Engineering flexible machine learning systems by traversing functionally invariant paths in weight space

Guruprasad Raghavan,[1] Matt Thomson,[1*]

[1]Department of Bioengineering, Caltech
1200 E California Blvd, Pasadena, CA 91125
*To whom correspondence should be addressed;
mthomson@caltech.edu
graghava@caltech.edu

Abstract

Deep neural networks achieve human-like performance on a variety of perceptual and decision making tasks. However, deep networks perform poorly when confronted with changing tasks or goals, and broadly fail to match the flexibility and robustness of human intelligence. Here, we develop a mathematical and algorithmic framework that enables continual training of deep neural networks on a broad range of objectives by defining path connected sets of neural networks that achieve equivalent functional performance on a given machine learning task while modulating network weights to achieve high-performance on a secondary objective. We view the weight space of a neural network as a curved Riemannian manifold and move a neural network along a functionally invariant path in weight space while searching for networks that satisfy a secondary objective. We introduce a path-sampling algorithm that trains networks with millions of weight parameters to learn a series of image classification tasks without performance loss. The algorithm generalizes to accommodate a range of secondary objectives including weight-pruning and weight diversification and exhibits state of the art performance on network compression and adversarial robustness benchmarks. Broadly, we demonstrate how the intrinsic geometry of machine learning problems can be harnessed to construct flexible and robust neural networks.

Introduction

Artificial neural networks can now out-perform humans on tasks ranging from image recognition, game playing, to predicting three-dimensional structures of proteins [1, 2, 3]. However, in many respects, artificial neural networks fail to replicate the flexibility and robustness that are defining features of human intelligence [4, 5, 6]. Humans can learn new tasks and accommodate novel goals with minimal instruction and without loss of performance on existing tasks. Unlike humans, deep neural networks suffer significant performance decay when trained to perform additional tasks or integrate new information, a phenomenon known as catastrophic forgetting [7, 8]. For example, a network trained to recognize images of hand-written digits [9] will 'forget' the digit recognition task when trained to recognize additional objects like letters or faces. In addition to well-known flexibility limits, deep neural networks have other pathologies, like vulnerability to targeted corruption of input data or adversarial fragility. Small, imperceptible changes in the input data can cause complete failure of network performance, a phenomenon known as adversarial fragility.

The current limitations of deep learning present challenges for the application of deep neural networks in industrial settings that require neural networks to adapt to changing data streams and to be robust to perturbation and corruption of input data. Fundamentally, the disparity between biological and artificial neural networks motivates the development of theoretical frameworks that define underlying principles and circuit architectures that enable flexibility and robustness in both

artificial and natural systems. While a range of specialized algorithms have emerged to separately address challenges like catastrophic forgetting and adversarial fragility in artificial neural networks, the strategies are often heuristic and specific and do not explain the missing principles of intelligence that could more broadly address known limits of deep learning. Conceptual and mathematical frameworks that provide a unified perspective on the pathologies of deep learning could provide new insights into the mathematical principles of flexible intelligence in both machines and humans.

In artificial neural networks, network function is encoded in the mathematical weights that determine the strength of connections between neural units (Fig. 1). Modern deep learning procedures train multi-layered neural networks to solve problems by performing gradient descent to adjust the weights of a network based on a mathematical function known as an objective function that encodes the performance of a network on a specific task. Standard learning methods, like back-propagation and gradient descent [10], adjust network weights to define a single, optimal weight configuration to maximize performance on a task specific objective function using training data. Training the network on new tasks through the traditional paradigm of stepping along the gradient of the task-specific objective function adjusts the networks' weights, inevitably resulting in the loss of information from previous tasks.

The weight adjustment problem underlies other challenges in modern machine learning. As an example, in many applications it is advantageous to prune or sparsify a network to minimize the number of non-zero weights and thus reduce the memory and power consumption of a network. Just like multi-task learning, network sparsification requires the adjustment of network weights while maintaining function, and sparsification procedures often proceed through heuristic weight pruning strategies. In the case of adversarial robustness, a central goal is to identify ensembles of networks that perform a task with distinct weight configurations and thus avoid vulnerabilities associated with a single weight configuration.

Unlike contemporary artificial neural nets, neural networks in the human brain, which are well known ingredients of natural intelligence, perform multiple functions and can flexibly switch between different functional configurations based on context, goals or memory [11]. Neural networks in the brain are hypothesized to overcome the limitations of a single, optimal weight configuration and perform flexible tasks by continuously 'drifting' their neural firing states and neural weight configurations, effectively generating large ensembles of degenerate networks [12, 13].Fluctuations might enable flexibility in biological systems by allowing neural networks to explore a series of network configurations while responding to sensory input.

Here, inspired by the notion of network ensembles, we develop a geometric framework and algorithm to construct path connected sets of neural networks that solve a given machine learning task. By building sets of networks rather than single networks, we can search within an ensemble of weight space for networks that accommodate a secondary goal. The central conceptual shift is that we consider path-connected sets of neural networks, rather than single-networks (isolated points in weight space) to be the central objects of study and application. We develop a core algorithm for identifying sets of networks that solve a given machine learning problem and search within the set to identify networks that accommodate secondary tasks without loss of functional performance. Previous work has demonstrated that large models often contain significant parameter degeneracy [14], and our path ensembles exploit weight degeneracy in neural networks to create functionally similar networks with distinct internal weights.

We view a neural networks' weight space as a pseudo-Riemannian manifold equipped with a distance metric that represents task performance. Rather than focusing on single, optimal networks, we move along the Riemannian manifold to identify path connected sets of networks that minimize the change in functional performance on one task while simultaneously capturing a secondary objective. We formalize the "search" for a suitable network as a dynamic movement on the curved pseudo-Riemannian manifold. Our path-sampling algorithm identifies functionally invariant

paths in weight space that maintain network performance while 'searching-out' for other networks that satisfy additional objectives like sparsification or mitigating catastrophic interference. We demonstrate that the path sampling algorithm achieves state of the art performance on three core challenges: sequential task learning, network sparsification, and adversarial robustness. In each case, our path-sampling algorithm achieves state of the art results on large networks with millions of parameters obtaining performance similar to domain specific approaches.

Using the framework, we cast three core challenges in modern machine learning within the language of differential geometry and develop a single, unified theoretical and algorithmic framework to address all three challenges. Beyond the algorithmic results, our work provides a mathematical framework that unifies a variety of contemporary machine learning problems in a single, geometric language. We demonstrate that each deep learning pathology has a geometric interpretation as finding a path that minimizes distance traveled along a curved manifold while also following the flow field defined by a secondary goal. Biologically, our work motivates the search for biological strategies that might allow networks to explore path-connected sets through weight fluctuations or explicit weight modulation of perceptual networks by higher level networks. Broadly, we suggest that the geometric perspective might provide a unified framework for linking distinct problems associated with understanding mechanisms of intelligence, both natural and artificial [5, 15].

Construction of functionally invariant paths in neural network's weight space

We develop a mathematical framework to quantify how the output of an artificial network changes as the network's weights are adjusted. We apply the framework to explore path-connected sets of neural networks that have divergent weight values but similar output on training data. To construct path-connected neural network ensembles, we view the weight-space of a neural network as a Riemannian manifold equipped with a local distance metric. Riemannian manifolds are used in physics and geometry to study the properties of curved spaces and to formalize notions of distance, velocity, and acceleration on curved manifolds [16, 17]. We construct a distance metric on weight space that measures the change in the output of a network given infinitesimal changes in network weights. By using basic notions from differential geometry, we construct paths through weight space that maintain the functional performance of a neural network while adjusting network weights to flow along a secondary goal. While the secondary goal can be general, we initially consider the secondary goal to be performance on additional classification tasks. For instance, we convert a neural network that only recognizes hand-written digits to a network that can recognize both, digits (MNIST) and images of garments (Fashion-MNIST).

