Stochastic Wiring of Cell Types Enhances Fitness by Generating Phenotypic Variability

Divyansha Lachi[1,6*], Ann Huang[2,3,7*], Augustine N. Mavor-Parker[4,*†]
Arna Ghosh[2,3], Blake Richards[2,3,5§], Anthony Zador[1,§]

Abstract

The development of neural connectivity is a crucial biological process that gives rise to diverse brain circuits and behaviors. Neural development is a stochastic process, but this stochasticity is often treated as a nuisance to overcome rather than as a functional advantage. Here we use a computational model, in which connection probabilities between discrete cell types are genetically specified, to investigate the benefits of stochasticity in the development of neural wiring. We show that this model can be viewed as a generalization of a powerful class of artificial neural networks—Bayesian neural networks—where each network parameter is a sample from a distribution. Our results reveal that stochasticity confers a greater benefit in large networks and variable environments, which may explain its role in organisms with larger brains. Surprisingly, we find that the average fitness over a population of agents is higher than a single agent defined by the average connection probability. Our model reveals how developmental stochasticity, by inducing a form of non-heritable phenotypic variability, can increase the probability that at least some individuals will survive in rapidly changing, unpredictable environments. Our results suggest how stochasticity may be an important feature rather than a bug in neural development.

1 Introduction

Animals are born with a remarkable array of innate abilities, which enable them to perform essential behaviors from the moment they enter the world, often even in the absence of instructive experience. Colts can stand and walk within hours of birth, spiders can hunt and capture prey, and ducklings can swim almost immediately after hatching—all without parental instruction or extensive trial-and-error. When old field mice, which build long burrows with an emergency exit, are reared from birth by closely related deer mice that build simpler burrows without a second exit, the adopted pups build burrows similar to those of their genetic parents rather than their foster parents (Weber et al., 2013). Even more striking, fish in which all neuronal activity has been blocked during development can swim and respond appropriately to visual stimuli from the moment neuronal activity is restored, demonstrating that these behaviors can develop without any activity-dependent learning (Barabási et al., 2024). These innate abilities, present at birth and developed independently of experience,

[1]Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, USA
[2]McGill University, Montreal, QC, Canada
[3]Mila, Montreal, QC, Canada
[4]University College London, London, UK
[5]CIFAR, Toronto, ON, Canada
[6]Current address for D.L: Georgia Institute of Technology, Atlanta, GA, US
[7]Current address for A.H: Harvard University, Cambridge, MA, US

[*]These first authors contributed equally to this work.
[§]These last authors contributed equally to this work.
[†]Work partially done during internship at Cold Spring Harbor Laboratory.

must be controlled by neural circuitry whose wiring is encoded in the genome and read out by the process of development.

Large brains can contain an enormous number of neurons connected by an even larger number of synapses. Naively, specifying the connectivity matrix of a human brain with $\sim 10^{11}$ neurons would require storing connection information for each of the $\sim 10^{14}$ synapses, amounting to at least 10^{15} bits of information. In stark contrast, the human genome comprises only about 3 billion base pairs, or roughly 10^9 bits, revealing a discrepancy of about six orders of magnitude (Wei et al., 2013; Zador, 2019). Thus, even if every nucleotide in the human genome were fully dedicated to optimally and efficiently specifying the precise connections in the brain, the genome would still fall far short of the required information capacity. The limited information capacity of the genome in comparison to the complexity of the connectome, a mismatch that has been referred to as the "genomic bottleneck" (Zador, 2019), implies some form of compression in the developmental specification of brain wiring. The need for such compression underscores the fact that brain development follows orderly rules and is not merely a "look-up table" (Mitchell & Cheney, 2024). This principle is implicit in developmental neuroscience, which seeks to characterize the nature of those rules.

Recent theoretical work has begun to explore how the genome might efficiently encode the connectivity of large brains despite its limited information capacity (Barabási et al., 2023; Koulakov et al., 2021). In the "deterministic genomic bottleneck" (DGB) model (Koulakov et al., 2021), inspired by receptor-ligand based wiring rules, a smaller "genomic" neural network was used to compress the weights of a much larger "phenotypic" artificial neural network model. The model, which was deterministic in the sense that a given model genome reliably and reproducibly generates exactly one connectome, demonstrated excellent "zero-shot" learning—good performance upon initialization, without further training—corresponding to animals' advanced innate abilities at birth.

Here, we consider the implications of developmental stochasticity (Mitchell, 2018; Ballouz et al., 2023; Linneweber et al., 2020; Vogt, 2015; Koulakov & Tsigankov, 2004; Tsigankov & Koulakov, 2009; Mitchell, 2024)—the fact that a given genome specifies an ensemble of different connectomes—by extending the deterministic genomic bottleneck framework. Building on the "probabilistic skeleton" approach (Stöckl et al., 2021), we propose a model in which the genome encodes the parameters of probability distributions over synaptic strengths between different cell types. Individual connectomes are then sampled from these genetically-encoded distributions. Mathematically, this model can be seen as a generalization of Bayesian neural networks (Blundell et al., 2015), with the genome encoding priors over weights rather than encoding the weights themselves.

We find that this stochastic genomic bottleneck (SGB) model offers several advantages over the DGB model. First, for high levels of compression (i.e., as the number of connections is large relative to the amount of genetic information), the SGB achieves better performance than the DGB, suggesting that stochasticity might be particularly important in larger brains. Second, in simulated physics environments, the SGB model produces individual networks that are specialized to unseen environment niches and body morphologies. Finally, we prove analytically, and confirm empirically, that the average performance of networks sampled from an SGB is higher than the performance of the average network encoded by the genome, providing a population-level evolutionary advantage to stochasticity.

These results suggest that stochasticity may be beneficial for neural development, particularly for animals with large brains that must compress a large amount of connectivity information into a relatively small genome. Stochasticity allows for the generation of diverse networks, some of which may be particularly well-suited to the specific environmental niche of the organism. This work thus provides a theoretical foundation for understanding the role of noise in brain development, and suggests that embracing and harnessing developmental stochasticity may be a key principle in the evolution of complex brains.

2 Results

We begin by describing the Stochastic Genomic Bottleneck (SGB) framework in Section 2.1. We then test the SGB framework by applying it to common supervised learning benchmarks in Section 2.2. We focus on what we will refer to as "innate performance" (i.e., performance of the initialized network corresponding to an agent "at birth", without further training), which in the machine learning literature is often referred to as "zero-shot" learning. Then, we consider the innate performance of the SGB framework in a reinforcement learning setting, on standard continuous control environments (Freeman et al., 2021) in Section 2.3.1, and then in a high-throughput robotics simulator (Makoviychuk et al., 2021) in Section 2.3.2.

