Supporting Information

Kirkpatrick et al. 10.1073/pnas.1611835114

Random Patterns

In this section we show that using EWC it is possible to recover a power-law decay for the SNR of random patterns. The task consists of associating random n-dimensional binary vectors x_t to a random binary output y_t by learning a weight vector W. The continual-learning aspect of the problem arises from the fact that at time step i, only the ith pattern is accessible to the learning algorithm. Before providing a detailed derivation of the learning behavior, we provide a sketch of the main ideas. Learning consists of minimizing an objective function at each time step. This objective function contains the square loss for the current pattern plus a penalty that minimizes the distance of the weight vector to its old value. This corresponds to EWC if the distance metric used is the diagonal of the total Fisher information matrix. Conversely, if a fixed metric is used, we recover gradient descent. In this particular case, the diagonal of the Fisher information matrix is proportional the number of patterns observed, so EWC simply consists of lowering the learning rate at each time step. We then obtain an exact solution for the average response (signal) of a pattern observed at time i at a later time t in both the gradient descent (constant learning rate) and the EWC cases. We find that EWC leads to a power-law decay in the signal whereas gradient descent leads to an exponential one. Our analysis of the variance in the response (noise) shows that in both the EWC and gradient descent cases, for small time, the noise increases as $\sqrt{t/n}$. Conversely, for large t the noise tends to 1 in the gradient descent case, and it decays as $\sqrt{t/(t+n)}$ in the EWC case.

We assume that $||x_t||_2 = 1$, such that each element is $\pm 1/\sqrt{n}$. We regress to the targets y_t with the dot product $W_t x_t$ and optimize the parameters W, using least-squares minimization. Therefore, the loss at step i can be written as

$$\mathcal{L}_i = \frac{1}{2}\left((W_i x_i - y_i)^2 + |W_i - W_{i-1}|_{M_i}\right). \qquad \textbf{[S1]}$$

The first term on the right-hand side is simply a squared error, and the second term makes the problem well defined by constraining the parameters to be close to the old ones under a metric M_i. We consider two possible choices for the metric: In the steepest descent case, we use a fixed, uniform metric, and in the EWC case we use the diagonal of the Fisher information matrix. Note that in this problem, the diagonal of the Fisher information matrix at time step i is simply an identity matrix multiplied by i/n. To cover both cases, we refer to the norm of the metric at time step i as λ_i. This quantity is fixed in the case of steepest descent and is increasing with time as i/n for the EWC case. Let us take the derivative of Eq. **S1** with respect to W_i and set it to 0. We then find that

$$W_i = W_{i-1} - \frac{1}{\lambda_i}(W_i x_i - y_i)x_i^T. \qquad \textbf{[S2]}$$

Let us solve Eq. **S2** for W_i to find

$$W_i = \left(W_{i-1} + \frac{y_i x_i^T}{\lambda_i}\right)\left(1 + \frac{x_i x_i^T}{\lambda_i}\right)^{-1}. \qquad \textbf{[S3]}$$

We simplify the previous equation by using the Sherman–Morrison formula for matrix inverses and by defining $\bar{x}_i = x_i y_i$:

$$W_i = W_{i-1}\left(1 - \frac{\bar{x}_i \bar{x}_i^T}{\lambda_i + 1}\right) + \frac{\bar{x}_i^T}{\lambda_i + 1}. \qquad \textbf{[S4]}$$

Note that this is exactly the same equation that one would obtain if performing gradient descent with a learning rate $\frac{1}{\lambda_i+1}$.

If we assume that the initial conditions W_0 are a null vector and define the matrix $A_i = \left(1 - \frac{\bar{x}_i \bar{x}_i^T}{\lambda_i+1}\right)$ and the vector $X_i = \frac{\bar{x}_i^T}{\lambda_i+1}$, then unfolding the recurrence relation from **[S4]** leads to

$$W_i = X_1 A_2 A_3 \ldots A_i + X_2 A_3 A_4 \ldots A_i + \cdots + X_i. \qquad \textbf{[S5]}$$

To measure the average memory strength of a memory from time i at a later time t, we need to compute the signal that is the mean response to a stimulus from time step i,

$$S(i,t) = \langle W_t \bar{x}_i \rangle_{\bar{x}_j, j \neq i}, \qquad \textbf{[S6]}$$

where the angle brackets denote averaging over all of the patterns seen, except x_i.