The defining feature of a Riemannian manifold is the existence of a local distance metric. We construct a distance metric in weight space that defines the distance between two nearby networks to be their difference in output. We consider a neural network to be a smooth function, $f(\mathbf{x}; \mathbf{w})$, that maps an input vector, $\mathbf{x} \in \mathbb{R}^{\mathrm{k}}$, to an output vector, $f(\mathbf{x}; \mathbf{w}) = \mathbf{y} \in \mathbb{R}^{\mathrm{m}}$, where the map is parameterized by a vector of weights, $\mathbf{w} \in \mathbb{R}^{\mathrm{n}}$, that are typically set in training to solve a specific task. We refer to $W = \mathbb{R}^n$ as the *weight space* of the network, and we refer to $\mathcal{Y} = \mathbb{R}^m$ as the *output space* as shown in Fig. 1B [18]. For pedagogical purposes, we will consider the action of f on a single input, $\mathbf{x}$. In the supplement we show that our results extend naturally to an arbitrary number of inputs $\mathbf{x_i}$.

We initially ask how the output, $f(\mathbf{x}; \mathbf{w})$, of a given neural network changes for small changes in network weights. Given a neural network with weights $\mathbf{w_t}$, a fixed input $\mathbf{x}$, we can compute the output of the perturbed network, $\mathbf{w_t} + \mathbf{dw}$ for an infinitesimal weight perturbation, $\mathbf{dw}$ as

$$f(\mathbf{x}, \mathbf{w_t} + \mathbf{dw}) \approx f(\mathbf{x}, \mathbf{w_t}) + \mathbf{J_{w_t}}\ \mathbf{dw}, \tag{1}$$

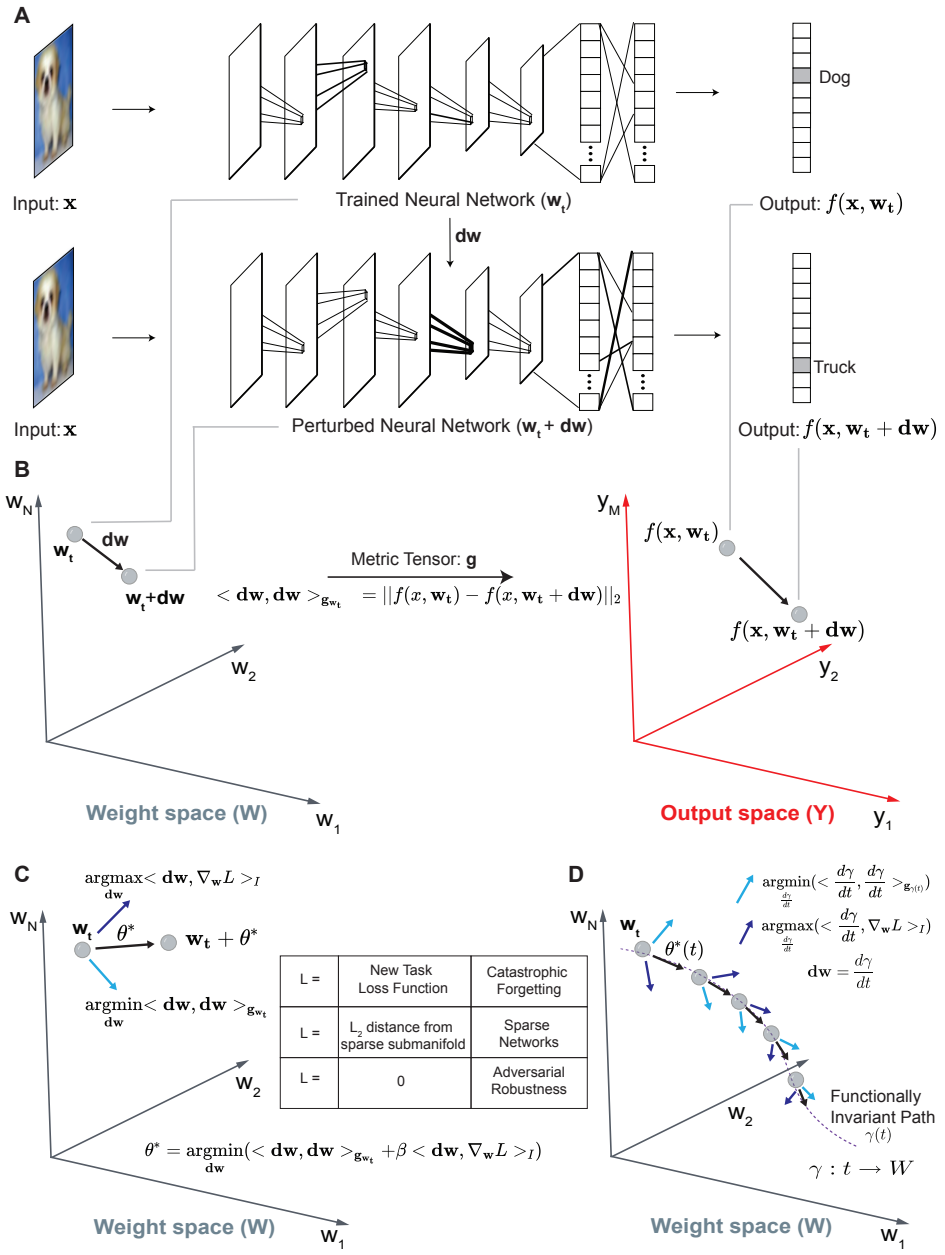


Figure 1: **Geometric Framework for constructing functionally invariant paths (FIP) in weight space.** (A) (Top) A trained convolutional neural network with weight configuration ($\mathbf{w_t}$), represented by lines connecting different layers of the network, accepts an input image, $\mathbf{x}$, and produces a 10-element output vector, $f(\mathbf{x}, \mathbf{w_t})$. (Below) Perturbation of network weights by $\mathbf{dw}$ results in a new network with weight configuration $\mathbf{w_t} + \mathbf{dw}$ with an altered output vector, $f(\mathbf{x}, \mathbf{w_t} + \mathbf{dw})$, for the same input, $\mathbf{x}$. (B) Mathematically, perturbation of the networks' weight configuration shift the network from $\mathbf{w_t}$ to $\mathbf{w_t} + \mathbf{dw}$ in weight space (W). The network's output, $f(\mathbf{x}, \mathbf{w_t})$, shifts to $f(\mathbf{x}, \mathbf{w_t} + \mathbf{dw})$ in output space (Y). The metric tensor ($\mathbf{g_{w_t}}$) transforms perturbation in weight space ($\mathbf{dw}$) to distance moved in the output space ($||f(\mathbf{x}, \mathbf{w_t}) - f(\mathbf{x}, \mathbf{w_t} + \mathbf{dw}||_2$). (C) The FIP algorithm identifies weight perturbations, θ^* that minimize distance moved in output space while maximizing alignment with gradient of a secondary objective function ($\nabla_{\mathbf{w}} L$). The light-blue arrow indicates ϵ-norm weight perturbation that minimizes distance moved in output space, dark-blue arrow is ϵ-norm weight perturbation that maximizes alignment with gradient of objective function, $L(\mathbf{x}, \mathbf{w})$. The secondary objective function $L(\mathbf{x}, \mathbf{w})$ is varied to solve distinct machine learning challenges. (D) The path algorithm defines functionally invariant paths, $\gamma(t)$, through iterative identification of ϵ-norm perturbations ($\theta^*(t)$) in the weight space.

where $\mathbf{J}_{\mathbf{w_t}}$ is the Jacobian of $f(\mathbf{x}, \mathbf{w_t})$ for a fixed $\mathbf{x}$, $J_{ij} = \frac{\partial f_i}{\partial w_j}$, evaluated at $\mathbf{w_t}$.

Thus, the total change in network output for a given weight perturbation $\mathbf{dw}$ is

$$|f(\mathbf{x}, \mathbf{w_t} + \mathbf{dw}) - f(\mathbf{x}, \mathbf{w_t})|^2 = \mathbf{dw}^T \ (\mathbf{J}_{\mathbf{w_t}}(\mathbf{x})^T \ \mathbf{J}_{\mathbf{w_t}}(\mathbf{x})) \ \mathbf{dw} \tag{2}$$
$$|\langle \mathbf{dw}, \mathbf{dw} \rangle_{\mathbf{g}_{\mathbf{w_t}}}|^2 = \mathbf{dw}^T \ \mathbf{g}_{\mathbf{w_t}}(\mathbf{x}) \ \mathbf{dw}$$

where $\mathbf{g}_{\mathbf{w_t}}(\mathbf{x}) = \mathbf{J}_{\mathbf{w_t}}(\mathbf{x})^T \ \mathbf{J}_{\mathbf{w_t}}(\mathbf{x})$ is the metric tensor evaluated at the point $\mathbf{w_t} \in W$ for a single data point, $\mathbf{x}$. The metric tensor is an $n \times n$ symmetric matrix that allows us to compute the change in network output for a perturbation along any direction in weight space as $\langle \mathbf{dw}, \mathbf{dw} \rangle_{\mathbf{g}_{\mathbf{w_t}}(\mathbf{x})}$. The metric also allows us to compute the infinitesimal change in network output while moving along a path $\gamma(\mathbf{t})$ in weight space as $\theta(t) = \frac{\mathrm{d}\gamma(t)}{\mathrm{d}t}$.

how does it allow that?