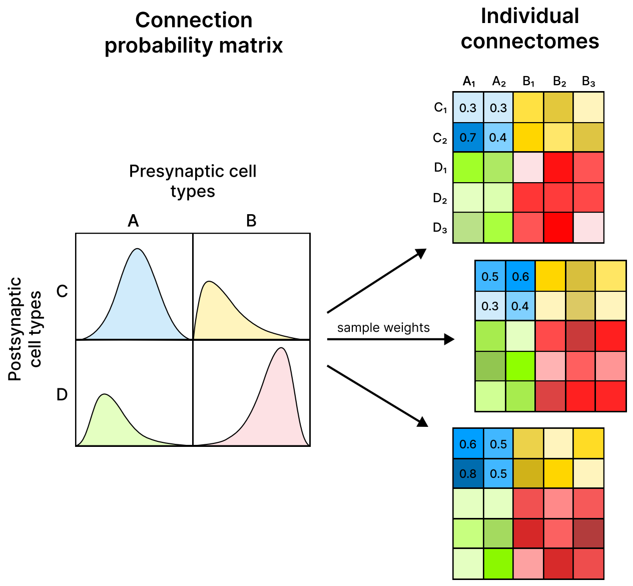

Figure 1: **Schematic of the Stochastic Genomic Bottleneck (SGB)**: The SGB framework uses a low-dimensional parameter set to specify the mean and variance of the synaptic weight distribution between neuronal cell types, allowing individual weights to be sampled stochastically. Unlike the deterministic approach of the DGB, the SGB leverages stochastic sampling to introduce variability in neural wiring. Neurons are grouped into distinct cell types, with their synaptic connections described by a connection probability matrix. This schematic shows the basic setup of generating individual connectomes from the connection probability matrix.

2.1 Stochastic Genomic Bottleneck Model

In the Deterministic Genomic Bottleneck (DGB), a "genomic" neural network learns to deterministically specify the weight of the synaptic connections between any two neurons in a "phenotypic" network (Koulakov et al., 2021). This phenotypic network is first trained in order to ensure that the genomic network learns connectomes that provide good performance on a task. The genomic network can be thought of as a compressed version of the phenotypic network because it has fewer parameters (refer to Section 4.3.2 for more details).

To study the impact of stochasticity on development, we introduce the Stochastic Genomic Bottleneck (SGB) algorithm. The SGB algorithm, inspired by the probabilistic skeleton of Stöckl et al. (2021), prespecifies neurons in the network as belonging to different cell types and models the connection strength between neurons via cell-type specific connection probabilities. The original model of Stöckl et al. (2021), which employed spiking neurons, was not differentiable and used an evolutionary strategy-based algorithms for optimization. In addition, the probabilistic skeleton had structured connectivity. Here, we reformulate the model to learn the distribution of connection weights between each pair of cell types under the Bayesian framework (Figure 1). Compression occurs as long as the number of cell types is less than the number of neurons. Conceptually, the SGB is a generalization of the standard Bayesian neural network (see Section 4), but rather than specifying a unique weight distribution between every pair of neurons, here instead the weight distribution is shared across all neurons of a given cell type.

To generate individual connectomes within the SGB framework, we begin with a predefined set of k distinct neuronal cell types, denoted as $C = \{c_1, c_2, \ldots c_k\}$. When applying SGB to feedforward layers, each neuronal cell type is either presynaptic or postsynaptic. The synaptic connections between these cell types are therefore described by a two-dimensional connection probability matrix $\mathbf{P}$, where each element P_{ij} , indicates the connection probability between the presynaptic neuronal types i and postsynaptic neuronal type j. Each element P_{ij} of the connection probability matrix $\mathbf{P}$ is parameterized by a prior distribution as described in Section 4.3.

Each individual's connectome $\mathbf{W}_q$ is then generated by sampling from the connection probability matrix $\mathbf{P}$. The matrix $\mathbf{W}_q$ is larger than $\mathbf{P}$ because it represents the actual synaptic weights between individual neurons, not just between cell types. The generation of $\mathbf{W}_q$ involves creating a block structure where each block corresponds to the connection strength between neurons of a specific pair of cell types. This process is formalized as follows:

$$W_{qij}^{ab} \sim \text{Normal}(\mu_{ij}, \sigma_{ij}^2) \quad \text{for each neuron } a \text{ of type } i \text{ and each neuron } b \text{ of type } j$$

where a and b index neurons within their respective cell types, and μ_{ij} and σ_{ij} are the mean and variance of the Gaussian distributions derived from P_{ij}.

When applied to convolutional layers, we generalize the notion of having distinct cell types to each dimension of the weight tensor. Therefore, the connection probability matrix $\mathbf{P}$ is four-dimensional, enabling weight sharing along both dimensions of the filters, the input channels, and the output channels. Similarly, we can sample individual connectomes $\mathbf{W}_q$ from the connection probability matrix $\mathbf{P}$.

The parameters of the SGB are optimized with gradient descent using the reparameterization trick on an evidence lower-bound loss (ELBO) to generate phenotypic networks that perform well at a given task (refer more details in Section 4). In the SGB framework, network complexity and architecture are parametrized by the number of distinct neuronal cell types, k. For $k = 1$, the framework reduces to a conventional neural network initialization (except it does not scale weight distributions by the number of input/output connections (Glorot & Bengio, 2010)). Conversely, at the upper limit where k equals the number of neurons—effectively treating each neuron as its own distinct cell type—the model transitions to a full standard Bayesian neural network.

2.2 Stochastic compression outperforms deterministic compression in supervised learning

We first compared the performance of the SGB to DGB algorithms on two well-studied supervised learning tasks, the MNIST and CIFAR image classification benchmarks (Section 2.1). As noted above, both the DGB and SGB seek to compress a phenotypic network into a smaller set of parameters, but whereas the DGB deterministically sets the synaptic weights in the phenotypic network, in the SGB the parameters specify probability distributions over the synaptic weights of the phenotypic network. We evaluate the performance of these models under varying levels of compression, defined as the ratio between the number of parameters in a standard uncompressed network and an equivalent compressed network Figure 2.2. For both models, "innate performance" (defined as performance of the SGB- or DGB-initialized model on held-out test data before any further training data are presented) is plotted as a function of compression. As expected, performance drops monotonically with increasing compression for both models, analogous to how lossy image compression techniques such as jpeg result in blurrier reconstructions as compression increases.

Comparing the two models, we find that for small compression levels the DGB performs better, but at higher levels of compression the SGB outperforms the DGB on both MNIST and CIFAR. Because the DGB is a special case of the SGB, in which the standard deviation of the weight distribution is zero, it might seem surprising that the DGB could ever outperform the SGB. However, the KL divergence penalty imposed on the posterior in the SGB prevents the standard deviation from reaching zero, ensuring the posterior remains a valid probabilistic distribution (refer to equation 1). If it did reach zero, the KL divergence would become infinite. This accounts for the observed performance differences at lower compression levels.

Furthermore, the SGB maintains remarkably stable performance across a wide range of compression levels, demonstrating its robustness and efficiency in handling high compression. These results suggest that stochastic developmental processes may be particularly beneficial for organisms with large brains, which face a greater challenge in genetically encoding their vast neural connectivity patterns (Zador, 2019).

