

491 Appendix

492 Table of contents 493

494	A Details of the theoretical results	15
495	A.1 Overview of the assumptions	15
496	A.2 Derivation of the expected loss and its gradients	15
497	A.3 Analysis of the dynamics without cross-sample repetition	16
498	A.3.1 Reducing the learning dynamics to two dimensions	16
499	A.3.2 Approximate dynamics around initialization	18
500	A.3.3 Phase transition and late phase dynamics	20
501	A.3.4 Empirical validation	21
502	A.4 Analysis of the dynamics with cross-sample repetition	21
503	A.4.1 Reducing the learning dynamics to three dimensions	22
504	A.4.2 Approximate dynamics around initialization	23
505	B Details and additional analysis for the linear regression task	24
506	B.1 Implementation of the task	24
507	B.2 Examples of learning curves	25
508	B.3 Additional ablations on the impact of task variation, model size and optimizer on 509 plateau length	25
510	C Details and additional experiments for the in-context associative recall task	28
511	C.1 Implementation of the task	28
512	C.2 Learning dynamics and additional analysis	28
513	D Methods	30
514	D.1 Measurement of the plateau length	30
515	D.2 Fitting power laws on plateau length	30
516	D.3 Architectural and training details	31
517	D.4 Reproducibility	31

518
519
520

A Details of the theoretical results

A.1 Overview of the assumptions

Throughout our theoretical analysis, we introduce a few assumptions to simplify it. We summarize them here:

- Asm. 1 **Gradient flow dynamics.** To study learning dynamics, we focus on the gradient flow of the expected loss. Compared to the standard deep learning regime, this removes stochastic noise induced by sampling, uses gradient descent as optimizer instead of optimizer with adaptive learning rates such as Adam, and requires an infinitely small learning rate. Due to the last point, phenomena such as the edge of stability [60] are out of the picture. Yet, this is a very standard assumption (e.g. [28]) and will enable us to use tools from dynamical systems to understand how the behavior of the model changes over the course of learning.
- Asm. 2 **Uniform attention at initialization.** At initialization, the attention patterns of Transformers generally display remarkable uniformity at initialization. We take advantage of this observation in our model by initializing a as a vector of zeros. This enables us to reduce the dynamics of the attention part of the model to a single scalar.
- Asm. 3 **W^* has norm 1 columns.** We assume that the columns of W^* all have norm 1. While this is not strictly necessary for performing our analysis, it simplifies the formulas that we obtain and helps to keep the notation concise. Additionally, this is the expected behavior when drawing the entire of W^* i.i.d. from a zero-mean normal distribution with variance $\frac{1}{d}$ when d goes to infinity.
- Asm. 4 **Initialize the weights W at 0.** When drawing the entries of W i.i.d. and independently of those of W^* , W is almost surely orthogonal to W^* as $d \rightarrow \infty$. The components of W not aligned with W^* exhibit a fast decay towards 0, so this assumption corresponds to having already completed this process. With such an assumption, we can reduce the dynamics of the weights to a single scalar, w , which will be the projection of W on a normalized W^* .
- Asm. 5 **Large sequences and large dimension.** Finally, we assume that we are in the large T limit and B is negligible in front of T . This makes sense in our context as we are interested in modeling the learning of sparse attention patterns. We additionally assume that d is large, which is a reasonable assumption given that we are interested in modeling the high-dimensional data neural networks have to process. These assumptions will enable us to ignore some negligible terms and keep our calculations more concise.

A.2 Derivation of the expected loss and its gradients

Recall that the loss is given by

$$L = \frac{1}{2} \mathbb{E}_x [\|y - y^*\|^2].$$

with

$$y = W \sum_{t=1}^T \text{softmax}(a)_t x_t$$

and

$$y^* = W^* x_T.$$

For notational clarity, we define attention patterns as $\alpha_t := \text{softmax}(a)_t$. Without loss of generality, we assume that the tokens repeated in tokens appear at positions $\{T - B + 1, \dots, T\}$ so that the last B tokens are always the same, and that the repeated input \tilde{x} is the first vector of the canonical basis³ of \mathbb{R}^d , that is $\tilde{x} = [1, 0, \dots, 0]^\top$. For the sake of conciseness, we denote by

$$\Sigma_t := \mathbb{E}_x [x_t x_t^\top]$$

the input covariance of each token. Note that for $t \leq T - B$, it is equal to $\Sigma_t = \frac{1}{d} \text{Id}$.

³This amounts to a right multiplication of the weights W by an orthogonal matrix.

561 Using the insights mentioned above and the properties of the data distribution, the loss reduces to:

$$\begin{aligned}
L &= \frac{1}{2} \mathbb{E}_x [\|y - y^*\|^2] \\
&= \frac{1}{2} \mathbb{E}_x \left[\left\| \sum_t \alpha_t W x_t - W^* x_T \right\|^2 \right] \\
&= \frac{1}{2} \mathbb{E}_x \left[\sum_{t \leq T-B} \alpha_t^2 \|W x_t\|^2 + \left\| \sum_{t > T-B} \alpha_t W x_t - W^* x_T \right\|^2 \right] \\
&= \frac{1}{2} \mathbb{E}_x \left[\sum_{t \leq T-B} \alpha_t^2 \|W x_t\|^2 + \left\| \sum_{t > T-B} \alpha_t W x_T - W^* x_T \right\|^2 \right] \\
&= \frac{1}{2} \left[\sum_{t \leq T-B} \alpha_t^2 \text{tr}(W \Sigma_t W^\top) + \text{tr} \left(\left(\sum_{t > T-B} \alpha_t W - W^* \right) \Sigma_T \left(\sum_{t > T-B} \alpha_t W - W^* \right)^\top \right) \right] \\
&= \frac{1}{2} \left[\sum_{t \leq T-B} \frac{\alpha_t^2}{d} \|W\|_F^2 + \text{tr} \left(\left(\sum_{t > T-B} \alpha_t W - W^* \right) \Sigma_T \left(\sum_{t > T-B} \alpha_t W - W^* \right)^\top \right) \right].
\end{aligned}$$

562 In the third line, we used the fact that the first $T - B$ tokens are independent of each other and
563 independent of the last B ones, in the fourth line that the last B tokens are all equal to x_T and in the
564 fifth line the equality $\mathbb{E}_x[\|Ax\|] = \text{tr}(A\mathbb{E}[xx^\top]A^\top)$.

565 Leveraging this formula, we get the following gradients for the loss:

$$\nabla_W L = \sum_{t > T-B} \alpha_t \left(\sum_{t > T-B} \alpha_t W - W^* \right) \Sigma_T + \sum_{t \leq T-B} \frac{\alpha_t^2}{d} W \quad (7)$$

$$\nabla_{\alpha_t} L = \begin{cases} \text{tr} \left(W \Sigma_T \left(\sum_{t > T-B} \alpha_t W - W^* \right)^\top \right) & \text{if } t > T - B \\ \frac{\alpha_t}{d} \|W\|^2 & \text{otherwise.} \end{cases} \quad (8)$$

566 Note that we have not calculated the gradients with respect to a here, but only with respect to α .

567 A.3 Analysis of the dynamics without cross-sample repetition

568 In this section, we study the dynamics without cross-sample repetition, that is, with $p_{\text{repeat}} = 0$.

569 A.3.1 Reducing the learning dynamics to two dimensions

570 In general, the parameters evolve in a high-dimensional space. However, with a few reasonable
571 assumptions, it is possible to show that they evolve in a 2-dimensional subspace:

