The Neural Race Reduction: Dynamics of Abstraction in Gated Networks

Andrew M. Saxe* [1 2 3]   Shagun Sodhani* [2]   Sam Lewallen [1]

Abstract

Our theoretical understanding of deep learning has not kept pace with its empirical success. While network architecture is known to be critical, we do not yet understand its effect on learned representations and network behavior, or how this architecture should reflect task structure.In this work, we begin to address this gap by introducing the Gated Deep Linear Network framework that schematizes how pathways of information flow impact learning dynamics within an architecture. Crucially, because of the gating, these networks can compute nonlinear functions of their input. We derive an exact reduction and, for certain cases, exact solutions to the dynamics of learning. Our analysis demonstrates that the learning dynamics in structured networks can be conceptualized as a neural race with an implicit bias towards shared representations, which then govern the model's ability to systematically generalize, multi-task, and transfer. We validate our key insights on naturalistic datasets and with relaxed assumptions. Taken together, our work gives rise to general hypotheses relating neural architecture to learning and provides a mathematical approach towards understanding the design of more complex architectures and the role of modularity and compositionality in solving real-world problems. The code and results are available at https://www.saxelab.org/gated-dln.

1. Introduction

While neural networks have led to numerous impressive breakthroughs (He et al., 2016; Vaswani et al., 2017; Amodei et al., 2016; Baevski et al., 2020; Mnih et al., 2015; Silver et al., 2017), our theoretical understanding of these

*Equal contribution [1]Gatsby Computational Neuroscience Unit & Sainsbury Wellcome Centre, UCL [2]FAIR, Meta AI [3]CIFAR Azrieli Global Scholar, CIFAR. Correspondence to: Andrew Saxe <a.saxe@ucl.ac.uk>.
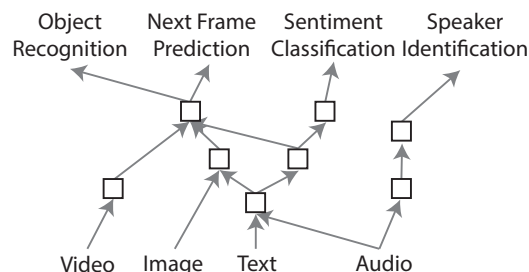
Figure 1. A multi-modal network composed of simple modules shared across modalities and tasks. How do shared modules and pathways impact representation learning and generalization?

models has not advanced at the same pace. Although we have gained some insight into general principles of network optimization (Patel et al., 2015; Carleo et al., 2019; Bahri et al., 2020; Arora et al., 2020; Roberts et al., 2021), we have a limited understanding of how the specific choice of architecture—that is, the mesoscale pattern of connectivity between hidden layers (Fig. 1)—affects a network's behavior (Zagoruyko & Komodakis, 2016; Raghu et al., 2017; Chizat et al., 2019; Saxe et al., 2019; Tian et al., 2019). For example, when training networks on the ImageNet dataset, wide networks perform slightly better on classes reflecting scenes, whereas deep networks are slightly more accurate on classes related to consumer goods (Nguyen et al., 2020). Understanding the reasons behind these behaviors may lead to more systematic techniques for designing neural networks.

Here, we address this gap by introducing and analyzing the Gated Deep Linear Network (GDLN) framework, which illuminates the interplay between data statistics, architecture, and learning dynamics. We ground the framework in the Deep Linear Network (DLN) setting as it is amenable to mathematical analysis, and previous works (Saxe et al., 2014; Lampinen & Ganguli, 2019; Arora et al., 2018; Saxe et al., 2019) have observed that DLNs exhibit several nonlinear phenomena that are observed in deep neural networks. Because of the gating mechanism, however, GDLNs can compute nonlinear functions of their input, making them more expressive than standard deep linear networks.

Our main contributions are: (i) We introduce the GDLN framework (Section 2), which schematizes how pathways of information flow impact learning dynamics within an architecture. (ii) We derive an exact reduction and, for certain cases, exact solutions to the dynamics of learning

(Section 3). (iii) Our analyses reveal the dynamics of learning in structured networks can be conceptualized as a neural race with an implicit bias towards shared representations (Section 4.2). (iv) We validate our findings on naturalistic datasets, with some relaxed assumptions (Section 5).

2. Gated Deep Linear Network Framework

A fundamental principle of neural network design is that powerful networks can be composed out of simple modules (Salakhutdinov, 2014a;b), and a key intuition is that a network's compositional architecture should resonate in some way with the task to be performed. For instance, a multimodal network might process each modality independently before merging these streams for further processing (Ngiam et al., 2011; Girdhar et al., 2022), and a multi-tasking NLP model might process its input through a shared encoder before it splits into task-specific pathways (Collobert et al., 2011; Liu et al., 2019b; Standley et al., 2020). Frequently, a network's compositional structure can be conceptualized as an "architecture graph" Γ, decorated by network modules, additional interactions, and learning mechanisms.

Here we introduce a class of networks, *Gated Deep Linear Networks* (GDLNs), depicted in Fig. 2a, for which we can analytically study the effect of the architecture graph on learning and generalization. A GDLN is defined as follows. Let Γ denote a directed graph with nodes $\mathbf{V}$ and edges $\mathbf{E}$, with the structure of Γ encoded by functions $s, t : \mathbf{E} \to \mathbf{V}$ mapping an edge to its source and target node, respectively. For each $v \in \mathbf{V}$, let $|v| \in \mathbb{N}$ denote the number of neurons assign to that node, and let $h_v \in \mathbb{R}^{|v|}$ denote neural activations for the corresponding network layer. For each edge $e \in \mathbf{E}$, let W_e denote a $|t(e)| \times |s(e)|$ weight matrix assigned to e. An *input node* of Γ is a node with only outgoing edges, and an *output* node is a node with only incoming edges. Let $\text{In}(\Gamma), \text{Out}(\Gamma) \subset V$ denote the sets of input nodes and output nodes of Γ, respectively.

The GDLN associated with Γ computes a function as follows: An input example specifies values $x_v \in \mathbb{R}^{|v|}$ for all input nodes $v \in \text{In}(\Gamma)$, and the input nodes are fixed to their values $h_v = x_v$ for $v \in \text{In}(\Gamma)$. Then, activation propagates to subsequent layers according to $h_v = g_v \sum_{q \in \mathbf{E}:t(q)=v} g_q W_q h_{s(q)}$ where g_v is the *node gate* and g_q is the *edge gate*. That is, activity propagates through the network as in standard neural networks, but modulated by *gating variables* that can act at the node and edge level.

In essence, these gating variables enable nonlinear computation from input to output, and can be interpreted in several ways: The node level gating can be viewed as an approximate reduction of ReLU dynamics as a ReLU neuron's activity can be written as $\max(0, wx) = \text{step}(wx)wx$. The gating variables can also be viewed as context-dependent
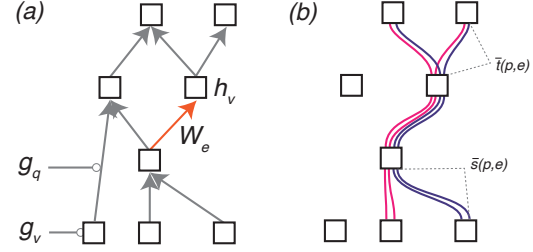

Figure 2. Formalism and notation. (a) The Gated Deep Linear Network applies gating variables to nodes and edges in an otherwise linear network. (b) The gradient for an edge (here using the orange edge in Panel (a) as an example) can be written in terms of paths through that edge (colored lines). Each path is broken into the component antecedent $\bar{s}(p, e)$ and subsequent $\bar{t}(p, e)$ to the edge.

control signals that modulate processing in the network. We discuss the extensive connections between the Gated Deep Linear Network and other approaches in Appendix 6.

In order to keep the analysis mathematically tractable, we assume that the gating variables are simply specified directly for each input that will be processed and we consider them to be a part of the dataset.

3. Gradient Flow Dynamics and Reduction

Having described the design of the network, we now exploit its special properties to understand learning dynamics and their relationship to network structure. In particular, we consider training all weights in the network to minimize the L_2 loss averaged over a dataset,

$$\mathcal{L}(\{W\}) = \left\langle \frac{1}{2} \sum_{v \in \text{Out}(\Gamma)} ||y_v - h_v||_2^2 \right\rangle_{x,y,g} \quad (1)$$

where $y_v \in \mathbb{R}^{|v|}$ for $v \in \text{Out}(\Gamma)$ are the target outputs for the output layers in the network, and $\langle \cdot \rangle_{x,y,g}$ denotes the average over the training dataset and gating structures. The weights in the network can be updated using gradient descent. A key virtue of the GDLN formalism is that the gradient flow equations can be compactly expressed in terms of the paths through the network.

We first lay out our notation, as illustrated in Fig. 2b for an example network. A path p is a sequence of edges that joins a sequence of nodes in the network graph Γ. Let $\mathcal{P}(e)$ be the set of all paths from any input node to any output node that pass through edge e. Let $\mathcal{T}(v)$ be the set of all paths terminating at node v. We denote the source and the target node of the path p as $s(p)$ and $t(p)$, respectively. We denote the component of path p that is subsequent to edge e (i.e., the path whose source node is the target of e, and that otherwise follows p) as $\bar{t}(p, e)$ (for the 'target' path of e). Similarly, we denote the component of path p that precedes edge e (i.e., the path whose target node is the source of e, and that otherwise follows p) as $\bar{s}(p, e)$ (for the 'source' path

of e). Overloading the notation, we will write W_p where p is a path to indicate the ordered product of all weights along the path p, with the target of p on the left and the source of p on the right. Similarly, we write g_p where p is a path to denote the product of the (node and edge) gating variables along the path.

With this notation, the gradient flow equations can be shown to be (full derivation in Appendix B),

$$\tau \frac{d}{dt} W_e = -\frac{\partial \mathcal{L}(\{W\})}{\partial W_e} \quad \forall e \in E, \tag{2}$$

$$= \sum_{p \in \mathcal{P}(e)} W_{\bar{t}(p,e)}^T \mathcal{E}(p) W_{\bar{s}(p,e)}^T \tag{3}$$

where the error term for path p is

$$\mathcal{E}(p) = \Sigma^{yx}(p) - \sum_{j \in \mathcal{T}(t(p))} W_j \Sigma^x(j,p). \tag{4}$$

Here the dataset statistics which drive learning are collected in the correlation matrices

$$\Sigma^{yx}(p) = \left\langle g_p y_{t(p)} x_{s(p)}^T \right\rangle_{y,x,g} \tag{5}$$

$$\Sigma^x(j,p) = \left\langle g_j x_{s(j)} x_{s(p)}^T g_p \right\rangle_{y,x,g} \tag{6}$$

where j and p index two paths. Hence if there are N paths through the graph from input nodes to output nodes, there are potentially N distinct input-output correlation matrices and N^2 distinct input correlation matrices that are relevant to the dynamics. Remarkably, no other statistics of the dataset are considered by the gradient descent dynamics.

