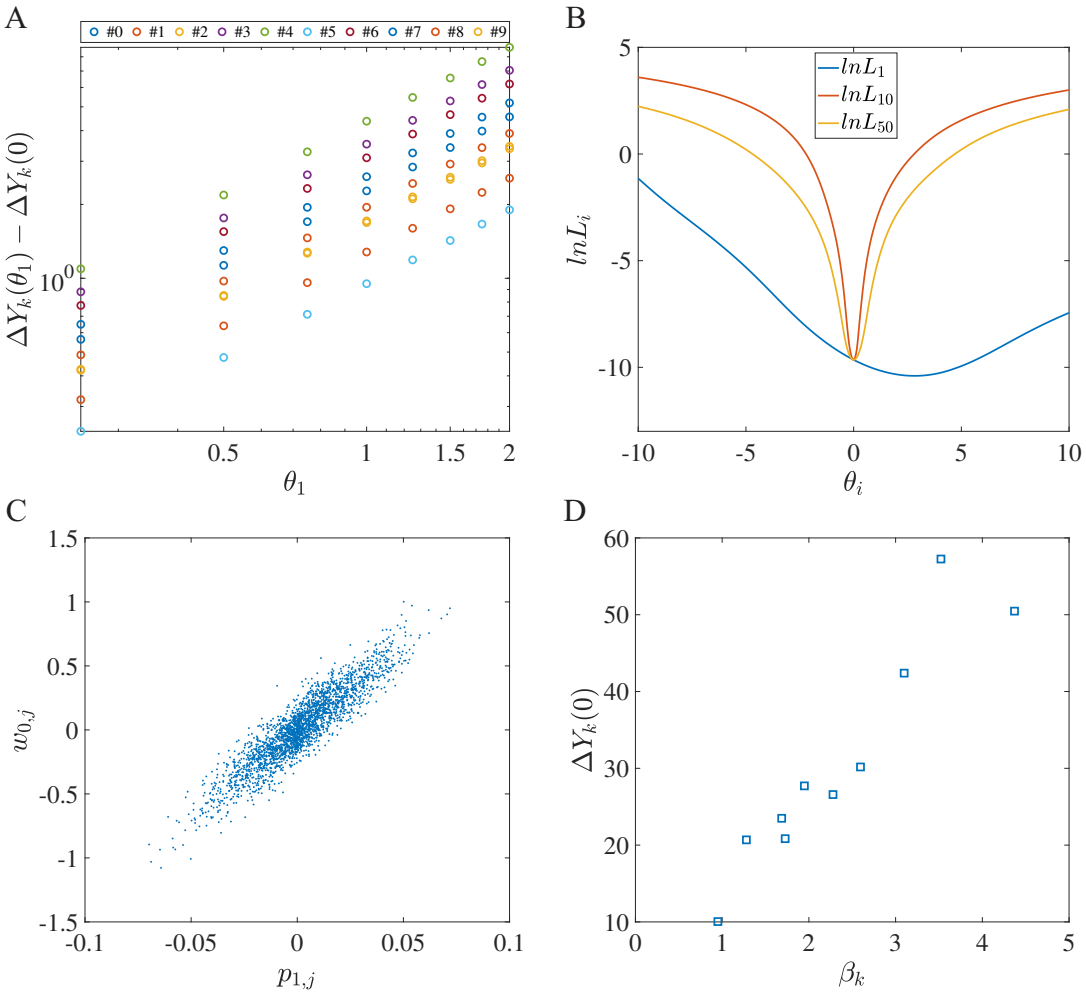

FIG. S3: The effect of changing θ_1 along the first PC direction $\vec{p}_1$. (A) The dependence of ΔY_k, the difference between the correct output and the maximum incorrect output, on the displacement θ_1 along the first PC for different sample k. ΔY_k increases linearly with θ_1. Each symbol corresponds to an average over 5 samples of the same digit. (B) The landscape of $\ln L_i(\theta_i)$ along the i-th PC direction. Only $\ln(L_1)$ has a finite gradient at $\theta_1 = 0$, which indicates a drift motion in $\vec{p}_1$. In all other PC directions ($i \geq 2$), $\ln(L_i)$ has a zero gradient at $\theta_i = 0$, which indicates a diffusive motion in these directions. (C) The components $p_{1,j}$ of $\vec{p}$ versus the corresponding components $w_{0,j}$ of $vecw_0$ for all $j = 1, 2, ..., 2500$ weight components between the two hidden layers. This shows that the two weight vectors $\vec{p}_1$ and $\vec{w}_0$ are highly aligned. (D) The coefficient β_k versus $\Delta Y_k(0)$ for different samples k shown in (A).