572 – **Attention is initialized uniformly.** The first assumption we make is that attention is perfectly
573 uniform (Assumption 2 from Section A.1), that is $\alpha_t = \frac{1}{T}$ or $a_t = 0$. Given that the gradients of
574 these parameters are the same for $t \leq T - B$ and $t > T - B$, cf. the calculation in the previous
575 section, all attention values will have two possible values. If we additionally leverage the fact
576 that $\sum_t \alpha_t = 1$, everything is captured by a single scalar Δa , that is defined as $a_T - a_1$. Indeed

$$\begin{aligned}
\alpha_t &= \frac{\exp(a_t)}{\sum_{t'} \exp(a_{t'})} \\
&= \frac{1}{(T - B) \exp(a_1 - a_t) + B \exp(a_T - a_t)}
\end{aligned}$$

577 so that, for $t > T - B$,

$$\alpha_t = \frac{1}{(T - B) \exp(-\Delta a) + B} \quad (9)$$

578 and for $t \leq T - B$,

$$\alpha_t = \frac{(1 - B\alpha_T)}{(T - B)}. \quad (10)$$

579 – **Weights are initialized at 0.** We assume that $W = 0$ at initialization, following Assumption 4 of
 580 Section A.1. As $\Sigma_T = \frac{1}{d}\text{Id}$, we have

$$\nabla_W L = \sum_{t>T-B} \frac{\alpha_t}{d} \left(\sum_{t>T-B} \alpha_t W - W^* \right) + \sum_{t \leq T-B} \frac{\alpha_t^2}{d} W.$$

581 This implies that whenever W is aligned with W^* , which is the case when $W = 0$, it remains
 582 aligned for the rest of learning. Yet, we are not entirely done as the precise parametrization
 583 is important to ensure that the dynamics match. Such a property is achieved when $W =$
 584 $wW^*/\|W^*\|_F$ as, under the gradient flow dynamics on W , we have

$$w = \frac{\langle W^*, W \rangle_F}{\|W^*\|_F}$$

$$\dot{w} = \frac{\langle W^*, \dot{W} \rangle}{\|W^*\|_F} = -\frac{\langle W^*, \nabla_W L \rangle_F}{\|W^*\|_F}$$

585 and under the gradient flow dynamics directly on w , we get

$$\dot{w} = -\nabla_w L = -\left\langle \frac{dW}{dw}, \nabla_W L \right\rangle_F = -\frac{\langle W^*, \nabla_W L \rangle_F}{\|W^*\|_F}.$$

586 In the calculations above, we used $\langle \cdot, \cdot \rangle_F$ to denote the element-wise dot product (i.e.,
 587 $\langle A, B \rangle_F = \sum_{ij} A_{ij} B_{ij}$). The corresponding norm is the Froebenius norm $\|\cdot\|_F$.

588 – **Each column of the target weights W^* has norm 1.** It follows that the Froebenius norm of W^*
 589 satisfies $\|W^*\|_F^2 = d$. This is Assumption 3 from Section A.1.

590 Under these assumptions, studying the gradient flow dynamics on the original loss therefore reduce
 591 to study the gradient flow dynamics on the simplified loss

$$L = \frac{1}{2} \left[\sum_{t \leq T-B} \frac{(1 - B\alpha)^2}{d(T-B)^2} \frac{w^2}{\|W^*\|_F^2} \|W^*\|_F^2 + \frac{1}{d} \left(\frac{B\alpha w}{\|W^*\|} - 1 \right)^2 \text{tr}(W^* W^{*\top}) \right] \quad (11)$$

$$= \frac{1}{2} \left(\frac{(1 - B\alpha)^2 w^2}{d(T-B)} + \frac{(B\alpha w - \sqrt{d})^2}{d} \right) \quad (12)$$

592 with the attention given to token T being equal to

$$\alpha = \frac{1}{(T - B) \exp(-\Delta a) + B}.$$

593 Note that we have dropped the T subscript for notational conciseness. We plot this reduced loss
 594 landscape in Figure 6, as well as how the parameters evolve under the gradient flow dynamics.

595 As $d_{\Delta a} \alpha = \alpha(1 - B\alpha)$, the gradient of this loss with respect to w and Δa are

$$\nabla_w L = \frac{(1 - B\alpha)^2 w}{d(T - B)} + \frac{B\alpha(B\alpha w - \sqrt{d})}{d} \quad (13)$$

$$\nabla_{\Delta a} L = \alpha(1 - B\alpha) \left(-\frac{B(1 - B\alpha)w^2}{d(T - B)} + \frac{Bw(B\alpha w - \sqrt{d})}{d} \right). \quad (14)$$

596 The entries of the Hessian are

$$\frac{d^2 L}{dw^2} = \frac{(1 - B\alpha)^2}{d(T - B)} + \frac{B^2 \alpha^2}{d}$$

$$\frac{d^2 L}{dw d\Delta a} = \alpha(1 - B\alpha) \left(-\frac{2B(1 - B\alpha)w}{d(T - B)} + \frac{B(2B\alpha w - \sqrt{d})}{d} \right)$$

$$\frac{d^2 L}{d\Delta a^2} = \frac{1 - 2B\alpha}{\alpha(1 - B\alpha)} \nabla_{\Delta a} L + \alpha(1 - B\alpha) \left(\frac{B^2 w^2}{d(T - B)} + \frac{B^2 w^2}{d} \right).$$

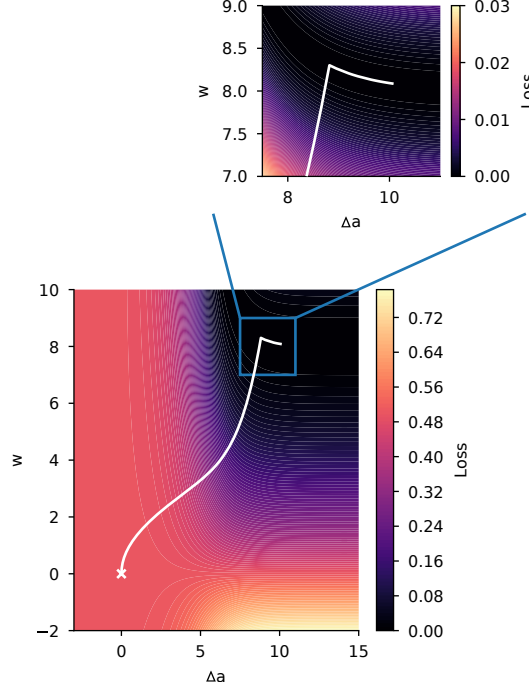


Figure 6: Loss landscape for the reduced model ($T = 256$, $d = 64$, without any repetition). The white line corresponds to the gradient flow on this loss, initialized in $(0, 0)$ (white cross).

597 A.3.2 Approximate dynamics around initialization

598 Despite having reduced the dynamics to two dimensions, it is still too complex to be analyzed
 599 mathematically. In order to gain insight into the early behavior of the dynamics, we linearize the
 600 dynamics around initialization and compute the time it will require to leave the initial loss plateau.