To be retrieved, it is important for this average signal to exceed the noise by a certain margin. The noise ν is simply defined as the standard deviation of the quantity above; that is,

$$\nu(i,t)^2 = \langle (W_t \bar{x}_i)^2 \rangle_{\bar{x}_j, j\neq i} - S(i,t)^2. \qquad \textbf{[S7]}$$

In what follows, we first derive the expression for the signal and then that for the noise term in both the steepest-descent and EWC cases.

Because each term A_j in each term of the sum of Eq. **S5** pertains to one time step only, averaging Eq. **S6** factors out, and it is sufficient to be able to compute the following average,

$$\langle A_j \rangle_{x_j} = \left(1 - \frac{1}{n(\lambda_j + 1)}\right) I, \qquad \textbf{[S8]}$$

where I is the identity matrix. Of each term in the sum of Eq. **S5**, all of the terms starting with an X_j with $j \neq i$ will average to 0, and the only term remaining will therefore be

$$S(i,t) = \langle X_i A_{i+1} \ldots A_t \bar{x}_i \rangle, \qquad \textbf{[S9]}$$

which, using Eq. **S8**, will result in

$$S(i,t) = \frac{1}{\lambda_i + 1} \Pi_{j=i+1}^{t} \left(1 - \frac{1}{n(\lambda_j + 1)}\right). \qquad \textbf{[S10]}$$

We consider two cases: In the steepest-descent case $\lambda_j = \frac{1}{n}$ and in the EWC case $\lambda_j = \frac{j}{n}$. Note that we chose the magnitude for the regularization for the steepest-descent case in such a way that they both agree for the first time step. In the steepest-descent case, after some manipulation we reach

$$S(i,t) = \left(\frac{n}{n+1}\right)^{1+t-i}, \text{ for } i < t. \qquad \textbf{[S11]}$$

If we are interested in cases when $n >> 1$, the previous expression can be rearranged into

$$S(i,t) = \exp\left(\frac{i-t-1}{n}\right). \qquad \textbf{[S12]}$$

In the EWC case we have

$$S(i,t) = \frac{n}{n+t}. \qquad \textbf{[S13]}$$

To compute the noise term, we assume that the weight vector W_t is uncorrelated from the observed vector $\bar{x}_i$, and we can simplify Eq. **S7**:

$$\nu(i,t)^2 = \left\langle \frac{1}{n}|W_t|^2 \right\rangle. \qquad \textbf{[S14]}$$

Note that this assumption is tantamount to averaging the quantity over all $\bar{x}_j$, including $\bar{x}_i$.

Therefore, the key to computing the noise term is to be able to predict the average norm of the weight vector. To deduce what form this must take, let us rearrange Eq. **S3** slightly to yield

$$W_t = W_{t-1} + \bar{x}_t^T \frac{1}{\lambda_t + 1}(1 - W_{t-1}\bar{x}_t). \qquad \textbf{[S15]}$$

Define the vector δ_t as

$$\delta_t = \bar{x}_t^T \frac{1}{\lambda_t + 1}(1 - W_{t-1}\bar{x}_t). \qquad \textbf{[S16]}$$

Then the expected change in the norm of the weight vector can we written as

$$\langle |W_t|^2 - |W_{t-1}|^2 \rangle = \langle |\delta_t|^2 \rangle + 2\langle \delta_t W_{t-1}^T \rangle. \qquad \textbf{[S17]}$$

These two terms are readily computed as

$$\langle |\delta_t|^2 \rangle = \left(\frac{1}{\lambda_t + 1}\right)^2 \left(1 + \frac{|W_{t-1}|^2}{n}\right) \qquad \textbf{[S18]}$$

$$\langle \delta_t W_{t-1}^T \rangle = -\frac{1}{\lambda_t + 1}\frac{|W_{t-1}|^2}{n}. \qquad \textbf{[S19]}$$

