Universal limits to parallel processing capability of network architectures

Giovanni Petri,1,∗ Sebastian Musslick,2 Biswadip Dey,2,3 Kayhan Özçimder,2,3 Nesreen K. Ahmed,4 Theodore Willke,4 and Jonathan D. Cohen2

1ISI Foundation, via Chisola 5, 10126 Turin, IT.
2Princeton Neuroscience Institute, Princeton University, Princeton, NJ 08540, USA.
3Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08544, USA.
4Intel Labs, Santa Clara, CA 95054, USA.

(Dated: July 22, 2018)

The ability to learn new tasks and generalize performance to others is one of the most remarkable characteristics of the human brain and of recent AI systems. The ability to perform multiple tasks simultaneously is also a signature characteristic of large-scale parallel architectures, that is evident in the human brain, and has been exploited effectively in more traditional, massively parallel computational architectures. Here, we show that these two characteristics are in tension, reflecting a fundamental tradeoff between interactive parallelism that supports learning and generalization, and independent parallelism that supports processing efficiency through concurrent multitasking. We formally show that, while the maximum number of tasks that can be performed simultaneously grows linearly with network size, under realistic scenarios (e.g. in an unpredictable environment), the expected number that can be performed concurrently grows radically sub-linearly with network size. Hence, even modest reliance on shared representation strictly constrains the number of tasks that can be performed simultaneously, implying profound consequences for the development of artificial intelligence that optimally manages the tradeoff between learning and processing, and for understanding the human brains remarkably puzzling mix of sequential and parallel capabilities.

I. INTRODUCTION

There is a fundamental tension between two kinds of use for parallel distributed computing in network architectures. The first focuses on incorporating a variety of interacting constraints in the learning and processing of complex representations (‘interactive parallelism’). This has been profitably exploited in theories of human cognitive function [1] and most recently in the design of “deep learning” artificial systems [2]. In contrast, a second kind of use focuses on the capacity of a network to carry out multiple processes independently (‘independent parallelism’). This approach has been exploited by massively parallel systems used in most modern computing clusters, and optimized by message-passing systems such as MPI [3] that seek to identify and distribute independent components of computation. What has been less well explored is the relationship between these two types of parallelism, and the consequences that this has for the design of adaptive systems. Recent work has begun to suggest that this tradeoff lies at the heart of human cognitive function [4]. Human behavior presents an interesting puzzle with regard to the capacity to perform multiple tasks simultaneously. On the one hand, we can effortlessly perform many kinds of tasks at the same time, such as walking, taking, and responding to our surroundings, all of which presumably involve extensive simultaneous computations. On the other hand, we are radically constrained in our ability to perform other kinds of tasks concurrently, such as planning a grocery list while simultaneously carrying out multidigit mental arithmetic. In cognitive psychology, this is attributed to a fundamental distinction between automatic and controlled processing, and the latter subject to cross-task interference and constraints on simultaneous performance [5, 6]. Early theorists proposed that controlled processing relies on a centralized, limited capacity processing mechanism (akin to the core of a traditional computer), thus explaining the dramatic limitation in the human ability to simultaneously perform multiple control-dependent tasks. Early on, however, an alternative interpretation of this limitation was offered: that constraints in parallel processing reflect local competition among the resources required to perform specific combinations of tasks (based on the overlapping, shared use of representations) [7–10]. While compelling, this proposal was not undergirded by formal analysis on the extent to which process overlap (i.e., shared use of representations) would constrain processing. Moreover, process overlap was thought to diminish as the overall size of the system increases, providing at best a weak account of the constraints observed for a system as large as the human brain. Recently, however, numerical work has suggested instead that constraints imposed by overlap may be more radical and, under some conditions, approach scale-invariance [11, 12]. Here, we give a general theoretical analysis of the problem and show that even modest degrees of shared representations impose radical constraints on the number of tasks that agents can perform simultaneously without the risk of interference from crosstalk. This defines a fundamental tension in network architectures between

* giosanni.petri@isi.it
the benefits that accrue from shared representations (i.e. flexibility of processing and generalization [2, 13, 14]) and their costs in terms of processing efficiency (i.e., the number of independent tasks that can be performed simultaneously [4]).

This explains the constraints in the human controlled processing capacity, suggesting that it reflects the purpose of control— to limit cross-talk among processes that share representations— rather than a limitation intrinsic to control mechanisms themselves [4, 11, 12]. The balance between controlled and automatic processes in human behavior may, in fact, represent an evolutionary and/or developmental optimization of the human brain between interactive and independent parallelism— an optimization that may be central to our remarkable ability for adaptive behavior. Hence, understanding this trade-off, and how the brain manages it, may be crucial in the design of human-level adaptive artificial systems.

II. RESULTS

A. Measures of task dependency predict parallel processing capability in a trained neural network

To consider the problem of concurrent parallel processing (multitasking) analytically, we first provide a formal definition of a task. Given an input space \( I \) of stimuli (e.g. colors) and an output space \( O \) of responses (e.g. verbal response), a task \( T : I \rightarrow O \) represents a mapping between the two (e.g. naming the color of a stimulus), such that the mapping is independent of any other, and that selection of a feature from its input space can be made independent of any other. Different tasks can share an input space, output space, or both (e.g., reading a color word such as ”red” and naming the color in which it is printed share an output space). When this occurs, there is the potential for the tasks to interfere with one another [11, 15]. Such interference can be made explicit by describing the task structure in the form of a (bipartite) task structure graph \( G_{TS} = G(I, O, T) \). \( G_{TS} \) makes the sharing of representations across tasks explicit (Figure 1a), in which \( I, O \) and \( T \) are respectively the sets of input spaces, output spaces and tasks. Whenever two tasks share an input node \( I \in I \) or an output node \( O \in O \) we assume that they are at risk of interference due to cross-talk and therefore should not be executed in parallel; we call this dependency structural because of the direct reliance on common resources [12]. In Figure 1a this is represented as the dependency between tasks \( a \rightarrow b \) and between \( b \rightarrow c \). Importantly, in addition to structural dependence, there can also be functional dependence between two tasks: this is the case whenever maps the input space (i.e. connects the input node) of the first to the output space (i.e. output node) of the second. In Fig. 1a \( a \) and \( c \) are functionally dependent via \( b \), because activating a stimulus in \( c \)’s input space does the same for \( b \) thus invoking a response to \( b \) that may conflict with the response to \( a \). These dependencies can be made explicit in a task dependency graph \( G_D \) derived from the bipartite graph, in which nodes represent tasks and edges their (structural or functional) dependence (see Methods and SI). This, in turn, can be used to determine the maximum parallel processing capacity of the network (i.e., the largest number of tasks it can perform simultaneously) by finding the largest set of independent tasks. In graph-theoretic terms, this corresponds to finding a maximum induced edge matching of \( G_{TS} \): a subset of tasks in which none of the tasks are either structurally or functionally dependent on one another. In Fig. 1b we show an example of induced matching (in orange). This is equivalent to finding the maximum independent set (MIS) of \( G_D \) [16]. The size of the MIS is the independence number \( \alpha \) of \( G_D \), which provides a measure of the maximum parallel processing capacity of a given task structure \( G_{TS} \).