601 **Linearized dynamics at initialization.** Linearizing the dynamics at initialization corresponds to
 602 considering the gradient flow on the quadratic approximation to the loss:

$$L(w, \Delta a) = L(0, 0) + \nabla L(0, 0)^\top \begin{pmatrix} w \\ \Delta a \end{pmatrix} + \frac{1}{2} \begin{pmatrix} w \\ \Delta a \end{pmatrix}^\top H(0, 0) \begin{pmatrix} w \\ \Delta a \end{pmatrix} + o(\|w\|^2 + \|\Delta a\|^2),$$

603 with $\nabla L(0, 0)$ the gradient of the loss and $H(0, 0)$ the Hessian of the loss at initialization. Plugging
 604 in the values the different variable take at initialization gives

$$\begin{aligned} \nabla_w L &= -\frac{B}{\sqrt{dT}} \\ \nabla_{\Delta a} L &= 0 \end{aligned}$$

605 and

$$\begin{aligned} \frac{d^2 L}{dw^2} &= \left(\frac{T-B}{dT^2} + \frac{B^2}{dT^2} \right) \\ \frac{d^2 L}{dw d\Delta a} &= -\frac{T-B}{T^2} \frac{B}{\sqrt{d}} \\ \frac{d^2 L}{d\Delta a^2} &= 0. \end{aligned}$$

606 Under Assumption 5 from A.1, $T \rightarrow \infty$, B stays negligible compared to T , and d is large enough
 607 so that \sqrt{d} is negligible in front of d , so that $\frac{T-1}{T^2}$ becomes $\frac{1}{T}$, $\frac{d^2 L}{dw^2}$ becomes $\frac{1}{dT}$ and it is therefore

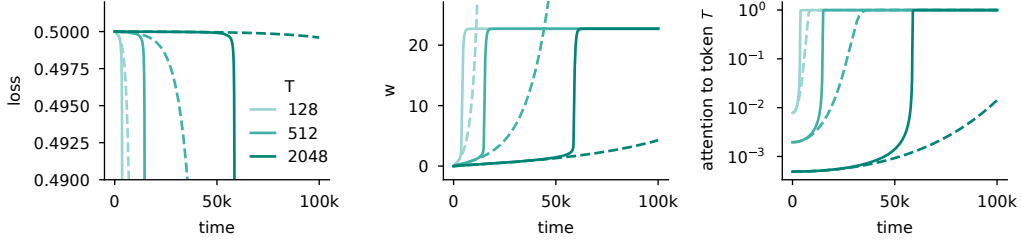


Figure 7: **Comparison of the early dynamics (plain lines) with their corresponding linear approximation around linear conditions (dashed lines).** d is here fixed to 512.

negligible in front of the cross-term second-order derivative. The gradient flow dynamics around initialization becomes

$$\begin{pmatrix} \dot{w} \\ \dot{\Delta a} \end{pmatrix} = \begin{pmatrix} \frac{B}{\sqrt{dT}} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & \frac{B}{\sqrt{dT}} \\ \frac{B}{\sqrt{dT}} & 0 \end{pmatrix} \begin{pmatrix} w \\ \Delta a \end{pmatrix} + o(\|w\| + \|\Delta a\|). \quad (15)$$

Figure 7 provides a visualization of how well these approximate dynamics match the original one.

The Hessian matrix has a positive eigenvalue $\frac{B}{\sqrt{dT}}$ and one negative one $-\frac{B}{\sqrt{dT}}$. This implies that the initial parameters are in the vicinity of an unstable fixed point and that the negative eigenvalue of the Hessian will dictate how fast we are escaping these initial conditions. Its eigenvalues are

$$\lambda_{\pm} = \pm \frac{B}{\sqrt{dT}}$$

with eigenvectors

$$v_{\pm} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ \pm 1 \end{pmatrix}.$$

Let P be the change of basis matrix defined by these eigenvectors. This matrix P transforms the original coordinates $(w, \Delta a)$ to the eigenbasis coordinates, (z_+, z_-) . The dynamics in the new basis is

$$\begin{pmatrix} \dot{z}_+ \\ \dot{z}_- \end{pmatrix} = \frac{B}{\sqrt{2dT}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \frac{B}{\sqrt{dT}} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} z_+ \\ z_- \end{pmatrix}.$$

This yields

$$\begin{aligned} z_+ &= \frac{1}{\sqrt{2}} (\exp(\lambda t) - 1) \\ z_- &= \frac{1}{\sqrt{2}} (1 - \exp(-\lambda t)) \end{aligned}$$

and

$$\begin{aligned} w &= \sinh(\lambda t) \\ \Delta a &= \cosh(\lambda t) - 1 \end{aligned}$$

Initial evolution of the loss. We can now derive a close-form equation for the temporal evolution of the quadratic approximation of the loss: as

$$L(w, \Delta a) = 0.5 - \lambda w - \lambda w \Delta a + o(\|w\|^2 + \|\Delta a\|^2),$$

we get that the loss is initially approximately equal to

$$\begin{aligned} L &\approx \frac{1}{2} - \lambda \sinh(\lambda t) - \lambda \sinh(\lambda t)(\cosh(\lambda t) - 1) \\ &= \frac{1}{2} - \lambda \sinh(\lambda t) \cosh(\lambda t) \\ &= \frac{1}{2} (1 - \lambda \sinh(2\lambda t)) \end{aligned}$$

using the identity $\sinh(x) \cosh(x) = \frac{1}{2} \sinh(2x)$.

624 **Time needed to escape the initialization plateau.** We can now compute the time T_ε it will take
 625 for the (approximate) loss to be equal to $(1 - \varepsilon)$ its initial value. From this definition, we get that T_ε
 626 satisfies

$$\frac{1}{2}(1 - \lambda \sinh(2\lambda t)) = \frac{1 - \varepsilon}{2},$$

627 that is

$$T_\varepsilon = \frac{\sqrt{d}T}{2B} \operatorname{arcsinh} \left(\frac{\varepsilon \sqrt{d}T}{B} \right).$$

628 Given that we are in the regime of large T and d , we finally get

$$T_\varepsilon = \frac{\sqrt{d}T}{2B} \log \left(\frac{2\varepsilon \sqrt{d}T}{B} \right)$$

629 by using the fact that $\operatorname{arcsinh}(x) \sim \frac{1}{2} \log(x)$ as $x \rightarrow \infty$.

630 A.3.3 Phase transition and late phase dynamics

631 Once we escape initial conditions and attention to the relevant token(s) starts increasing, the impact
 632 of the other tokens on the loss starts becoming negligible, the loss approximately becomes

$$L \approx \frac{1}{2d} (B\alpha w - \sqrt{d})^2. \quad (16)$$

633 This loss features multiplicative interactions akin to a one-hidden-layer neural network, and therefore
 634 it comes as no surprise that the loss exhibits similar sharp phase transitions as the ones observed
 635 when learning these networks, cf. Saxe et al. [28] for an in depth analysis of these dynamics (one
 636 needs to linearize the mapping $\Delta a \mapsto \alpha$ to get the exact mapping to this set of results).

637 The learning dynamics exhibits an interesting behavior after the phase transition: the projection of w
 638 on W^* starts decreasing, as seen on Figure 6 for example. In the following, we will argue that w is
 639 close to equilibrium in that phase and that learning consists mainly in slowly pushing Δa to infinity
 640 and that the corresponding equilibria decrease. We will show this by investigating the structure of the
 641 Hessian when w is at equilibrium.