At every point in weight space, the metric allows us to discover directions $\mathbf{dw}$ that have large or small impact on the output of a network. As we move along a path in weight space, we sample a series of neural networks over time, t. Using the metric we can define a notion of 'output velocity', $\mathbf{v} = \frac{\mathrm{d}f(\mathbf{x},\gamma(t))}{\mathrm{d}t}$, that quantifies the distance a network moves in output space along the path. We seek to identify 'Functionally invariant paths (FIPs)' ($\gamma : [0, 1] \to W$) in weight space along which the output velocity is minimized for a fixed change in weight. To do so, we solve the following optimization problem

$$\psi^*(t) = \underset{\frac{d\gamma}{dt}}{\operatorname{argmin}} \langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_{\mathbf{g}_{\gamma(\mathbf{t})}}$$
$$\text{with} \quad \langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_I = \epsilon \tag{3}$$

where we attempt to find a direction to perturb the network, such that it is ϵ units away in the weight space (in the euclidean sense) ($\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_I = \epsilon$) while minimizing the distance moved in the networks' output space, given by $\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_{\mathbf{g}_{\gamma(\mathbf{t})}}$. Here, I is an identity matrix, with the inner product $\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_I$ capturing the euclidean distance in the weight space [19]. The optimization problem is a quadratic program at each point in weight space. The metric $\mathbf{g}$ is a matrix that takes on a specific value at each point in weight space, and we aim to identify vectors $\psi^*(t) = \frac{d\gamma(t)}{dt}$, that minimize the change in functional output of the network.

We will often want to amend the optimization problem with a second objective function $L(\mathbf{x}, \mathbf{w})$. We can enumerate paths that minimize the functional velocity in the output space while moving along the gradient of the second objective ($\nabla_{\mathbf{w}} L$). We define a path-finding algorithm that captures the trade off between these two terms.

$$\theta^*(t) = \underset{\frac{d\gamma}{dt}}{\operatorname{argmin}} (\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_{\mathbf{g}_{\gamma(t)}} + \beta \langle \frac{d\gamma}{dt}, \nabla_{\mathbf{w}} L \rangle_I)$$
$$\text{with } \langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_I = \epsilon \tag{4}$$

where now the first term, $\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_{\mathbf{g}_{\gamma(t)}}$, identifies functionally invariant directions while the second term, $\langle \frac{d\gamma}{dt}, \nabla_{\mathbf{w}} L \rangle_I$, biases the direction of motion along the gradient of a second objective. When $L = 0$, the algorithm merely constructs paths in weight space that are approximately isofunctional ($\theta^*(t) = \psi^*(t)$), i.e. the path is generated by steps in the weight space comprising of networks with different weight configurations while preserving the input-output map. $L(\mathbf{x}, \mathbf{w})$ can also represent the loss function of a second task, for example a second input classification problem. In this case,

we identify vectors that simultaneously maintain performance on an existing task (via term 1) while also improving performance on a second task by moving along the negative gradient of the second task loss function, $\nabla_{\mathbf{w}} L$.

To approximate the solution to Eq-4, in large neural networks, we developed a numerical strategy that samples points in an ϵ ball around a given weight configuration, and then performs gradient descent to identify vectors $\theta^*(t)$. In the appendix, we extend the metric formulation to cases where we consider a set of N training data points, $\mathbf{X}$, and view $\mathbf{g}$ as the average of metrics derived from individual training examples. $\mathbf{g_w} = \mathbf{g_w}(\mathbf{X}) = \Sigma_{i=1}^{N} \mathbf{g_w}(\mathbf{x_i})/N$. The metric, $\mathbf{g}$, provides a local measure of *output distance* on the Riemannian manifold $(W, \mathbf{g_w})$. At each point in weight space, the metric defines the length, $\langle \mathbf{dw}, \mathbf{dw} \rangle_{\mathbf{g_w}}$, of a local perturbation by its impact on the functional output of the network (Fig. 1A).

Functionally invariant paths alleviate catastrophic forgetting

We apply the geometric framework to a series of catastrophic forgetting problems. Catastrophic forgetting (CF) problems have important applications in industrial settings where it is often advantageous to extend an existing neural network to accommodate additional labels or training data while maintaining its ability to perform an existing function. A series of algorithms including the elastic weight consolidation (EWC) [20], Gradient Episodic memory (GEM) [21], Optimal Relevance Mapping (ORM)[22] have been developed to address CF. Our geometric algorithm allows us to extend current strategies to achieve high performance on CF problems and also to address a series of CF problems through iteration of our approach. In the CF problems our goal is to modulate the weights of an existing neural network to achieve high performance on an additional classification task while maintaining performance on earlier tasks. While our algorithm is general, we work in the context of image classification tasks as a canonical setting for analyzing deep learning algorithms generally and the phenomena of CF specifically.

To circumvent catastrophic forgetting while learning two sequential tasks, we solve the optimization problem in Eq-4, by setting $L(\mathbf{x}, \mathbf{w})$ as the classification loss function specified by the second task and $\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_{\mathbf{g}_{\text{Task1}}}$ as the distance moved in the networks' output space for a small number of inputs sampled from the first (earlier) task. As demonstrated in Fig. 2A(ii) we solve the optimization problem (in Eq-4) for a fixed value of β by simultaneously minimizing the distance moved in the networks' output space (light-blue arrow) corresponding to inputs from the first task while maximizing alignment with the gradient of $L(\mathbf{x}, \mathbf{w})$ (dark-blue arrow) encoding the classification loss from the second task. In this manner, we construct a Functionally invariant path (FIP) (purple dotted line) in weight space from N_1 (blue node), trained on the first task, to N_2 (yellow-blue node) that retains performance on the first task while gaining performance on a second task (Fig. 2A(ii)).

We apply our path-finding algorithm to mitigate catastrophic forgetting in a large convolutional network (Supporting information) with one million weights and twenty output classes for learning two sequential tasks: (Task-1) recognizing images of 10 classes of handwritten digits from MNIST and (Task-2) recognizing images of 10 classes of fashion-apparel from the Fashion-MNIST dataset (Fig. 2A(i)). Convolutional neural networks (CNN's) are widely used in the ML community for large scale visual recognition challenges [23, 3]. Our specific CNN has 5 layers: (2 convolutional layers, with 32 and 64 convolutional filters each, and 3 fully connected layers - with 600, 120 and 20 nodes each and a total of 1476548 weights in all. The MNIST data set is a canonical data set representing $60,000$ examples of human hand written digits [24] and the Fashion-MNIST data contains $60,000$ images of fashion items. [25]

Naive training of the CNN on MNIST and retraining on FMNIST leads to significant performance decays on the MNIST classification task. Specifically, we randomize the initial CNN