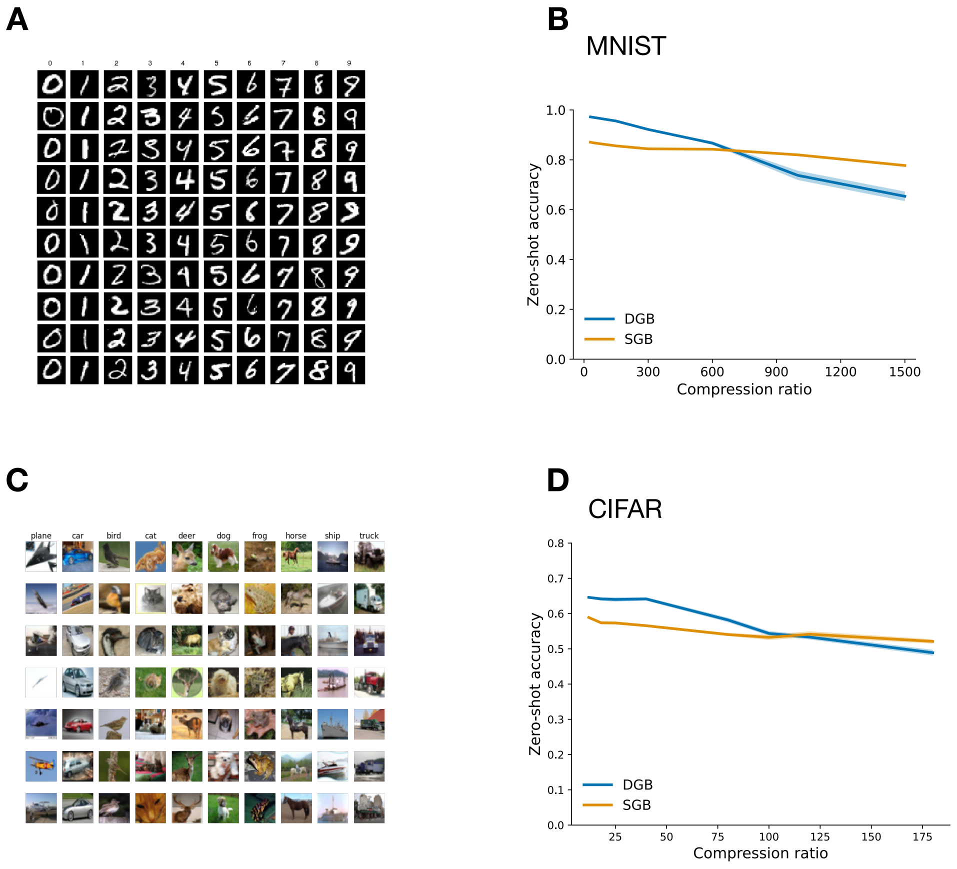

Figure 2: **Stochastic genomic gottleneck (SGB) leads to better innate performance at high compression compared to the deterministic genomic bottleneck (DGB). (A)** Examples of MNIST dataset of handwritten digits. **(B)** Innate performance on MNIST, measured by the zero-shot classification accuracy, for both the DGB and SGB models across various levels of compression. Solid lines represent the mean zero-shot accuracy across five seeds with shaded areas representing the standard error of the mean. **(C)** Examples of CIFAR dataset of images. **(D)** Innate performance on CIFAR for both DGB and SGB model for various levels of compression across five seeds.

2.3 SGB for Reinforcement Learning

We next tested the SGB framework on reinforcement learning tasks, which are more relevant to the kinds of learning and decision-making problems faced by animals in their natural environments. In contrast to supervised learning, where the goal is to learn a mapping from inputs to outputs based on a labeled dataset, reinforcement learning involves learning to make a sequence of actions in an environment to maximize a cumulative reward signal (Sutton & Barto, 2018). This trial-and-error process, guided by rewards, more closely resembles how animals learn from experience to navigate their world and acquire complex behaviors.

We used reinforcement learning tasks inspired by the motor control challenges faced by animals. Just as organisms must learn to efficiently control their bodies to navigate environments and acquire resources, our RL agents must learn to map sensory inputs to motor outputs to maximize reward. This allows us to study how an agent's learning and performance are shaped by its embodiment and environment (Pfeifer & Bongard, 2006), key factors in both biological and artificial intelligence.

We test SGB on continuous control tasks in a simulated physics environment, BRAX (Freeman et al., 2021), which loosely mimics the motor control challenges of legged animals. In Section 2.3.1, we benchmark SGB on the Ant and Halfcheetah environments from Freeman et al. (2021). In Section 2.3.2, we experiment with controlling the Ant agent with different body morphologies, and controlling a simulated quadruped ANYmal robot (Hutter et al., 2016) on different terrains.

For our reinforcement learning experiments, we employ Proximal Policy Optimization (PPO) (Schulman et al., 2017), a standard on-policy actor-critic algorithm that offers a good trade-off between sample efficiency and wall-clock training time. We use SGB layers throughout the agent's policy and value networks, except for the output layers which remain deterministic. Full architectural details and hyperparameters are provided in Appendix 4.5. We train SGB agents with varying numbers of cell types, as shown in Figure 3, and measure their performance as a fraction of the return achieved by an uncompressed baseline agent. For the Brax experiments, the baseline agent has a hidden layer width of 128, so an SGB agent with 128 cell types experiences no compression. For the ANYmal experiments, the SGB agent has 256 cell types compared to the baseline's hidden layer width of 1024.

2.3.1 Stochastic compression on motor control tasks

In both the Ant and Halfcheetah tasks, the goal for the corresponding agent is to learn to move its body forward, while minimizing constraints such as the energy expended. The state space of both tasks consists of the positions and velocities of the agent's limbs, and the action space is a continuous vector of forces it can apply to its different limbs. The performance of the Ant and Halfcheetah agents as a proportion of a vanilla PPO agent (non-compressed, deterministic) is shown in Figure 3. When the hidden layer width of the SGB agent is equal to that of the vanilla agent, the SGB agent is able to achieve a significant proportion of the return achieved by the vanilla agent ($\approx 75\%$ for Ant and $\approx 45\%$ for Halfcheetah). As the number of cell types decreases, the performance of the SGB decreases gradually until the region of 16 cell types, where the SGB performance falls off quickly, indicating that it is in this region where the SGB network does not have enough expressive capacity to represent a reasonable policy for the SGB. These results suggest that there is a minimal number of cell types needed to endow a network with the needed flexibility.

2.3.2 SGB induces population-level diversity in behavioral capacity

Ant motor control with varied leg lengths: Next, we sought to examine the influence of developmental stochasticity on the phenotypic diversity and adaptability of reinforcement learning agents to variations in body morphologies. We optimized the SGB on the Ant environment and sampled 100 agents generated by the SGB. These networks were tested for their innate (zero-shot) performance within the same environment and on a modified version of the Ant environment where the four legs of the agent were lengthened Figure 4B. Although on long time scales it seems plausible