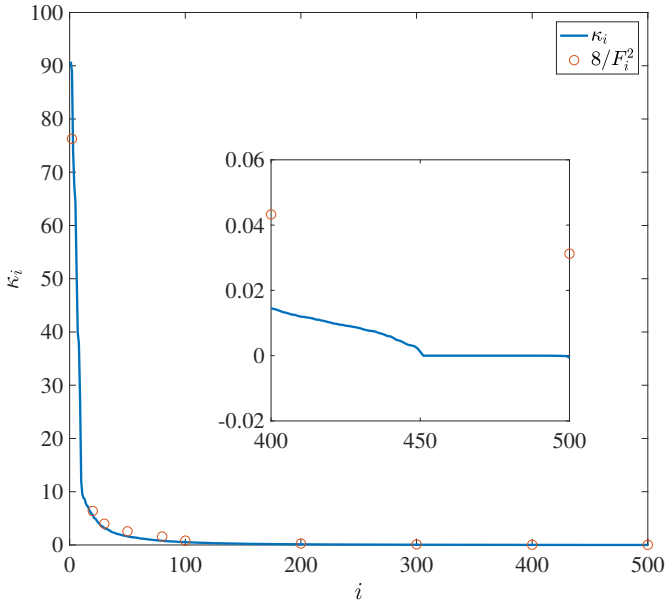

FIG. S4: This figure shows the comparison between flatness and eigenvalues of Hessian matrix. Here the eigenvalue of Hessian matrix and flatness are calculated from the last layer of fully connect network with 500 parameters.For the lower rank modes, Hessian eigenvalue (κ_i) is the same as the curvatures calculated from flatness ($8/F_i^2$). For the higher rank modes, the eigenvalues can reach zero and even become negative, but M_{ii} still remain positive (see inset) because F_i is a measure of the landscape flatness in a finite region near the minimum.

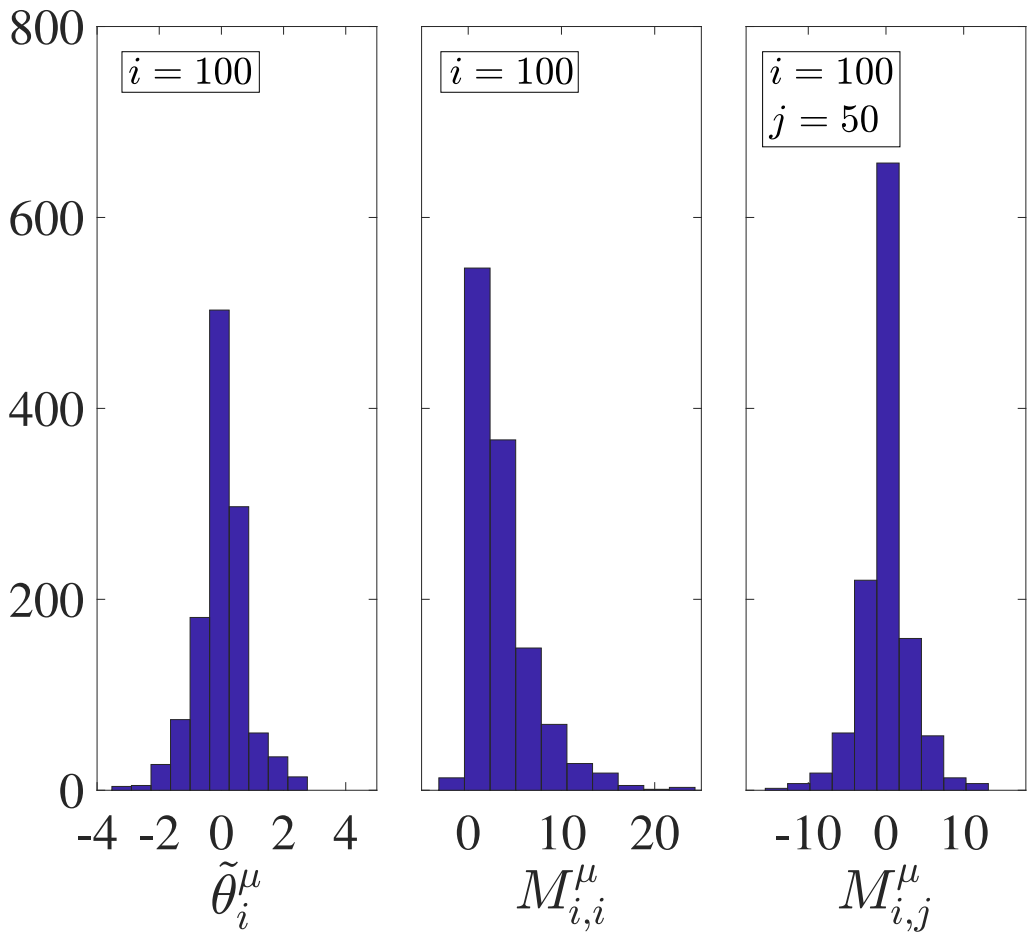

FIG. S5: The distributions of θ_i^μ, the diagonal Hessian matrix element M_{ii}^μ, the off-diagonal Hessian element M_{ij}^μ $(i \neq j)$. M_{ii}^μ and M_{ij}^μ are obtained by taking the second derivatives of the MLF (L^μ) at a given solution in the exploration phase.