Notably, these correlation matrices depend not just on the dataset statistics (x and y), but also on the gating structure g. The possible gating structures are limited by the architecture. In this way, the architecture of the network influences its learning dynamics.

In essence, the core simplification enabled by the GDLN formalism is that the gating variables g appear only in these data correlation matrices. They do not appear elsewhere in Eqns. (3)-(4), which otherwise resemble the gradient flow for a deep linear network (Saxe et al., 2014; 2019). The effect of the nonlinear gating can thus be viewed as constructing pathway-dependent dataset statistics that are fed to deep linear subnetworks (pathways).

As a simple example of the power of this framework relative to deep linear networks, consider the XoR task (Fig. 3a), a canonical nonlinear task that cannot be solved by linear networks. By choosing the gating structures to activate a different pathway on each example (Fig. 3b), the gated deep linear network can solve this task (Fig. 3c blue). Crucially, its dynamics (analytically obtained in Appendix A based on the reduction in the following sections) closely approximates the dynamics of a standard ReLU network trained
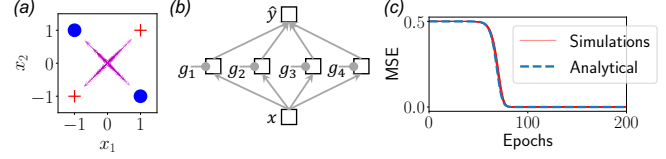

Figure 3. XoR solution dynamics. (a) The XoR task with positive (red) and negative (blue) examples. Input-to-hidden weights from ReLU simulations (magenta) reveal four functional cell types. (b) GDLN with four paths, each active on one example. (c) Simulations of ReLU dynamics from small weights (red, 10 repetitions) closely track analytical solutions in the GDLN. *Parameters:* $N_h = 128, \tau = 5/2, \sigma_0 = .0002$.

with backprop (Fig. 3c red). This result demonstrates that the gated networks are more expressive than their non-gated counterpart, and that gated networks can provide insight into ReLU dynamics in certain settings. We note that so far, our analysis does not provide a mechanism to select the gating structure. We will return to this point in Section 4.2, which provides a perspective on the gating structures likely to emerge in large networks.

3.1. Exact reduction from decoupled initial conditions

Our fundamental goal is to understand the dynamics of learning as a function of architecture and dataset statistics. In this section, we exploit the simplified form of the gradient flow equations to obtain an exact reduction of the dynamics. Our reduction builds on prior work in deep linear networks, and intuitively, shows that the dynamics of gating networks can be expressed succinctly in terms of effective independent 1D networks that govern the singular value dynamics of each weight matrix in the network. The reduced dynamics can be substantially more compact, as for instance, a weight matrix of size $N \times M$ has NM entries but only $\min(M, N)$ singular values.

To accomplish this, we introduce a change of variables based on the singular value decomposition of the relevant dataset statistics. Suppose that the dataset correlation matrices are mutually diagonalizable, such that their singular value decompositions have the form
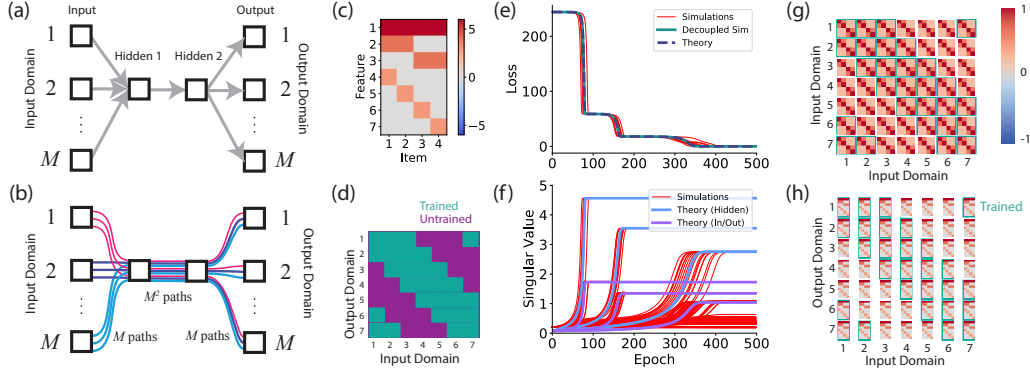
$$\Sigma^{yx}(p) = U_{t(p)} S(p) V_{s(p)}^T \tag{7}$$

$$\Sigma^x(j,p) = V_{s(j)} D(j,p) V_{s(p)}^T \tag{8}$$

where the set of U and V matrices are orthogonal, and the set of S and D matrices are diagonal. That is, there is a distinct orthogonal matrix U_l for each output layer, a distinct orthogonal matrix V_l for each input layer, and diagonal matrices $S(p), D(p)$ for each path through the network.

Then, following analyses in deep linear networks (Saxe et al., 2014), we consider the following change of variables. We rewrite the weight matrix on each edge as

$$W_e(t) = R_{t(e)} B_e(t) R_{s(e)}^T \quad \forall e, \tag{9}$$

Figure 4. Pathway network solution dynamics. (a) The network contains M different input domains (each consisting of a bank of neurons), M different output domains, and two hidden layers. The task is to learn a mapping from each input domain to each output domain. The gating structure gates on one input and one output pathway. The hidden pathway is always on. (b) Gated network formalism. There are M^2 pathways through the network from input to output. All M^2 flow through the hidden weight matrix, while only M flow through each input or output weight matrix. This fact causes the hidden layer to learn faster. (c) Small example dataset with hierarchical structure. The task of the network is to produce the 7-dimensional output vector for each of four items. Inputs are random orthogonal vectors for each item. (d) Each input domain is trained with K output domains (here $K = 4$), such that some input-output routes are never seen in training. (e) Training loss dynamics for simulated networks from small random weights (red, 10 repetitions), simulated networks from decoupled initial conditions (green), and theoretical prediction from Eqn. 15 (blue). The theory matches the decoupled simulations exactly, and is a good approximation for small random weights. (f) The singular values of the hidden weight matrix (blue) are larger than those in input or output matrices by a factor $\sqrt{M}$. Theoretical predictions match simulations well, particularly for larger singular values. (g) Representational similarity (or kernel) matrix at the first hidden layer. Inputs from different domains are mapped to similar internal representations, revealing a shared representation even for input domains that are never trained with a common output. (h) Predicted output at the end of training. The network generalizes perfectly to input-output routes that were never seen during training. *Parameters:* $M = 7, K = 4, \lambda = .02, \sigma_0 = .2, N_h = 64$.

where the matrices $B_e(t)$ are the new dynamical variables, and the matrix R_v associated to each node v in the graph satisfies $R_v^T R_v = I$. Further, for output nodes v, we require $R_v = U_v$, the output singular vectors in the diagonalizability assumption. Similarly, for input nodes, we require $R_v = V_v$.

Inserting (7)-(9) into (3)-(4) shows that the dynamics for B_e decouple: if all $B_e(0)$ are initially diagonal, they will remain so under the dynamics (full derivation in Appendix B). For this decoupled initialization, the dynamics are

$$\tau \frac{d}{dt} B_e = \sum_{p \in \mathcal{P}(e)} B_{p \backslash e} \left[S(p) - \sum_{j \in \mathcal{T}(t(p))} B_j D(j, p) \right] \tag{10}$$

where $B_{p \backslash e} = B_{\bar{t}(p,e)} B_{\bar{s}(p,e)}$ is the product of all B matrices on path p after removing edge e (see Appendix B).

In essence, this reduction removes competitive interactions between singular value modes, such that the dynamics of the overall network can be described by summing together several "1D networks," one for each singular value. Intuitively, this reduction shows that learning dynamics depend on several factors.

Input-output correlations Other things being equal, a pathway learning from a dataset with larger input-output singular values will learn faster. This fact is well known from prior work on deep linear networks

(Saxe et al., 2014).

Pathway counting Other things being equal, a weight matrix corresponding to an edge that participates in many paths (such that the sum contains many terms) will learn faster. This fact is less obvious, as it becomes relevant only if one moves beyond simple feed-forward chains to study complex architectures and gating.

We now turn to examples that verify and illustrate the rich behavior and consequences of these dynamics.

4. Applications and consequences

To fix a specific scenario with rich opportunities for generalization, we consider a "routing" setting, as depicted in Fig. 4a. In this setting, a network receives inputs from M different input domains and produces outputs across M different output domains. The goal is to learn to map inputs from a specific input domain to a specific output domain, with no negative-interference from other input-output domain pairs. There are thus M^2 possible tasks which can be performed, each corresponding to mapping one of the M input domains to one of the M output domains.

We assume that the target input-output mapping from the active input domain to the active output domain is the same for all pathways, and defined by a dataset with input correlations $\langle xx^T \rangle = VDV^T$ and input-output correlations $\langle yx^T \rangle = USV^T$. For the simulations in this section, we

take the dataset to contain four examples, and the target output to be a 7-dimensional feature vector with hierarchical structure (Fig. 4c), but note that the theory is more general.

To investigate the possibility of structured generalization, we consider a setting where only a subset of input-output pathways are trained. That is, each input domain is trained with only $K \leq M$ output domains, as depicted in Fig. 4d, such that some input-output pathways are never observed during training.

We consider solving this task with a two-hidden layer gated deep linear network depicted in Fig. 4a. We emphasize that this task is fundamentally nonlinear, because inputs on irrelevant input domains must be ignored. We take the gating structure to gate off all first layer pathways except the relevant input domain, and to gate off all third layer pathways but the one to the relevant output domain. As shown in Fig. 4b, this scheme results in M^2 pathways through this network that must be considered in the reduction. The resulting pathway correlations are simply given by the original dataset, scaled be the probability that each path is active (see Appendix C)

$$\Sigma^{yx}(p) \quad = \quad \frac{1}{KM} USV^T \qquad (11)$$

$$\Sigma^x(j,p) \quad = \quad \begin{cases} \frac{1}{KM} VDV^T & \text{if } j = p \\ 0 & \text{otherwise} \end{cases} \qquad . \quad (12)$$

Crucially, we have a simple "pathway counting" logic behind the reduction: the first and third layer weights are active in K paths (all tasks originating from a given input domain or terminating at a given output domain, respectively), while second layer weights are active in KM trained paths. This fact causes the second layer weights to learn more rapidly.