642 First, let us compute the value that w takes at equilibrium, which requires solving the equation
 643 $\nabla_w L = 0$. Using Equation 13, it gives

$$\left(\frac{(1 - B\alpha)^2}{d(T - B)} + \frac{(B\alpha)^2}{d} \right) w = \frac{B\alpha}{\sqrt{d}},$$

644 that is

$$w^\infty(\alpha) := \left(\frac{(1 - B\alpha)^2}{d(T - B)} + \frac{(B\alpha)^2}{d} \right)^{-1} \frac{B\alpha}{\sqrt{d}}.$$

645 We plot w^∞ as a function of α in Figure 8.left.

646 Let us now consider the case in which $B\alpha$ converges to 1, that is $B\alpha = 1 - \varepsilon$ with $\varepsilon \rightarrow 0$. This gives

$$w^\infty = \frac{B\alpha d}{(B\alpha)^2 \sqrt{d}} + o(\varepsilon^2) = \frac{\sqrt{d}}{1 - \varepsilon} + o(\varepsilon^2).$$

647 We can then plug this value into the different components of the Hessian using the second derivatives
 648 we calculated for the simplified loss:

$$\begin{aligned} \frac{d^2 L}{dw^2} &= \frac{1}{d} - \frac{2\varepsilon}{d} + o(\varepsilon^2) \\ \frac{d^2 L}{dw d\Delta a} &= \frac{B\varepsilon}{\sqrt{d}} + o(\varepsilon^2) \\ \frac{d^2 L}{d\Delta a^2} &= \left(\frac{B + B^2}{T - B} + B^2 \right) \varepsilon + o(\varepsilon^2). \end{aligned}$$

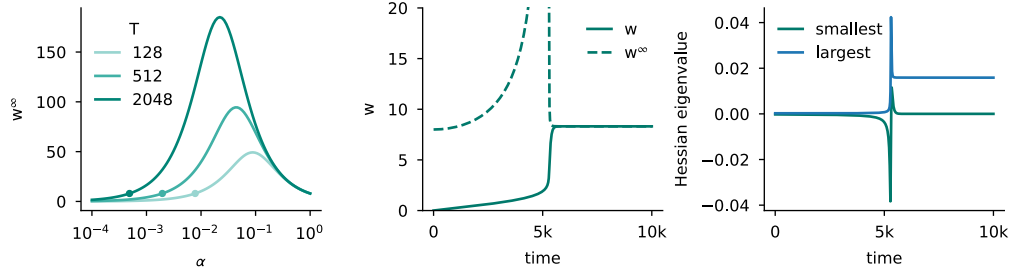


Figure 8: (left) w^∞ as a function of α for different T values ($d = 64$, $B = 1$). (middle) Evolution of w on the gradient flow dynamics ($T = 512$, $d = 64$, $B = 1$) and comparison with its instantaneous equilibrium value w^* . (right) Evolution of the smallest and largest eigenvalue of the Hessian of the loss along the gradient flow dynamics.

649 The Hessian is thus equal to

$$H(w^\infty, \Delta a) = \begin{pmatrix} \frac{1-2\varepsilon}{d} & \frac{B\varepsilon}{\sqrt{d}} \\ \frac{B\varepsilon}{\sqrt{d}} & \left(\frac{B+B^2}{T-B} + B^2 \right) \varepsilon \end{pmatrix} + o(\varepsilon^2)$$

650 when w^∞ is at optimality and $B\alpha$ close to 1 (i.e. $\Delta a \rightarrow \infty$). As ε goes to 0, the loss becomes
651 increasingly flat in the Δa dimension, highlighting that Δa will be the bottleneck in terms of learning
652 and w will always be at its equilibrium values w^∞ . We confirm this in simulation in the middle panel
653 of Figure 8.

654 It is now worth comparing the loss landscape structure in this late phase compared to the one early on
655 during learning. In the early stages, the eigenvalues are small (B/\sqrt{dT}) and w and Δ both contribute
656 to them. At the end of learning, the loss is much sharper in the w direction ($1/d$) than in the Δa
657 direction dimension (ε), because of the softmax within the attention. From this analysis of the two
658 extremes of the learning dynamics, in discrete time, w would be the bottleneck in terms of the learning
659 rate, due to the final sharpness of the loss. A complete picture is hard to obtain analytically because
660 calculations become more involved in the middle of learning, so we resort to simulations and plot the
661 results in Figure 8.right. We find that the behavior we described analytically holds for reasonably
662 long in the early and late dynamics. In the middle of the dynamics, the geometry of the loss landscape
663 changes drastically and the loss is the sharpest at this time (and thus the maximum learning rate we
664 can take in discrete time will be dependent on geometry of the loss around the phase transition).

665 A.3.4 Empirical validation

666 In these sections, our aim is to verify whether our theory still captures the behavior of the model
667 when the assumptions we made are not met. In particular, we will relax Assumption 3 (W^* has unit
668 norm columns), 4 (W initialized to 0), 5 (long sequences) and partially Assumption 1 (gradient flow
669 dynamics). Indeed, we sample W and W^* from $\mathcal{N}(0, 1/d)$, take T to be 256 (thus finite), d to be
670 256 and consider stochastic gradient descent dynamics with batch size 32 and learning rate 1. We
671 report the comparison of the evolution of the loss, the attention α to the relevant to the T -th token,
672 and the projection w of the weights W on W^* in Figure 9 for both the reduced dynamics (Theory
673 line, as in Equations 13 and 14) and for simulations (with the parameters described above). Overall,
674 we find that the simplified dynamics is able to depict the ones of the actual model quite accurately.

675 A.4 Analysis of the dynamics with cross-sample repetition

676 When considering cross-sample repetition, Σ_T will have a larger magnitude in the direction of the
677 token \tilde{x} that appears more frequently during learning. As a result, weights learn faster in that direction
678 and having a single scalar to capture the behavior of the entire weight matrix is no longer possible. In
679 the following, we show how to reduce it to two scalars using similar techniques as before, as well as
680 proceed to the analysis.

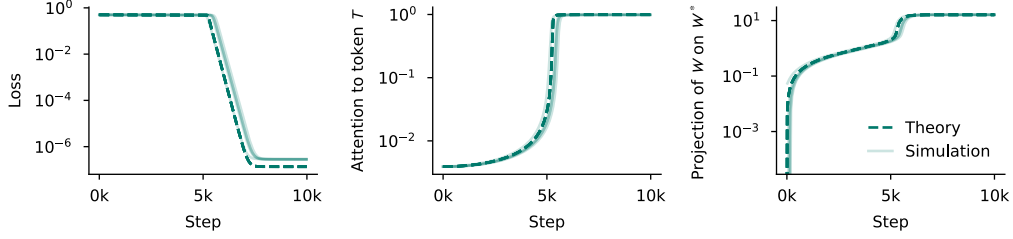


Figure 9: **The gradient flow dynamics on the simplified loss of Equation 12 match the stochastic gradient descent dynamics of the original objective.** We report the dynamics of 5 different seeds for the simulation lines. Changing the seed modified the target mapping, the sampling process, as well as the model initialization. Additional details are provided in Section A.3.4.

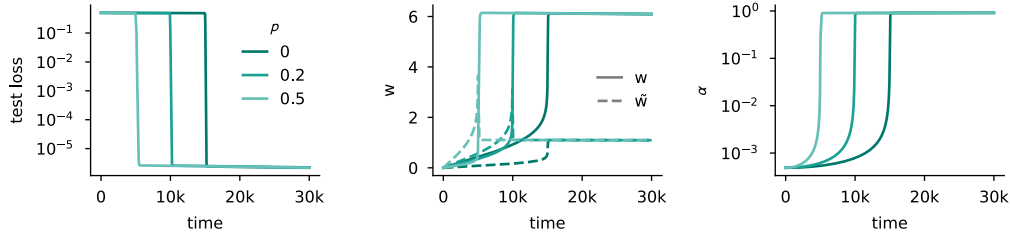


Figure 10: **Cross-sample repetition speeds up emergence.** This is because the repeated component \tilde{w} (dashed lines in the middle plot) learns faster, which leads to an earlier increase of attention α to the relevant token (right) and faster learning overall (left). The results reported here were obtained by simulating the gradient flow on the loss of Equation 17 for $T = 512$ and $d = 64$.