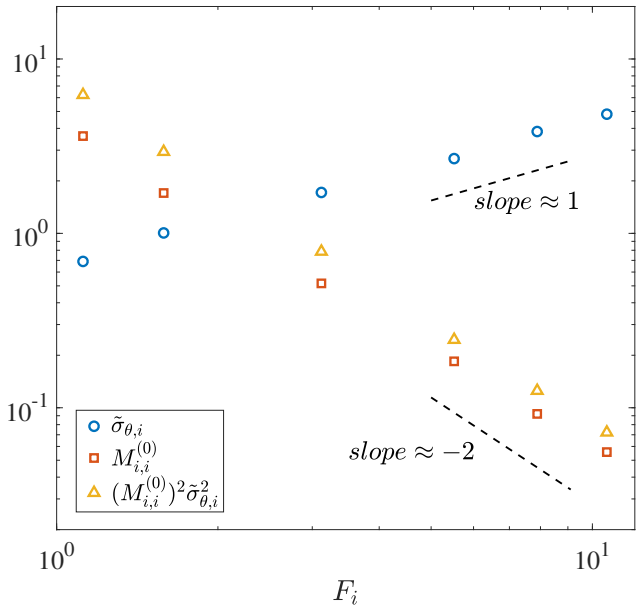

FIG. S6: The variance $\tilde{\sigma}_i^2$ of the minimum positions and the average diagonal element $M_{ii}^{(0)}$ of the Hessian matrices of the MLF ensemble versus the flatness F_i of the overall loss function. The diffusion constant is $D_i \propto (M_{ii}^{(0)})^2\tilde{\sigma}_i^2$, and the combination $(M_{ii}^{(0)})^2\tilde{\sigma}_i^2$ versus F_i is also shown. $i = 30$ used here.

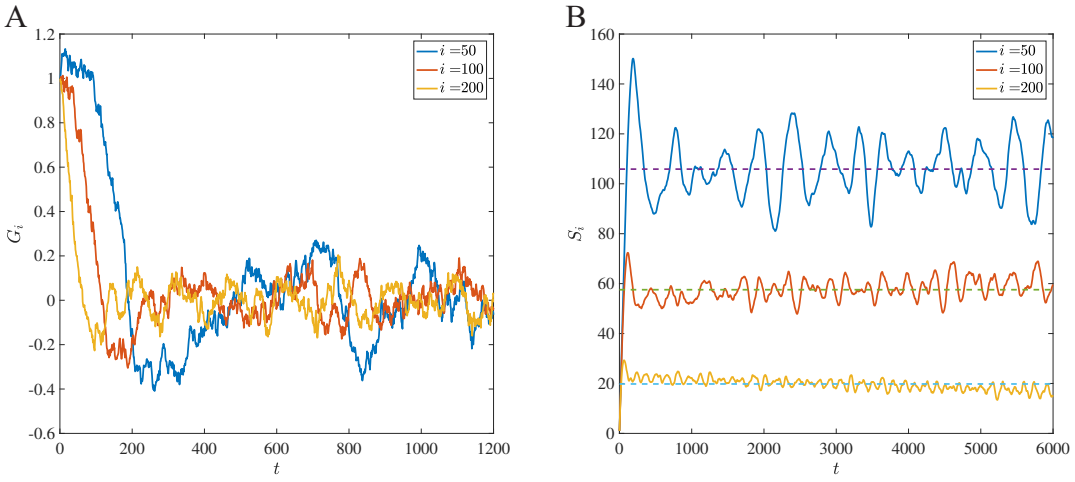

FIG. S7: (A) $G_i(t)$ and (B) $S_i(t)$ versus training time $t$for different principal components. The dotted lines in (B) indicate the values of τ_i.