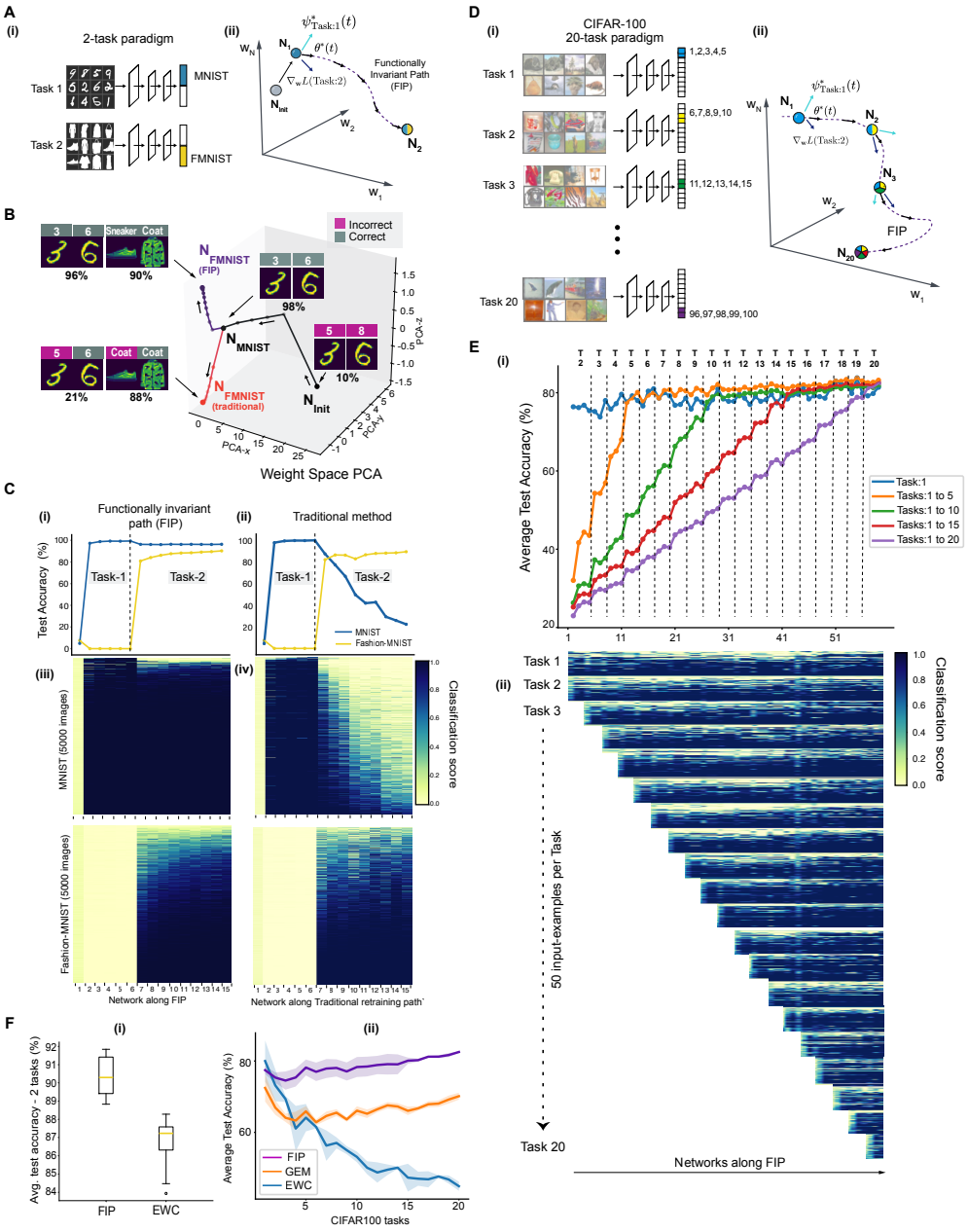


Figure 2: **Networks learn sequential tasks without catastrophic forgetting by traversing FIPs.** (A)(i) Training neural networks on a 2 task paradigm, with Task-1 being 10-digit recognition from MNIST and Task-2 being 10-item recognition from Fashion-MNIST. (ii) Schematic to construct FIPs in weight space to train networks on two tasks sequentially. (B) 3D lineplot where dots are weight configurations of 5-layered convolutional neural networks (CNNs) in PCA space. Training on two tasks sequentially via conventional approach takes the black followed by red path to reach N-FMNIST(traditional), while the path-finding algorithm takes the black followed by purple path to reach N-FMNIST(FIP). Images of digits-3,6 are from MNIST and sneaker, coat images are from Fashion-MNIST. Text labels above the image are networks' predictions and numbers below are the networks' test accuracy on MNIST and Fashion-MNIST. (C) Test accuracy of networks learning two tasks sequentially by traversing FIP (i) and by traditional retraining (ii). Heatmaps capture classification score on 10k test images (5k images from each task) for networks obtained through FIP (iii) and traditional retraining strategy (iv). (D)(i) Neural network with 100 output classes are trained on 20 task paradigm, with every task containing 5 non-overlapping classes of natural images sampled from CIFAR100 dataset. (ii) Schematic to construct FIPs in weight space to train neural networks on 20 sequential tasks. (E)(i) Average test accuracy of networks along FIP while learning 20 sequential tasks. The networks to the right of a dashed line encounter a new task (T-i), referring to the i'th task. (ii) Heatmap displays classification score for networks along FIP on 1k test images, with 50 images sampled from every task. (F) FIP surpasses state-of-art methods in mitigating catastrophic forgetting in 2-task paradigm (i) and 20-task CIFAR100 paradigm (ii). Error bars indicate standard deviation over 5 trials.

network weights, and obtain 98% on MNIST image classification following gradient descent based weight optimization (Fig. 2C). When we train the MNIST classification network to classify Fashion-MNIST garments, the network rapidly loses accuracy on the MNIST task as performance increases on Fashion-MNIST (Fig. 2C). The traditional training/retraining paradigm results in discovering networks that drop its accuracy on the first task, from 98% to 21% (blue line, from network-7 onwards in Fig. 2C(i)) while gaining performance on the second task from 10% to 88% (yellow line from network-7 in Fig. 2C(i)). The networks along the red path in the weight space PCA (Fig. 2B) initially are able to recognize image '3' as the number-3, but as they step along the loss gradient from the second task, they lose performance on the first task and misclassify the image of digit '3' as a '5'. The heatmaps (Fig. 2C(iv)) show that networks obtained by the traditional method for retraining on the second task have a uniformly low classification score (yellow pixels) for MNIST images, while having a high classification score (dark-blue pixels) for Fashion-MNIST images.

While naive training and retraining paradigm induces catastrophic forgetting of the MNIST classification task, our FIP algorithm discovers networks that simultaneously retain performance on the first task (MNIST), staying between 98% to 96% test accuracy (blue line, from network-7 onwards in Fig. 2C(i)) while gaining performance on the second task (Fashion-MNIST) reaching 89% (yellow line, from network-7 onwards in Fig. 2C(i)). The networks along the purple path in the weight space PCA (Fig. 2B) retain their ability to correctly recognize images of digits '3' and '6', while additionally becoming capable of classifying images of fashion-apparel 'Sneaker' and 'Coat' from Fashion-MNIST. The heatmaps (Fig. 2C(iii)) show that networks along the FIP have a high classification score (dark-blue pixels) for both, MNIST and Fashion-MNIST images.

The FIP algorithm induces curved weight space paths that differ considerably from the naive training/retraining paradigm. The curved training paths allow the network to achieve high performance broadly across the held out testing set as indicated in the heatmaps in Fig. 2C. The introduction of the second task causes a small number of MNIST examples to decline in classification performance while the network improves monotonically on the Fashion-MNIST examples (yellow band in Fig. 2C(iii)). In comparison, the naive training and retraining strategy causes monotonic performance decay with increasing loss of performance across MNIST images as performance on Fashion-MNIST increases.

Functionally invariant paths enable continuous learning in large networks

Owing to the generality of our framework, we can naturally scale the FIP approach to learn a series of tasks rather than two through iterative application of Eq-4. To demonstrate sequential task learning, we execute a standard sequential CIFAR100 task where a network is asked to identify new sub-sets of CIFAR100 images at each round [26]. We applied our framework to a large convolutional network (ResNet18) with 18 layers, 100 output classes and a total of 1.1 million parameters, by subjecting them to 20 sequential tasks derived from the CIFAR100 dataset [27]. As shown in Fig. 2D(i), each task requires the network to identify 5 distinct image-classes from the CIFAR100 dataset. Task-1 comprises identification of images from 5 CIFAR100 classes, namely dolphin, whale, otter, baby, wardrobe, Task-2 comprises of a different set of 5 CIFAR100 image-classes, namely boy, man, bowls, cups, cattle, and so on.

To circumvent catastrophic forgetting in the twenty sequential task paradigm, we solve the optimization problem in Eq-4 by setting $L(\mathbf{x}, \mathbf{w})$ as the loss function specified by the incoming new task (Task-i), while $\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_{\mathbf{g}_{\text{Task}-1:i-1}}$ is set to be the distance moved in output space for a small number of inputs sampled from all the previous tasks (from Task:1 to Task:i-1).