Assuming that weights start out roughly balanced in each first layer and third layer weigh matrix (a reasonable assumption when starting from small random weights), this yields the reduced dynamics (Appendix C)

$$\tau \frac{d}{dt} B_1 \quad = \quad \frac{1}{M} B_2 B_1 \left[S - B_2 B_1^2 D \right] \qquad (13)$$

$$\tau \frac{d}{dt} B_2 \quad = \quad B_1^2 \left[S - B_2 B_1^2 D \right] \qquad (14)$$

where B_1 describes the input and output pathway weights singular values, and B_2 describes the hidden layer weight singular values.

We note that the quantity $MB_1^2 - B_2^2$ is conserved under the dynamics. Defining the constant $C = MB_1(0)^2 - B_2(0)^2$, we can therefore write the dynamics as

$$\tau \frac{d}{dt} B_2 = \frac{1}{M}(B_2^2 + C) \left[S - \frac{1}{M} B_2(B_2^2 + C)D \right]. \quad (15)$$

Remarkably, this equation reveals that the dynamics of this

potentially large, gated, multilayer network with arbitrary numbers of hidden neurons can be reduced to a single scalar for each singular value in the dataset. Each diagonal element of this equation provides a separable differential equation that may be integrated to give an exact formal solution.

Figure 4e compares the training error dynamics predicted by Eqn. 15 to full simulations starting from small random weights (i.e. scaling Xavier initialiation weights by 0.2), verifying that our reduction is a good description of dynamics starting from small random weights. Furthermore, as shown in Appendix C, the hidden layer weight singular values change by a factor of $\sqrt{M}$ more than the input or output weights, as verified in Fig 4f.

4.1. Shared representations and generalization

With this description of the training dynamics of the network, we can then ask what representations emerge in the network over training. One way of interrogating the nature of representations in the network is to compute the representational similarity between different input examples from the same input domain, and across different input domains. Specifically, we compute the dot product between the neural activity in the first hidden layer in response to different inputs. As shown in Fig. 4g, the pathway network learns a *shared representation*, in which each individual example maps to the same representation regardless of what input domain it arrives on. That is, the gating and learning dynamics enable the network to learn a representation that is *invariant* to input domain, and which is abstract in the sense that the representation contains no information about what input domain produced it. This shared representation supports zero-shot generalization to untrained input-output pathways, as shown in Fig. 4h.

The intuition behind obtaining zero-shot generalization is as follows: Say that we are evaluating the network on a new input-output pair of domains. As long as the network has been trained on examples from the current input domain (in conjunction with any output domain), the network will map it to the shared representation. Similarly, as long as the network has been trained on examples from the current output domain, it will be able to map this shared representation to the output. In this way, training on a subset of M^2 tasks is enough to obtain strong generalization to all M^2 tasks.

In essence, this solution accomplishes a factorization of the problem into two interacting but distinct components: the gating variables represent what input domain links to what output domain, providing information about "where" signals should go; while the neural activity represents "what" task-relevant input was presented, regardless of where it came from or where it should be routed to. This factorization can permit generalization to untrained pathways provided the gating structure is configured appropriately.
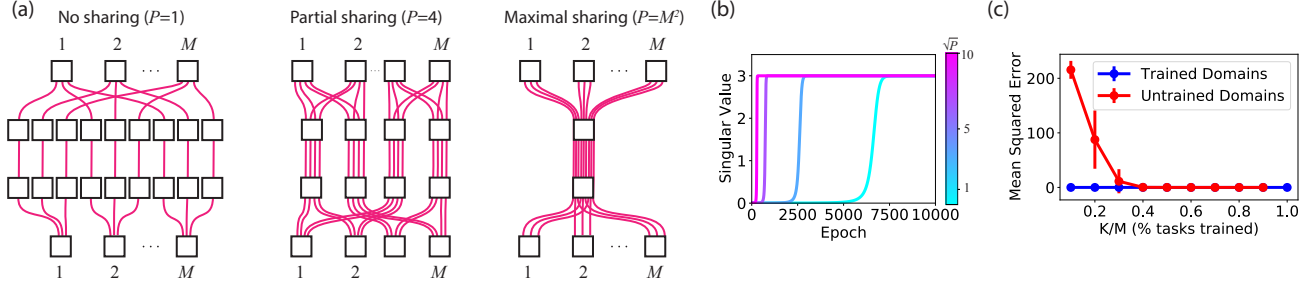
Figure 5. Pathway race dynamics. (a) The same routing task can be solved using a variety of gating schemes that differ in their use of shared representations. Every input-output combination can be given a dedicated pathway such that $P = 1$ tasks flow through each (left), groups of two input domains and two output domains can share a pathway such that $P = 4$ tasks flow through each (middle), or all $P = M^2$ tasks can run through a shared representation (right). (b) Singular value dynamics as a function of the number of pathways P flowing through the hidden layers. Networks that share more structure learn faster. Consequently, in a single network where subparts share structure to different degrees, the maximally shared dynamics dominate the race between pathways. (c) Error on trained (blue) and untrained (red) pathways as a function of the fraction of output domains K trained with each input domain M. When few outputs are trained per input domain, the race dynamics do not strongly favor shared structure and so error on untrained domains is large. When $\sim 40\%$ of output domains are trained with each input, the shared structure solutions are sufficiently faster to reliably dominate the race and yield generalization to untrained domains. *Parameters:* $M = 10, \lambda = .05/K, \sigma_0 = .2, N_h = 64$.

4.2. Neural race dynamics: Implicit bias toward shared representations

The reductions so far have assumed that the gating structure is specified *a priori*, and furthermore, that different domains connect to the same singular value modes in the hidden layer. That is, the gating structure provides the opportunity for learning a shared representation, but this is not obligatory: different parts of the hidden representation could learn distinct pathways, despite all being gated on. Remarkably, the dynamics from small random weights track the trajectory predicted for maximally shared representation, suggesting that the full solution dynamics rapidly converge to the submanifold of decoupled weights. Why are shared representations favored under the dynamics?

To investigate this, we note that the same task and architecture typically permit several gating schemes and singular value mode connectivity patterns that each would obtain zero training error. As shown in Fig. 5a (left), for instance, the routing task could be solved with an alternative gating structure in which each input-output route receives a dedicated pathway that is gated on only when the task is to connect that specific input-output route. This gating scheme would still obtain zero training error, but does not yield any representation sharing. Other partial sharing schemes are possible; for instance, representations could be shared across groups of two input and two output domains (Fig. 5a, middle). What is the impact of these choices on learning dynamics and generalization?

Taking the case where all routes are trained ($K = M$) for simplicity, with no sharing, each pathway participates in just one of the M^2 total trained pathways, compared to the fully shared solution where the input and output layers participate in M pathways and the hidden layer participates

in M^2. From this, we can see that greater sharing leads to faster learning. In a combined network that produces its output using both shared and non-shared representations, the dynamics of each pathway will race each other to solve the task; and hence the most-shared structure will dominate.

To see this quantitatively, we repeat a similar derivation to the preceding section for networks with varying degrees of pathway overlap. In particular, we parameterize the degree of pathway overlap with the parameter P that counts the number of pathways flowing through a given hidden layer. The resulting reduction is

$$\tau \frac{d}{dt} B_1 = \frac{\sqrt{P}}{M^2} B_2 B_1 \left[S - B_2 B_1^2 D \right] \qquad (16)$$

$$\tau \frac{d}{dt} B_2 = \frac{P}{M^2} B_1^2 \left[S - B_2 B_1^2 D \right], \qquad (17)$$

which shows that the learning rate in all layers increases as P increases. Solution dynamics for a range of degrees of sharing P are plotted in Fig. 5b, which show that greater degrees of sharing reliably leads to faster singular value dynamics.

Hence, dynamics in GDLNs take the form of a *pathway race*: when many gating schemes coexist in the same network, the ones that share the most structure–and hence learn the fastest–will come to dominate the solution. Therefore gradient flow dynamics in complex network architectures has an implicit bias toward extracting shared representations where possible. The strength of this bias increases as each input domain is trained with more output domains. As shown in Fig. 5c, shared representations begin to dominate reliably when roughly 40% of input-output routes are trained, enabling generalization to unseen input-output routes.
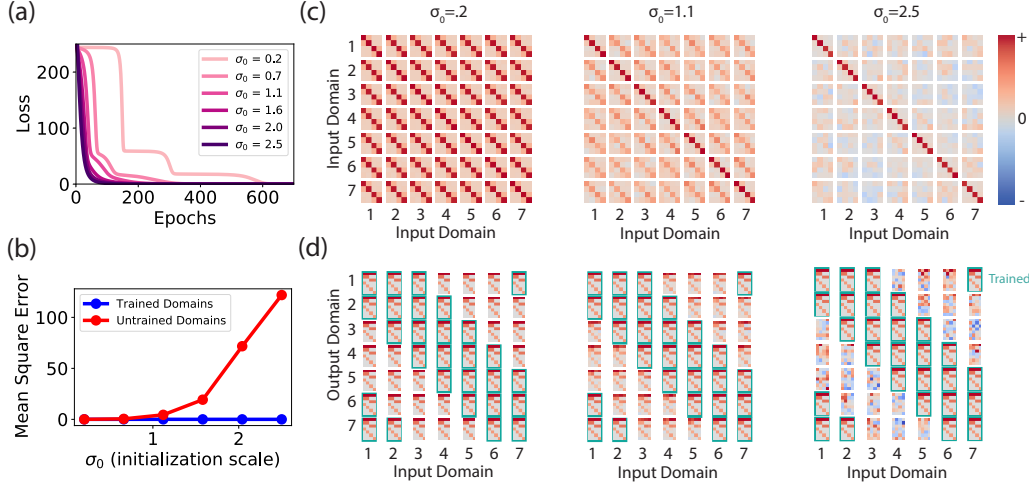
Figure 6. Neural race vs. NTK regime: Initialization, shared representation formation, and zero-shot generalization. (a) Training loss for pathway networks with different initialization scales σ_0. Small σ_0 yields pathway race dynamics with stage-like drops through training. Large σ_0 yields NTK-like dynamics with rapid, exponential learning curves. (b) Error for trained (blue) and untrained (red) input-output domain combinations as a function of initialization scale. While performance on trained domains is excellent for all scales, zero-shot generalization only emerges in the neural race regime. (c) Representational similarity between inputs presented to different domains, for small (left column), medium (middle column) and large (right column) initialization scales. At small initialization scales, internal representations in the first hidden layer are similar even across domains, indicating one common shared representation. Large initialization scales place the network in the NTK regime where random initial connectivity persists throughout learning, yielding distinct high-dimensional random representations for each domain. (d) Network output for all input-output combinations for three initialization scales (labeled in panel c). Because networks in the NTK regime do not learn a shared representation for different input domains, they do not generalize to untrained pathways.