This equivalence is key to our first main contribution: a neural network trained on a task structure characterized by graph \( G_{TS} \) with given topology and parallel capacity exhibits a maximum parallel capacity given by the independence number of the corresponding \( G_D \). To assess the correspondence of this theoretical measure of parallel processing capacity to the performance of an actual network, we trained a simple non-linear feed-forward network (see Figure 1c), with four layers, that has been used previously to simulate a wide array of empirical findings concerning human cognitive performance (e.g. [17–19]). The network architecture entails two input layers, one that encodes the current stimulus (stimulus layer) and another one that encodes the task to be performed on the stimulus (task layer). Both input layers project to a hidden layer that computes an internal representation of task-relevant input features of the stimulus. Finally, information encoded at the hidden layer is projected together with the task layer input to an output layer at which the response of the network is computed. Note that the projections from the task layer allow it to bias processing towards task-relevant stimulus information represented at the hidden layer, as well as task-relevant responses at the output layer, thereby shaping the representations of the input and output space respectively for each task.

We trained 480 networks on a range of different task environments \( G_{TS} \), varying both the total number of tasks (between 4 – 30) and the associated task structure graph \( G_{TS} \) (Figure 1c). Weight projections from the task layer to the hidden layer were fixed and impose similarity between tasks with respect to their shared input spaces. For each network \( N \), trained on task environment \( G_{TS}^N \), we constructed an empirical task dependency graph \( G_D \) by studying the similarity between single task representations encoded in the hidden and output layers [12]. A single task representation for a given layer corresponds to the average activity vector at that layer for a given task. Two tasks were considered to be structurally dependent if either their hidden layer representation or their output layer representation exceeded a Pearson correlation
FIG. 1. **Graph-theoretic measures predict parallel processing capacity.** (a) A bipartite task structure graph $G_{TS}$ describes tasks in terms of mappings from an input space to an output space. Each task corresponds to an edge in $G_{TS}$ from an input node to an output node. Two tasks are: i) structurally dependent if they converge to the same output node (left) or originate from the same input node (middle); ii) functionally dependent if their edges are connected by a third task (right). (b) Dependencies between tasks in $G_{TS}$ are expressed by the corresponding task dependency graph $G_{D}$: tasks are now represented as nodes, and they are connected if the two tasks are structurally or functionally dependent. The MIS of $G_{D}$ corresponds to the largest set of tasks that a network can execute in parallel without interference[12]. Its cardinality $\alpha = |\text{MIS}|$ is the maximal parallel processing capacity of the network $\Pi$. (c) A neural network is trained on a set of tasks, in which each task requires the network to map a set of features from the stimulus layer via a hidden layer to a set of features on the output layer. Each task is designated by a unit in an additional (task) input layer that projects to both the hidden and output layers. All tasks in the environment can be expressed in terms of a task structure graph $G_{TS}$ (as shown in (a)). The network is trained on all tasks by activating, on each trial, a particular task unit and an input unit corresponding to a stimulus feature in the set for that task, and requiring the network to activate the corresponding output unit. The average activity patterns at the hidden and output layers across all inputs under a given task are taken as the network’s representation of that task. The two resulting similarity matrices (for the hidden and output layers) are used to infer dependencies between tasks based on shared task representations, and to construct the empirical task dependency graph $G_{D}$, which we use to predict the empirical $\bar{\alpha}^N$. (d) $\bar{\alpha}^N$ matches the maximum number of tasks $\Pi^N$ that the network can perform with a mean accuracy above 90% (empirical maximum parallel processing capability) across 480 simulations with different task environments.

threshold of $\theta = 0.8$, indicating that they shared either a common input or output space respectively. A pair of tasks were assessed to be functionally dependent if there was a third task for which the hidden layer representation was similar to one task in the pair and the output layer representation was similar to the other task in the pair. The resulting MIS $\bar{\alpha}^N$ of the empirical dependency graph $G_{D}$ matches the MIS $\alpha^N$ of the theoretical dependency graph describing the task environment $G_{D}$. Critically, $\bar{\alpha}^N$ yields an almost perfect agreement with the largest task subset $\Pi^N$ that the network can perform with a mean absolute error below 0.1, corresponding to 90% accuracy (Figure 1d). We refer to $\Pi^N$ as the empirical maximum parallel processing capability of the network.

These results also highlight the crucial possibility of faithfully extracting $G_{D}$ from network activation data when the original $G_{TS}$ is not known. For example, it would be possible in principle to use correlation matrix of patterns of neural activity elicited by individual tasks (e.g., the pattern of firing of a neuronal population, or the multi-voxel pattern of activity in an fMRI image) to construct a dependency graph analogous to the one constructed for the neural network as described above, without the need of knowing the task structure graph a priori, suggesting the possibility of observing representation learning and predicting parallel processing capability in vivo.
FIG. 2. Graph-theoretic results for $\rho_\alpha$ and $\rho_\phi$. a) Comparison of estimated $\bar{k}_D$ and actual $k_D$ for dependency graphs obtained from task structure graphs with a range of average degrees $z$. For each value $z$, the task dependency graph $G_D$ was computed from its bipartite $G_{TS}$ with a fixed degree on the input nodes, and a binomial distribution of degrees for the output nodes (similar to an Erdős-Rényi graph, and following [11]). The lower plot shows the same information in distribution form, with discrete bins corresponding to $\bar{k}_D$ and the solid line corresponding to the actual distribution. b) MIS densities $\rho_\alpha$ for a set of generic networks with Gaussian degree distributions of varying widths, comparing the exact computation (dots) with the values predicted from Eq. 2 (solid lines) as a function of the Gaussian distribution mean $\mu$. The plots show that, for fixed $\mu$, increasing heterogeneity ($\sigma^2$) is associated with increased $\rho_\alpha$ (the inset shows a wider range of heterogeneities on a log-log scale). c) MIS densities $\rho_\alpha$ for dependency graphs, comparing the explicit calculation on $G_D$ with the analytical results obtained using Eq. 1 and Eq. 2 for a set of task structure graphs with binomial degree distribution on the output layer ($p = z/N$ where $z$ is the average layer degree and $N$ the layer size) and regular degree on the input layer. We find good association between the numerical and theoretical results. d) For various $\gamma$ (cardinality of the task subset of interest) and fixed average degree $\langle k \rangle = 5$, we show that the fractional mean expected performance (MEP) $\rho_\phi = \phi/N$ strongly decreases with the layer size $N$ of the task structure graphs. Dots are values computed directly on the graphs, while the solid line represents the values obtained summing Eq. 4. The predicted $\phi/N$ provides an upper bound because it is obtained in a locally tree-like approximation, while dependency graphs are more clustered by construction. The dashed line represents the size-independent $\rho_\alpha$.