We first train the network to obtain high accuracy on the first task by gradient descent. Specifically, we initialize the ResNet18 with random weights using the Xavier initialization [28] protocol and train the network by stepping along the loss gradient of the first task to achieve a performance

of 78% on the first task (to recognize CIFAR100 images pertaining to Task:1). In Fig. 2E(i), the first point on the x-axis corresponds to the network trained on the first task alone, and the blue line records test accuracy of networks along the FIP on the first task.

Having trained ResNet18 on the first task, we train the network on subsequent tasks (e.g. recognizing images from disjoint sets of 5 image-classes sampled from CIFAR100) by constructing FIPs in the weight space and obtain networks that simultaneously retain performance on all previous tasks while learning a new task. In Fig. 2E(i), we find that networks along the FIP do not merely retain task performance on previous tasks, but also increase their test performance on previous tasks while learning new CIFAR100 tasks. For instance, the networks performance on Task:1 (as seen from the blue line) remains constant at 76% while being introduced to tasks 2 through 15, but increases its performance to 80% when exposed to tasks 16 through 20. Similarly, the networks average test performance on Tasks 1 through 5 (as seen from the orange line) remains constant at 77% until Task-14 and increases to 81% while learning Tasks 15 through 20. Similar trends of retaining previous task performance while learning new tasks are observed for the networks average test performance on Tasks 1 thru 10 (green line) and Tasks 1 thru 15 (red line). On introducing the last task (Task-20), we find that the networks along the FIP have a mean performance of 82% on the entire CIFAR100 dataset having shown 5 classes at a time.

The heatmaps in Fig. 2E(ii) show that the FIP strategy achieves high performance broadly across the held-out testing set. The introduction of a new task (along the heatmaps' y-axis) causes a small number of examples from the previous task to decline in classification performance while the network improves monotonically on the new task. In Fig. 2E(ii), the first 4 networks (1-4) retain their classification score on Task-1, while increasing their classification score on Task-2. The subsequent set of 3 networks (5-7, 8-10, ...) retain their classification score on previous tasks (Tasks 1-2, Tasks 1-3, ...) while increasing their classification score on (Task-3, Task-4 ...) respectively. Having presented all 20 tasks to the network, our path-finding framework discovers networks that perform at 82.54±0.17% accuracy on the 20 tasks, while the conventional method performs at 26.86±1.05% (supplementary).

The FIP approach outperforms other methods that have been introduced to mitigate catastrophic forgetting, specifically Elastic weight consolidation on the 2 task paradigm (Fig. 2F(i)) (FIP: 91±1.1 %, EWC: 87±1.6%) and 20-task paradigm (FIP: 82.54±0.17%, EWC:44.9±0.01%) (Fig. 2F(ii)). Elastic Weight consolidation (EWC) [20] is a state of the art algorithm that adjusts a neural network by identifying weights that are rarely used in a given task and adjusting those weights to achieve high performance on a second task. Our path finding algorithm has conceptual and mathematical similarities with EWC. Both the procedures find weights (directions), along which changes in weights accrue little change in network output. The key difference is that our approach scales to overcoming catastrophic forgetting on multiple tasks as we explicitly construct functionally invariant paths over long distances in the weight space while EWC relies on finding suitable networks in the vicinity of a previously trained network (which may or may not exist) by computing a local Fisher Information metric.

In addition to performing better than regularization based methods (like EWC) on the 20-task paradigm (Fig. 2F(ii)), our approach performs better than Gradient episodic memory (GEM) with a memory budget of 500 memories from each task previously encountered [21]. GEM is a method that stores gradients from previous tasks while being trained on new tasks (GEM: 70.14±0.1%). In this comparison, we chose a 500 example memory budget for GEM as our path-finding algorithm uses 500 randomly sampled data points from the previous tasks for constructing the FIPs in weight space.

In summary, we demonstrate that the FIP strategy allows training of neural networks on a series of sequential image classification tasks. By stepping along a path defined through a trade-off between function invariance and a secondary loss function, we train networks that achieve high

performance on multiple tasks with similar or greater performance than existing state of the art methods.

Network sparsification by traversing path connected network sets

One of the critical aspects of the FIP framework is that it can be generalized beyond sequential task training to address a broad range of machine learning problems by considering a more general set of secondary objective functions. In particular, we focus on sparsification of neural networks, which is important for reducing the memory and computational footprint of a network [29]. Sparsification refers to the problem of reducing the number of non-zero weights in a network. The sparsification of networks decreases the required memory and computational foot-print of a network and is therefore of great practical interest for applying neural networks on memory limited devices.

We can apply the FIP framework to sparsify neural networks by discovering functionally invariant paths in weight space that also decrease the number of non-zero network weights. We discover $p\%$ sparse neural networks (wherein $p\%$ of the networks' weights are set to zero) with a high performance on the task of interest by recasting the optimization problem in Eq-4 as a problem of constructing FIPs in the weight space from a high-performance densely connected neural network (all non-zero network weights) to a $p\%$ sparse submanifold in the weight space. The $p\%$ sparse submanifold comprises of a set of networks that have an architectural constraint of having $p\%$ of their weights set to zero, while having no constraint on the networks' task performance. To solve the optimization problem in Eq-4, we set $L(\mathbf{x}, \mathbf{w})$ to be the euclidean distance in the weight space between the dense network and a network in the $p\%$ sparse submanifold, obtained by setting $p\%$ of the smallest weights in the dense network to zero.

We discover a series of sparsified LeNet-300-100 networks with sparsities ranging from 20% to 99.4% that exhibit a high performance on the task of classifying images of handwritten digits from MNIST [24] by traversing FIPs in the weight space. LeNet-300-100 [23] is a multilayer perceptron with two hidden layers consisting of 300 and 100 nodes each, and a total of 484000 non-zero weights. Although most networks randomly sampled from the 99.1% sparse submanifold in the weight space perform poorly on the MNIST task (with test accuracies ranging from 6 to 10%), the FIP algorithm finds a curved path in the weight space from densely connected LeNet (with test accuracy of 98% on MNIST) to networks in the 99.1% sparse submanifold that perform at test accuracies between 96.3% to 96.8% on the MNIST classification task (Fig. 3D).

In addition to identifying high performance sparse networks in the sparse submanifold, the FIP-discovered sparse networks have diverse inter-layer connectivity structures. Fig. 3F(top) illustrates the connectivity structure of 6 high performance sparse networks in the 99.1% sparse submanifold performing at accuracies ranging from 96.3% to 96.8%. Vertical bars in the figure indicate the position of non-zero weights. Non-zero weights occur in different positions across the six networks indicating that the FIP algorithm discovers architecturally diverse sparse solutions.

While the FIP solutions vary locally, there are also patterns observed across networks. Specifically, 99.2±0.2% and 98.4±0.3 % of the weights between layers 1-2 and 2-3 respectively are zeroed out, while only 52±4% of the weights between layers 3-4 are zeroed out. The differential sparsification across different layers indicates that connections between the first few layers contain more redundancy than the later layers.

The sparse networks discovered by traversing FIPs from dense network trained on MNIST to the sparse sub-manifold have a higher task-performance than the ones discovered by the lottery ticket hypothesis (LTH) [30], which entails subjecting trained dense networks to multiple prune-train cycles (Fig. 3B). Across a wide range of sparsities (from 20% to 99.4%), the sparse networks discovered by FIP are comparable in test accuracy on MNIST to those obtained by LTH, while our method succeeds in discovering extremely sparse networks with high-performance. In Fig. 3B, the

FIP method finds a 99.4% sparse network performing at an accuracy of 96±0.6%, while the LTH strategy finds a 99.4% sparse network performing at 91±3% on the MNIST dataset.