Therefore, the expected change in the norm can be written as

$$|W_t|^2 - |W_{t-1}|^2 = \left(\frac{1}{\lambda_t + 1}\right)^2 - \frac{2\lambda_t + 1}{n(\lambda_t + 1)^2}|W_{t-1}|^2. \qquad \textbf{[S20]}$$

To proceed, we take a continuous approximation of this difference equation to yield a linear, first-order, inhomogeneous ordinary differential equation for the norm of the weight vector,

$$\frac{d}{dt}|W|^2 = \left(\frac{1}{\lambda(t) + 1}\right)^2 - \frac{2\lambda(t) + 1}{n(\lambda(t) + 1)^2}|W|^2, \qquad \textbf{[S21]}$$

with boundary condition

$$|W|^2(t = 0) = 0. \qquad \textbf{[S22]}$$

In the case of steepest descent, the function $\lambda(t)$ is simply a constant $1/n$ and the above equation simplifies to

$$\frac{d}{dt}|W|^2 = \frac{n^2}{(n+1)^2} - \frac{2+n}{(n+1)^2}|W|^2. \qquad \textbf{[S23]}$$

Let us assume that $n >> 1$ to simplify that expression to

$$\frac{d}{dt}|W|^2 \approx 1 - \frac{1}{n}|W|^2 \qquad \textbf{[S24]}$$

with solution

$$|W|^2(t) = n\nu(t)^2 = n\left(1 - \exp\left(-\frac{t}{n}\right)\right). \qquad \textbf{[S25]}$$

If, instead, we are in the EWC case, the function $\lambda(t)$ is t/n and Eq. **S21** becomes

$$\frac{d}{dt}|W|^2 = \frac{n^2}{(n+t)^2} - \frac{2t+n}{(n+t)^2}|W|^2. \qquad \textbf{[S26]}$$

It is possible to obtain an analytic solution to this ordinary differential equation,

$$|W|^2 = n\nu(t)^2 = \frac{n^2}{(t+n)^2}\exp\left(-\frac{n}{n+t}\right)\left[-n\mathrm{Ei}\left(\frac{n}{n+t}\right)\right.$$
$$\left. + n\mathrm{Ei}(1) + (n+t)\exp\left(\frac{n}{n+t}\right) - n\exp(1)\right], \qquad \textbf{[S27]}$$

where Ei is the exponential integral function.

For the gradient descent case the analytical form of the SNR is simple. Using Eqs. **S12** and **S25** we find

$$SNR(i,t) = \frac{\exp\left(-\frac{t+1-i}{n}\right)}{\sqrt{1 - \exp(-\frac{t}{n})}}, \quad \text{gradient descent case} \qquad \textbf{[S28]}$$

This expression is a power law when $t << n$ and an exponential when t is the same order of magnitude as n or greater.

The expression for the noise in Eq. **S27**, however, is harder to interpret. If $t << n$, it is equivalent to the solution for the steepest descent, that is, Eq. **S25**. If conversely $t >> n$, noting that $\mathrm{Ei}(x) \approx \exp(-x)$ for large x, we can get a simplified form for the norm of the weight as a function of time:

$$|W|^2 = \frac{n^2}{(t+n)}. \qquad \textbf{[S29]}$$

Thus, we can obtain the expression for the behavior of the SNR in the EWC case in the small time and large time regimes. When $t << n$, the signal expressed in **[S13]** is $\sim$1 and the noise term from **[27]** can be approximated as $\sqrt{t/n}$. So the SNR can be written as

$$SNR(i,t) = \sqrt{\frac{n}{t}}, \text{ for } t << n. \qquad \textbf{[S30]}$$

If, however, the time is of the same magnitude as n different outcomes are observed for steepest descent and for EWC, as the time step approaches n in the EWC case, the signal from Eq. **S13** will fall as $(t/n)^{-1}$ and the noise from Eq. **S29** will also as $\sqrt{n/(n+t)}$; therefore the overall SNR will take the form

$$SNR(i,t) = \sqrt{\frac{n}{n+t}}, \text{ for } t >> n \text{ EWC case.} \qquad \textbf{[S31]}$$

The main distinction that we expect between using steepest descent and EWC is therefore that EWC should show a power law in the SNR with an exponent of -0.5 for both the small time and large time regimes. Gradient descent, conversely, will show a power law only for times shorter than the capacity of the network, and subsequently the memories are forgotten exponentially.