B. Maximum parallel capacity estimation for dependency graphs or arbitrary size

The results above are encouraging. However, it is very hard to scale up the direct computation of $\Pi$, as for a network with $N$ tasks it requires the enumeration of all task subsets and thus scales as $O(N!)$. Using the MIS size computation affords a significant computational advantage, as the algorithmic complexity of computing $\alpha$ explicitly for a graph with $M$ nodes is $O(2^M)$ and thus $M$ scales as $O(N^2)$ in our case [20, 21]. Although efficient algorithms for specific classes of graphs exist [22–24], measuring $\Pi$ directly is impractical for graphs relevant to most real-world applications. More importantly, algorithmic solutions do not provide insights about the features of the task structure that are responsible for limiting parallel processing capacity. To gain such insight, we provide a graph ensemble formulation of the MIS problem in terms of the degree distribution of the task structure. This allows us to isolate apart the roles of graph density and heterogeneity, independently from the network size and make general observations about the relationship between task structure and task encoding policies in determining parallel processing capacity.

The first step is to link the task graph $G_{TS}$ with the corresponding task dependency graph $G_D$ by calculating the degree distribution of $G_D$ from that of $G_{TS}$. This can be done in a manner similar to the standard calculation of the number of second neighbours [25]. Consider a task graph $G_{TS}$ with $N$ input and output nodes, and a degree distribution $p_e(k_i, k_o)$ (or $p_{k_i k_o}(e)$ in short) for the input ($k_i$) and output ($k_o$) node degrees at the endpoints of task edge $e$ in $G_{TS}$. Following the dependency rules (see Fig. 1), the task dependency graph $G_D$ is the square of the line graph $L(G_{TS})$ of $G_{TS}$. This allows us to calculate the estimated degree $\bar{k}_e^D$ of task $e$ in $G_D$ as
\[ \bar{k}^D_e \approx \frac{(k_i - 1)(k_i^2)}{\langle k_i \rangle} + \frac{(k_o - 1)(k_o^2)}{\langle k_o \rangle} - \frac{(k_i - 1)(k_o - 1)(\langle k_i \rangle - 1)(\langle k_o \rangle - 1)}{M - 1} \]

where \( \langle \bullet \rangle \) are the expectation values of \( k_i \) and \( k_o \), and \( M \) is the number of edges in \( G_D \) (we refer the reader to the SI for full details on the approximation and the calculation). In Figure 2a (and the SI) we show that Eq. 1 gives good results for graphs of various densities and for various degree distributions. Note that \( \bar{k}^D \) is written in terms of the first two moments of \( p_{\alpha} \), recovering the previously observed connection between the heterogeneity of \( G_{TS} \) graph and that of the corresponding dependency graph. This will play an important role in the estimation of the density of MIS \( \rho_\alpha = \alpha/N \) of \( G_D \).

The second step is to write \( \rho_\alpha \) as a function of the degree distribution of \( G_D \). We do this by building on recent work by Lucibello and Ricci-Tersenghi [26], we calculated an estimation for \( \rho_\alpha \) based on a factor graph description of the maximum set packing problem, of which the independence number problem is a particular instance. Crucially, these expressions depend only on the graph’s degree distribution, which in turn allows to study the effect that different network topologies have on their parallel capacity to be studied. Exploiting the properties of the degree and excess degree (the degree a node reached following an edge) distributions [27], the \( \rho_\alpha \) estimate can be rewritten in terms of generating functions and takes the form:

\[ \rho_\alpha = \frac{\langle k \rangle}{\langle c \rangle} \left( 1 - p_\gamma/c \right) + \left[ M_k(t) - M'_k(t) \right] \]

where \( t = \log p_\alpha \), and \( p_\alpha \) needs to satisfy the self-consistent equation \( p_\alpha = E_{c'} \left( 1 - \frac{1}{\langle k \rangle} M'_k(t)^{\tilde{c}} \right) \). Here \( k \) is the node degree in \( G_D \), and \( \tilde{c} \) refers to the factor nodes’ degrees and excess degrees, which in the case of the MIS are fixed to \( c = 2, \tilde{c} = 1 \) (see Methods and SI), and \( M_k(t) \) is the generating function for the degree distribution \( p(k) \). For a generic \( G_{TS} \), Eq. 1 can be used to calculate the degree distribution of the corresponding \( G_D \), which, substituted in Eq. 2, gives \( \rho_\alpha \).

For a generic Gaussian-like distribution with mean \( < k > \) and variance \( \sigma^2 \), the moment generating function takes the form \( M_k(t) = e^{(k)t + \sigma^2t^2/2} \). Following the expression above, we obtain \( p_\alpha = \left[ \frac{1 - \langle k \rangle}{\langle k \rangle} \right] \left( \frac{\langle k \rangle}{\langle k \rangle} + \sigma^2 \ln p_\alpha \right) M(\ln p_\alpha) \) the numerical solution of which can be computed:

\[ \rho_\alpha = \frac{\langle k \rangle}{\langle c \rangle} \left( 1 - p_\gamma/c \right) + M(\ln p_\alpha)(1 - \langle k \rangle - \sigma^2 \ln p_\alpha) \]

In Figure 2b we show that this expression provides a good approximation of the behaviour of \( \rho_\alpha \) for increasing network density and for various levels of degree heterogeneity. Importantly, it provides an analytical grounding for the previous empirical observations that increased heterogeneity of task overlap for a given average density results in a higher \( \rho_\alpha \) [11, 12]. Here, we used Gaussian degree distributions to illustrate the impact of the density of the dependency graph, which depends, in turn, on the density of the task structure graph (referred to as task overlap in [11, 12]) and its degree heterogeneity: for a fixed size, dense and uniform graphs have a smaller MIS than sparse, heterogeneous ones. In summary, Eq. 1 can be applied to a \( G_{TS} \) of arbitrary size, providing an estimate of the degree distribution of the corresponding \( G_D \), which can be used in turn to estimate \( \rho_\alpha \). In Figure 2c we show that our analytical method accurately predicts the result of exact computation of \( \rho_\alpha \) for a wide range of \( G_D \) in which this was carried out (see SI for other \( G_{TS} \) topologies). These results justify the use of our size-independent predictions of Eq. 2 for \( G_{TS} \) of arbitrary sizes.