4.3. Impact of initialization

The training and generalization dynamics of deep networks are known to depend on the weight initialization. Here we show that initial weight variance exerts a pronounced effect on the emergence of shared representations, and hence generalization abilities. As observed in a number of theoretical and empirical works, neural networks can operate in two different initialization regimes (Chizat et al., 2019; Bahri et al., 2020; Flesch et al., 2022). Sufficiently wide networks initialized with large variance initializations enter the Neural Tangent Kernel regime, where training dynamics follow a simple linear dynamical system and error trajectories exhibit exponential approach to their asymptote (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019c). Intuitively, in this regime, the initial strong random connectivity in the network provides sufficiently rich features to learn the task without substantially changing internal representations. In this setting, deep networks behave like kernel machines with a *fixed* kernel (the neural tangent kernel). By contrast, networks initialized with sufficiently small variance initializations learn rich task-specific representations, and their dynamics as we have seen can be more complex (Mei et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2020; Saxe et al., 2019).

To show the effect of this transition in our setting, we train pathway networks starting from different random matrices with singular value σ_0. For a range of initialization scales,

all networks converge to zero training error (Fig. 6a). As expected, large initialization scales lead to NTK-like exponential dynamics, while small initialization scales lead to progressive stage-like drops in the error consistent with prior analyses of deep linear networks in the rich regime. Critically, initialization has a dramatic impact on generalization (Fig. 6b), and only small initializations are capable of zero-shot generalization to untrained routes. To understand why, we visualize the representational similarity structure for several networks in Fig. 6c, which shows a transition from shared to independent representations as networks move from the rich to the lazy regime. Finally, Fig. 6d shows the breakdown in generalization in the NTK regime. Hence our neural race reduction can describe learning in the rich feature learning regime, with non-trivial generalization behavior.

5. Experiments

So far, we have described mathematical principles relating network architecture to learning dynamics and the nature of learned representations in GDLNs. We demonstrated how the network architecture affects generalization in pathway networks when applied to a simple toy dataset. In this section, we qualitatively validate our key findings on naturalistic datasets. Specifically, we test if GDLNs can exhibit strong zero-shot generalization performance on *untrained* input-output domain pairs (Section 4.1) and if the
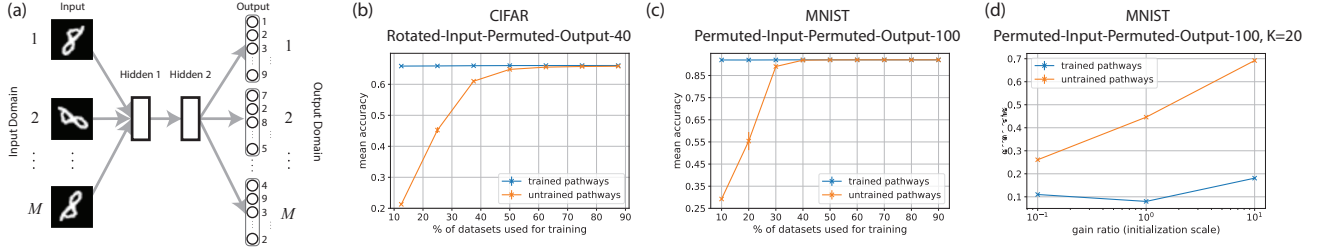
Figure 7. Experimental results. (a) Each input domain receives inputs that have been subjected to one of M input transformations. The target output for each output domain is also transformed by one of M output transformations. Here the visualization uses rotations in the input and permutations in the output. Only a subset of input-output transformation pairs are seen during training. (b) Error for trained (blue) and untrained (orange) input-output domain pairs as a function of the percentage of trained pathways (K/M) on the CIFAR dataset with $M^2 = 1600$ total tasks. (c) Error on MNIST with $M^2 = 10^4$ total tasks. Training accuracy is always high while zero-shot transfer to untrained pathways becomes as good as the training performance when $\approx$40% of pathways are trained. (d) Error as a function of initialization scale. While performance on trained domains is good for all scales, zero-shot generalization only emerges at small inits.

link between initialization scale and zero-shot generalization (Section 4.3) holds when training on naturalistic datasets.

Several datasets and benchmarks have been proposed to evaluate systematic generalization in neural networks (Johnson et al., 2017; Bahdanau et al., 2019b; Sinha et al., 2019; Lake, 2019; Ruis et al., 2020; Sinha et al., 2020). However, these naturalistic datasets generally require the network to learn multiple capacities (like spatial reasoning, logical induction, etc.) and come with a fixed (and often unclear) extent of training in different domains. To develop a setting that remains close to the theory, we create new datasets by *composing transformations* on popular vision datasets. Having fine-grained control over the dataset generation mechanism enables us to understand the effect of parameters like the number of input/output domains.

Briefly, as depicted in Figure 7a, starting from a base dataset with n inputs and outputs $\{(x^\mu, y^\mu)\}, \mu = 1, \cdots, n$, we generate new tasks by applying one of M input transformations $f_i^{\text{in}}, i = 1, \cdots, M$ and one of M output transformations $f_j^{\text{out}}, j = 1, \cdots, M$ (full details deferred to Appendix D due to space constraints). The task from input domain i to output domain j thus has samples $\{(f_i^{\text{in}}(x^\mu), f_j^{\text{out}}(y^\mu))\}$. We use the MNIST (Deng, 2012) and CIFAR-10 datasets (Krizhevsky et al., 2009) as base datasets, and rotations and permutations as transformations. Relative to our previous experiments, these datasets add real data correlations and distinct transformations on each domain that make finding a shared representation challenging.

5.1. Results

In Figure 7, we evaluate the zero-shot generalization performance of gated deep linear networks on *untrained* pathways and study the effect of initialization on their performance. Full model and training details are given in Appendix D. Fig. 7(b,c) shows mean accuracy, over trained and untrained pathways, as a function of the fraction of datasets that the model used for training, for CIFAR ($M = 40$) and MNIST

($M = 100$) respectively. Training accuracy is always high while the zero-shot transfer to untrained pathways becomes near-perfect when $\approx$40% of pathways are trained. In Fig. 7d, we report the error for trained (blue) and untrained (orange) input-output domain combinations as a function of initialization scale (gain ratio). While performance on trained domains is good for all scales, zero-shot generalization only emerges for smaller scales, as in the neural race regime. These observations validate our findings (from Section 4) on naturalistic datasets.

6. Related Work

Our work is closely related to several areas in machine learning: Deep Linear Networks, the study of dynamics of learning in neural networks, and modular neural networks.

Deep Linear Networks: Baldi & Hornik (1989); Fukumizu (1998); Saxe et al. (2014); Arora et al. (2018); Lampinen & Ganguli (2019); Saxe et al. (2019) showed that deep linear networks exhibit several nonlinear phenomena that are observed in deep neural networks and proposed studying the dynamics of deep linear networks as a surrogate for understanding the dynamics in the deep neural networks. (Baldi & Hornik, 1989) described the loss landscape, while (Saxe et al., 2014) developed the theory of gradient descent learning in deep linear neural networks and provided exact solutions to the nonlinear dynamics. Motivated by their observation about the similarity in dynamics of linear and non-linear networks and the feasibility of analyzing the gradients in the linear network, we ground the proposed framework in the deep linear network setting.

Many works have studied the **dynamics of deep networks** under specific assumptions like linear separability of data (Combes et al., 2018), deep ReLU networks (Tian et al., 2019; Straat & Biehl, 2019), Tensor Switching Networks (Tsai et al., 2016) (generalization of ReLU to tensor-valued hidden units), the Neural Tangent Kernel limit (Jacot et al., 2018; Fort et al., 2020), the Mean Field limit (Mei

et al., 2018; Rotskoff & Vanden-Eijnden, 2018; Sirignano & Spiliopoulos, 2020) and Ensemble Networks (Fort et al., 2019) to name a few (see (Carleo et al., 2019; Bahri et al., 2020; Arora et al., 2020) for reviews). Similar to these works, we also focus on a specific subset of deep networks, gated deep linear networks, that captures a nonlinear relationship between a network's input-output maps and its parameters, while being amenable to theoretical analysis.

Within the setup of deep linear networks, several works have focused on the analysis of convergence rate (Saxe et al., 2014; Arora et al., 2019a; 2018; Du & Hu, 2019), on understanding inductive biases like implicit regularization (Laurent & von Brecht, 2018; Ji & Telgarsky, 2019; Gunasekar et al., 2018; Saxe et al., 2019; Arora et al., 2019b) and understanding generalization dynamics (Lampinen & Ganguli, 2019; Poggio et al., 2018; Huh, 2020). Veness et al. (2021); Budden et al. (2020) propose and study the Gated Linear Networks (GLNs) as a class of backpropagation-free neural architectures using geometric mixing. While GLNs appear similar to GDLNs, there are several differences. GDLN is a framework that schematizes how pathways of information flow impact learning dynamics within architecture and studies networks trained using back-propagation. Additionally, GLNs are good at online learning and continual learning, while in this work, we use the GDLN framework for understanding zero-shot generalization capabilities. In the GLN model, a neuron is defined as a gated geometric mixer of the output of linear networks in the previous layer, while in the GDLN model, the neurons are linear networks where the input is the output of the linear network along the previous path. In the GLN setup, multiple input-to-output connections (in successive layers) can be active for the same input, while in the GDLN setup, only one input-output connection (in successive layers) is active for one input.

In this work, we propose modeling the model architecture as a graph and study how pathways of information flow impact learning dynamics within an architecture. Previous works have also proposed analyzing neural networks as directed graphs using the Complex Network Theory (Boccaletti et al., 2006). Scabini & Bruno (2021) analyzed the structure and performance of fully connected neural networks, Zambra et al. (2020) focused on the *emergence* of motifs in fully connected networks, Testolin et al. (2020) study deep belief networks using techniques from Complex Network Theory literature and La Malfa et al. (2021) focused on convolution and fully connected networks, with ReLU non-linearity.