Our analytic computation of the noise term is approximate, because of the approximation that weights and input can be considered to be uncorrelated and also because we make a continuous approximation. To validate these assumptions, and to check our computations, we compare the analytic expression with numerical simulations. The numerical simulations were obtained by making several (400) simulations of the weights and patterns observed with a value of $n = 1{,}000$. We then simply computed the signal $S(i,t)$ as the mean response at time t from a pattern observed at time i and the noise as the SD. In Fig. S1 we show the value of the signal and noise obtained numerically and for the analytic expressions in Eqs. **S11**, **S13**, **S25**, **S27**, and **S29**. Note that agreement between the observed signal and the numerical one is always very good. For the noise term agreement is good except when $t - i$ is small and our assumptions that weights and the patterns are uncorrelated does not hold. When our assumption on the noise breaks down, we observe that the magnitude of the noise is smaller than we predict and the SNR is higher than expected by our analysis.

It should be noted that in this analysis we have chosen a particular value for the regularization term $\lambda = 1/n$. This is equivalent to the value used for the first pattern observed in EWC. Because this regularization term effectively sets the learning rate, this means we have picked a learning rate for gradient descent equivalent to that used at the start of learning in EWC. To make a fair comparison we should also test gradient descent using the smaller learning rates used at later stages by EWC. First, we generalize the expressions for the signal and noise terms in the gradient descent case with arbitrary learning rate α. Remembering that $\alpha = \frac{1}{1+\lambda}$, the generalization of Eqs. **S12** and **S25** becomes

$$S(i,t) = \alpha\exp\left(-\alpha\frac{(t-i)}{n}\right) \qquad \textbf{[S32]}$$

$$\nu(t) = \sqrt{\frac{\alpha}{2-\alpha}\left(1 - \exp\left(-\alpha(2-\alpha)\frac{t}{n}\right)\right)}. \quad \textbf{[S33]}$$

In Fig. S2 we show a comparison of the SNR in the EWC case (red curves) and in the plain gradient descent case with different learning rates (green lines, $\alpha = 1.0$; blue lines, $\alpha = 0.5$). These learning rates are chosen to match the learning rate in EWC at the beginning of training ($\alpha = 1$) and at capacity, that is, when $t = n$, at which point $\alpha = 0.5$. The SNR plot (Fig. S2, *Top*) shows that irrespective of the learning rate used, the SNR in the gradient descent case eventually follows an exponential decay, albeit with a different rate. EWC, conversely, maintains a power-law decay. Fig. S2, *Middle* shows that the fraction of memories retained is higher using EWC than with either the learning rates, although a higher percentage is retained with the lower learning rate.

MNIST Experiments

We carried out all MNIST experiments with fully connected networks with rectified linear units, using the Torch neural network framework. To replicate the results of ref. 24, we compared with results obtained using dropout regularization. As suggested in ref. 24, we applied dropout with a probability of 0.2 to the input and of 0.5 to the other hidden layers. To give SGD with dropout the best possible chance, we also used early stopping. Early stopping was implemented by computing the test error on the validation set for all pixel permutations seen to date. Here, if the validation error was observed to increase for more than five subsequent steps, we terminated this training segment and proceeded to the next dataset; at this point, we reset the network weights to the values that had the lowest average validation error on all previous datasets. Table S1 shows a list of all hyperparameters used to produce the three graphs in Fig. 3 of the main text. Where a range is present, the parameter was randomly varied and the reported results were obtained using the best hyperparameter setting. When random hyperparameter search was used, 50 combinations of parameters were attempted for each number experiment.

Atari Experiments

Atari experiments were carried out in the Torch framework. The agent architecture used is almost identical to that used in ref. 42. In this section we provide details on all of the parameters used.

Images are preprocessed in the same way as in ref. 25, namely the 210×160 images from the Atari emulator are down-sampled to 84×84, using bilinear interpolation. We then convert the red green blue (RGB) images to YUV and use the grayscale channel alone. The state used by the agent consists of the four latest down-sampled, grayscale observations concatenated together.