Our FIP algorithm scales to finding sparser counterparts of large convolutional networks with skip, like the ResNet-20 architecture [31], which has a series of 20 convolutional layers, trained to recognize images of automobiles, animals and man-made structures from the CIFAR-10 dataset. Although most networks sampled from the 93% sparse submanifold of ResNet20 networks perform at accuracies between 18 to 30% on CIFAR10, an FIP constructed from a dense, trained ResNet20 network to the 93% sparse submanifold is successful in picking out high performance sparse ResNet20 networks functioning at accuracies between 82 to 84.7% on CIFAR10. In Fig. 3E, we visualize the 93% sparse submanifold of ResNet20 networks (blue and red dots), the dense trained ResNet20 (N_1, grey dot), and the curved FIP (purple dots) constructed between the two in the weight space PCA. The 93% sparse submanifold comprises of networks that have 19073 non-zero weights (out of 272474 weights), wherein the blue dots represent networks that have been randomly sampled from the sparse submanifold while the red dots correspond to networks uncovered by traversing the FIP. We also find that the random sparse networks (blue dots) perform poorly on the task, incorrectly recognizing the images of 'deer', 'frog', 'plane', 'ship' as 'plane', 'deer', 'truck' and 'bird' respectively.

Like the LeNet example, the inter-layer connectivity of discovered sparse ResNet-20 networks are distinct locally, but have globally conserved patterns. For instance, in Fig. 3G, we find that the weights between layers 2 to 19 have an average inter-layer sparsity of 85% while having a maximum sparsity of 99.2%, present between layers 18-19 (penultimate layer), across all sparsified ResNet20 networks. On the other hand, the weights between layer 1-2 (first 2 layers) and the layers 19-20 (last 2 layers) are least sparsified, with them being 41% and 24% sparse respectively. The differential sparsification across different layers points to the fact that the redundancy in ResNet-20 architectures is encoded majorly between layers 3 and 18.

The FIPs are successful in discovering high performance sparse ResNet20 networks on a wide range of sparse submanifolds (from 20% to 95%) that are at par with the state-of-art technique Lottery ticket hypothesis (LTH) [30]. In Fig. 3C, we find that our path-finding algorithm discovers sparse networks that perform at par with the LTH strategy for sparsities ranging from 20% to 93%, while performing 3% lower than sparse networks discovered by LTH in the 95% sparse submanifold. Lottery ticket hypothesis adopts the iterative prune-train strategy to discover sparse networks. Here, a densely connected network is initially trained, then a part of their weights are pruned (or set to zero), before commencing the next cycle of training and pruning. As each cycle of training maintains the L_0 norm of the weights, it forces the pruned network to train on smaller subspaces after every prune-train cycle.

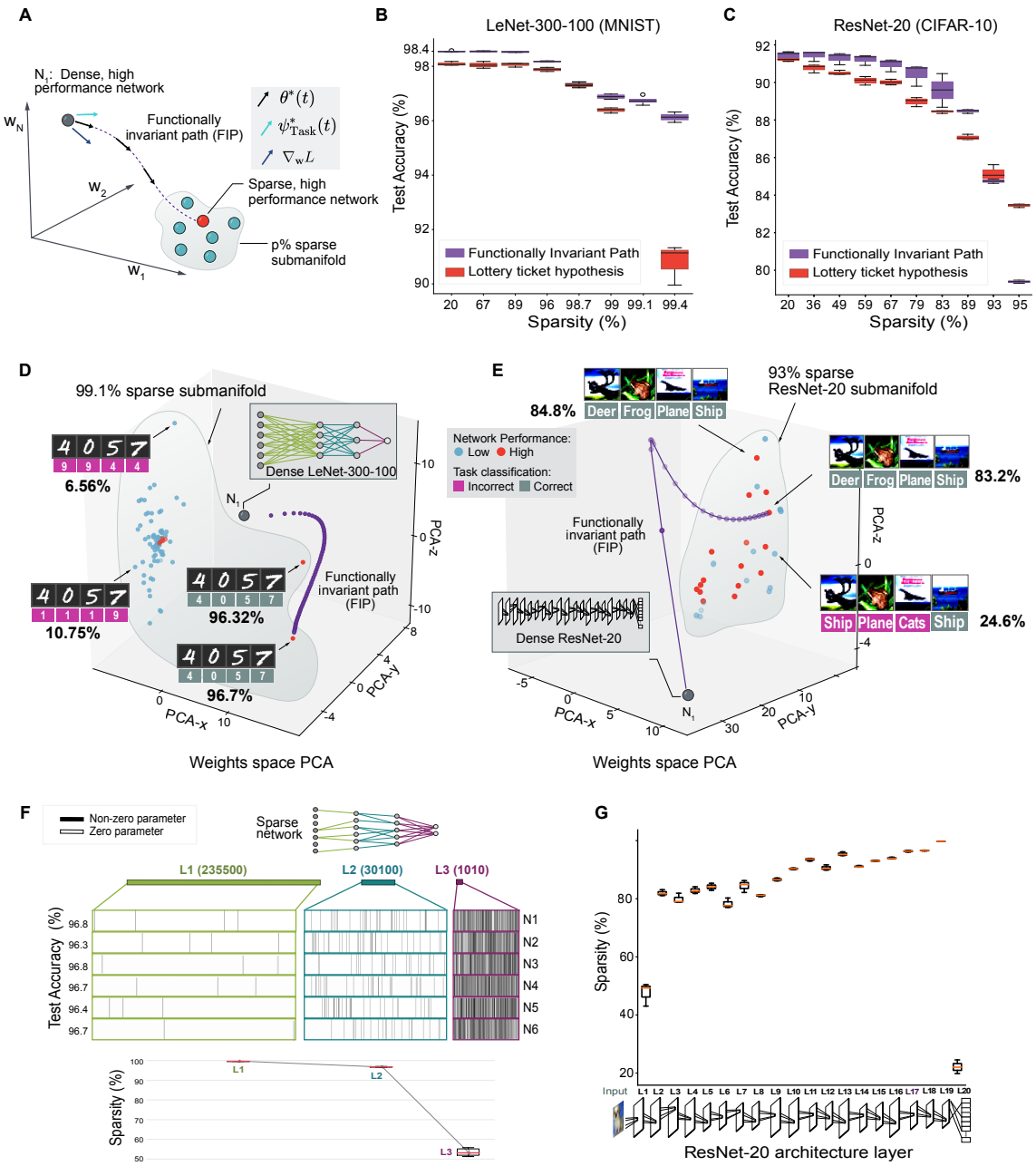


Figure 3: **Sparse networks discovered by traversing FIPs in the weight space.** (A) Schematic to construct FIP from N_1 to p% sparse submanifold. (B, C) Performance of sparse networks discovered by FIPs (purple) and Lottery ticket hypothesis (red) across a wide range of sparsities on MNIST (B) and CIFAR-10 (C). (D) Scatterplot where the dots are weight configurations of LeNet-300-100 networks in PCA space. The FIP (purple line) beginning from N_1 (grey dot) discovers high-performance LeNet's in the 99.1% sparse submanifold (red dots). Blue dots are random sparse networks in the 99.1% sparse submanifold. Digits-4,0,5,7 are from MNIST, text-labels below the image are network predictions and the number below is the networks' test accuracy on MNIST. (E) Scatterplot where the dots are weight configurations of ResNet-20 networks in PCA space. The FIP beginning at N_1 (grey dot) discovers high-performance ResNet-20 networks in the 93% sparse submanifold (red dots). Blue dots are random sparse networks in 93% sparse submanifold. Deer, frog, plane, ship images are from CIFAR-10, text-labels below the image are network predictions and the number adjacent is the networks' test accuracy on CIFAR10. (F) Sparse LeNet connectivity visualized by plotting vertical lines in color-coded rows to represent non-zero weights. (Below) Boxplot shows sparsity across LeNet's layers. (G) Boxplot shows the sparsity across ResNet-20's layers (over n=6 sparsified ResNet's). The cartoon below depicts the ResNet-20 architecture.

Path-connected sets of networks confer robustness against adversarial attack

Adversarial perturbations to neural networks involve the selection of data perturbations that lead to incorrect network decisions. Although deep networks have achieved remarkable performance on image-recognition tasks, they remain extremely vulnerable to small perturbations of the input image. That is, human-imperceptible additive perturbations to the input image are successful in fooling deep networks, ultimately resulting in a failed image recognition (Fig. 4B). The intentionally crafted human-imperceptible additive perturbations to the input image are called adversarial examples [32]. Classical work has demonstrated that data perturbations that are imperceptible to humans can lead to catastrophic performance loss in deep neural networks. Adversarial robustness is a major goal for deep learning in industrial settings where catastrophic performance declines could compromise user safety in autonomous vehicle and medical applications of Deep Neural Networks.