C. Effective parallel processing capacity

We showed that using the degree distribution of a dependency graph, it is possible to approximate well the independence density of the graph \( \rho_\alpha \) and, using that, the corresponding independence number \( \alpha_{G} = \rho_\alpha[p(k)] * N \) (where \( N \) is the number of nodes in \( G \)) for networks of arbitrary size.

It is important to note here that the independence number is specific to a particular set (or sets) of tasks; that is, the specified level of parallelism can only be achieved for the particular tasks that comprise a maximal independence set. The size of this set tells us what is the maximum number of tasks that a given neural architecture can possibly perform simultaneously, assuming that there are no constraints on task selection. Critically, however, this does not address a question that is likely to arise in naturalistic settings: how many tasks can the system be expected to perform simultaneously on average, given a probabilistically determined subset of tasks that are currently viable (i.e. desirable and/or possible). That is, what is the expected number \( \phi \) of independent tasks across all task subsets \( T' = \{t_0, \ldots, t_{\gamma-1}\} \) with cardinality \( |T'| = \gamma \)? This effective parallel processing capacity \( \phi \) can be written as

\[ \phi(\gamma, G_D) = \sum_{\beta=1}^{\gamma} p(\beta, \gamma, G_D) \beta \]

where \( p(\beta, \gamma, M) \) is the probability that exactly \( \beta \) out of \( \gamma \) nodes are independent from each other in \( G_D \). The latter requires the \( \beta \) nodes in \( G_D \) to be not linked with each other, and the remaining \( \gamma - \beta \) nodes to be connected to at least one of the first \( \beta \) tasks. For a graph \( G_D \) with \( M \) edges, we can write the probability of successfully executing \( 1 \leq \beta \leq \gamma \) tasks in \( T \) as:

\[ p_k(\beta; \gamma, G_D) \approx \left( 1 - \frac{\langle k \rangle}{2M} \right)^{\binom{\beta}{2}} \frac{(\beta(\gamma)^{\gamma-\beta+1})}{2M} \gamma^{-\beta} \]

For \( \gamma = 1, \)
the probability of executing the task is always 1 since a single task cannot interfere with itself. For comparison with \( \rho_\alpha \), in Figure 2d we plot the results for the mean expected performance (MEP) density \( \rho_\phi = \phi/N \), obtained by numerically integrating \( \phi(\gamma, G_D) \) for a range of task structure sizes. As expected, \( \rho_\phi \) decreases with increasing network density, as it becomes progressively less likely to randomly find independent tasks. Critically, the increase in \( \phi \) as a function of \( N \) is strongly sub-linear (see SI). As a consequence, increasing network size is associated with a rapid decline in the average expected fraction \( \rho_\phi \) of simultaneously executable tasks as opposed to the size-independent \( \rho_\alpha \). This highlights a fundamental limitation of network architecture: the very network fabric that supports interactive parallelism by sharing representations between tasks (e.g., for learning and/or generalization) induces limits on independent parallelism — that is, their ability to perform multiple tasks simultaneously.

III. DISCUSSION

The ability of neural network architectures to support parallel distributed processing has been exploited in a variety of domains, ranging from cognitive neuroscience [28–30] to recent advances in machine learning (e.g., “deep learning” [31–33]). Most approaches focused on the learning and processing of complex representations by simultaneously integrating a large number of constraints through fine-grained interactions among many units processing the same input in parallel — what we call interactive parallelism. In contrast, the traditional computer science approach to parallelism focused on exploiting large numbers of processors by separating jobs into as many independent components as possible, and then process them in isolation what we call here independent (or also “embarrassing”) parallelism. Remarkably, the human brain has the ability to integrate the use of both forms of parallelism — a feature that we suspect may be fundamental to its capacity for adaptation. However, the factors that influence the tradeoff between each, and how the brain decides which form to use, remain a mystery. Here, we contribute to unraveling this mystery by providing a formal description of the constraints that the degree of interaction among processes imposes on the capacity for independent parallelism in simple networks.

Our findings provide support for the idea that the striking limit in the number of control-dependent processes that humans can execute at one time [5, 6] may in fact reflect a constraint imposed by the shared representation among task processes encoded in the network — a constraint that engages the function of control mechanisms (to mitigate crosstalk) rather than reflecting a limitation that is intrinsic to control mechanisms themselves. One might ask why the human brain did not evolve or develop to avoid such crosstalk, for example by encoding tasks independently. In point of fact, it does exhibit the capacity for such parallelism in many domains (such as the realtime regulation of the many homeostatic systems for which it is responsible), and the ability to develop it in others (i.e., the automaticity associated with skill acquisition, such as learning to play a musical instrument, or to drive a vehicle). Still, the question remains: why do so many important behaviors (e.g. planning and language) exhibit strict constraints on simultaneous multitasking indicative of control-dependent processing?

One compelling response to this question made clear by advances in deep learning [2, 13, 14] is that shared representations are a powerful means for promoting efficient learning and generalization. Indeed, we have previously shown that there is a direct tradeoff in networks between speed of learning, on the one hand, and concurrent parallel processing capability on the other [4]. The findings we report here reinforce and further illuminate this point, making it clear that the tradeoff is not relaxed by increasing network size: while the maximum parallel capacity density \( \rho_\alpha \) that is, the theoretical limit in the number of simultaneously executable tasks grows linearly with the size of a network, the maximum expected parallelism \( \rho_\phi \) is subject to strict asymptotic limits. The problem is compounded with deeper networks, in which the likelihood of interaction increases across layers. An obvious means of mitigating the limits imposed by the development of shared representations would be the use of learning policies that weigh the advantages of speed of acquisition and/or generalization against the value of processing efficiency. This however demands an understanding of how different policies impact task structure and its effects on performance. Our results provide the tools and language to assess the impact of different learning policies and set a foundation for exploring the relative costs and benefits of independent vs. interactive parallelism, and how adaptive learning systems might be regulated to favor one vs. the other [34]. Finally, our findings are relevant to other formally similar problems similar, e.g. maximum channel capacity for coding problems [35], jobs with interfering schedules [36, 37] and register allocation [38].