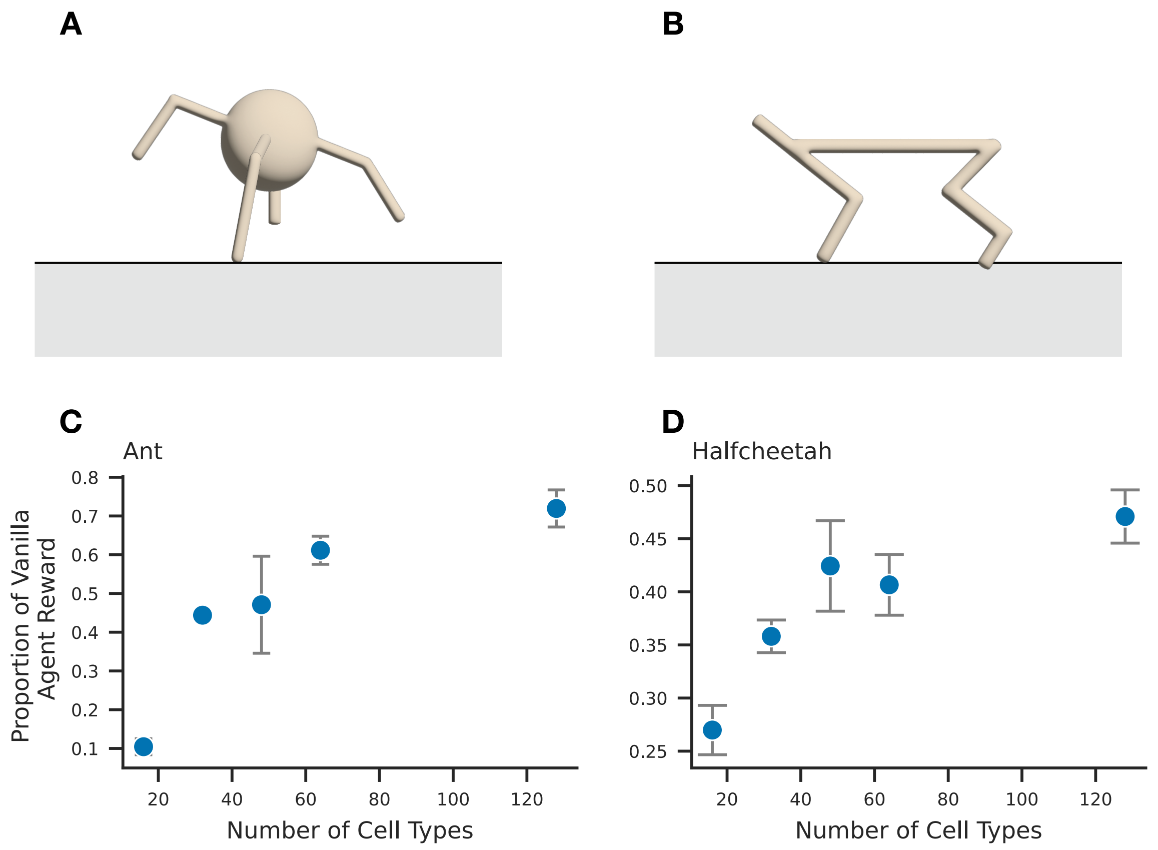

Figure 3: **SGB endows reinforcement learning agents with innate motor control capacities**: **(A)** Illustration of the Ant reinforcement learning environment where the network learns to coordinate the four legs of the ant to move forward as fast as possible. **(B)** Illustration of Halfcheetah environment where the goal is to apply appropriate torques on the joints to make the halfcheetah run as fast as possible. **(C)** Zero-shot performance as a function of different prespecified number of cell types in the network. Here, zero-shot performance is measured by the reward obtained by the compressed stochastic networks (the SGB models) normalized by that of networks without compression. Error bars represent standard error of the mean across five seeds. **(D)** Similar to (C) but for the HalfCheetah environment.

that brains and bodies co-evolve, on short time scales the brain specified by a given genetic program must learn to control whatever body happens to develop, which can be shaped by forces in an animal's lifetime (e.g. the availability of nutrients for growth).

Figure 4E shows a scatterplot of the performance of each network controlling a standard body (normal legs) vs a body with long legs. Strikingly, there is considerable scatter around the mean for both conditions, indicating that different networks drawn from the same distribution can achieve very different levels of performance with different body morphologies. Thus, developmental stochasticity can lead to marked phenotypic diversity of behavioral capacities over the population. Furthermore, performance on the two conditions are positively correlated (Pearson's r = 0.64, p $<$ 0.001), suggesting that certain "athletic" neural phenotypes are intrinsically better at performing the locomotion task, regardless of variations in body morphology.

ANYmal Robot Control on Varied Terrains: In nature, animals must navigate a wide variety of terrains, from flat plains to rocky hillsides. From an evolutionary perspective, this diversity of environments may favor the development of specialized individuals within a population, each adapted to a particular niche. We hypothesize that stochastic development, by introducing variability in neural wiring, could be a mechanism for generating this specialization. To test this idea in a controlled setting, we use the ANYmal robot environment from Rudin et al. (2021), which challenges a reinforcement learning agent to control a quadruped robot in traversing different terrains, including rough terrain and stairs. The agent must learn to map sensory inputs (the state space, containing information about the terrain and past actions) to motor outputs (the action space, dictating the torques applied to the robot's joints) in order to navigate each terrain successfully.

We train the SGB on all terrains simultaneously, mimicking the diversity of environments encountered in nature. Then, at test time, we evaluate the innate performance of 100 networks sampled from the SGB on the "rough" and "stairs" terrains separately. Strikingly, we find that the sampled networks show considerable individual differences in their ability to navigate the two terrains. Some networks excel at traversing rough terrain but struggle with stairs, while others show the opposite specialization. Quantitatively, we find no significant correlation between a network's performance on rough terrain and its performance on stairs (Pearson's $r = 0.065$, $p = 0.52$), unlike the strong positive correlation between long-leg and short-leg performance in the Ant task (Section 2.3.2). This result suggests that developmental noise can indeed lead to specialization to environmental niches, with different individuals within a population being better adapted to different niches.

2.4 Stochastic sampling gives better average performance

To investigate the possibility that the stochasticity of brain development may actually be advantageous, we compared the performance of networks created via stochastic sampling to that of a “mean network”, i.e. where the synaptic weights are simply set to be the means of the specified distributions, which represents the limit case of sampling with zero variance. We conducted this comparison in the reinforcement learning setting, using the Ant environment, and evaluated the mean performance over 100 rollouts for both the sampled networks and the mean network.

Strikingly, we found that the mean performance of a population of stochastically sampled networks exceeds the performance of the mean network. We found that the mean performance over 100 rollouts was substantially higher for the sampled networks than for the mean network ($t(99) = 12, p < 0.001$) Figure 5). Assuming a monotonic relationship between the task-specific loss function and performance (where lower loss implies higher performance), these results can be summarized as follows: the loss of the mean network is greater than the mean loss of the sampled networks. According to Jensen's inequality (Jensen, 1906), this outcome suggests a concave loss function. In Appendix (A.2), we provide a mathematical proof demonstrating how gradient-based optimization leads to sampling networks over a concave slice of the loss landscape. The proof offers a mechanistic explanation for the seemingly counterintuitive result: A diverse population of brains, each wired

[1]Adapted from (Azabou et al., 2023)