We generate adversarially perturbed images for deep networks trained on an image-classification task and find that, although humans perceive the adversarial images identical to the original images, it causes significant performance loss to deep neural networks. Specifically, we generate adversarial examples for an instance of the VGG16 network (with 16 layers and 130 million parameters) trained to recognize images from the CIFAR-10 dataset, by applying the projected gradient descent (PGD) attack. The PGD attack computes the best direction (in image space) to perturb the image such that it maximizes the trained networks' loss on the image while constraining the L_{inf} norm of the perturbation. Prior to the adversarial attack, the VGG16 network performs at an accuracy of 92% on the held-out test dataset consisting of 10k original CIFAR-10 images. Fig. 4B (left) shows 16 images sampled from the CIFAR-10 dataset where the trained network recognizes 15 out of 16 images correctly, as shown by the text-labels displayed above each image. Following the PGD attack,the network performs at an accuracy of 37% on the held-out adversarial test set consisting of 10k images. Fig. 4B (right) shows 16 adversarially perturbed images that look identical to the images in Fig. 4B (left) to humans, but the trained network recognizes only 4 out of 16 images correctly, as shown by the text-labels above each image. The adversarial attack paradigm clearly demonstrates deep networks' susceptibility to adversarial failure.

Our FIP algorithm provides an efficient strategy to increase network robustness and mitigate adversarial failure by generating path-connected sets of network with diverse weights that can be used collectively for image-classification tasks. Initially, we construct the FIP computationally by setting L=0, in the optimization problem in Eq-4, while $\langle \frac{d\gamma}{dt}, \frac{d\gamma}{dt} \rangle_{\mathbf{g}_{\text{CIFAR10}}}$ is set to the distance moved in the networks' output space for images obtained from the original CIFAR-10 dataset. Following the FIP construction, we sample 10 networks along the FIP in the weight space and construct an ensemble. The ensemble's prediction is computed by feeding an image-input through all the networks in the ensemble followed by summing their 'softmaxed' output across all networks in the ensemble. The adversarial performance of individual networks from the FIP ensemble (composed of 10 networks sampled from the FIP) is 55.61 $\pm$ 1.1 %, with a joint ensemble accuracy of 57.8% (purple solid, dashed line in Fig. 4C), while the DeepNet ensemble (composed of 10 independently trained deep networks) has an adversarial performance of 38.12 $\pm$ 0.44 % for individual networks in the ensemble and a joint ensemble accuracy of 37.18% (orange solid, dashed line in Fig. 4C).

The FIP ensemble has a significantly higher adversarial accuracy than other state of the art ensemble methods, like the DeepNet (DN) ensemble, Adaptive Diversity promoting (ADP) ensemble and the Fast Geometric Ensembling (FGE) method. In Fig. 4D, we compare the adversarial performance of three state-of-art techniques with the FIP ensemble, wherein each ensemble contains 10 deep networks. The FIP ensembles perform at an accuracy of 55.61 $\pm$ 1.1 % on the adversarial examples, ADP at an accuracy of 43.84 $\pm$ 7.8 %, FGE performs at 41.7 $\pm$ 0.34 % and DeepNet (DN) ensemble at an accuracy 38.12 $\pm$ 0.44 % on adversarial input.

As FIP ensembles have a higher intra-ensemble diversity as measured by the representation

diversity score, they are more robust in mitigating adversarial failure. We compute a diversity score for both ensembles (FIP, DN) by evaluating the standard deviation of the L_2 norm of the network's activation across all networks in the ensemble along each layer for a set of image-inputs. In Fig. 4F, we show that the FIP ensemble has a much higher diversity score than the DeepNet ensemble, with a more pronounced difference between the two in the earlier layers (from Layer 1 to Layer 6) and the later layers (Layer 15 and Layer 16).

In addition to having a high intra-ensemble diversity, we find that the networks in the FIP ensemble have low coherence with a trained surrogate network (network instance used to generate adversarial images) as compared to the DN ensemble (Supplementary). In Fig. 4E, the left-shifted coherence distribution of the FIP ensemble (purple histogram) when compared to the DN ensemble (orange histogram) suggests that the adversarial examples generated against the surrogate network are less likely to attack the FIP ensemble, improving robustness of the FIP ensemble against adversarial attacks.

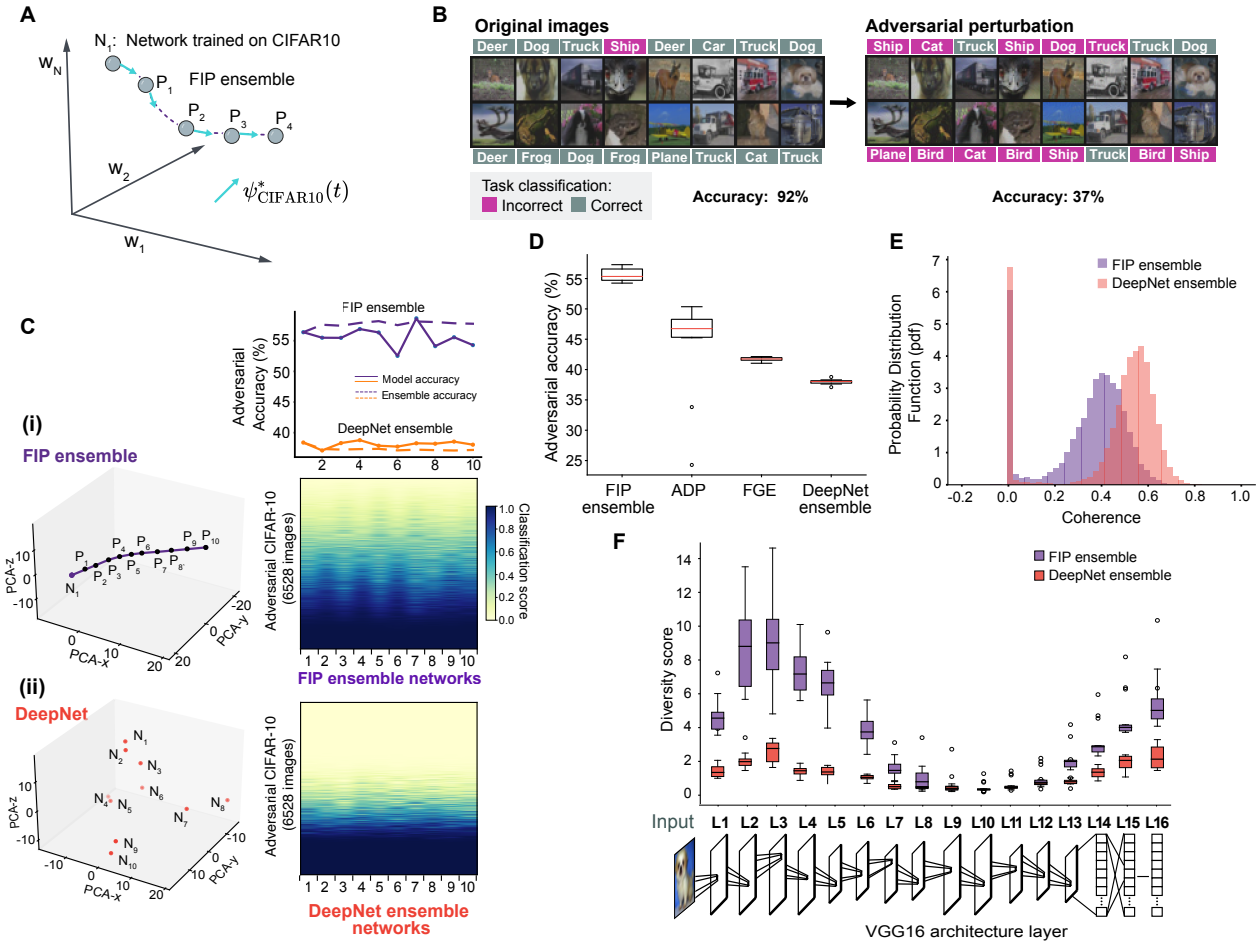
Discussion

We have introduced a mathematical theory and algorithm for training path connected sets of neural networks to solve machine learning problems. Specifically, we show that path connected sets of networks can be applied to diversify the functional behavior of a network, enabling the network to accommodate additional tasks, to prune weights, or generate diverse ensembles of networks for preventing failure to adversarial attack.