681 A.4.1 Reducing the learning dynamics to three dimensions

682 Cross-sample repetition does not affect attention directly, so we can still capture there entire behavior
 683 of attention with

$$\alpha = \frac{1}{(T-1)\exp(-\Delta a) + 1}.$$

684 Note that we do not consider any in-context repetition here for simplicity, but all our results can be
 685 extended to this case. Let us now look at the weights. Without loss of generality, we can assume
 686 $\tilde{x} = [1, 0, \dots, 0]^\top$, which yields

$$\Sigma_T = \text{diag}\left(p + \frac{1-p}{d}, \frac{1-p}{d}, \dots, \frac{1-p}{d}\right).$$

687 The gradient of the loss with respect to the i -th column $W_{:i}$ of the weight matrix therefore becomes

$$\nabla_{W_{:i}} L = \sum_{t>T-B} \alpha_t \left(p\delta_{i=1} + \frac{1-p}{d} \right) \left(\sum_{t>T-B} \alpha_t W_{:i} - W_{:i}^* \right) + \sum_{t\leq T-B} \frac{\alpha_t^2}{d} W_{:i}.$$

688 Importantly, when initialized at 0, each column of W will evolve on the line spanned by the
 689 corresponding column of W^* . However, they will not move at the same speed as the first column has
 690 different dynamics than the other columns. We therefore introduce two scalars \tilde{w} and w such that
 691 $W_{:1} = \tilde{w}W_{:1}^*$ and $W_{:i} = \frac{w}{\sqrt{d-1}}W_{:i}^*$. As for the analysis in the previous section, the $\sqrt{d-1}$ scaling
 692 factor is here to ensure that the learning speed coincide. To keep the notation as concise as possible,
 693 we introduce $\tilde{\sigma} := p + \frac{1-p}{d}$ and $\sigma := \frac{1-p}{d}$.

694 Under these assumptions, the loss simplifies to

$$L = \frac{1}{2} \left(\frac{(1-\alpha)^2(w^2 + \tilde{w}^2)}{d(T-1)} + \tilde{\sigma}(\alpha\tilde{w} - 1)^2 + \sigma(\alpha w - \sqrt{d-1})^2 \right). \quad (17)$$

695 The gradients flow dynamics on this loss are reported on Figure 10. The gradients are

$$\nabla_{\tilde{w}} L = \frac{(1-\alpha)^2 \tilde{w}}{d(T-1)} + \alpha \tilde{\sigma}(\alpha \tilde{w} - 1) \quad (18)$$

$$\nabla_w L = \frac{(1-\alpha)^2 w}{d(T-1)} + \frac{\alpha(1-p)(\alpha w - \sqrt{d-1})}{d} \quad (19)$$

$$\nabla_{\Delta a} L = \alpha(1-\alpha) \left(-\frac{(1-\alpha)(w^2 + \tilde{w}^2)}{d(T-1)} + \tilde{w} \tilde{\sigma}(\alpha \tilde{w} - 1) \right. \quad (20)$$

$$\left. + w \sigma(\alpha w - \sqrt{d-1}) \right). \quad (21)$$

696 The entries of the Hessian are

$$\begin{aligned} \frac{d^2 L}{d\tilde{w}^2} &= \frac{(1-\alpha)^2}{d(T-1)} + \alpha^2 \tilde{\sigma} \\ \frac{d^2 L}{dw^2} &= \frac{(1-\alpha)^2}{d(T-1)} + \alpha^2 \sigma \\ \frac{d^2 L}{dw d\tilde{w}} &= 0 \\ \frac{d^2 L}{d\tilde{w} d\Delta a} &= \alpha(1-\alpha) \left(-\frac{(1-\alpha)\tilde{w}}{d(T-1)} + (2\alpha\tilde{w} - 1) \tilde{\sigma} \right) \\ \frac{d^2 L}{dw d\Delta a} &= \alpha(1-\alpha) \left(-\frac{(1-\alpha)w}{d(T-1)} + (2\alpha w - \sqrt{d-1}) \sigma \right) \\ \frac{d^2 L}{d\Delta a^2} &= \frac{1-2\alpha}{\alpha(1-\alpha)} \nabla_{\Delta a} L + \alpha(1-\alpha) \left(\frac{(w^2 + \tilde{w}^2)}{d(T-1)} + (\tilde{\sigma}\tilde{w}^2 + \sigma w^2) \right) \end{aligned}$$

697 A.4.2 Approximate dynamics around initialization

698 Around initialization we get

$$\frac{d}{dt} \begin{pmatrix} \tilde{w} \\ w \\ \Delta a \end{pmatrix} = \begin{pmatrix} \frac{\tilde{\sigma}}{T} \\ \frac{\sigma\sqrt{d-1}}{T} \\ 0 \end{pmatrix} + \begin{pmatrix} -\frac{T-1}{dT^2} - \frac{\tilde{\sigma}}{T^2} & 0 & \frac{\tilde{\sigma}(T-1)}{T^2} \\ 0 & -\frac{T-1}{dT^2} - \frac{\sigma}{T^2} & \frac{\sigma(T-1)\sqrt{d-1}}{T^2} \\ \frac{\tilde{\sigma}(T-1)}{T^2} & \frac{\sigma(T-1)\sqrt{d-1}}{T^2} & 0 \end{pmatrix} \begin{pmatrix} \tilde{w} \\ w \\ \Delta a \end{pmatrix}$$

699 Under Assumption 5, this dynamics becomes

$$\frac{d}{dt} \begin{pmatrix} \tilde{w} \\ w \\ \Delta a \end{pmatrix} = \begin{pmatrix} \frac{p}{T} \\ \frac{1-p}{\sqrt{dT}} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & \frac{p}{T} \\ 0 & 0 & \frac{1-p}{\sqrt{dT}} \\ \frac{p}{T} & \frac{1-p}{\sqrt{dT}} & 0 \end{pmatrix} \begin{pmatrix} \tilde{w} \\ w \\ \Delta a \end{pmatrix}.$$

700 The characteristic polynomial of the Hessian is

$$X^3 - X \left(\frac{(1-p)^2}{dT^2} + \frac{p^2}{T^2} \right)$$

701 which means that the eigenvalues of the Hessian are 0 and

$$\pm \frac{1}{\sqrt{dT}} \sqrt{p^2 d + (1-p)^2},$$

702 which provides a scaling factor for the exit time of

$$\frac{\sqrt{dT}}{\sqrt{p^2 d + (1-p)^2}}.$$

703 As a sanity check, we can remark that when there is no cross-sample repetition, we recover the same
704 scaling factor as in our previous analysis. As p increases, it progressively removes the effect of the
705 dimension d in the escape time.

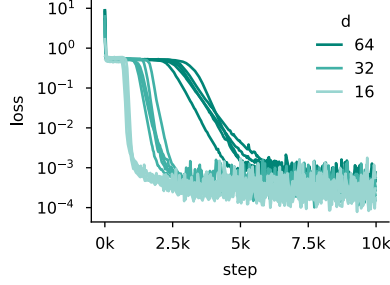


Figure 11: **The learning dynamics of Transformers exhibit sharp phase transitions in the single-location task.** Here, $T = 128$ and the architecture and training details are the one described in Appendix D.3.