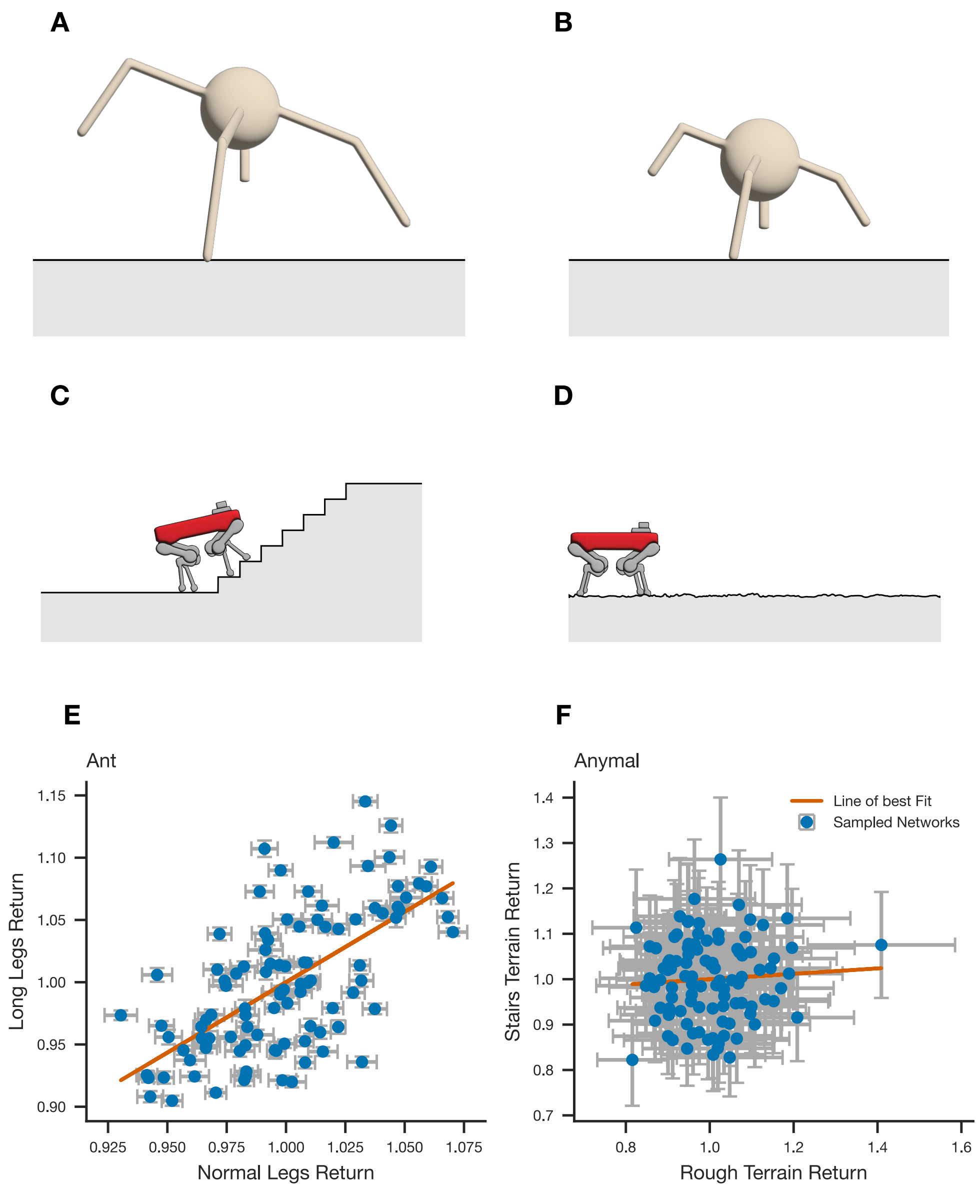

Figure 4: **SGB induces phenotypic diversity for different body morphologies and environmental terrains**: **(A)** Illustration of the Ant environment with long legs, contrasted with **(B)** the Ant environment with normal legs. **(C)** Illustration showing Anymal robot[1] on the stairs terrain versus **(D)** the rough terrain. **(E)** Innate performance of sampled networks for the long leg versus normal leg Ant environment. The SGB is first evolved on the normal leg environment. After the SGB has been optimized, we sampled networks from the trained SGB and tested their zero-shot performance without any learning on both environments. Each dot is one sampled network from the evolved SGB tested on the two environments. The networks' returns are positively correlated on the two body morphologies (Pearson's $r = 0.64$, $p < 0.001$), while also exhibiting phenotypic variability in their capacities for each body morphology. The error bars on each dot represent the standard error of the mean in the return (the reward) obtained by the sampled network across multiple episodes on the corresponding environment. **(F)** Similar to (E), where the SGB was evolved on only the stair terrain, while the subsequently sampled networks were tested for their zero-shot performance on both the stairs and the rough terrains. The sampled networks exhibit substantial phenotypic diversity such that on each environmental terrain, there are individual networks performing notable better than the rest of the population. There is no significant correlation between a network's performance on rough terrain and its performance on stairs (Pearson's $r = 0.065$, $p = 0.52$).

slightly differently due to developmental noise, collectively outperforms a population of identical brains wired according to the mean connectivity. Thus, at the population level, stochasticity yields better performance.

These results suggest that the stochasticity of brain development may not be merely a bug, but rather a feature that evolution has harnessed to enhance behavioral adaptability. A population of diverse brains, collectively exploring a wider range of cognitive and behavioral strategies, may be more resilient to environmental challenges than a phenotypically identical population. This diversification may be especially advantageous in the face of compression.

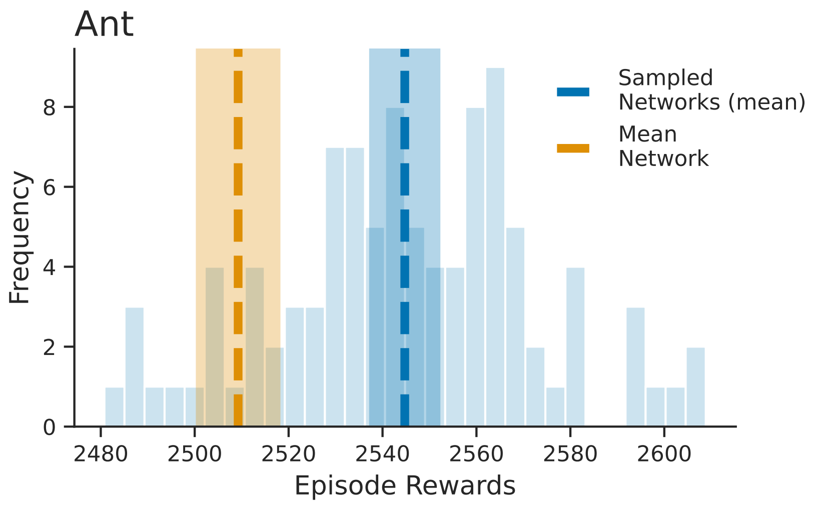

Figure 5: **Stochastic sampling gives better average performance across a population of sampled networks**: Zero-shot performance of sampled networks and mean network with 64 cell types on the Ant environment. Shaded region indicates 99% confidence intervals.

3 Discussion

In this study, we introduced the Stochastic Genomic Bottleneck (SGB) model, a differentiable algorithm that incorporates stochasticity into the process of generating neural network architectures from compressed genomic representations (Stöckl et al., 2021). Inspired by the noisy nature of biological development (Mitchell, 2018; Ballouz et al., 2023; Linneweber et al., 2020), our model suggests that stochastic rules provide a powerful approach to overcoming the limited information contained in the genome, resulting in the generation of diverse and complex neural connectomes. Our model can be seen as a generalization of Bayesian neural networks and offers a new perspective on how brains can efficiently develop from genetic information.

Through experiments on both supervised learning and reinforcement learning tasks, we demonstrated that SGB maintains stability across a wide range of compression levels and significantly outperforms deterministic models in high compression settings. We find that SGB excels in large networks and variable environments, exhibiting superior adaptability and robustness due to the incorporation of stochasticity. Furthermore, we provide analytical proofs and empirical validation showing that the mean performance of connectomes sampled through our stochastic model surpasses the performance of the average network generated without stochasticity in complex loss landscapes. This indicates that stochastic sampling around the mean provides an advantage over generating each synaptic weight precisely according to a deterministic rule.