Our Gated Deep Linear Network framework is closely related to areas like *modular networks* (Happel & Murre, 1994; Sharkey, 1997; Auda & Kamel, 1999; Johnson et al., 2017; Santoro et al., 2017), *routing networks* (Rosenbaum et al., 2019) and *mixture of experts* (Jacobs et al., 1991; Jordan & Jacobs, 1994; Chen et al., 1999; Yuksel et al., 2012). In

these works, the common theme is to learn a set of *modules* (or *experts*) that can be composed (or selected) using a *controller* (or a router). The modules are generally instantiated as neural networks, while the controller can either be a neural network or a hand-designed policy. These approaches have been prominently used in natural language processing (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021; Lewis et al., 2021), computer vision (Ahmed et al., 2016; Gross et al., 2017; Yang et al., 2019; Wang et al., 2019) and reinforcement learning (Yang et al., 2020; Sodhani et al., 2021; Andreas et al., 2017; Goyal et al., 2020; He & Boyd-Graber, 2016; Goyal et al., 2021).

Our work is also related to previous works in **systematic generalization** (Bahdanau et al., 2019b;a; Lake, 2019; Ruis et al., 2020; Gontier et al., 2020) and multi-task learning (Caruana, 1997; Zhang et al., 2014; Kokkinos, 2017; Radford et al., 2019; Ruder, 2017; Liu et al., 2019a; Mott et al., 2019; Vithayathil Varghese & Mahmoud, 2020). Specifically, we explore the role of model architecture and weight initialization on models' ability to exhibit systematic generalization and multi-task learning.

7. Conclusion

A key intuition in deep learning holds that a network's architecture influences learned representations, and should relate to task structure in order to achieve good performance and generalization. Here, we have introduced the Gated Deep Linear Network framework, which reveals how architecture–reflected by a simple nonlinear gating scheme along the edges of an architecture graph–controls pathways of information flow that govern learning dynamics, representation learning, and ultimately generalization. Our exact reductions and solutions show that learning dynamics take the form of a race, with greater representational reuse causing faster learning, imparting a bias toward shared representations. We validate our key insights on naturalistic datasets and with relaxed assumptions. An interesting future research direction will be to explore mechanisms for inferring the optimal architecture and gating for a given setup.

Acknowledgements

References

Ahmed, K., Baig, M. H., and Torresani, L. Network of experts for large-scale image categorization. In *European Conference On Computer Vision*, pp. 516–532. Springer, 2016.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J. H., Fan, L., Fougner, C., Hannun, A. Y., Jun, B., Han, T., LeGresley, P., Li, X., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Qian, S., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, C., Wang, Y., Wang, Z., Xiao, B., Xie, Y., Yogatama, D., Zhan, J., and Zhu, Z. Deep speech 2 : End-to-end speech recognition in english and mandarin. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 173–182. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/amodei16.html.

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 166–175. PMLR, 2017. URL http://proceedings.mlr.press/v70/andreas17a.html.

Arora, R., Arora, S., Bruna, J., Cohen, N., Du, S., Ge, R., Gunasekar, S., Jin, C., Lee, J., Ma, T., and Others. Theory of deep learning, 2020.

Arora, S., Cohen, N., and Hazan, E. On the optimization of deep networks: Implicit acceleration by overparameterization. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 244–253. PMLR, 2018. URL http://proceedings.mlr.press/v80/arora18a.html.

Arora, S., Cohen, N., Golowich, N., and Hu, W. A convergence analysis of gradient descent for deep linear neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=SkMQg3C5K7.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F.,

Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7411–7422, 2019b. URL https://proceedings.neurips.cc/paper/2019/file/c0c783b5fc0d7d808f1d14a6e9c8280d-Paper.pdf.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8139–8148, 2019c. URL https://proceedings.neurips.cc/paper/2019/file/dbc4d84bfcfe2284ba11beffb853a8c4-Paper.pdf.

Auda, G. and Kamel, M. Modular neural networks: A survey. *International Journal Of Neural Systems*, 9(02): 129–151, 1999.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf.

Bahdanau, D., de Vries, H., O'Donnell, T. J., Murty, S., Beaudoin, P., Bengio, Y., and Courville, A. Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*, 2019a.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. C. Systematic generalization: What is required and can it be learned? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=HkezXnA9YX.

Bahri, Y., Kadmon, J., Pennington, J., Schoenholz, S. S., Sohl-dickstein, J., and Ganguli, S. Statistical Mechanics of Deep Learning. *Annual Review Of Condensed Matter Physics*, 11(1):501–528, 2020. doi: 10.1146/annurev-conmatphys-031119-050745. _eprint:

Https://doi.org/10.1146/annurev-conmatphys-031119-050745.

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989. ISSN 08936080. doi: 10.1016/0893-6080(89) 90014-2. URL http://linkinghub.elsevier.com/retrieve/pii/0893608089900142.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-u. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

Budden, D., Marblestone, A. H., Sezener, E., Lattimore, T., Wayne, G., and Veness, J. Gaussian gated linear networks. *ArXiv*, abs/2006.05964, 2020.

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-maranto, L., and Zdeborová, L. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91: 045002, 2019. doi: 10.1103/RevModPhys.91.045002.

Caruana, R. Multitask learning. *Machine Learning*, 28(1): 41–75, 1997.

Chen, K., Xu, L., and Chi, H. Improved learning algorithms for mixture of experts in multiclass classification. *Neural Networks*, 12(9):1229–1252, 1999.

Chizat, L., Oyallon, E., and Bach, F. R. On lazy training in differentiable programming. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 2933–2943, 2019. URL https://proceedings.neurips.cc/paper/2019/file/ae614c557843b1df326cb29c57225459-Paper.pdf.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *Journal Of Machine Learning Research*, 12(Article):2493–2537, 2011.

Combes, R. T., Pezeshki, M., Shabanian, S., Courville, A. C., and Bengio, Y. On the learning dynamics of deep neural networks. *ArXiv*, abs/1809.06848, 2018.

Deng, L. The mnist database of handwritten digit images for machine learning research. *Ieee Signal Processing Magazine*, 29(6):141–142, 2012.

Du, S. S. and Hu, W. Width provably matters in optimization for deep linear neural networks. In Chaudhuri, K. and

Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1655–1664. PMLR, 2019. URL http://proceedings.mlr.press/v97/du19a.html.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv preprint*, abs/2101.03961, 2021. URL https://arxiv.org/abs/2101.03961.

Flesch, T., Juechems, K., Dumbalska, T., Saxe, A., and Summerfield, C. Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, 0(0), 2022. ISSN 0896-6273. doi: 10.1016/j.neuron.2022.01. 005. URL https://www.cell.com/neuron/abstract/S0896-6273(22)00005-8. Publisher: Elsevier.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *ArXiv preprint*, abs/1912.02757, 2019. URL https://arxiv.org/abs/1912.02757.

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/file/405075699f065e43581f27d67bb68478-Paper.pdf.

Fukumizu, K. Effect of Batch Learning In Multilayer Neural Networks. In *Proceedings of the 5th International Conference on Neural Information Processing*, pp. 67–70, 1998.

Girdhar, R., Singh, M., Ravi, N., Van Der Maaten, L., Joulin, A., and Misra, I. Omnivore: A single model for many visual modalities. *ArXiv preprint*, abs/2201.08377, 2022. URL https://arxiv.org/abs/2201.08377.

Gontier, N., Sinha, K., Reddy, S., and Pal, C. Measuring systematic generalization in neural proof generation with transformers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020,*

virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/file/fc84ad56f9f547eb89c72b9bac209312-Paper.pdf.

Goyal, A., Sodhani, S., Binas, J., Peng, X. B., Levine, S., and Bengio, Y. Reinforcement learning with competitive ensembles of information-constrained primitives. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=ryxgJTEYDr.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. Recurrent independent mechanisms. *ArXiv*, abs/1909.10893, 2021.

Gross, S., Ranzato, M., and Szlam, A. Hard mixtures of experts for large scale weakly supervised vision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5085–5093. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.540. URL https://doi.org/10.1109/CVPR.2017.540.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9482–9491, 2018. URL https://proceedings.neurips.cc/paper/2018/file/0e98aeeb54acf612b9eb4e48a269814c-Paper.pdf.

Happel, B. L. and Murre, J. M. Design and evolution of modular neural network architectures. *Neural Networks*, 7(6-7):985–1004, 1994.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

He, H. and Boyd-Graber, J. L. Opponent modeling in deep reinforcement learning. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York*

City, NY, USA, June 19-24, 2016, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1804–1813. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/he16.html.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Huh, D. Curvature-corrected learning dynamics in deep neural networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4552–4560. PMLR, 2020. URL http://proceedings.mlr.press/v119/huh20a.html.

Idelbayev, Y. Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. https://github.com/akamaster/pytorch_resnet_cifar10, 2020.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Jacot, A., Hongler, C., and Gabriel, F. Neural tangent kernel: Convergence and generalization in neural networks. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8580–8589, 2018. URL https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HJflg30qKX.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu,*

HI, USA, July 21-26, 2017, pp. 1988–1997. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.215. URL https://doi.org/10.1109/CVPR.2017.215.

Jordan, M. I. and Jacobs, R. A. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2): 181–214, 1994.

Kokkinos, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5454–5463. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.579. URL https://doi.org/10.1109/CVPR.2017.579.

Krizhevsky, A., Hinton, G., and Others. Learning multiple layers of features from tiny images. 2009.

La Malfa, E., La Malfa, G., Nicosia, G., and Latora, V. Characterizing learning dynamics of deep neural networks via complex networks. In *2021 Ieee 33rd International Conference On Tools With Artificial Intelligence (ictai)*, pp. 344–351. Ieee, 2021.

Lake, B. M. Compositional generalization through meta sequence-to-sequence learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9788–9798, 2019. URL https://proceedings.neurips.cc/paper/2019/file/f4d0e2e7fc057a58f7ca4a391f01940a-Paper.pdf.

Lampinen, A. K. and Ganguli, S. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryfMLoCqtQ.

Laurent, T. and von Brecht, J. Deep linear networks with arbitrary loss: All local minima are global. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2908–2913. PMLR, 2018. URL http://proceedings.mlr.press/v80/laurent18a.html.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8570–8581, 2019. URL https://proceedings.neurips.cc/paper/2019/file/0d1a9651497a38d8b1c3871c84528bd4-Paper.pdf.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2020.

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. Base layers: Simplifying training of large, sparse models. *ArXiv preprint*, abs/2103.16716, 2021. URL https://arxiv.org/abs/2103.16716.

Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 1871–1880. Computer Vision Foundation / IEEE, 2019a. doi: 10.1109/CVPR.2019.00197. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Liu_End-To-End_Multi-Task_Learning_With_Attention_CVPR_2019_paper.html.

Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4487–4496, Florence, Italy, 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1441. URL https://aclanthology.org/P19-1441.

Mei, S., Montanari, A., and Nguyen, P.-m. A mean field view of the landscape of two-layer neural networks. *Proceedings Of The National Academy Of Sciences*, 115 (33):E7665–e7671, 2018. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1806579115. URL https://www.pnas.org/content/115/33/E7665. Publisher: National Academy Of Sciences Section: Pnas Plus.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and Others. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., and Rezende, D. J. Towards interpretable reinforcement learning using attention augmented agents. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12329–12338, 2019. URL https://proceedings.neurips.cc/paper/2019/file/e9510081ac30ffa83f10b68cde1cac07-Paper.pdf.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. Multimodal deep learning. In Getoor, L. and Scheffer, T. (eds.), *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 689–696. Omnipress, 2011. URL https://icml.cc/2011/papers/399_icmlpaper.pdf.

Nguyen, T., Raghu, M., and Kornblith, S. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Patel, A. B., Nguyen, T., and Baraniuk, R. G. A probabilistic theory of deep learning. *ArXiv preprint*, abs/1504.00641, 2015. URL https://arxiv.org/abs/1504.00641.

Poggio, T., Liao, Q., Miranda, B., Burbanski, A., and Hidary, J. Theory iiib: Generalization in deep networks. Technical report, Center for Brains, Minds and Machines (CBMM), arXiv. org, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *Openai Blog*, 1(8):9, 2019.

Raghu, M., Poole, B., Kleinberg, J. M., Ganguli, S., and Sohl-Dickstein, J. On the expressive power of deep neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2847–2854. PMLR, 2017. URL http://proceedings.mlr.press/v70/raghu17a.html.

Roberts, D. A., Yaida, S., and Hanin, B. The principles of deep learning theory. *ArXiv preprint*, abs/2106.10165, 2021. URL https://arxiv.org/abs/2106.10165.

Rosenbaum, C., Cases, I., Riemer, M., and Klinger, T. Routing networks and the challenges of modular and compositional computation. *ArXiv preprint*, abs/1904.12774, 2019. URL https://arxiv.org/abs/1904.12774.

Rotskoff, G. M. and Vanden-Eijnden, E. Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 7146–7155, 2018. URL https://proceedings.neurips.cc/paper/2018/file/196f5641aa9dc87067da4ff90fd81e7b-Paper.pdf.

Ruder, S. An overview of multi-task learning in deep neural networks. *ArXiv preprint*, abs/1706.05098, 2017. URL https://arxiv.org/abs/1706.05098.

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., and Lake, B. M. A benchmark for systematic generalization in grounded language understanding. *Advances in Neural Information Processing Systems*, 33, 2020.

Salakhutdinov, R. Deep learning. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., and Ghani, R. (eds.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 1973. ACM, 2014a. doi: 10.1145/2623330.2630809. URL https://doi.org/10.1145/2623330.2630809.

Salakhutdinov, R. Deep learning. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., and Ghani, R. (eds.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 1973. ACM,

2014b. doi: 10.1145/2623330.2630809. URL https://doi.org/10.1145/2623330.2630809.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P. W., and Lillicrap, T. A simple neural network module for relational reasoning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4967–4976, 2017. URL https://proceedings.neurips.cc/paper/2017/file/e6acf4b0f69f6f6e60e9a815938aa1ff-Paper.pdf.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6120.

Saxe, A. M., Mcclelland, J. L., and Ganguli, S. A mathematical theory of semantic development in deep neural networks. *Proceedings Of The National Academy Of Sciences*, 116(23):11537–11546, 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1820226116.

Scabini, L. F. and Bruno, O. M. Structure and performance of fully connected neural networks: Emerging complex network properties. *ArXiv preprint*, abs/2107.14062, 2021. URL https://arxiv.org/abs/2107.14062.

Sharkey, A. J. C. Modularity, combining and artificial neural nets. *Connection Science*, 9(1):3–10, 1997.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., and Others. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Sinha, K., Sodhani, S., Dong, J., Pineau, J., and Hamilton, W. L. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4506–4515, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1458. URL https://aclanthology.org/D19-1458.

Sinha, K., Sodhani, S., Pineau, J., and Hamilton, W. L. Evaluating logical generalization in graph neural networks. *ArXiv preprint*, abs/2003.06560, 2020. URL https://arxiv.org/abs/2003.06560.

Sirignano, J. and Spiliopoulos, K. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes And Their Applications*, 130(3):1820–1852, 2020. ISSN 0304-4149. doi: 10.1016/j.spa.2019.06.003. URL https://www.sciencedirect.com/science/article/pii/S0304414918306197.

Sodhani, S. xplogger: Logging utility for ML experiments, 2 2022. URL https://github.com/shagunsodhani/xplogger.

Sodhani, S., Zhang, A., and Pineau, J. Multi-task reinforcement learning with context-based representations. In *International Conference On Machine Learning (icml)*, 2021.

Standley, T., Zamir, A. R., Chen, D., Guibas, L. J., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9120–9132. PMLR, 2020. URL http://proceedings.mlr.press/v119/standley20a.html.

Straat, M. and Biehl, M. On-line learning dynamics of relu neural networks using statistical physics techniques. *ArXiv preprint*, abs/1903.07378, 2019. URL https://arxiv.org/abs/1903.07378.

Team, T. P. D. pandas-dev/pandas: Pandas, 2020. URL https://doi.org/10.5281/zenodo.3509134.

Testolin, A., Piccolini, M., and Suweis, S. Deep learning systems as complex networks. *Journal Of Complex Networks*, 8(1):Cnz018, 2020.

Tian, Y., Jiang, T., Gong, Q., and Morcos, A. Luck matters: Understanding training dynamics of deep relu networks. *ArXiv preprint*, abs/1905.13405, 2019. URL https://arxiv.org/abs/1905.13405.

Tsai, C.-Y., Saxe, A. M., Saxe, A. M., and Cox, D. Tensor switching networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural*

Information Processing Systems, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/b1563a78ec59337587f6ab6397699afc-Paper.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Veness, J., Lattimore, T., Bhoopchand, A., Budden, D., Mattern, C., Grabska-Barwinska, A., Toth, P., Schmitt, S., and Hutter, M. Gated linear networks. In *AAAI*, 2021.

Vithayathil Varghese, N. and Mahmoud, Q. H. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9), 2020. ISSN 2079-9292. doi: 10.3390/electronics9091363. URL https://www.mdpi.com/2079-9292/9/9/1363.

Wang, X., Yu, F., Dunlap, L., Ma, Y., Wang, R., Mirhoseini, A., Darrell, T., and Gonzalez, J. E. Deep mixture of experts via shallow embedding. In Globerson, A. and Silva, R. (eds.), *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, volume 115 of *Proceedings of Machine Learning Research*, pp. 552–562. AUAI Press, 2019. URL http://proceedings.mlr.press/v115/wang20d.html.

Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

Yang, B., Bender, G., Le, Q. V., and Ngiam, J. Condconv: Conditionally parameterized convolutions for efficient inference. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1305–1316, 2019. URL https://proceedings.neurips.cc/paper/2019/file/f2201f5191c4e92cc5af043eebfd0946-Paper.pdf.

Yang, R., Xu, H., WU, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4767–4777. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/32cfdce9631d8c7906e8e9d6e68b514b-Paper.pdf.

Yuksel, S. E., Wilson, J. N., and Gader, P. D. Twenty years of mixture of experts. *Ieee Transactions On Neural Networks And Learning Systems*, 23(8):1177–1193, 2012.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In Wilson, R. C., Hancock, E. R., and Smith, W. A. P. (eds.), *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

Zambra, M., Maritan, A., and Testolin, A. Emergence of network motifs in deep neural networks. *Entropy*, 22(2):204, 2020.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. Facial landmark detection by deep multi-task learning. In *European Conference On Computer Vision*, pp. 94–108. Springer, 2014.

A. Simple Nonlinear Classification

A.1. The XoR task

We consider the XoR task with $P = 4$ data points that lie at $[\pm 1 \;\; \pm 1]^T$ as depicted in Fig. 3a. The task is exclusive-or on the input bits, with a target output of $y = 1$ if true and $y = -1$ otherwise. We solve this with a GDLN containing four pathways (Fig. 3b), with each pathway active on exactly one of the four examples. By symmetry, all pathways will have the same loss dynamics and so we need only solve one. Consider the pathway active for the example $x = [1\;1]^T, y = 1$, that is, whose gating variable $g_1 = 1$ when this example is presented and $g_1 = 0$ on the remaining three examples. The relevant dataset correlations are

$$\Sigma^{yx} = \langle g_1 y x^T \rangle = 1/P \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{18}$$

$$\Sigma^{x} = \langle g_1 x x^T \rangle = 1/P \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{19}$$

$$\Sigma^{y} = \langle g_1 y y^T \rangle = 1/P \tag{20}$$

The singular value decomposition of the input-output correlations yields the nonlinear singular value $s = \sqrt{2}/P$ and input singular vector $v = [1/\sqrt{2} \; 1/\sqrt{2}]^T$. Applying this singular vector to diagonalize the input correlations yields the associated input variance $d = v^T \Sigma^x v = 2/P$. The effective singular value dynamics of this pathway is given by the deep linear network dynamics with these correlations (see Saxe et al. (2014; 2019)), yielding

$$a(t) = \frac{s/d}{1 - (1 - \frac{s}{da_0})e^{-2st/\tau}} \tag{21}$$

where $a(t)$ is the singular value in the product of both weight matrices in the pathway, and a_0 is the initial effective singular value, related to the initialization variance. Finally the loss for this pathway is the loss trajectory $l(t)$ of the associated deep linear network. The total loss, by symmetry, is the loss from all four pathways $\mathcal{L}(t) = Pl(t)$,

$$\mathcal{L}(t) = \frac{1}{2} - Psa(t) + \frac{P}{2}da(t)^2 \tag{22}$$

$$= \frac{1}{2} - \sqrt{2}a(t) + a(t)^2. \tag{23}$$

This analytical expression is exact for GDLNs initialized in the decoupled regime, and it agrees closely with the dynamics of standard ReLU networks trained end-to-end on the task starting from small random weights (Fig. 3c). Hence GDLNs can learn nonlinear input-output tasks, and in certain settings, describe the dynamics of standard ReLU networks when the gating structure is chosen appropriately.