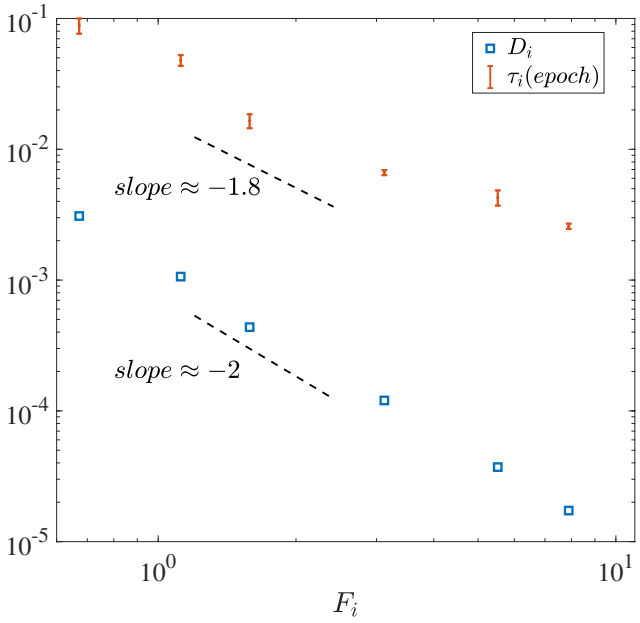

FIG. S8: The inverse dependence of τ_i (in unit of epoch) and D_i on the flatness F_i for the 2-hidden-layer NN on MNIST used in the main text. Each epoch has $1,200$ minibathes.

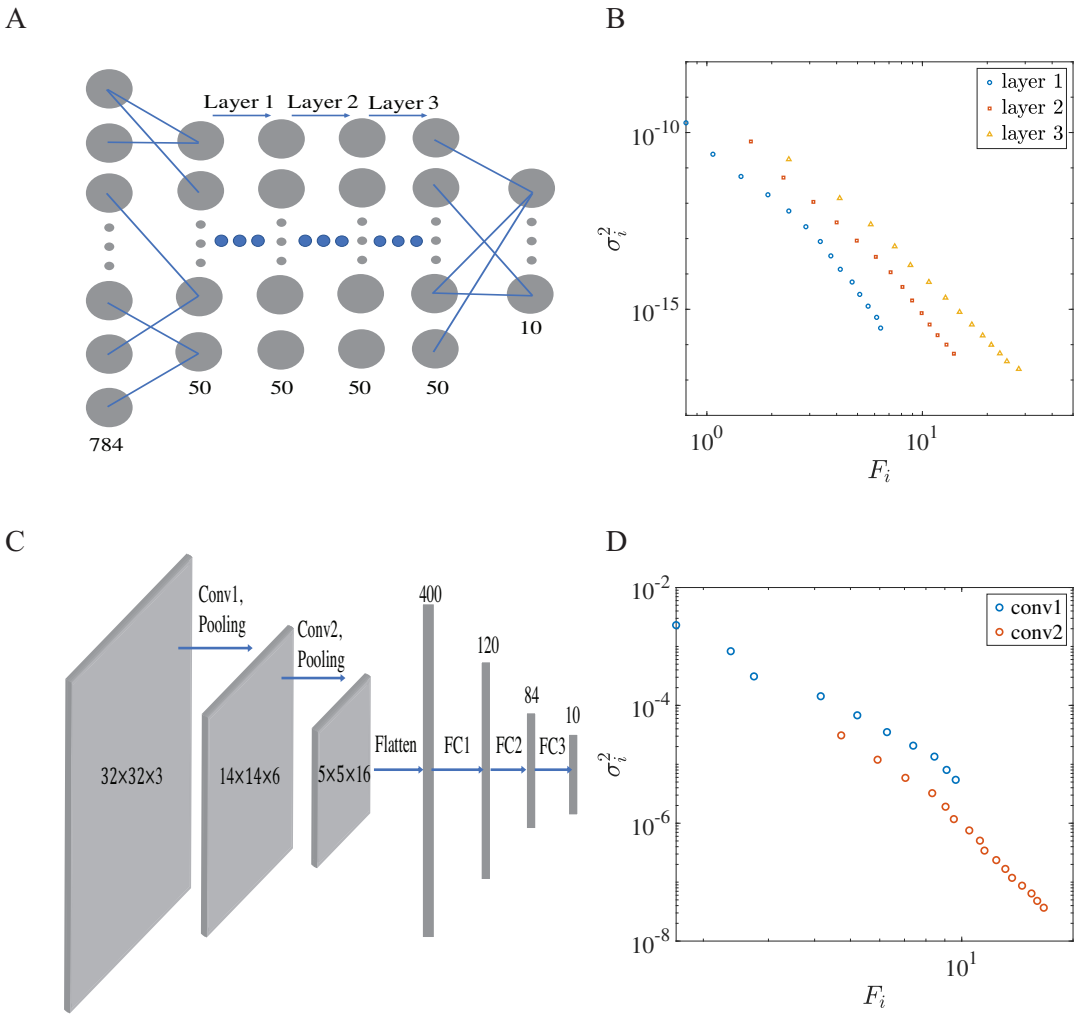

FIG. S9: The variance-flatness relation for different network, data set and optimization method. (A) A four hidden layer neural network is used on the MNIST dataset. (B) The inverse relation between variance and flatness holds between any two adjacent hidden layers. (C) Illustration of LeNet which has two convolution layer with sizes of $3 \times 5 \times 5 \times 6$ and $6 \times 5 \times 5 \times 16$, and three fully connected layers with sizes of 400×120,120×84,84×10. (D) The relation between variance and flatness in the two convolution layers clearly shows an inverse dependence. The LeNet convolution network shown in (C) is used on the CIFAR10 dataset and the network is optimized by using SGD with momentum.