Our model draws an analogy between the processes of evolution and development. The optimization of cell type connectivities in our model represents the role of evolution in shaping the genetic blueprint for brain development. For computational efficiency, we have constructed our model to be differentiable, which greatly facilitates navigation in high-dimensional spaces. However, it is important to note that at the level of an individual, Darwinian evolution is not differentiable: the specific genetic mutations and recombinations that appear in the offspring do not benefit from the life experiences of the parents. Thus, we are not positing that our model is an accurate reflection of evolution and neural development. Rather, it provides a proof-of-concept that stochastic development can be functionally useful.

The phenotypic variability discussed in our study differs from traditional sources of phenotypic variability, which arise from genotypic variability and environmental factors (nature and nurture). The importance of stochasticity (chance) as a third source of phenotypic variability is increasingly recognized (Mitchell, 2018). While phenotypic variability arising from genotypic variability is heritable, variability resulting from stochasticity is not. Growing evidence supports the significance of stochastically induced phenotypic variability. For example, in the nine-banded armadillo, a species which uniquely among mammals reproduces with genetically identical quadruplets, minor stochastic variations in gene expression during early development have lasting impacts. This highlights the profound influence of developmental noise on individual diversity beyond genetic and environmental determinants (Ballouz et al., 2023). In Drosophila melanogaster, nonheritable, stochastic variations in brain wiring within the dorsal cluster neurons (DCN) contribute to individual differences in object orientation behavior, demonstrating a direct link between brain development noise and behavioral variation. Furthermore, in neurodevelopmental disorders like schizophrenia and autism concordance can be as high as 50% (Imamura et al., 2020), raising the possibility that early stochastic development may play a role in the remaining variation.

In engineering, noise is often considered a nuisance, and great efforts are made to manufacture parts with high precision. However, our findings suggest that in some biological contexts, noise may actually be advantageous. The incorporation of stochasticity in neural development can lead to increased adaptability and robustness, especially in large networks and variable environments.

Our work builds on a long tradition of computational models exploring how patterned neural connectivity emerges through development. Early pioneers like Turing proposed models of how simple local interactions could give rise to structured patterns in biological systems (Turing, 1952). In the domain of neural development, Hebbian learning rules can explain the emergence of center-surround receptive fields and ocular dominance columns in the visual system (Linsker, 1988; Miller et al., 1989; Miller & MacKay, 1994). More recently, researchers have explored how structured connectivity can arise through various mechanisms, including synaptic plasticity (Clopath et al., 2010), axon guidance (Goodhill, 2007), and spontaneous activity (Albert et al., 2008). However, these models typically focus on recapitulating observed connectivity patterns rather than directly optimizing for computational function. In contrast, our approach starts from the assumption that the network must perform a specific task and asks how this computational goal constrains the developmental process. By directly optimizing for performance, our model provides a complementary perspective on how evolution and development might shape neural connectivity to support adaptive behavior.

The significance of our findings goes beyond biological inspiration. They suggest new directions for developing AI models that are more flexible and capable of dealing with uncertain environments. For example, ANN researchers could first optimize an SGB in a generic environment, sample individual networks with good zero-shot performance from the SGB, and then train those individual networks on various tasks with much less data than would be required to train them from scratch. This could prove particularly advantageous in reinforcement learning settings where data collection can be expensive and time-consuming.

Our results highlight the power of incorporating randomness into the development of neural networks, offering a more nuanced approach to bridging genetics and brain structure. Our work not only advances our understanding of biological processes but also opens up new possibilities for creating more adaptable and resilient artificial intelligence systems. By embracing the inherent stochasticity in developmental processes, we hope to gain insights not only into the evolution of natural intelligence but also the design of artificial ones.

4 Methods

4.1 Preliminaries

First, we describe the model in the supervised learning setting. Using the notation of (Goodfellow et al., 2016), let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M\}$ denote our data and $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_M\}$ be the targets. Typically, deep neural networks train on a maximum likelihood loss, which aims to maximise the likelihood of the targets given the input data and neural network parameters

$$\theta^* = \arg\max_{\theta} p(\mathbf{Y}|\mathbf{X}; \theta),$$

where θ^* is the set of optimal parameters (Goodfellow et al., 2016; Blundell et al., 2015). In the case of the stochastic genomic bottleneck, we instead aim to learn a posterior distribution over weights with approximate Bayesian inference.

4.2 Bayesian Neural Networks

Bayesian neural networks (BNNs) have gained significant attention in the machine learning community due to their ability to capture uncertainty in model predictions by treating the weights of a neural network as probability distributions rather than fixed values (Blundell et al., 2015; Wilson & Izmailov, 2022; Ghahramani, 2015; Kendall & Gal, 2017). This allows BNNs to express the confidence of their predictions, which is particularly valuable in decision-making scenarios where the cost of an incorrect prediction is high (Kendall & Gal, 2017; Leibig et al., 2017; Kahn et al., 2017). Furthermore, uncertainty quantification makes BNNs more robust to overfitting, enables them to generalize better to new data, and provides a principled way to incorporate prior knowledge into the model, allowing for more efficient learning and better performance in data-scarce settings Blundell et al. (2015); Fortunato et al. (2017); Louizos & Welling (2017). The Bayesian framework also facilitates model comparison and selection, as it naturally penalizes overly complex models through the marginal likelihood (MacKay, 1992). These advantages have led to the application of BNNs in various domains, such as healthcare, finance, and autonomous systems, where reliable uncertainty estimates and robust decision-making are crucial (Kahn et al., 2017; Leibig et al., 2017).

Instead of learning a single scalar value for each weight and bias parameter, BNNs aim to learn a posterior distribution over weights by employing Bayes rule, meaning weight matrices are sampled from an underlying distribution parametrized by θ: $\mathbf{W} \sim P(\mathbf{W}|\theta)$ (Blundell et al., 2015). However, when the input data $\mathbf{X}$ is high-dimensional, computing the full posterior distribution over weights becomes computationally expensive. As a result, practitioners typically learn an approximate posterior with "Bayes by Backprop", where variational inference is performed by maximizing the ELBO (Blundell et al., 2015):

$$\text{ELBO}(\mathbf{X}, \mathbf{Y}, \theta) = \mathbb{E}_{q(\mathbf{W}|\theta)}\left[\log P(\mathbf{X}, \mathbf{Y}|\mathbf{W})\right] - d_{KL}\left[q(\mathbf{W}|\theta) \| P(\mathbf{W})\right] \tag{1}$$

where $\mathbf{W}$ refers to the set of weights of the BNN, and $q(\mathbf{W}|\theta)$ is the distribution that approximates the true posterior. Intuitively, maximizing the ELBO requires optimizing the posterior to match the prior distribution $P(\mathbf{W})$ while also satisfying the likelihood of data under network weights. In practice, each conventional neural network weight is represented by a mean and a variance parameter. The ELBO is then maximized by averaging losses over multiple samples from the weight distributions using the reparameterization trick (Kingma & Welling, 2013), which allows for backpropagating gradients through to the underlying distribution parameters when stochastically sampling network weights.