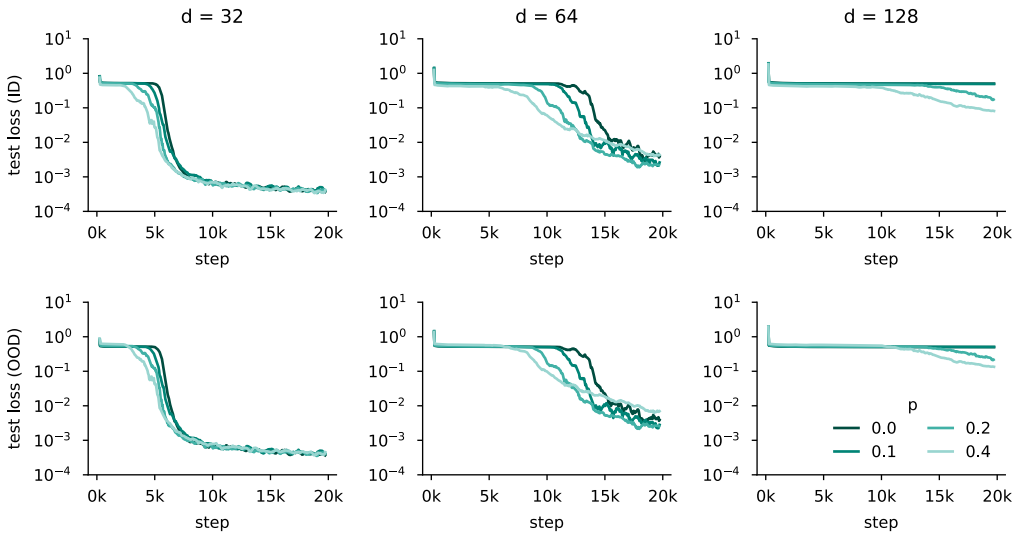


Figure 12: **Effects of cross-sample repetition on the learning dynamics of a Transformer in the linear regression task.** The top line corresponds to the test loss on the same data distribution than the one on which the network is trained, that is with repetition. The bottom line is data without any repetition. The results are obtained for $T = 256$. See Appendix D.3 for the rest of the training details.

706 B Details and additional analysis for the linear regression task

707 B.1 Implementation of the task

708 For our experiments, we implement a slightly different version of the task than the one we introduced
 709 in the main text and that we used for our theoretical analysis. The difference lies in where the relevant
 710 token is and whether there exists a feature to indicate it:

- 711 – **Random relevant token position.** In the simple version of the task, the relevant token was always
 712 presented at position T for convenience, as the output of the toy attention model we consider does
 713 not depend on the specific position. For experiments with more realistic Transformer models, we
 714 make this position random to avoid skip connections playing a specific role, and we resample it
 715 for every new sequence. The only exception to this is the task specific ablation of Figure 14, in
 716 which we sometimes fixed it throughout the entire training run (the relevant position being still
 717 randomly sampled at the beginning of training).
- 718 – **Relevant token feature.** As the position of the relevant token is resampled for every new
 719 sequence, we need to provide information to the network about which token is relevant for the

task. To this end, we append an extra feature to each input token that is 1 whenever the token is relevant. As in the previous point, the only experiment in which this feature is not added to the input is the ablation of Figure 14.

B.2 Examples of learning curves

Figures 11 and 12 provide examples of the learning dynamics of Transformers on the variation of the single-location linear regression task described in the previous section.

B.3 Additional ablations on the impact of task variation, model size and optimizer on plateau length

In the main text, we claim that the architecture, the details of the task, as well as the optimizer change the dependence of the plateau length on the different data parameters (here we study d and T). These changes collectively explain why the theoretical exponents of the power laws we obtained in our simplified regime do not directly translate to practice. This section provides the ablation underlying this claim; we detail them by increasing importance.

- **Architecture.** The toy model we consider has a single head and a single layer whereas the Transformer architecture we consider has 2 layers and 4 heads. In Figure 13, we ablate the number of layers and number of heads. We find that they have some importance, by slowing the learning process (note that we did not tune the learning rate to each architecture size). However, they do not significantly alter the dependency in d and T captured through the power law exponents.
- **Task specifics.** In the more realistic version of the task, the Transformer is provided with an extra input feature which indicates whether the token is relevant for the task, and the relevant token position is random. However, the toy model does not have access to this feature (it cannot use it) and the position is fixed. In Figure 14, we ablate these different choices and find that randomizing the position significantly increase the dependence in T and, to a smaller extent, the dependence on d . Interestingly, the power laws are almost the same for fixed position when the relevant feature is given to the model or not. We argue that this may be because the relevant feature provides a weaker statistical signal, likely because of the initial embedding layer from dimension d to 256, than the positional encoding. Learning to use would therefore be slower, and the network eventually ignores this feature.
- **Optimizer.** Our theoretical were obtained by analyzing the gradient flow dynamics. While gradient flow has tight links with gradient descent, and thus the emergence time under the gradient flow is approximately proportional to the number of steps before emergence (assuming small enough learning rates), this does not hold for Adam. To test how much Adam changes emergence time, we ablate the choice of the optimizer in Figure 15. We find that Adam to be significantly faster than SGD (up to almost two order of magnitudes for the hard versions of the task), both in absolute terms and in sensitivity to the difficulty of the task.