Our present findings are limited by a number of factors, including the use of undirected binary dependency graphs, corresponding to the assumption that interference between processes is symmetric and all or none. In natural systems, of course, interactions can be both asymmetric and graded, a feature that can be captured by the use of directed weighted graphs. The generalization of our methods to such graphs present considerable challenges, and is an important direction for future research. Nevertheless, the correspondence of our theoretical results with the numerical analyses (implementing asymmetric and graded forms of interference) suggests that our findings may provide useful approximations for current applications, and a valuable foundation for future theoretical work.
[1] J. L. McClelland, D. E. Rumelhart, and G. E. Hinton, *The appeal of parallel distributed processing* (Cambridge, MA: MIT Press, 1986).
[2] Y. Bengio, A. Courville, and P. Vincent, IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 1798 (2013).
[3] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, Parallel computing 22, 789 (1996).
[4] S. Musslick, A. Saxe, K. Özczmder, B. Dey, G. Henselman, and J. D. Cohen, in 39th cognitive science society conference, GB (2017).
[5] M. Posner and C. Snyder, in *Information processing and cognition: The Loyola symposium* (1975) pp. 55–85.
[6] R. M. Shiffrin and W. Schneider, Psychological review 84, 127 (1977).
[7] C. D. Wickens, *Processing resources and attention*, Vol. 1991 (1991) pp. 3–34.
[8] D. A. Allport, Cognitive psychology: New directions 1, 12 (1980).
[9] D. E. Meyer and D. E. Kieras, Psychological review 104, 3 (1997).
[10] D. Navon and D. Gopher, Psychological review 86, 214 (1979).
[11] S. F. Feng, M. Schwemmer, S. J. Gershman, and J. D. Cohen, Cognitive, Affective, & Behavioral Neuroscience 14, 129 (2014).
[12] S. Musslick, B. Dey, K. Özçmider, M. M. A. Patwary, T. L. Willke, and J. D. Cohen, in 38th cognitive science society conference, PA (2016).
[13] R. Caruana, Machine learning 28, 41 (1997).
[14] J. Baxter, in *Proceedings of the eighth annual conference on Computational learning theory* (ACM, 1995) pp. 31–120.
[15] J. R. Stroop, Journal of experimental psychology 18, 643 (1935).
[16] F. Gavril, Networks 3, 261 (1973).
[17] J. D. Cohen, K. Dunbar, and J. L. McClelland, Psychological Review 97, 322 (1990).
[18] J. D. Cohen, D. Servan-Schreiber, and J. L. McClelland, The American journal of psychology 230 (1992).
[19] M. M. Botvinick, T. S. Braver, D. M. Barch, C. S. Carter, and J. D. Cohen, Psychological review 108, 624 (2001).
[20] B. Bollobás, *Modern Graph Theory*, Vol. 184 (Springer Science & Business Media, 1998).
[21] R. E. Tarjan and A. E. Trojanowski, SIAM Journal on Computing 6, 537 (1977).
[22] R. Quaddoura, World of Computer Science and Information Technology Journal (WCSIT) 4, 34 (2014).
[23] M. Grotschel, L. Lovasz, and A. Schrijver, in *Topics on Perfect Graphs*, North-Holland Mathematics Studies, Vol. 88, edited by C. Berge and V. Chvatal (North-Holland, 1984) pp. 325 – 356.
[24] F. Gavril, SIAM Journal on Computing 1, 180 (1972).
[25] M. Newman, *Networks: An Introduction* (Oxford University Press, Inc., New York, NY, USA, 2010).
[26] C. Lucibello and F. Ricci-Tersenghi, *International Journal of Statistical Mechanics* 2014, 1 (2014).
[27] M. E. Newman, Physical review letters 103, 058701 (2009).
[28] D. E. Rumelhart, J. L. McClelland, P. R. Group, et al., Cambridge, MA (1986).
[29] J. L. McClelland, D. E. Rumelhart, P. R. Group, et al., *Parallel distributed processing*, Vol. 2 (MIT press Cambridge, MA, 1987).
[30] R. C. O’Reilly and Y. Munakata, *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain* (MIT press, 2000).
[31] Y. LeCun, Y. Bengio, and G. Hinton, Nature 521, 436 (2015).
[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Nature 518, 529 (2015).
[33] E. Gibney, Nature 529, 445 (2016).
[34] K. Özcimder, B. Dey, S. Musslick, G. Petri, K. N. Ahmed, T. L. Willke, and J. D. Cohen, in *Proceedings of the 38th Annual Conference of the Cognitive Science Society* (London, GB, in press).
[35] S. Butenko, P. Pardalos, I. Sergienko, V. Shylo, and P. Stetsyuk, in *Proceedings of the 2002 ACM symposium on Applied computing* (ACM, 2002) pp. 542–546.
[36] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, in *Handbook of combinatorial optimization* (Springer, 1999) pp. 1–74.
[37] P.-J. Wan, O. Frieiher, X. Jia, F. Yao, X. Xu, and S. Tang, *Wireless link scheduling under physical interference model* (IEEE, 2011).
[38] S. Hack, D. Grund, and G. Goos, in *International Conference on Compiler Construction* (Springer, 2006) pp. 247–262.
[39] T. T. Rogers and J. L. McClelland, *Semantic cognition: A parallel distributed processing approach* (MIT press, 2004).
[40] G. E. H. David E. Rumelhart and R. J. Williams, Nature 323, 533 (1986).
[41] E. Hazan, S. Safra, and O. Schwartz, computational complexity 15, 20 (2006).
[42] S. Musslick, B. Dey, K. Özçmider, M. A. Patwary, T. L. Willke, and J. D. Cohen, “Controlled vs. automatic processing: A graph-theoretic approach to the analysis of serial vs. parallel processing in neural network architectures,” https://github.com/musslick/Multitasking (2016).
Appendix A: Materials and Methods