A.2. Nonlinear Contextual Classification

As another simple example, consider a **nonlinear contextual classification** problem that cannot be solved using deep linear networks but can be solved using the gated deep linear network, again highlighting that the gated networks are more expressive than their non-gated counterpart.

Consider receiving two-dimensional inputs $x \in \mathbb{R}^2$ where each component $x_i, i = 1, 2$ is drawn from a uniform distribution between -1 and 1. The task of the network is to classify stimuli based either on the first or second input component. That is, the target output is $y = x_c$ in context $c \in 1, 2$, and each context appears with probability 1/2. In this simple scenario (a variant of the XoR task), the same input must be treated in two different ways depending on context, and nonlinearity is required for solving it correctly.

Now we must choose a gating structure. If we choose a single pathway that is always active ($g = 1$ for all samples), then we recover a deep linear network. The resulting correlation matrices are

$$\Sigma^{yx} = \langle yx^T \rangle = [1/6 \; 1/6] \tag{24}$$

$$\Sigma^{x} = \langle xx^T \rangle = 1/3I \tag{25}$$

where I is the 2×2 identity matrix. Under the resulting dynamics, the total weights converge to the linear least squares solution $W^{tot} = \Sigma^{yx}(\Sigma^x)^{-1} = [1/2 \; 1/2]$, the best solution attainable by the linear network.

Alternatively, we can set the gating variables such that a different pathway is active in each context. We then have the collection of correlation matrices

$$\Sigma^{yx}(1) = [1/6 \; 0] \tag{26}$$

$$\Sigma^{yx}(2) = [0 \; 1/6] \tag{27}$$

$$\Sigma^{x}(1,1) = \Sigma^{x}(2,2) = 1/6I \tag{28}$$

$$\Sigma^{x}(1,2) = \Sigma^{x}(2,1) = 0. \tag{29}$$

We thus see that each pathway faces a subproblem defined by just one context. For this simple case, the pathways converge to their respective linear least squares solutions. In particular, $W^{tot}(1) = [1 \; 0]$ and $W^{tot}(2) = [0 \; 1]$, such that each pathway picks out the correct input coordinate. In combination with the gating scheme, these weights exactly solve this nonlinear task, showing that gated linear networks are more expressive than linear networks. Interestingly, neuroimaging and electrophysiological recordings from this paradigm suggest that this type of solution is observed in the human and primate brain, as well as in standard ReLU networks trained in the "rich" feature learning regime (Flesch et al., 2022).

B. Gradient flow dynamics

The gradient flow equations are

$$\tau \frac{d}{dt} W_e = -\frac{\partial \mathcal{L}(\{W\})}{\partial W_e} \quad \forall e \in E \tag{30}$$

$$= \left\langle \sum_{p \in \mathcal{P}(e)} g_p W_{\bar{t}(p,e)}^T \left[y_{t(p)} x_{s(p)}^T - h_{t(p)} x_{s(p)}^T \right] W_{\bar{s}(p,e)}^T \right\rangle_{y,x,g} \tag{31}$$

$$= \left\langle \sum_{p \in \mathcal{P}(e)} g_p W_{\bar{t}(p,e)}^T \left[y_{t(p)} x_{s(p)}^T - \sum_{j \in \mathcal{T}(t(p))} g_j W_j x_{s(j)} x_{s(p)}^T \right] W_{\bar{s}(p,e)}^T \right\rangle_{y,x,g} \tag{32}$$

$$= \sum_{p \in \mathcal{P}(e)} W_{\bar{t}(p,e)}^T \left[\left\langle g_p y_{t(p)} x_{s(p)}^T \right\rangle_{y,x,g} - \sum_{j \in \mathcal{T}(t(p))} W_j \left\langle g_j x_{s(j)} x_{s(p)}^T g_p \right\rangle_{y,x,g} \right] W_{\bar{s}(p,e)}^T \tag{33}$$

$$= \sum_{p \in \mathcal{P}(e)} W_{\bar{t}(p,e)}^T \left[\Sigma^{yx}(p) - \sum_{j \in \mathcal{T}(t(p))} W_j \Sigma^x(j,p) \right] W_{\bar{s}(p,e)}^T, \tag{34}$$

where we have simply rearranged terms and used the linearity of expectation.

The dynamics reduction can then be obtained by applying the change of variables,

$$\tau \frac{d}{dt} W_e = \sum_{p \in \mathcal{P}(e)} W_{\bar{t}(p,e)}^T \left[\Sigma^{yx}(p) - \sum_{j \in \mathcal{T}(t(p))} W_j \Sigma^x(j,p) \right] W_{\bar{s}(p,e)}^T \tag{35}$$

$$\tau \frac{d}{dt} \left(R_{t(e)} B_e R_{s(e)}^T \right) = \sum_{p \in \mathcal{P}(e)} \left(U_{t(p)} B_{\bar{t}(p,e)} R_{t(e)}^T \right)^T \left[U_{t(p)} S(p) V_{s(p)}^T - \right. \tag{36}$$

$$\left. \sum_{j \in \mathcal{T}(t(p))} U_{t(j)} B_j V_{s(j)}^T V_{s(j)} D(j,p) V_{s(p)}^T \right] \left(R_{s(e)} B_{\bar{s}(p,e)} V_{s(p)}^T \right)^T \tag{37}$$

$$\tau \frac{d}{dt} B_e = \sum_{p \in \mathcal{P}(e)} B_{\bar{t}(p,e)} \left[S(p) - \sum_{j \in \mathcal{T}(t(p))} B_j D(j,p) \right] B_{\bar{s}(p,e)} \tag{38}$$

where we have used the fact that $R_v^T R_v = I$ for all nodes, and the fact that $U_{t(j)} = U_{t(p)}$ by definition of the set $\mathcal{T}(t(p))$. From this we see that if the B variables are initially diagonal they remain so under the dynamics. In this case, the dynamics decouple and each element along the diagonal of the B matrices evolves independently of the rest.

C. Routing task and network reduction

To understand the dynamics in the pathway network, we first collect the relevant input statistics. We have

$$\Sigma^{yx}(p) = \left\langle g_p y_{t(p)} x_{s(p)}^T \right\rangle \tag{39}$$

$$= \Pr(g_p = 1) U S V^T \tag{40}$$

$$= \frac{1}{KM} U S V^T \tag{41}$$

$$\Sigma^x(j,p) = \left\langle g_j x_{s(j)} x_{s(p)}^T g_p \right\rangle \tag{42}$$

$$= \begin{cases} \frac{1}{KM} V D V^T & \text{if } j = p \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

because there are KM total trained paths from input to output and all pathways are gated off except for the active pathway.

Inserting these data statistics into (10), and assuming that initial singular values are equal for all input domains and all output domains (a reasonable approximation when starting from small random weights), we can track only the variables B_1, B_2, and B_3 encoding the singular values in the input, hidden, and output weights respectively.

Next, we note that the first and third layer weights are active in K tasks (all tasks originating from a given input or terminating at a given output, respectively), while second layer weights are active in all KM tasks. This yields the reduced dynamics

$$\tau \frac{d}{dt} B_1 = \frac{1}{M} B_3 B_2 \left[S - B_3 B_2 B_1 D \right] \tag{44}$$

$$\tau \frac{d}{dt} B_2 = B_3 B_1 \left[S - B_3 B_2 B_1 D \right] \tag{45}$$

$$\tau \frac{d}{dt} B_3 = \frac{1}{M} B_2 B_1 \left[S - B_3 B_2 B_1 D \right]. \tag{46}$$

If we consider 'balanced' initial conditions where $B_1(0) = B_3(0)$, we have

$$\tau \frac{d}{dt} B_1 = \frac{1}{M} B_2 B_1 \left[S - B_2 B_1^2 D \right] \tag{47}$$

$$\tau \frac{d}{dt} B_2 = B_1^2 \left[S - B_2 B_1^2 D \right], \tag{48}$$

recovering Eqns. (13)-(14) of the main text.

To estimate the ratio of singular values in the first layer to that in the second, we consider its time derivative and calculate the steady state. We have

$$\frac{d}{dt} B_2 / B_1 = B_1 \left[S - B_2 B_1^2 D \right] - \frac{1}{M} B_2^2 / B_1 \left[S - B_2 B_1^2 D \right] \tag{49}$$

$$0 = B_1 - \frac{1}{M} B_2^2 / B_1 \tag{50}$$

$$B_2 = \sqrt{M} B_1. \tag{51}$$

Hence if training continues for long times (such that the error term does not become zero), the shared portion of the pathway changes more by a factor $\sqrt{M}$ (and this ratio does not depend on K).

We can extend this analysis to the case where all input-output routes are trained but the gating structure sends only P paths through each hidden weight matrix, as considered in Section 4.2. With this gating scheme, the reduction is

$$\tau \frac{d}{dt} B_1 = \frac{\sqrt{P}}{M^2} B_2 B_1 \left[S - B_2 B_1^2 D \right] \tag{52}$$

$$\tau \frac{d}{dt} B_2 = \frac{P}{M^2} B_1^2 \left[S - B_2 B_1^2 D \right], \tag{53}$$

and the singular value ratio is

$$\frac{d}{dt}B_2/B_1 = \frac{P}{M^2}B_1\left[S - B_2 B_1^2 D\right] - \frac{\sqrt{P}}{M^2}B_2^2/B_1\left[S - B_2 B_1^2 D\right] \tag{54}$$

$$0 = PB_1 - \sqrt{P}B_2^2/B_1 \tag{55}$$

$$B_2 = P^{\frac{1}{4}}B_1. \tag{56}$$

This ratio scales from 1 to $\sqrt{M}$ as the number of shared paths goes from $P = 1$ (no sharing) to $P = M^2$ (full sharing). Hence for this architecture, greater sharing causes larger weight changes in the hidden pathway.

D. Experimental details and further results

This section contains details and hyperparameter settings for the simulations reported in Section 5, as well as additional visualization of results in Figures 9 and E.1.

We start by explaining the general procedure for transforming the existing datasets and then describe the new dataset instances that we create.

Consider a dataset $D = (X, Y)$, defined as a tuple of inputs X and targets Y. The dataset D has n datapoints, that is, $X \in R^{n \times n_{dim}}$, where n_{dim} is the dimensionality of each input[1]. The dataset has n_{class} unique classes, referred by their indices $\{0, 1, \cdots, n_{class} - 1\}$. We want to create new datasets by transforming the given dataset D. We assume that we have a list of M input transformations $f_i^{input} : R^{n_{dim}} \to R^{n_{dim}} \forall i \in \{0, \cdots, M - 1\}$ and M output transformations $f_j^{output} : R^{n_{class}} \to R^{n_{class}} \forall j \in \{0, \cdots, M - 1\}$. Now, we can define a new dataset, $D_{i,j}$, as (X_i, Y_j), where $X_i = f_i^{input}(X)$, $Y_j = f_j^{output}(Y)$ i.e any transformation of the given dataset is a new dataset. We can apply M transformations on the input and M transformations on the output to obtain M^2 datasets.