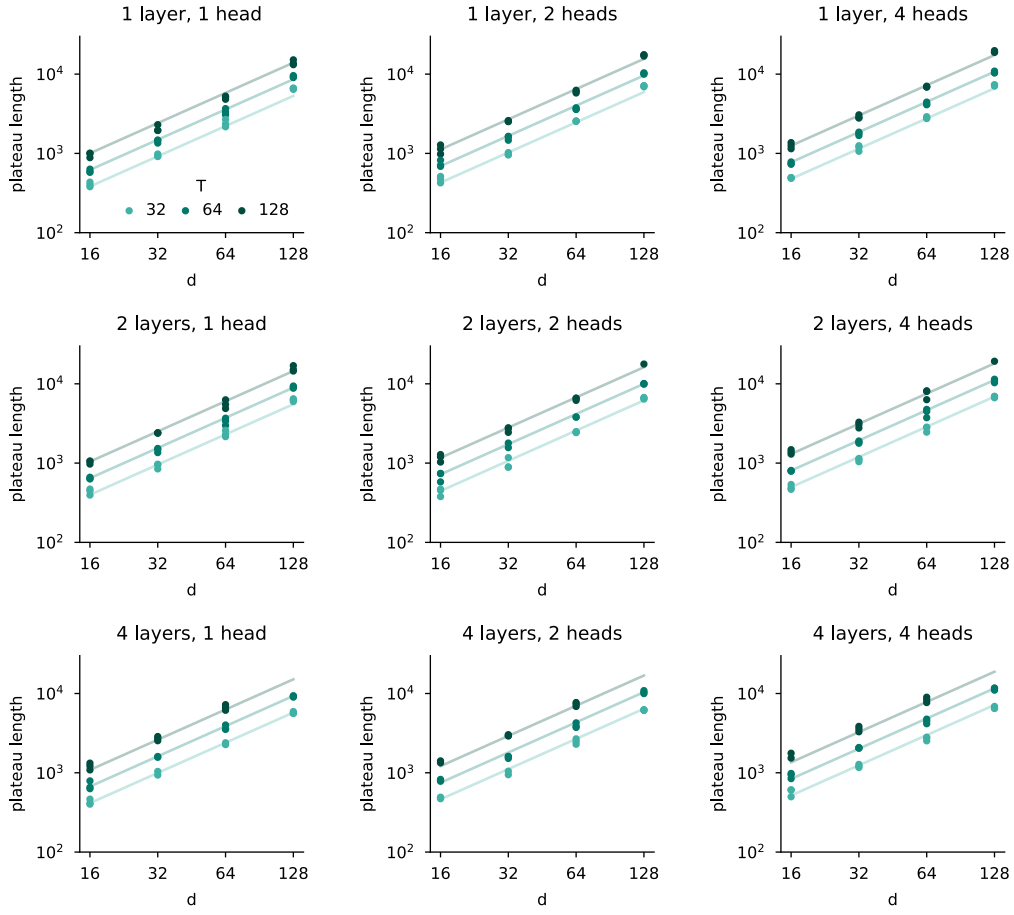


Figure 13: **Changing the width of the network (by increasing the number of heads) increases time spent on the initial plateau more significantly than increasing the depth of the network.** We plot the evolution of the plateau length, for Transformers with different number of layers and number of heads. We obtain the following scaling law: $T_{\text{plateau}} = 1.03 d^{1.27} T^{0.70} N_{\text{layers}}^{0.06} N_{\text{heads}}^{0.16}$ ($R^2 = 0.995$), which corresponds to the lines in the plots above.

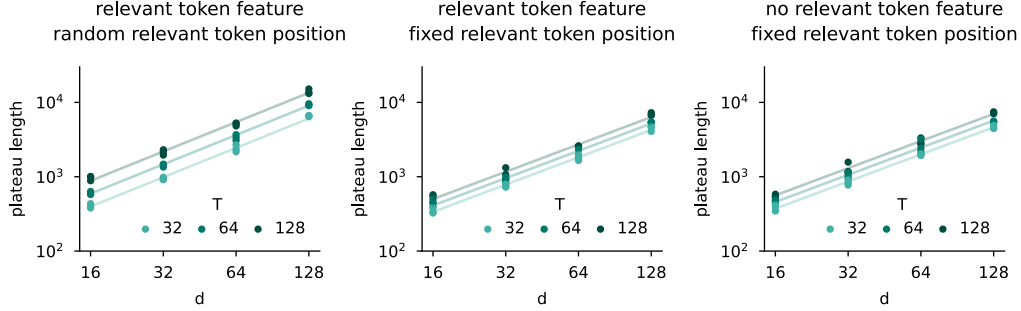


Figure 14: **Emergence is faster when positional information is available than when the network has to use semantic information.** In this set of experiments, we vary the nature of the task. The network can be given a "relevant token feature" indicating whether the token is relevant for the task is given (that is semantic information about the nature of the token). The position of the relevant token for the task can be either fixed or random. If it is random, the network must use semantic information to solve the task. If not, it (may) use positional information. In particular, this means that the network cannot solve the task when it has no access to the relevant token feature and the positions are random; this is why we do not report results in this configuration here. The different scaling laws plotted are:
(left) $T_{\text{plateau}} = 1.43 d^{1.31} T^{0.58}$, $R^2 = 0.998$ (relevant feature, random position).
(middle) $T_{\text{plateau}} = 4.20 d^{1.22} T^{0.29}$, $R^2 = 0.996$ (relevant feature, fixed position).
(right) $T_{\text{plateau}} = 4.61 d^{1.21} T^{0.30}$, $R^2 = 0.999$ (no relevant feature, fixed position).

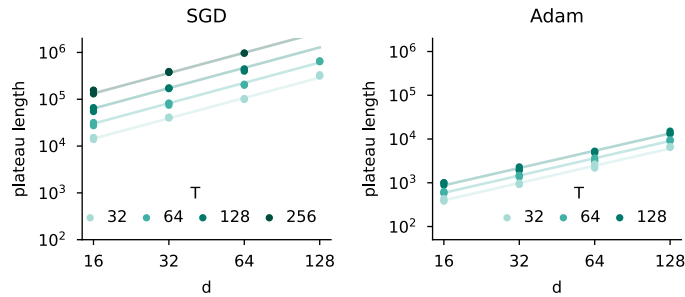


Figure 15: **Switching SGD for Adam leads to an important acceleration of learning.** Here, we train a single head single layer transformer with (left) SGD with a learning rate of $5 \cdot 10^{-3}$ and (right) Adam with a learning rate of 10^{-4} (these learning rates are manually tuned). Changing the optimizer has a huge effect on emergence time, both in terms of absolute time but also of scaling as the problem gets harder: $T_{\text{plateau}} = 6.42 d^{1.44} T^{1.07}$ ($R^2 = 0.997$) for SGD and $T_{\text{plateau}} = 1.45 d^{1.31} T^{0.57}$ for Adam ($R^2 = 0.998$).

C Details and additional experiments for the in-context associative recall task

C.1 Implementation of the task

Recall that each sequence consists of N_{pairs} key-value token pairs (5 such pairs in the example below) followed by a query token (Z below), as shown in the following example:

Y I A X U R Z Y C A Z ?

The number of possible tokens is the vocabulary size N_{tokens} and each of these tokens has a corresponding one-hot encoding. Both keys and values are sampled from the same pool of tokens.

The generative process behind the task is as follows:

- **Set up query distribution.** We define a query distribution p_{query} which corresponds to uniformly sampling one of the 2-repeated tokens with probability p and uniformly sampling all the tokens with probability $1 - p$ (recall that p is the cross-sample repetition probability). Said otherwise, this probability distribution has mass $\frac{p}{2} + \frac{1-p}{N_{\text{tokens}}}$ on the repeated tokens and mass $\frac{1-p}{N_{\text{tokens}}}$ on the rest of the tokens.
- **Sample the in-context mapping.** For each new sequence, we sample the mapping between keys and queries. This mapping is injective, meaning that two different keys will always be associated with different values. In practice, we sample a random permutation for this mapping.
- **Sample the query.** For each new sequence, we sample the query following the probability distribution p_{query} defined when creating the task. The target output for the task, will be the value indicated by the in-context mapping.
- **Fill the context.** For each new sequence, we finally fill the context as follows. We first decide whether we put the query (and its corresponding value) in the N_{pairs} possible positions by sampling a Bernoulli variable with success probability $\frac{B}{N}$ (on expectation the query thus appears B times in the context). We then fill the rest of the context by sampling keys from the $N_{\text{tokens}} - 1$ remaining tokens without replacement, and appending their corresponding values directly after.

C.2 Learning dynamics and additional analysis

We here provide two additional results: the learning dynamics on the training data, which show that repetition accelerates emergence in similar ways to what is predicted in our theory (Figure 16) and the evolution of the attention scores over the course of learning, which confirms that emergence occurs jointly with attention to relevant tokens getting sparser (Figure 17). We can make additional comments regarding the last point: it is interesting to see that all heads in the same layer have similar attention to relevant tokens, and that the copying mechanism (to group together the key and the value) only occurs in layer 3 and the selection mechanism only in layer 4.