The network structure used is similar to the one from ref. 25, namely three convolutional layers followed by a fully connected layer. The first convolution had kernel size 8, stride 4, and 32 filters. The second convolution had kernel size 4, stride 2, and 64 filters. The final convolution had kernels size 3, stride 1, and 128 filters. The fully connected layer had 1,024 units. Note that this network has approximately four times as many parameters as the standard network, due to having twice as many fully connected units and twice as many filters in the final convolution. The other departure from the standard network is that each layer was allowed to have task-specific gains and biases. For each layer, the transformation $x \to y$ computed by the network is therefore

$$y_i = \left(\sum_j W_{ij} x_j + b_i^c\right) g_i^c, \quad \textbf{[S34]}$$

where the biases are b and the gains are g. The network weights and biases were initialized by setting them randomly with a uniform number between $-\sigma$ and σ, with σ set to the square root of the incoming hidden units (for a linear layer) or set to the area of the kernel times the number of incoming filters (for convolutional layers). Biases and gains were initialized to 0 and 1, respectively.

We used an ϵ-greedy exploration policy, where the probability of selecting random action, ϵ, decayed with training time. We kept a different timer for each of the tasks. We set $\epsilon = 1$ for 5×10^4 time steps and then decayed this linearly to a value of 0.01 for the next 10^6 time steps.

We trained the networks with the Double Q-learning algorithm (42). A training step is carried out on a minibatch of 32 experiences every four steps. The target network is updated every 3×10^4 time steps. We trained with RMSProp, with a momentum of 0, a decay of 0.95, a learning rate of 2.5×10^{-4}, and a maximum learning rate of 2.5×10^{-3}.

Other hyperparameters that we changed from the reference implementation were (*i*) using a smaller replay buffer (5×10^5 past experiences) and (*ii*) a scaling factor for the EWC penalty of 400. Another subtle difference is that we used the full action set in the Atari emulator. In fact, although many games support only a small subset of the 18 possible actions, to have a unified network structure for all games we used 18 actions in each game.

We randomly chose the 10 games for each experiment from a pool of 19 Atari games for which the standalone DQN could reach human-level performance in 50×10^6 frames. The scores for each of these games for the baseline algorithm, for EWC, and for plain SGD training, as a function of the number of steps played in that game, are shown in Fig. S3. To get an averaged performance, we chose 10 sets of 10 games and ran four different random seeds for each set.

The most significant departure from the published models is the automatic determination of the task. We model each task by a generative model of the environment. In this work, for simplicity, we model only the current observation. The current task is modeled as a categorical context c that is treated as the hidden variable in a hidden Markov model that explain observations. In such a model the probability of being in a particular context c at time t evolves according to

$$p(c, t+1) = \sum_{c'} p(c', t)\Gamma(c, c')$$

$$\Gamma(c, c') = \delta(c, c')(1-\alpha) + (1 - \delta(c, c'))\alpha,$$

where δ is the Kronecker delta function and α is the probability of switching context. The task context then conditions a generative model predicting the observation probability $p(o|c, t)$. Given such generative models, the probability of being in a task set at time t can be inferred by the observations seen so far as

$$p(c\,|o_1 ... o_t) \propto \sum_{c'} \Gamma(c, c')\; p(c', t-1) p(o|c, t).$$

The maximal probability context is then taken to be the current task label.

In our implementation, the generative models consist of factored multinomial distributions explaining the probability of the state of each pixel in the observation space. The model is a parameterized Dirichlet distribution, which summarizes the data seen so far using Bayesian updates. To encourage each model to specialize, we train the models as follows. We partition time into windows of a particular width W. During each window, all of the Dirichlet priors are updated with the evidence seen so far. At the end of the window, the model best corresponding to the current task set is selected. Because this model was the most useful to explain the current data, it keeps its prior, and all other priors are reverted to their state at the beginning of the time window. We ensure that one hold-out uniform (i.e., uninitialized) Dirichlet multinomial is always available. Whenever the hold-out model is selected, a new generative model is created and a new task context is therefore created. This model is Bayesian, in the sense that data are used to maintain beliefs over priors on the generative models, and is nonparametric, in the sense that the model can grow as a function of the observed data. It can be seen as an implementation of the flat FMN algorithm described in ref. 34. The parameter α is not learned. Instead we use the result from ref. 43 where it is shown that a time-decaying switch rate $\alpha = 1/t$ guarantees good worst-case asymptotic performance provided the number of tasks grows as $o\left(\frac{n}{\log n}\right)$.