The SGB framework combines the idea of Bayesian neural networks with the genomic bottleneck (Koulakov et al., 2021). The genomic bottleneck builds upon hyper-networks (Ha et al., 2017), which are (relatively) small networks that are used to generate the weights of a larger network. Krueger et al. (2017) also combine the ideas of Bayesian neural networks and hyper-networks, but generates a particular set of parameters instead of generating all parameters of the network (as we do in SGB).

4.3 Parameterization of Weight Distributions

Instead of defining the distribution of connection strength between each pair of neurons individually, SGB groups neurons into different cell types, and defines a distribution over connections between different cell types. As is common in Bayesian deep learning, we also assume zero covariance between the cell type connection distributions, meaning that the variations in connection strength between different cell types are assumed to be independent of each other. So, instead of defining a prior of the whole weight matrix, the stochastic genomic bottleneck defines distributions over connections between different cell types, whose joint probability can be expressed as the following.

$$p^{Normal}(\mathbf{W}) = \prod_{j,k} \mathcal{N}(\mathbf{W}_{jk}|0, \sigma^2)$$

Where $\mathbf{W}_{jk}$ represents the weight between j^{th} and k^{th} cell type within the network. Furthermore, instead of using a normal distribution, we follow Blundell et al. (2015) and use a Scale Mixture Prior (SMP). The SMP is characterized by two Gaussian components with zero mean but two distinct variances σ_1^2 and σ_2^2, and a mixing coefficient π. The SMP is formally defined as follows:

$$p^{SMP}(\mathbf{W}) = \prod_{j,k} \left[\pi \mathcal{N}(\mathbf{W}_{jk}|0, \sigma^2_{1,jk}) + (1-\pi)\mathcal{N}(\mathbf{W}_{jk}|0, \sigma^2_{2,jk})\right]. \tag{2}$$

The variances σ_1 and σ_2 allow the model to capture a broad range of weight magnitudes. In our experiments, the initialization of the mean parameters is set to zero, reflecting the assumption of no a priori preferred direction in the weight space. For the variance parameters, σ_1^2 is initialized to capture the larger scale behavior of weights, whereas σ_2^2 is set several orders of magnitude smaller to encourage sparsity (Blundell et al., 2015). The mixing coefficient π is typically initialized to favor the sparser Gaussian component (i.e closer to zero), thus initializing the network in a state that promotes regularization.

To obtain the weight matrix used for a forward pass of a given SGB layer, multiple samples are taken from each distribution $P(\mathbf{W}_{jk})$, where jk indexes the connection between two cell types. Then, these sampled connections are assigned to entries within a weight matrix that represent a scalar connection between two cell types in two successive layers (as shown in figure 1). Connections between two cell types are represented multiple times within a given weight matrix as there are multiple instantiations of the same cell type in each SGB layer—hence the compressive nature of SGB and the need for multiple samples of $p(\mathbf{W}_{jk})$. Gradient are backpropagated, again with reparametrization trick, through the stochastic sampling operations used to obtain weights between two given cell types.

4.3.1 Supervised Learning Loss Function

To optimize the SGB to perform image classification, we employ a standard cross entropy loss for the log-likelihood term in equation 1, (see e.g. Murphy (2012, p. 57)), and then compute the KL divergence between the SMP prior and its learned values for the first term in equation 1, resulting in the following loss function.

$$\mathcal{L}^{SL,SGB}(\theta) = \left[-\sum_{i=1}^{C} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) \right] + d_{KL}(q(\mathbf{W}|\theta) \| p^{SMP}(\mathbf{W})) \tag{3}$$

where C is the number of classes in the classification problem, $\mathbf{y}$ is a one-hot encoded vector describing the ground truth class and $\hat{\mathbf{y}}$ is the predicted probabilities for each class. For each gradient update, we aggregate gradients over multiple samples of the weight distribution whose loses are computed across a minibatch of multiple input-output pairs.

4.3.2 Deterministic Genomic Bottleneck

The genomic bottleneck framework, as proposed by Koulakov and colleagues (Koulakov et al., 2021), introduced a method to train ANN's that have good innate performance. It is inspired by the biological constraint that the genome cannot explicitly encode the vast number of synaptic connections in the brain. Instead, the genome must rely on a compact set of instructions or developmental rules to construct the neural connectome (Zador, 2019). They formulated this problem in terms of lossy compression of a weight matrix of a traditional ANN. The genomic bottleneck approach introduced two distinct networks: the phenotype network (p-net) and a genomic network (g-net). The p-net represents the functional ANN as it would be after traditional training, embodying the learned tasks through its weights and connections. The g-net is tasked with compressing this p-net, which aims to encode p-network's connectivity and functionality. The authors used an iterative algorithm to learn a g-net that can generate a p-net that has good innate performance. First they train the p-net using standard gradient descent based approach up to some level of performance. Subsequently, the trained weight matrix of the p-net is used to train the weights of the g-net. This process is repeated until the g-net converges to a stable solution i.e it can effectively generate p-nets that have good innate performance and are compressible. This is the training process we use in the experimental sections when reporting DGB performance.

4.3.3 Reinforcement Learning Loss Function

For our reinforcement learning experiments, we use the PPO loss function within the SGB framework. PPO is an actor-critic architecture consisting of two networks (one actor that selects actions and one critic that judges the value of states). The PPO loss function aims to select actions that maximise value. However, PPO is constrained in the size of the updates it can make in order to stabilise training. There is also an entropy term, which encourages exploration during policy learning. PPO works by collecting trajectories from parallel actors and then updating its networks weights with epochs of training on the collected data. The standard PPO objective from Schulman et al. (2017) is as follows:

$$\mathcal{L}^{PPO}(\phi) = \hat{\mathbb{E}}_t \left[-\min(r_t(\phi)\hat{A}_t, \text{clip}(r_t(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_t + c_1(V_\phi(s_t) - V_t^{\text{targ}})^2 - c_2 H(\pi_\phi(s_t)) \right]. \tag{4}$$

Where we use ϕ to represent the trainable weight parameters of a standard deterministic actor-critic network (to avoid confusion with the trainable parameters of a SGB network which we label as θ), $r_t(\phi)$ is the ratio of the new policy to the old policy, ϵ is a clipping parameter that limits the size of gradient updates to improve stability, $\hat{A}_t$ is the estimated advantage—which computes the difference between the experienced (bootstrapped) value of the current trajectory compared to its predicted value, V_ϕ is a value function that predicts the value of a given state, V_t^{targ} is the bootstrapped target for the value function, H computes the entropy of the network policy π_ϕ and c_1 and c_2 are constants that control relative contribution of the different terms to the loss function. Note, we use PPO in environments with continuous action spaces, meaning we use the standard approach of predict parameters of an action distribution, which is then sampled to get the action used by the agent.

In order to adapt PPO to the SGB framework, we use a SGB actor and critic parametrised by θ and we let the traditional PPO loss function represent the log-likelihood term in equation 1 and then compute the KL divergence between the agent's weight priors and its sampled weight values (for both the actor and the critic), resulting in the following loss function.

$$\mathcal{L}(\theta)^{RL,SGB} = \mathcal{L}^{PPO}(\theta) + c_3 d_{KL}\left[(q(\mathbf{W}|\theta)\|p^{SMP}(\mathbf{W})\right]$$

Where c_3 is a further hyperparameter controlling the relative weighting of the PPO loss to the KL divergence.