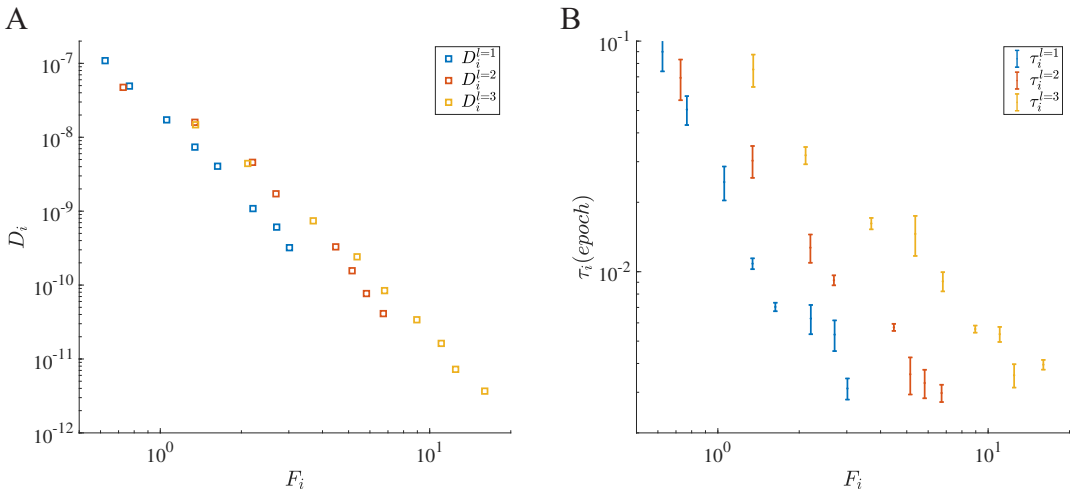

FIG. S10: The inverse relation of τ_i (in unit of epoch) and D_i on F_i for a multi-hidden-layer network (4 hidden layers) on MNIST the same as in Fig. S9A.

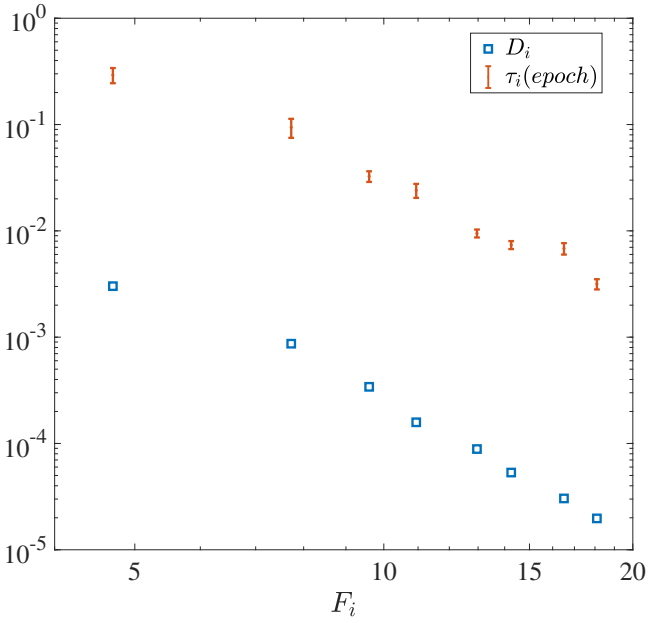

FIG. S11: The inverse dependence of τ_i (in unit of epoch) and D_i on F_i for CNN on CIFAR10 the same as in Fig S9C.