Table S2 summarizes all hyperparameters used for the Atari experiments. Except for the parameters pertaining to the EWC algorithm (Fisher multiplier, num. samples Fisher, EWC start) or pertaining to the task recognition models (model update period, model downscaling, and size window), all of the parameter values are the same as in ref. 42 and have not been tuned for these experiments.

Fisher Overlap

To assess whether different tasks solved in the same network use similar sets of weights (Fig. 3*C* in the main text), we measured the degree of overlap between the two tasks' Fisher matrices. Precisely, we computed the two tasks' Fishers, F_1 and F_2; normalized these to each have unit trace, $\hat{F}_1$ and $\hat{F}_2$; and then computed their Fréchet distance, a metric on the space of positive-semidefinite matrices (44),

$$d^2(\hat{F}_1, \hat{F}_2) = \frac{1}{2}\mathrm{tr}\left(\hat{F}_1 + \hat{F}_2 - 2(\hat{F}_1\hat{F}_2)^{1/2}\right)$$
$$= \frac{1}{2}||\hat{F}_1^{1/2} - \hat{F}_2^{1/2}||_F,$$

which is bounded between zero and one. We then define the overlap as $1 - d^2$, with a value of zero indicating that the two tasks depend on nonoverlapping sets of weights and a value of one indicating that $F_1 = \alpha F_2$ for some $\alpha > 0$.

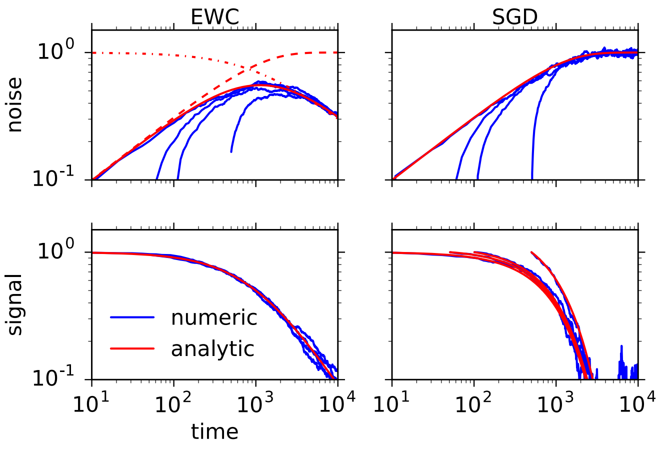

Fig. S1. Signal (*Bottom row*) and noise (*Top row*) terms for the EWC (*Left column*) and gradient descent (*Right column*) cases as a function of time *t*. The blue curves are the results of numeric simulations, whereas the red curves show the analytic results. Each panel contains stimuli observed at different times *i*. (*Top Left*) (noise in the EWC case) The solid red curve is the full form of Eq. **S27**, the dashed line is Eq. **S25** (which is valid for small times), and the dashed-dotted line is the long time approximation of the noise in Eq. **S29**. The different solid blue curves correspond to the noise from patterns observed at times 1, 50, 100, and 500. (*Top Right*) (noise in the gradient descent case) The red curve shows Eq. **S25**. (*Bottom Left*) (signal in the EWC case) The red curve is Eq. **S13**. (*Bottom Right*) (signal for the gradient descent case) The red curves are Eq. **S11**.