1. Neural Network Architecture and Processing.

We use a standard non-linear feed-forward network, with four layers, that has been used previously to simulate a wide array of empirical findings concerning human cognitive performance [17, 19, 39]. The network consists of two input layers, one of which represents the stimulus presented to the network and another that encodes the task that the network has to perform on this stimulus. Both input layers project to a hidden layer (compromising 100 units). The hidden layer and the task layer further project to an output layer that computes the network’s response. The real-valued activity of each input unit composes the current stimulus. Activated units in the task layer indicate the tasks that are being currently executed. Performing a single task corresponds to clamping the corresponding task unit to 1 (activated) while all other units are set to 0. Multitasking conditions are represented by activating multiple task units at the same time. Units in the hidden and output layers take values between 0 and 1, as determined by a logistic activation function applied to their net input. Stimulus input units are structured according to $D$ dimensions (subvectors of the stimulus pattern), each of which is comprised of a set of $D$ feature units with only one feature unit activated per dimension. Similarly, output units are organized into $D$ response dimensions, with only one of the $D$ response units permitted to be active within a response dimension. Each task is represented by a single task input unit that is associated with a set of unique, one-to-one mappings between the input units in one stimulus dimension and the output units in one response dimension, and that is independent of the mappings for all other tasks (see Figure 1c). The number of stimulus input dimensions and response dimensions was varied between 4 and 6 across environments. The task mappings were generated with the Erdős-Rényi model such that the number of overlapping tasks for a given stimulus input dimension $z$ varied between 1 and 5. For each environment $G_{NTS}$ a network was initialized with a set of small random weights and then trained using the backpropagation algorithm [40] to produce the task-specified response for all stimuli in each task until it reached a mean-squared error performance of 0.001. We constrained the learned representations of the network to reflect the task similarity structure of the environment $G_{NTS}$ by fixing the weights from the task units to the associative layer: Weight vectors for tasks relying on the same stimulus input dimensions were set to yield a Pearson correlation coefficient of value 1 whereas weight vectors for tasks of non-overlapping stimulus dimensions were uncorrelated.

2. Dependency graph extraction.

We follow the analysis described in [12] and focus on the representations (patterns of activity) over the hidden and output units, insofar as these reflect the computations carried out by the network required to perform each task. In particular, we are interested in the characteristics of these representations for each task, how they compare across tasks, and how these factors correspond to empirical parallel processing performance. The representations associated with each task can be characterized by calculating, for each unit in the hidden and output layers, the mean of its activity over all of the stimuli for a given task; this mean pattern of activity can then be used as a representation of the task. Correlating these patterns of activity across tasks yields a task similarity matrix that can be examined separately for the hidden and output layers of the network. This can then be used to assess the extent to which different tasks rely on similar or different representation within each layer of the network. Figure 1c provides an example of such similarity matrices (thresholded for similarity correlations above $r > 0.8$). Tasks that have similar representations over the hidden layer can be inferred to rely on the same input dimension – that is, they share an input component in the bipartite graph representation of the network – and tasks that are similar at the output layer can be inferred to share an output component. Accordingly, a task dependency graph $G_D$ (of the type shown in Figure 1b) can be constructed by measuring the patterns of activity observed in the network while it performs each individual task.

3. Computation of the MIS of the dependency graph

The Maximum Independent Set (MIS) problem[21] is a particular instance of a larger optimization problem class, called Maximum Packing Set (MPS) problem [41], which we introduce below. Given a set $S = \{1, \ldots, n\}$ and a collection of its subsets $S = \{S_i|S_i \subset S, c, e\}$ labeled by $= \{1, \ldots, k\}$. A set packing is a collection of the pairwise disjoint subsets $S_i$, its size is the packing number. The problem of finding the maximum packing number can be formulated as an
integer programming problem as follows:

$$\text{maximise } \sum_{i \in V} n_i, \quad \text{(A1)}$$

subject to

$$\sum_{i : r \in S_i} n_i \leq 1 \quad \forall r \in F \quad \text{(A2)}$$

$$n_i = \{0,1\} \forall i \in V \quad \text{(A3)}$$

We follow here the approach of [26], denote the variable nodes set to be $V$ and to each $i \in V$ associate a variable $n_i$ which takes values in $\{0,1\}$. Denote by $F$ be the factor factor nodes set, that contains the elements acting as constraints on the variables $n_i$. The edge set $E$ is then defined as $E = \{(i, r) | i \in V, r \in S_i \subset F\}$.

Denoting by $G$ the factor graph $G = (V, F, E)$ the problem specification can be rewritten as

$$\sum_{i \in \partial_r} n_i \leq 1 \quad \forall r \in F \quad \text{(A4)}$$

Summing the values of the $n_i$s for a configuration satisfying the condition above yields the size of the solution. Working in density this becomes:

$$\rho(n_i) = \frac{1}{N} \sum_{i \in V} n_i \quad \text{(A5)}$$

Note that all the results are defined in thermodynamic limit $N \to \infty$.

It is possible to write analytical expressions for the expected MIS density, which are exact in certain regimes (as defined by a ration between the average factor and variable nodes degrees, effectively amounting to local tree-likeness of the factor graph) and in the Replica Symmetry solution are given by [26]:

$$\rho^{RS} = \frac{\langle k \rangle}{\langle c \rangle} \left( 1 - \mathbb{E}_c (1 - \mathbb{E}_k \tilde{p}^k) \right)^\tilde{c} + \mathbb{E}_k (1 - k) \tilde{p}^k \quad \text{(A6)}$$

where $p_* = \mathbb{E} (1 - \mathbb{E}_k \tilde{p}^k)$

with $t = \ln p_*$ and $M_k(t)$ is the moment generating function of the degree distribution and the derivative of $M$ is intended in $t$.

$$\mathbb{E}_k [ (1 - k) p_*^k ] = \mathbb{E}_k p_*^k - \mathbb{E}_k [ k p_*^k ] = M_k(t) - M'_k(t) \quad \text{(A9)}$$

so the final expressions become:

$$p_* = \mathbb{E}_c \left( 1 - \frac{1}{\langle k \rangle} \frac{\partial t}{\partial p_*} M'_k(t) \right)^\tilde{c} \quad \text{(A10)}$$

These allow us to directly substitute the moment-generating functions corresponding to the degree distribution of interest. In the main text, we use a slightly different notation. This is due to the fact that for the case of MIS $c = 2$ for all factor nodes, so we can simplify the expressions.
FIG. 3. Effects of network heterogeneity on predicted $\rho$. The subplots shows $\rho_\alpha$ as a function of network average degree $<k>$ for a range of network degree heterogeneities. We consider here a Gaussian degree distribution with average $\mu \in (0, 20)$ and variance $\sigma^2$ proportional to $\mu$ as shown in the legends. The solid black line is the prediction for a Poissonian graph, while the dashed line is the corresponding analytical solution of Equation A12. We see very clearly that for larger $\sigma^2$, $\rho_\alpha$ systematically shifts to higher values, staying below the Poissonian solution for $\sigma^2 < 1$ and above it for $\sigma^2 > 1$.