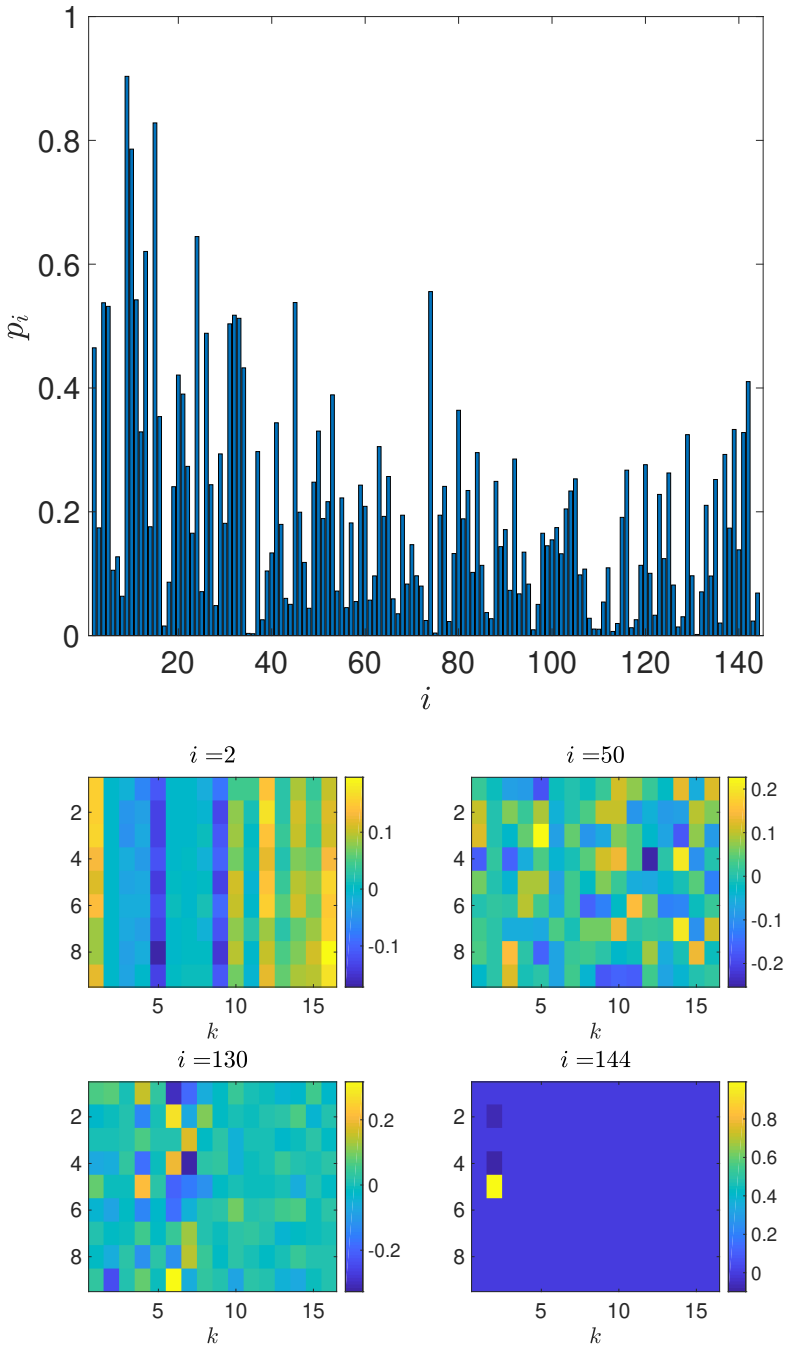

FIG. S12: (Upper panel) The amplitude of the solution weight matrix $w^*(k,l)$ projected onto different PCA weight matrix $w_i(k,l)$: $p_i \equiv \sum_{k,l}[w^*(k,l)w_i(k,l)]^2$ for different PCA direction i. (Lower panels) The filter weight matrices in CNN along different PCA directions, $w_i(l,k)$, for four different directions $i = 2, 50, 130, 144$. $k \in [1,16]$ is the filter index, and $l \in [1,9]$ labels the 3×3 filter weights. For larger i's (flatter directions), the filter matrix seems to become sparser.