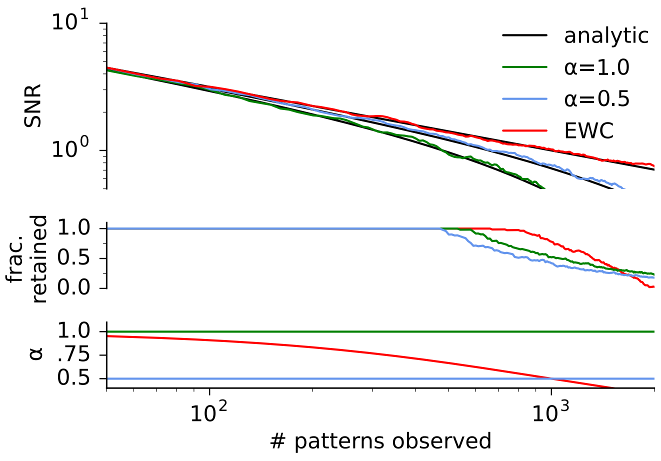

Fig. S2. Comparison of EWC (red lines) with gradient descent with different learning rates (green, $\alpha = 1.0$; blue, $\alpha = 0.5$). *Bottom* shows these two learning rates correspond to the learning rate used in EWC at the first pattern and at network capacity ($t = n = 1{,}000$) *Top* shows log-log plot of the SNR for the first pattern observed. The black lines show the analytic expressions from Eqs. **S13**, **S27**, and **S32**. Note that EWC has a power-law decay for the SNR, whereas gradient descent eventually decays exponentially, albeit at a later time for the lower learning rate. *Middle* shows the fraction of memories retained (i.e., with $SNR > 1$) in the three cases. Note that the lower rate has a moderately higher fraction of memories retained than the larger one, but that EWC still has a higher memory retention.