For a generic Gaussian-like distribution, the moment generating function takes the form $M_k(t) = e^{\langle k \rangle t + \sigma^2 t^2/2}$. Following the expression above, we obtain:

$$p_* = \left[ 1 - \frac{1}{\langle k \rangle p_*} (\langle k \rangle + \sigma^2 \ln p_*) M_k(\ln p_*) \right]$$  \hfill (A11)

$$\rho_\alpha = \frac{\langle k \rangle}{\langle c \rangle} (1 - p_*^{c/(c-1)}) + M_k(\ln p_*) (1 - \langle k \rangle - \sigma^2 \ln p_*)$$  \hfill (A12)

While for a Poisson degree distribution, the generating function for Poisson distribution $M_k(t) = e^{\nu(e^t - 1)}$ and in our case with $\nu = <k>$, $t = \ln p_*$, yielding:

$$p_* = \left(1 - e^{<k>(p_*^{-1})} \right)^{c-1} \text{ yielding}$$  \hfill (A13)

$$\rho = \frac{<k>}{<c>} \left(1 - p_*^{c/(c-1)}\right) + (1 - p_* <k>) e^{<k>(p_*^{-1})}$$  \hfill (A14)

In Figure 3 we show that this expression captures well the behavior of $\rho_\alpha$ for increasing network density and for various levels of degree heterogeneity. Moreover, it gives an analytical grounding to the previously empirical observation that a larger heterogeneity of task overlap at fixed density results in a higher $\rho_\alpha$ [11, 42].

In the manuscript we focused on unimodal degree distributions for simplicity of explanation and for consistence with previous work by Feng et al. However, Eq. A10 gives good results also with more structured degree distributions. We give here the solutions for Gamma and Pareto distribution.

For a gamma distribution $p(k) = \frac{1}{\Gamma(\gamma)} k^{\gamma - 1} e^{-k/\beta}$ we obtain:

$$p_* = 1 - \frac{1}{p_* (1 - \beta \log p_*)} \left( \frac{1}{1 - \beta \log p_*} \right)^\gamma$$  \hfill (A15)

$$\rho = \frac{<k>}{<c>} \left(1 - p_*^{c/(c-1)}\right) + \left(1 - \frac{\gamma \beta}{1 - \beta \log p_*} \right) \left( \frac{\gamma \beta}{1 - \beta \log p_*} \right)^\gamma$$  \hfill (A16)
While for a Pareto distribution \( p(k) = \frac{\gamma k^\gamma}{k_m^{\gamma+1}} \), the density expression becomes:

\[
p_* = \left(1 - \frac{1}{<k>} p_*(\gamma^2 k_m^\gamma \Gamma[-\gamma, k_m \log p_*(\gamma - 1 - \frac{\gamma}{\log p_*} p_m^k))\right)
\]

\[
\rho = \frac{\gamma k_m^\gamma \Gamma[-\gamma, -k_m \log p_*]}{<c>} \frac{(-t)^\gamma + \gamma(- \log p_*)^{\gamma-1} + \frac{\gamma}{\log p_*} p_m^k}{1 - p_*/(c-1)}
\]

(A17)

(A18)

In Figure 5 we show the comparison of simulated and predicted \( \rho_\alpha \).

4. Computation of the expected performance

When we calculate the maximum parallel capacity \( \alpha \) for a certain task graph \( G_D \) with task-set (i.e. nodeset) \( T \) we are focusing only on one (or a few, in some cases) maximum independence sets. It is however important to ask a different question: given a certain task set \( T' \subset T \) with cardinality \( \gamma = |T'| \leq \alpha \), what is the average number of tasks that can be performed in parallel? Since we do not want to specify which specific task subset we are interested in, it is useful to rewrite this as follows:

\[
\phi(\gamma, \mathcal{G}^D) = \sum_{\beta=1}^\gamma (\gamma) p(\beta, \gamma, \mathcal{G}^D) \beta
\]

(A19)
where \( p(\beta, \gamma, M) \) is the probability that \( \beta \) nodes are independent from each other, \( \gamma - \beta \) are attached at least to one of first \( \beta \) (otherwise they would be independent too) in \( G_D \). Again, we would like to estimate this quantity using the degree distribution of \( G_D \) in order to be able to compute the expected performance. This can be easily visualized as in Figure 6: denote the degree of the \( \gamma \) nodes as degrees \( \{k_0, \ldots, k_{\gamma - 1}\} \): i) we need the \( \beta \) nodes to be independent, which for each pair of nodes happens with probability \( 1 - \frac{k_i k_j}{2M} \) (we are prohibiting the red edge in the Figure); ii) we then need the remaining \( \gamma - \beta \) nodes to connect to at least one of the \( M_{in} = \sum_{i=0}^{\beta - 1} k_i \) stubs belonging to the independent \( \beta \) nodes, and this happens with probability \( M_{in} k_i / 2M \) per node (we impose the existence of at least one yellow edge). Clearly \( M_{out} = M - M_{in} \). Putting these contributions together we finally arrive at the probability of executing successfully \( \beta \) out of \( \gamma \) tasks with degrees \( \{k_0, \ldots, k_{\gamma - 1}\} \):

\[
p(\beta; \gamma, G^D) \propto \left[ \prod_{i<j,i=0}^{\beta - 1} \left( 1 - \frac{k_i k_j}{2M} \right) \right] \left[ \prod_{i=0}^{\gamma - \beta - 1} k_i \right] \left( \frac{M_{in}}{2M} \right)^{\gamma - \beta} (A20)
\]

Naturally, the full probability should include the probability of the degree configuration \( p(\{k_0, \ldots, k_{\gamma - 1}\}) \) which for uncorrelated random graphs factorises in the product of \( \gamma \) times \( G_D \)'s degree distribution. So finally becomes:

\[
p(\beta; \gamma, G^D) = \left[ \prod_{i<j,i=0}^{\beta - 1} \left( 1 - \frac{k_i k_j}{2M} \right) \right] \left[ \prod_{i=0}^{\gamma - \beta - 1} k_i \right] \left( \frac{M_{in}}{2M} \right)^{\gamma - \beta} \left[ p(k_{G^D}) \right]^\gamma (A21)
\]

Specifying to the case of \( z \)-regular networks for simplicity, the previous equation takes a very simple form:

\[
p_z(\beta; \gamma, G^D) = \left( 1 - \frac{z^2}{2M} \right)^{\binom{\beta}{2}} \left( \frac{\beta z^{\gamma - \beta + 1}}{2M} \right)^{\gamma - \beta} p_z(\gamma; \gamma, G^D) = \left( 1 - \frac{z^2}{2M} \right)^{\gamma} (A22)
\]

This form of the equation is interesting because it makes very easy to see the size dependence of the expected performance on both the average degree and size of the original \( G_{TS} \): indeed in this case \( 2M = z N_I \), where \( N_I \) here is the number of tasks in \( G_D \) which can grow both by enlarging the size of \( G_D \) or by increasing its average degree. For a generic degree distribution with finite first and second moment, if we consider the three main factors in the probability as independent variables (which is a reasonable assumption for \( \gamma \ll N \) or equivalently \( M_{in} \ll M \), we can approximate the expressions above with the following:

\[
p_k(\beta; \gamma, G^D) = \left( 1 - \frac{\langle k^2 \rangle}{2M} \right)^{\binom{\beta}{2}} \left( \frac{\beta \langle k \rangle^{\gamma - \beta + 1}}{2M} \right)^{\gamma - \beta} p_k(\gamma; \gamma, G^D) = \left( 1 - \frac{\langle k^2 \rangle}{2M} \right)^{\gamma} (A23)
\]

which are the ones we use in the main text.