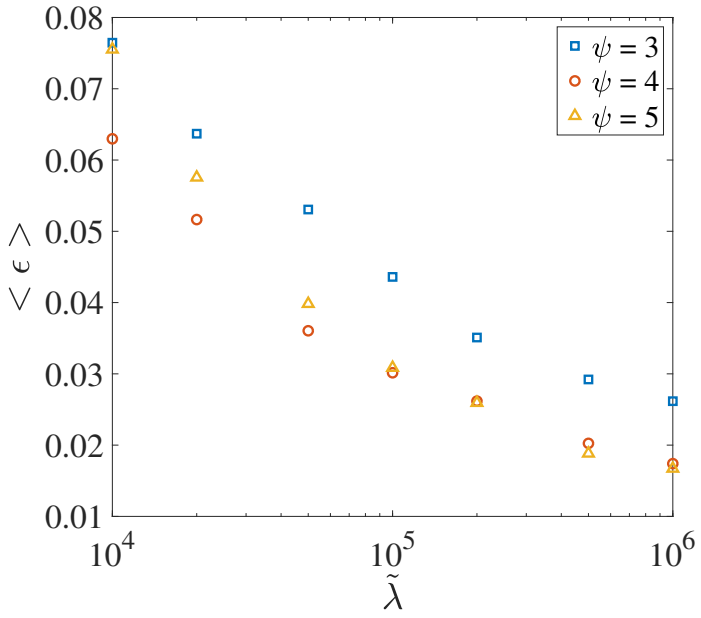

FIG. S13: The average error $\langle \epsilon \rangle = (\epsilon_1 + \epsilon_2)/2$ for the two tasks versus the normalized constraint strength $\tilde{\lambda} \equiv 10^{\psi}\lambda$ for different choices of $\psi = 3, 4, 5$. The simulations are done on the same network and data set as the one in Figure 4 in the main text. In the landscape dependent constraint (LDC) algorithm, we determine the flatness by using the inverse variance-flatness relationship $F_i \propto \sigma_i^{-2/\psi}$, where the variance σ_i^2 is determined by the PCA analysis. The LDC results seem to be insensitive to the precise choice of ψ.

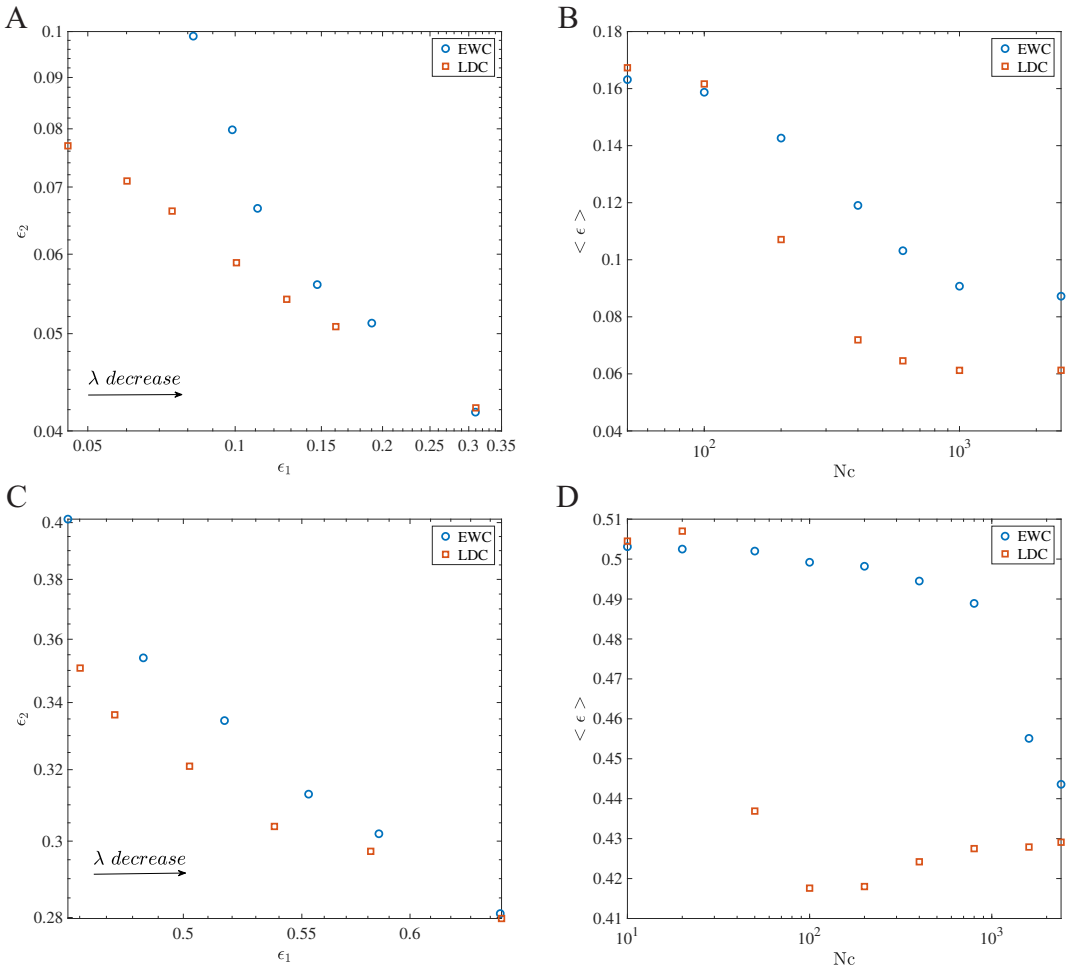

FIG. S14: The comparison of LDC and EWC for avoiding catastrophic forgetting in two more complex cases. (A)&(C) Task-1 error (ϵ_1) versus task-2 error (ϵ_2) as the constraint strength λ is varied, (B)&(D) the minimum average error ($\langle\epsilon\rangle$) versus the number of constraints N_c. For (A)&(B), learning digit sets, $[0, 1, 2, 3, 4]$ and $[5, 6, 7, 8, 9]$ in MNIST, are used as task-1 and task-2 respectively. For (C)&(D), we used the CIFAR10 dataset, and learning all the natural objects (birds,cats,deer,dogs,frogs,horses) is used as task-1 and learning all man-made objects (airplanes,cars,ships and trucks) is used as task-2 .