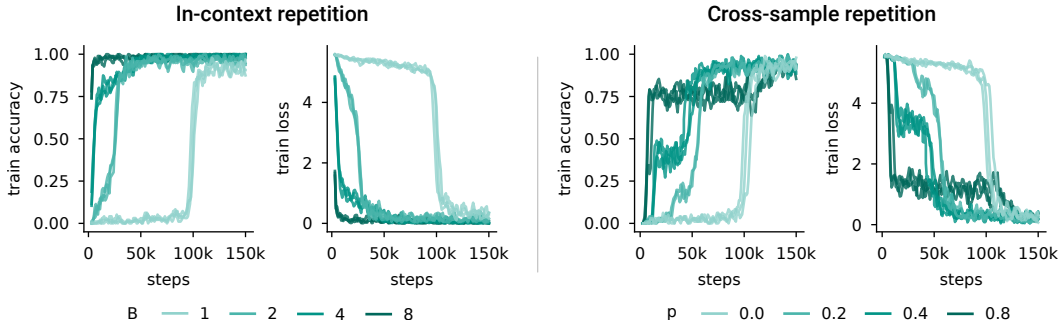


Figure 16: **Repetition accelerates emergence in the associative recall task as per the theory.** This plot is the training counterpart of the plots of Figure 5. As we measure the performance on the training data, there is no overfitting and repetition only speeds up emergence.

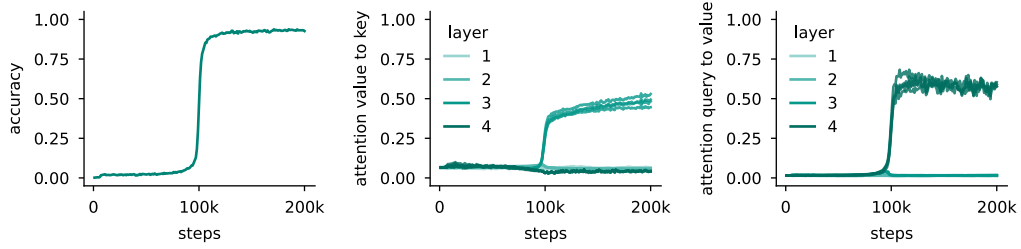


Figure 17: **The emergence of associative recall co-occurs with an increase of attention to relevant tokens.** (left) Evolution of the performance of the model over time, as in Figure 4 left ($N_{\text{pairs}} = 32$ and $N_{\text{tokens}} = 256$). There is no repetition in this experiment. (middle) Evolution of the attention from the relevant value token to the relevant key token (the first layer of a two-layers manually constructed attention head would have this attention value equal to 1). (right) Evolution of the attention from the query token to the relevant value token (the second layer of the induction head would get this value equal to 1).

D Methods

D.1 Measurement of the plateau length

To quantify the emergence time of sparse attention, we measure the duration of the initial plateau in the learning dynamics. We define the plateau length as the number of training steps required for the model to reach a specific threshold performance from initialization. This performance threshold depends on the task we consider:

- For the linear regression task with the toy model, we define the plateau length as the time required for the test loss to decrease to 0.2 of its initial value when following the gradient flow dynamics, that is 0.1. The test loss is the loss without any form of repetition for the vanilla dynamics (Figure 1.a) or for the dynamics with cross-sample repetition (Figure 2 right), and with the same B value as during training for in-context repetition. Note that we use a test loss with repetition for the latter as the toy model cannot generalize to the no repetition case.
- For the linear regression task learned with a Transformer, we use the same threshold as above, except for the cross-sample repetition where we used a threshold of 0.4. This threshold being different partially explains why the lighter lines in Figure 3 middle and right do not map (the other reason being that we additionally tuned the learning rate for the right plot). The reason behind this change comes from the shape of the learning curves: after emergence, the learning speed can be drastically different for different p values (see Figure 12, and hence we need it to be closer to 0.5, the initial loss value, to accurately capture the emergence time.
- For the in-context associative recall task, we pick an accuracy threshold, here 5%, as the random guess loss varies as a function of the vocabulary size.

It is important to note that in the first case, the plateau length corresponds to a physical time, whereas for the last two cases it corresponds to a number of optimization steps.

D.2 Fitting power laws on plateau length

To quantify the relationship between plateau length and various factors (sequence length T , input dimension d , in-context repetition B , cross-sample repetition probability p), we fit power laws to our empirical measurements using least squares regression on log-transformed data. We also fit power laws on the number of heads and the number of layers in 13 according to similar principles to what is detailed below.

For the general form of our scaling laws, we assume:

$$T_{\text{plateau}} = C d^{\alpha} \left(\frac{T}{B} \right)^{\beta} f(p, d)^{\gamma} \quad (22)$$

where C is a constant, α , β , and γ are the scaling exponents, and $f(p, d) = \sqrt{p^2 d + (1 - p)^2}$ is a function of the repetition probability (which is rooted in our theoretical analysis of Appendix A.4. Note that when considering the associative recall task, the vocabulary size is replacing d and the number of pairs is replacing T .

We perform the fitting by linearizing this relationship through a logarithmic transformation:

$$\log T_{\text{plateau}} = \log C + \alpha \log d + \beta \log \left(\frac{T}{B} \right) + \gamma \log f(p, d) \quad (23)$$

and performing a linear regression to find the missing coefficients. We report the score (R^2) of the linear fit as a measure of fit quality for all scaling laws. Most reported scaling laws had $R^2 > 0.98$, indicating excellent fit to the empirical data. For the cross-sample repetition results of Figure 3 right, where we could not fit accurately any power law on the obtained results.

More precisely, the exact parameters we fit for each plot are:

- Figure 1.c: C , α and β (B is fixed to 1).
- Figure 2 left: C , α and β (T is fixed to 4096).
- Figure 2 right: C , α , β and γ , such that $2\alpha = \beta = \gamma$ (T is fixed to 4096).

- Figure 3 left and middle: C , α and β . The same power law is fitted on the data of the two plots.
- Figure 4 right: C , α and β (B is fixed to 1).

D.3 Architectural and training details

We report the default hyperparameters we use in our experiments in Table 1 and report the deviations we make to this setup below:

- In Figure 3 right, we additionally tune the learning rate (in $\{10^{-4}, 3 \cdot 10^{-4}\}$) based on the shortest average plateau length as we found cross-sample repetition to significantly change the learning rate the Transformers we considered need.
- In Figure 4 right, we additionally tune the learning rate (in $\{10^{-5}, 3 \cdot 10^{-5}, 10^{-4}\}$) as large vocabulary sizes and number of pairs need smaller learning rates.
- In Figure 13, we ablate the number of layers and the number of heads in the Transformer architecture.
- In Figure 15, we ablate the choice of Adam as optimizer and replace it by stochastic gradient descent. Note that we here consider a single layer and a single head.

It should be additionally noted that we adapt the number of iterations depending on the difficulty of the task and roughly aim to end training significantly after emergence. However, for the most challenging versions of the task (e.g., large T values in Figure 3), the phase transition sometimes occurs after the maximum number of training steps we attribute to the task (in that case 50k steps).

Parameter Type	Theory	Linear regression	Associative recall
Architecture Parameters			
Model	Toy model	Transformer	Transformer
Layers	1	2	4
Number of heads	N/A	4	4
Embedding dimension	N/A	256	256
QK dimension	N/A	64	64
MLP factor	N/A	4	4
Positional encoding	N/A	sinusoidal	sinusoidal
Layer normalization	No	No	No
Skip connections	No	Yes	Yes
MLP between layers	No	Yes	Yes
Training Parameters			
Optimizer	GD	Adam	Adam
Learning rate	1.0	10^{-4}	10^{-4}
Scheduler	None	None	None
Batch size	Full	32	32
Seeds	N/A	5	3

Table 1: Default hyperparameters for the different types of experiments.

D.4 Reproducibility

Our theoretical simulations were run in JAX [61] and our Transformer experiments in PyTorch [62]. All our experiments were run on Nvidia RTX 3090 GPUs. The typical training run takes one hour, although training on shorter sequences can be significantly shorter. The code in the supplementary.