Fundamentally, our work exploits a parameter degeneracy that is intrinsic to large mathematical models. Recent work in physics has demonstrated that physical models with large numbers of parameters often contain parameter degeneracy such that model parameters can be set to any value within a sub-manifold of parameter space without loss of accuracy in predicting experimental data [14]. In such situations, input data constrains parameters to lie within a sub-manifold parameter space, but the models contain an intrinsic 'sloppiness' [14]. Sloppiness can emerge for a variety of reasons including low dimensional structure in the input data, blurring out of short-length scale behaviors, and non-linearity. Signatures of model degeneracy exist in the spectrum of the Fisher information matrix of the model which provides an analogous mathematical object to the metric tensor we developed in our path framework.

Modern deep neural networks contain large numbers of parameters that are fit based on training data, and so are similar mathematical objects to physical models with large numbers of parameters set by experimental data , and in fact, exact mappings between statistical mechanics models and neural networks exist [33]. Further, like large physical models, neural networks contain similar sources of degeneracy to physical models. Neural networks utilize non-linear weight functions like the ReLU or sigmoid function. Input data sets used in image classification like MNIST and FashionMNIST or CIFAR also have low dimensional structure [15]. Moreover, if data classes depend only on coarse grained information within an image, then pixel level changes in image interpretation can become insignificant, and entire space of models emerges that can solve a specific image classification problem.

Mathematically, in the supporting materials we show that the neural networks that we analyze have mathematical signatures of significant parameter degeneracy after training, through spectral analysis of the metric tensor. In models we consider, spectral analysis demonstrates that the weight space contains significant sub-spaces where movement of parameters causes insignificant change in network behavior. Our FIP algorithm explores these degenerate sub-spaces or sub-manifolds of parameter space. Implicitly we show that exploration of the degenerate sub-space can find regions of flexibility where parameters can accommodate a second task (a second image classification task) or goal like sparsification. We apply basic methods from differential geometry to identify and traverse



what's ensemble accuracy?

Figure 4: **FIPs in weight space generate ensembles of networks that confer adversarial robustness** (A) Schematic to generate FIP ensemble (P_1,...,P_4) by sampling networks along FIP (purple dotted line) beginning at network-N_1. FIP is constructed by identifying a series of weight perturbations that minimize the distance moved in networks' output space. (B) Original CIFAR10 images (left) and Adversarial CIFAR-10 images (right) are shown. The text-labels (left, right) above the images are predictions made by a network trained on CIFAR-10. Trained networks' accuracy on the original and adversarial images are shown below. (C) (Top) Line-plot (solid) shows the individual network performance on adversarial inputs, (dashed) shows the joint ensemble accuracy on adversarial inputs, for FIP ensemble (purple) and DeepNet ensemble (orange). (i,ii-Left) FIP ensemble in purple (P_1,P_2,...,P_{10}) and DeepNet ensemble in orange (N_1,N_2,...,N_{10}) are visualized on weight space PCA. (i,ii-Right) Heatmaps depict classification score of networks in FIP ensemble and DeepNet ensemble on 6528 adversarial CIFAR-10 examples. (D) Boxplot compares adversarial accuracy (over 10k adversarial examples) across different ensembling techniques (n=3 trials). (E) Histogram of coherence values for FIP (purple) and DeepNet ensemble (orange). (F) Boxplot shows the ensemble diversity score across VGG16 layers over n=1000 CIFAR10 image inputs. The cartoon below depicts the VGG16 network architecture.

these degenerate sub-spaces. In the Supporting Materials we show that additional concepts from differential geometry including the covariant derivative along a weight space path can be applied to refine paths by minimizing not only the velocity along a weight space path but also acceleration. The analysis leads to the concept of a weight space geodesic as a path along which the change in network output is constant.

Broadly, our results shift attention from single networks to the path-connected sets of neural networks that can emerge due to local variation in network weights. The weights within biological neural networks, synaptic strength, can fluctuate due to stochastic chemical effects and the impact of activity based regulation. It will be interesting to ask whether biological neural networks also explore paths of networks to increase their flexibility and robustness. Networks could explore paths through fluctuations or also through the influence of top-down activity. By traversing functionally invariant paths, networks could find routes to learning new tasks or secondary goals. However, the paths are even more broadly useful allowing a single network to take on a range of different functions. In this way, by shifting attention from networks as single points to exploring sub-manifolds of the weight space, we hope our work introduces a potential principle of intelligence and motivates the development of mathematical methods for studying the local and global geometry of functionally invariant solution sets to machine learning problems.

References

[1] D. Silver, *et al.*, *nature* **550**, 354 (2017).

[2] J. Jumper, *et al.*, *Nature* **596**, 583 (2021).

[3] A. Krizhevsky, I. Sutskever, G. E. Hinton, *Advances in neural information processing systems* **25** (2012).

[4] N. Sünderhauf, *et al.*, *The International journal of robotics research* **37**, 405 (2018).

[5] S. Smale, *The mathematical intelligencer* **20**, 7 (1998).

[6] T. S. Lee, D. Mumford, *JOSA A* **20**, 1434 (2003).

[7] M. McCloskey, N. J. Cohen, *Psychology of learning and motivation* (Elsevier, 1989), vol. 24, pp. 109–165.

[8] R. Ratcliff, *Psychological review* **97**, 285 (1990).

[9] P. Y. Simard, Y. A. LeCun, J. S. Denker, B. Victorri, *Neural networks: tricks of the trade* (Springer, 1998), pp. 239–274.

[10] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *nature* **323**, 533 (1986).

[11] J. Minxha, R. Adolphs, S. Fusi, A. N. Mamelak, U. Rutishauser, *Science* **368**, eaba3313 (2020).

[12] W. Mau, M. E. Hasselmo, D. J. Cai, *Elife* **9**, e63550 (2020).

[13] C. Stringer, *et al.*, *Science* **364**, eaav7893 (2019).

[14] B. B. Machta, R. Chachra, M. K. Transtrum, J. P. Sethna, *Science* **342**, 604 (2013).

[15] D. Mumford, A. Desolneux, *Pattern theory: the stochastic analysis of real-world signals* (CRC Press, 2010).

[16] S.-i. Amari, *Information geometry and its applications*, vol. 194 (Springer, 2016).

[17] I. Benn, R. Tucker, *An introduction to spinors and geometry with applications in physics* (Adam Hilger Ltd, 1987).

[18] D. H. Mache, J. Szabados, M. G. de Bruin, *Trends and Applications in Constructive Approximation*, vol. 151 (Springer Science & Business Media, 2006).

[19] E. W. Weisstein, *https://mathworld. wolfram. com/* (2014).

[20] J. Kirkpatrick, *et al.*, *Proceedings of the national academy of sciences* **114**, 3521 (2017).

[21] D. Lopez-Paz, M. Ranzato, *Advances in neural information processing systems* **30** (2017).

[22] P. Kaushik, A. Gain, A. Kortylewski, A. Yuille, *arXiv preprint arXiv:2102.11343* (2021).

[23] Y. LeCun, P. Haffner, L. Bottou, Y. Bengio, *Shape, contour and grouping in computer vision* (Springer, 1999), pp. 319–345.

[24] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *Proceedings of the IEEE* **86**, 2278 (1998).

[25] H. Xiao, K. Rasul, R. Vollgraf, *arXiv preprint arXiv:1708.07747* (2017).

[26] G. M. van de Ven, H. T. Siegelmann, A. S. Tolias, *Nature communications* **11**, 1 (2020).

[27] G. M. Van de Ven, A. S. Tolias, *arXiv preprint arXiv:1904.07734* (2019).

[28] X. Glorot, Y. Bengio, *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (JMLR Workshop and Conference Proceedings, 2010), pp. 249–256.

[29] D. Blalock, J. J. G. Ortiz, J. Frankle, J. Guttag, *arXiv preprint arXiv:2003.03033* (2020).

[30] J. Frankle, M. Carbin, *arXiv preprint arXiv:1803.03635* (2018).

[31] K. He, X. Zhang, S. Ren, J. Sun, *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[32] I. J. Goodfellow, J. Shlens, C. Szegedy, *arXiv preprint arXiv:1412.6572* (2014).

[33] P. Mehta, D. J. Schwab, *arXiv preprint arXiv:1410.3831* (2014).