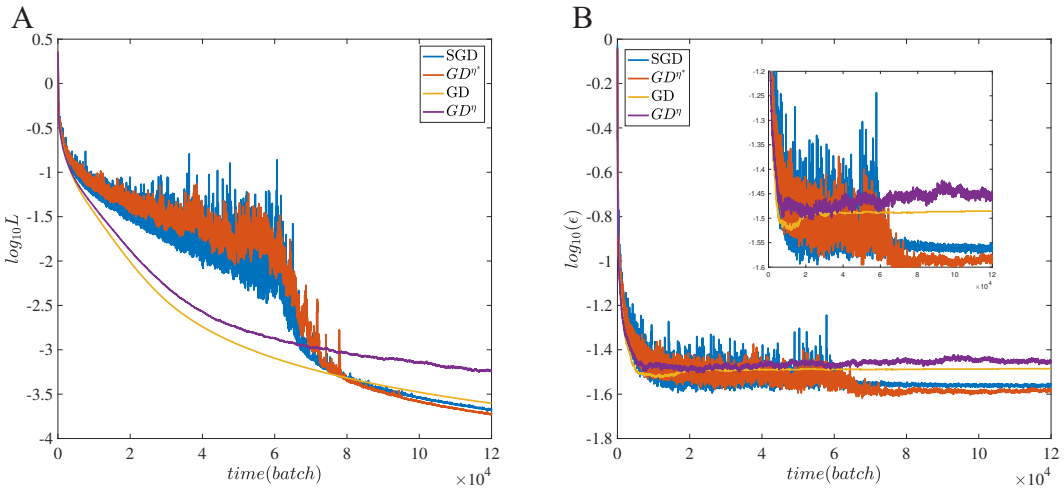

FIG. S15: (A) The loss error and (B) the test error versus training time for four algorithms. For gradient descent (GD) method, its loss decreases smoothly and the test error reaches a higher steady state value (3.27%) at the end of training. SGD have large fluctuations in the beginning of the training before it transitions to the exploration phase where the cross entropy loss is the same as in GD but the steady state test error (2.54%) is lower. When we add the "flatness detecting noise" to GD, which we call the GD$^{\eta*}$ algorithm, the behavior is similar to that of SGD and GD$^{\eta*}$ also leads to a low test error$\sim$ 2.41% (see inset in (B)). As a comparison, we also add an isotropic noise with constant strength to SD, which we call the GD$^{\eta}$ algorithm, we find that the behavior of GD$^{\eta}$ is similar to that of GD, which does not lead to a lower generalization error as in GD$^{\eta*}$, where anisotropic landscape dependent noise is used.

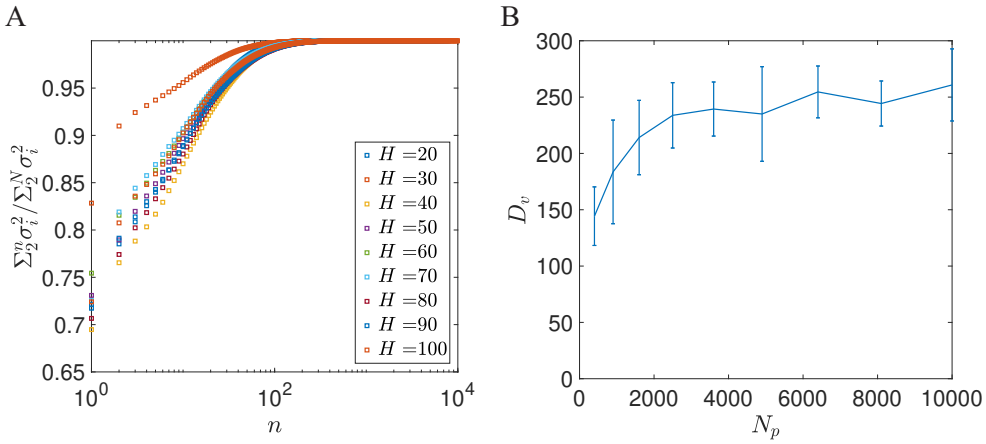

FIG. S16: The normalized accumulative variance spectrum and the effective SGD learning dimension (D_l). (A) The normalized accumulative variance versus PCA directions (i) for different network width H. (B) The learning dimension D_l, which is defined as the number of components (directions) which accounts for 99.9% total variance, is much smaller than the total number of parameters $N_p(\equiv H^2)$, and it only increases weakly as N_p increases. This means SGD only search for solutions in a relatively small sub-manifold and is not affected significantly by over-parametrization. The error bars are obtained by using 10 different solutions obtained by random initialization (with the same norm) for each network.