We consider the following two operations for input transformations: *rotation* of the input image and *permutation* of pixels. For the ith *rotation* transformation, we rotate the input images by an angle $\theta = 180i/M$ degrees. For the ith *permutation* transformation, we apply a random permutation matrix to the flattened input. Each of these transformations provides input to one input domain. We use the *permutation* operation as the output transformation, implying that each new output transformation corresponds to a n_{class}-way classification task.

We use the MNIST (Deng, 2012) and CIFAR-10 datasets (Krizhevsky et al., 2009) to create three datasets:

MNIST-Permuted-Input-Permuted-Output-40: 40 transformations on both the input and the output, leading to a total of 1600 datasets. The input transformation is permutation of pixels and the output transformation is permutation of the targets.

MNIST-Permuted-Input-Permuted-Output-100: 100 permutation transformations on both the input and the output, leading to a total of 10^4 datasets.

MNIST-Rotated-Input-Permuted-Output-40 40 rotation transformations on the input and 40 permutation transformations on the output.

CIFAR-Rotated-Input-Permuted-Output-40 40 rotation transformations on the input and 40 permutation transformations on the output.

In the case of *CIFAR-Rotated-Input-Permuted-Output-40* dataset, use a pre-trained ResNet18 (He et al., 2016) model[2] to map the images into 512 dimensional vectors. We pretrain the ResNet18 model on full CIFAR-10 dataset, freeze the pre-trained model and use the first two residual blocks to encode the images from the transformed datasets. The output of the (frozen) ResNet encoder is used as input to the gated network.

D.1. Model and training

The model consists of M encoders, denoted as $(\{\phi_i \forall i \in \{1, \cdots, M\}\})$ and M decoders, denoted as $(\{\psi_i \forall i \in \{1, \cdots, M\}\})$ and a shared hidden layer θ. In the GDLN framework, the encoders correspond to the *input nodes*, the decoders correspond to the *output nodes* and the connection from an encoder, to the hidden layer, to the decoder corresponds to a *path*. The

---

[1]While images are multi-dimensional arrays, they can be represented as flattened 1-d arrays.

[2]We use the following code for pre-training the models: https://github.com/akamaster/pytorch_resnet_cifar10
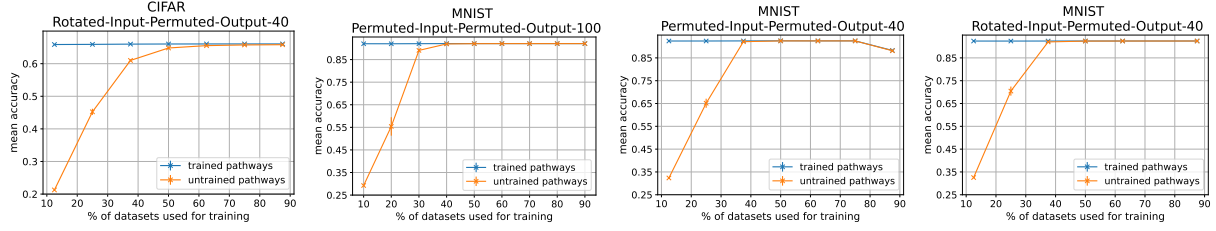
Figure 8. Experimental results. Error for trained (blue) and untrained (orange) input-output domain pairs as a function of the percentage of trained pathways (K/M) on: (i) CIFAR dataset, where input is rotated and output is permuted, with $M^2 = 1600$ total tasks, (ii) MNIST dataset, where input and output, both are permuted, with $M^2 = 10^4$ total tasks, (iii) MNIST dataset, where input and output, both are permuted, with $M^2 = 1600$ total tasks, and (iv) MNIST dataset, where input is rotated and output is permuted, with $M^2 = 1600$ total tasks. (in the order of left to right). Training accuracy is always high while zero-shot transfer to untrained pathways becomes as good as the training performance when $\approx$40% of pathways are trained.
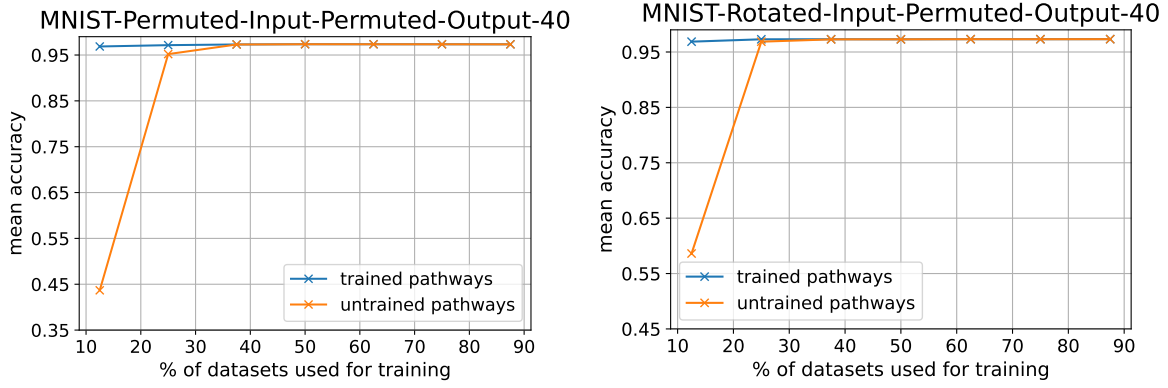

Figure 9. Experimental results. Error for trained (blue) and untrained (orange) input-output domain pairs as a function of the percentage of trained pathways (K/M) on: (i) MNIST dataset, where input and output, both are permuted, with $M^2 = 1600$ total tasks, and (ii) MNIST dataset, where input is rotated and output is permuted, with $M^2 = 1600$ total tasks (in the order of left to right). These models are trained with the leaky-ReLU non-linearity with the negative slope parameter set to be 0.01, thus making the model non-linear. The training accuracy is always high while zero-shot transfer to untrained pathways becomes as good as the training performance when $\approx$25% of pathways are trained.

encoders, decoders and the shared hidden layer are all instantiated as linear networks. Given a dataset, $D_{i,j}$, or (x_i, y_j), we compute the prediction using the following function: $\psi_j(\theta(\phi_i(x_i)))$ [3]

D.2. Training and Evaluation Setup

Following the setup in Section 4, we train a subset of input-output domains such that each input domain is trained with only $K \leq M$ output domains, resulting in $M \times K$ *trained* pathways and $M \times (M - K)$ *untrained* pathways. During evaluation, we report the performance on both the *trained* pathways and the *untrained* pathways.

---

[3] Note that we overload the notation to represent the components and the computation using the same symbol.
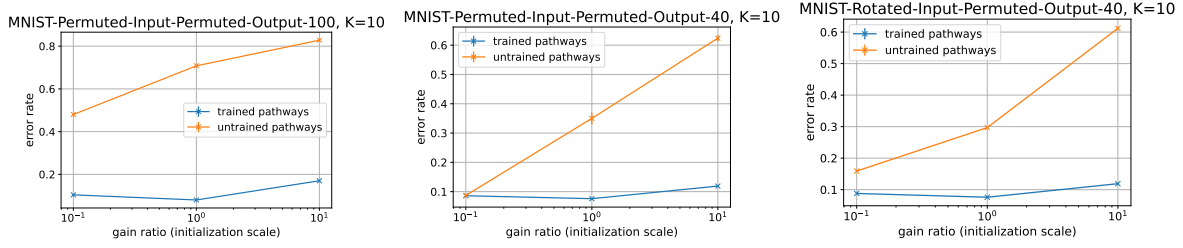
Figure 10. Experimental results. Error as a function of initialization scale for trained (blue) and untrained (orange) input-output domain pairs when training on K/M fraction of pathways, where $K = 10$ on (i) MNIST dataset, where input and output, both are permuted, with $M^2 = 10^4$ total tasks, (i) MNIST dataset, where input and output, both are permuted, with $M^2 = 1600$ total tasks, and (iii) MNIST dataset, where input is rotated and output is permuted, with $M^2 = 1600$ total tasks. (in the order of left to right). While performance on trained domains is reasonably good for all scales, zero-shot generalization only emerges at small inits.

E. Additional Implementation Details

E.1. Libraries
We use the following open-source libraries:

1. PyTorch (Paszke et al., 2019)[4]

2. Hydra (Yadan, 2019)[5]

3. Numpy (Harris et al., 2020)[6]

4. Pandas (Team, 2020)[7]

5. ResNet Implementation for CIFAR10/CIFAR100 in PyTorch (Idelbayev, 2020)[8]

6. Xplogger (Sodhani, 2022)[9]

We ran all the experiments with 10 seeds and report the mean as well as the standard error across the seeds.

Table 1. Hyperparameter values common across all the task distributions

Hyperparameter	Hyperparameter values
Batch size (per task)	8
Number of epochs	1000
Number of classes	10
Dimensionality of the hidden layer	128
Gain Ratio for weights	$10.0, 1.0, 0.1, 0.01, 0.001, 0.0001$
Initial value for bias	0.0
Optimizer	SGD
Learning Rate	0.0001
Momentum	0.9

Table 2. Hyperparameter values for *MNIST-Permuted-Input-Permuted-Output-40* (different from the values described in Table 1

Hyperparameter	Hyperparameter values
Number of input transformations	40
Number of output transformations	40
Number of encoders	40
Number of decoders	40

Table 3. Hyperparameter values for *MNIST-Permuted-Input-Permuted-Output-100* (different from the values described in Table 1

Hyperparameter	Hyperparameter values
Number of input transformations	100
Number of output transformations	100
Number of encoders	100
Number of decoders	100

Table 4. Hyperparameter values for *MNIST-Rotated-Input-Permuted-Output-100* (different from the values described in Table 1

Hyperparameter	Hyperparameter values
Number of input transformations	100
Number of output transformations	100
Number of encoders	100
Number of decoders	100

Table 5. Hyperparameter values for *CIFAR-Rotated-Input-Permuted-Output-40* (different from the values described in Table 1

Hyperparameter	Hyperparameter values
Number of input transformations	40
Number of output transformations	40
Number of encoders	40
Number of decoders	40
Dimensionality of the hidden layer	1024