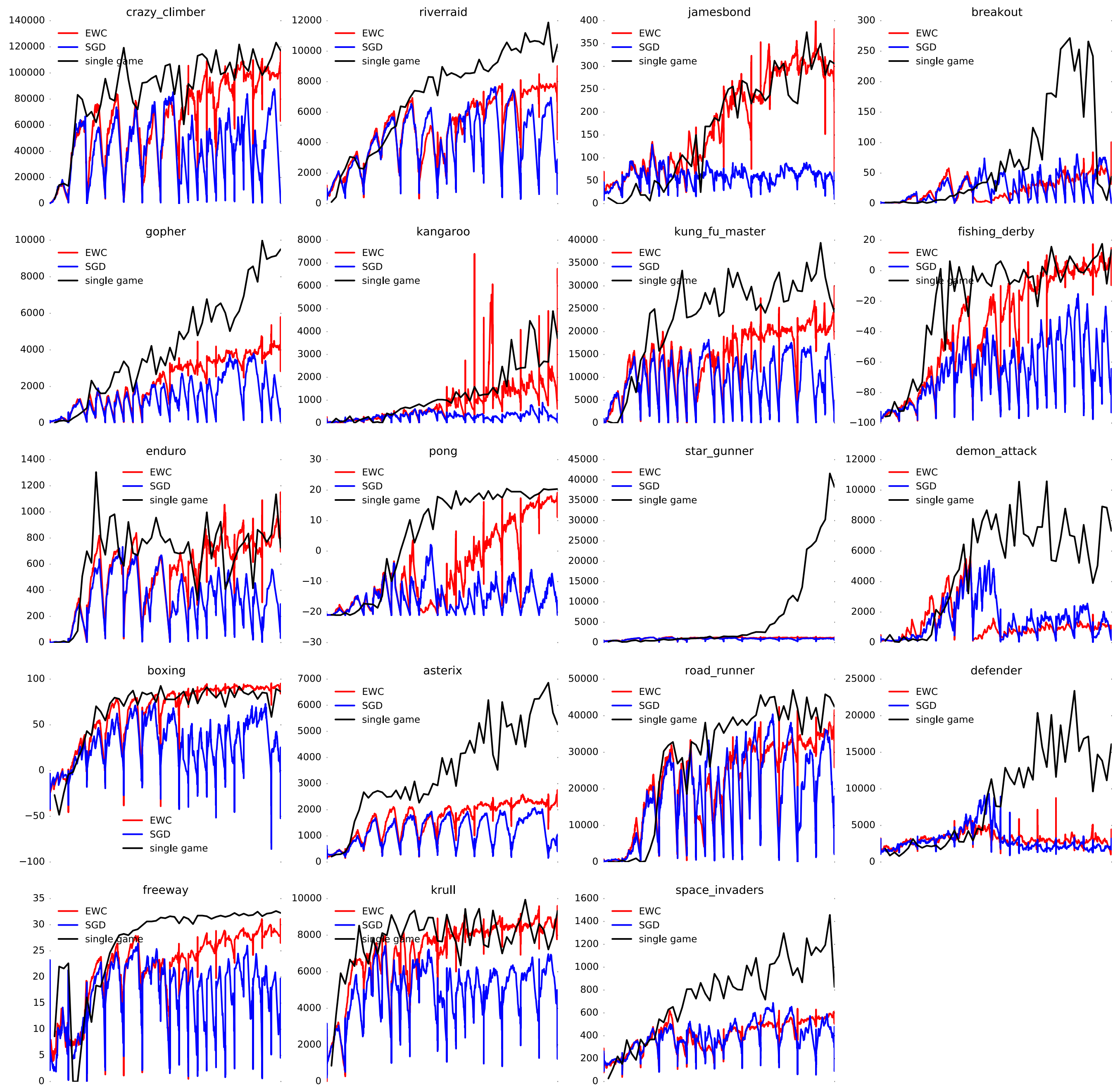

Fig. S3. Score in the individual games as a function of steps played in that game. The black baseline curves show learning on individual games alone.

Table S1. Hyperparameters for each of the MNIST figures

Hyperparameter	Fig. 3*A*	Fig. 3*B*	Fig. 3*C*
Learning rate	10^{-3}	10^{-5}–10^{-3}	10^{-3}
Dropout	No	Yes	No
Early stopping	No	Yes	No
No. hidden layers	2	2	6
Width hidden layers	400	400–2,000	100
Epochs/dataset	20	100	100

Table S2. Hyperparameters for the Atari experiment

Hyperparameter	Value	Brief description
Action repeat	4	Repeat the same action for four frames. Each agent step will occur every fourth frame.
Discount factor	0.99	Discount factor used in the Q-learning algorithm.
No-op max	30	Maximum number of do nothing operations carried out at the beginning of each training episode to provide a varied training set.
Max. reward	1	Rewards are clipped to 1.
Scaled input	84×84	Input images are scaled to 84×84 with bilinear interpolation.
Optimization algorithm	RMSprop	Optimization algorithm used.
Learning rate	0.00025	The learning rate in RMSprop.
Max. learning rate	0.0025	The maximum learning rate that RMSprop will apply.
Momentum	0.0	The momentum used in RMSprop.
Decay	0.95	The decay used in RMSProp.
Clipδ	1.0	Each gradient from Q-learning is clipped to ± 1.
Max. norm	50.	After clipping, if the norm of the gradient is greater than 50., the gradient is renormalized to 50.
History length	4	The four most recently experienced frames are taken to form a state for Q-learning.
Minibatch size	32	The number of elements taken from the replay buffer to form a minibatch training example.
Replay period	4	A minibatch is loaded from the replay buffer every four steps (16 frames including action repeat).
Memory size	50,000	The replay memory stores the last 50,000 transitions experienced.
Target update period	7,500	The target network in Q-learning is updated to the policy network every 7,500 steps.
Min. history	50,000	The agent will start learning only after 50,000 transitions have been stored into memory.
Initial exploration	1.0	The value of the initial exploration rate.
Exploration decay start	50,000	The exploration rate will start decaying after 50,000 frames.
Exploration decay end	1,050,000	The exploration rate will decay over 1 million frames.
Final exploration	0.01	The value of the final exploration rate.
Model update period	4	The Dirichlet model is updated every fourth step.
Model downscaling	2	The Dirichlet model is downscaled by a factor of 2; that is, an image of size 42×42 is being modeled.
Size window	4	The size of the window for the task recognition model learning.
Num. samples Fisher	100	Whenever the diagonal of the Fisher is recomputed for a task, 100 minibatches are drawn from the replay buffer.
Fisher multiplier	400	The Fisher is scaled by this number to form the EWC penalty.
Start EWC	20E6	The EWC penalty is applied only after 5 million steps (20 million frames).