**Appendix B: Estimation of dependency graph degree sequence and distribution**

Consider an input-output pairing bipartite graph \( G_{TS} \) with the same number of input and output nodes \( N \), and denote its input degree distribution as \( p(s) \) and the output degree distribution as \( p(t) \). The line graph \( L(G_{TS}) \) of \( G_{TS} \) has node set \( V(L(G_{TS})) = E(G_{TS}) \) of \( G_{TS} \). Hence to each edge \( e \) in \( G_{TS} \), or equivalently node in \( L(G_{TS}) \), we can associate its extremal nodes’ degrees \( (s_t, t_s) \), giving the degree of \( e \) \( k^L_{G(TS)} = s_t + t_s - 2 \). If we denote the probability for a input node with degree \( s \) to be linked to a output node with degree \( t \) as \( p_{st} \), we can use it to generate the degree distribution for \( L(G_{TS}) \) as \( p^L_{G(TS)} = \sum_{s,t} p_{st} \delta_{s+t-2} \). We write the generating function for \( p_{st} \) as \( g_p(x, y) = \sum_{s,t=0}^{\infty} p_{st} x^s y^t \), which by substitution in the expression for \( p_k^L(G_{TS}) \), yields \( f(z) = \sum_{k=0}^{\infty} p_k^L(G_{TS}) z^k = \frac{1}{z} g_p(z, z) \) [27]. The corresponding excess degree distributions are obtained easily as

\[
q_{st} = \frac{(s + 1)p_{s+t+1}}{\langle t \rangle} \quad r_{st} = \frac{(s + 1)p_{s+t+1}}{\langle s \rangle} (B1)
\]

which can be then rewritten in the standard generating function formalism.

In order to obtain an estimate \( k^D_e \) for the actual degree \( k^D_e \) in the dependency graph \( G_D \) of a node \( e \) in \( L(G_{TS}) \) characterized by \( (s, t) \), we need to calculate what is the contribution to the degree coming from the closure of open
FIG. 6. Cartoon of the necessary conditions for \( \beta \) (3 in this case) tasks \( t_0, t_1, \ldots, t_{\beta-1} \) belonging to a subset \( T' \subset T \) with cardinality \( |T'| = \gamma \) (\( \gamma = 7 \) in the example) of the entire task universe \( T \) to be independent.

wedges, that is how many second neighbours the node has. This calculation can be performed in a similar way to standard calculation of the number of second neighbours\[25\]. In this case, however, we need to take care of the potential effect of the joint degree distribution \( p_{st} \).

The degree of node \( e \) in the dependency graph \( G_D \) is the sum of the degree of \( e \) in \( G_{TS} \) and the number of second neighbours reached from the input excess edges and from the output excess edges:

\[
k^D_e \simeq (s + t - 2) + (s - 1) \sum_{t'} t' q_{s,t'=t} + (t - 1) \sum_{s'} s' q_{s',t'=s} = (s - 1) \frac{\langle t^2 \rangle}{\langle t \rangle} + (t - 1) \frac{\langle s^2 \rangle}{\langle s \rangle} \tag{B2}
\]

This result holds exactly for sparse graphs but we show that it also gives good results for graphs with intermediate densities. In that case, however, we need to keep track of the possibility that input and output edges of a node in \( L(G_{TS}) \) might connect to the same node. For a node \( e \in \mathcal{V}_{L(G_{TS})} \) characterized by \((s, t)\), the expression for the degree correction takes the form

\[
k^D_e \simeq (M - 1)(s - 1)(t - 1) \sum_{s' = s, t'} \frac{t'}{M - 1} \sum_{s', t' = t} r_{s', t'} s' = \frac{(s - 1)(t - 1)(\langle s \rangle - 1)(\langle t \rangle - 1)}{M - 1} \tag{B3}
\]

. Collecting all terms together we arrive at:

\[
\tilde{k}^D_e \simeq (s - 1) \frac{\langle t^2 \rangle}{\langle t \rangle} + (t - 1) \frac{\langle s^2 \rangle}{\langle s \rangle} - (s - 1)(t - 1)(\langle s \rangle - 1)(\langle t \rangle - 1) \frac{1}{M - 1} \tag{B3}
\]

Formally, we can write \( \tilde{k}^D = \tilde{k}^D(s, t) \) and, similarly to above, this generates the degree distribution for the dependency graph \( p^D_k = \sum_{s,t} p_{st} \delta_{k, \tilde{k}^D(s, t)} \). In practice, calculating the solution for this distribution can be cumbersome, especially when the \( p_{st} \) does not have a well specified functional form. However, Eq. B3 can directly be used to generate the degree sequence for \( G_D \) from \( G_{TS} \). Note also that the final expression for \( \tilde{k}^D \) is written in terms of the first two moments of the task structure graph’s degree distribution \( p_{st} \), which plays a crucial role in the estimation of \( \rho_\alpha \) of the dependency graph. In Figure 7 we show the performance of Eq. B3 for two different network topologies, exponential graphs and scale-free graphs. It is easy to see that in the exponential case, by virtue of the definiteness of the moments the estimation is quite accurate, while for scale-free graphs its accuracy is reduced due to the divergence of the second moment of the degree distribution (for slopes between 2 and 3).
FIG. 7. **Estimated degrees in** $G_D$. We show here the comparison of the estimated degrees for individual nodes in the dependency graph with the actual corresponding degree in $G_D$. We show the results for two network topologies for the underlying task structure graph $G_{TS}$, exponential graphs with slopes (0.5, 0.7, 1, 1.5, top row) and scale-free graphs (with power law exponents 2.5, 2.8, 3.1, bottom row), for various layer sizes of the underlying task structure graph $N = 100$ (left), $N = 150$ (center) and $N = 200$ (right).