4.4 Supervised Learning Hyperparameters

4.4.1 MNIST

For MNIST experiments, we used 3-layer feedforward networks with 800 hidden units. To obtain SGB's performance on MNIST across different compression ratios, we fixed the number of cell types in the input and output layer to be 30 and 10 respectively, and varied the number of cell types in the hidden layer. A list of hyperparameters used to train the SGB for the feedforward networks on the MNIST task has been summarized in Table 1.

4.4.2 CIFAR

For the CIFAR experiments, we used a 5-layer CNN based architecture consisting of two initial convolutional layers followed by three fully connected layers. To ensure consistency in experimentation we only compressed the fully connected layers in the network. To obtain SGB's performance on CIFAR across different compression ratios, we fixed the number of cell types in the first fully connected layer and output layer to be 100 and 10 respectively, and varied the number of cell types in the hidden layer. A list of hyperparameters used to train the SGB on the CIFAR task has been summarized in Table 1.

Hyperparameter	MNIST	CIFAR
Learning Rate	0.00005	0.001
Batch Size	64	128
Number of Epochs	50	200
Optimizer	Adam	Adam

Table 1: Hyperparameters for SGB for CIFAR and MNIST Experiments

4.5 RL Experiment Hyperparameters

For our Brax experiments we use the Brax implementation of PPO (Freeman et al., 2021), while for the robotic control experiments we use the cleanRL PPO implementation (Huang et al., 2022). Below we briefly list hyperparameters we found to be important for performance, the full set of hyperparameters are in the supplementary code.

Hyperparameter	Ant	Halfcheetah
Learning Rate	0.0003	0.0003
PPO clip coefficient	0.001	0.3
Number of parallel envs	2048	2048
Entropy coefficient	0.01	0.001

Table 2: Hyperparameters for Brax experiments.

Hyperparameter	ANYmal
Learning Rate	0.00085
PPO clip coefficient	0.001
Number of parallel envs	4096
Entropy coefficient	0.0

Table 3: Hyperparameters for ANYmal experiments.

A Proofs

A.1 Notation

Let us denote the loss function as $\mathcal{L}$ and the loss value at parameters w as $\mathcal{L}(w)$. Let us denote the parameters of the Bayesian network as (μ, σ) where μ and σ are vectors indicating the mean and variance respectively of the distribution of each parameter of the network. In a vanilla setting (no compression), each element of (μ, σ) corresponds to the distribution for a different parameter in the network. Let the network have P parameters. So, $\mu, \sigma \in \mathbb{R}^P$. We will also use $\nabla\mathcal{L}(w)$ and $\nabla^2\mathcal{L}(w)$ to denote the gradient and Hessian respectively of the loss function $\mathcal{L}$ at parameter values w. Given these notations, the parameters of a sampled network can be written as $w = (\mu + \sigma \odot \epsilon)$, where ϵ is a vector with elements drawn from a standard Gaussian distribution.

A.2 Key Theoretical Results

In this section, we provide an analytical framework to elucidate the advantages of stochasticity during development, with particular focus on explaining the empirical results in subsection 2.4. First, we show that the difference between the mean performance of sampled networks and the performance of the mean network depends on the variance in the SGB network parameters and the eigenvalues of the loss function Hessian. Next, we study the implicit bias of gradient descent in modulating this difference between the performances. Finally, we posit that the SGB network parameters would converge to saddle points in the loss landscape during optimization, thereby leading to a better mean performance of sampled networks, compared to the performance of the mean network.

Let us denote the network parameter space as $\mathbb{R}^p$, i.e. a p-dimensional space, the task-relevant loss function as $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$, the gradient of the loss function as $\nabla\mathcal{L} : \mathbb{R}^p \to \mathbb{R}^p$ and the Hessian as $\nabla^2\mathcal{L} : \mathbb{R}^p \to \mathbb{R}^{p\times p}$. The parameters of a sampled network are denoted as $\theta \in \mathbb{R}^p$, while the parameters for the mean and variance of the Bayesian network are denoted as $\mu \in \mathbb{R}^p$ and $\sigma \in \mathbb{R}^p$, respectively. Under this notation, we can write the parameters of a sampled network, $\theta = \mu + \sigma \odot \epsilon$, where $\epsilon \in \mathbb{R}^p$ is a vector with each element drawn i.i.d. from a unit Gaussian. Note that this is the simplest form of Bayesian neural network parameterization, i.e. without any compression. This framework can be easily extended to networks with a compression ratio more than 1 by enabling parameter sharing such that $\mu, \sigma \in \mathbb{R}^{\tilde{p}}$, where $\tilde{p} < p$.

Theorem A.1. *The difference between the mean loss of sampled networks,* $\mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right]$ *and the loss of the mean network (i.e. network with parameters,* $\theta = \mu$*),* $\mathcal{L}(\mu)$ *is as follows.*

$$\Delta\mathcal{L}(\mu, \sigma) := \mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right] - \mathcal{L}(\mu) = \frac{1}{2}\sum_i \lambda_i \mathbb{E}_\epsilon\left[\rho_i^2\right] \tag{5}$$

where (λ_i, ν_i)*'s denote the eigenvalue and eigenvector pairs of* $\nabla^2\mathcal{L}$ *and* $\rho_i = (\sigma \odot \epsilon)^T \nu_i$*.*

A direct corollary of theorem A.1 is that for convex loss landscapes, i.e. when $\lambda_i > 0\ \forall i \in \{1 \ldots p\}$, $\mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right] > \mathcal{L}(\mu)$. This would mean that the mean network would actually perform better than the average sampled network. But, given our empirical observations, we posit that the optimization loss landscape is non-convex for the sorts of tasks that animals must solve. Next, we consider the implicit bias of gradient descent during optimization of the Bayesian neural network parameters, μ and σ, within the SGB framework.

Theorem A.2. *Let* $\hat{\mathcal{L}}$ *denote the variational inference loss (computed using the reparameterization trick) for training the Bayesian neural network parameters* μ, σ*. The parameter updates following a gradient descent step on* $\hat{\mathcal{L}}$ *are as follows.*

$$\Delta\mu = -\eta\mathbb{E}_\epsilon\left[\nabla_\mu\hat{\mathcal{L}}(\mu + \sigma \odot \epsilon)\right] = -\eta\nabla\mathcal{L}(\mu) - \eta\hat{\lambda}\mu \tag{6}$$

$$\Delta\sigma = -\eta\mathbb{E}_\epsilon\left[\nabla_\sigma\hat{\mathcal{L}}(\mu + \sigma \odot \epsilon)\right] = -\eta\nabla^2\mathcal{L}(\mu)\sigma \tag{7}$$

where η is the learning rate and $\hat{\lambda}$ is the regularization weight for the KL divergence term in the variational inference (ELBO) loss, representing the weight for the prior.

Note that theorem A.2 implies that change in μ is akin to doing gradient descent on a single network with parameters μ. Thus, we can extrapolate the behavior of gradient descent (and its stochastic versions) in high-dimensional non-convex loss landscapes to the SGB setting and hypothesize that the mean parameters will converge to saddle points in the loss landscape (Chen et al., 2024). Another direct consequence of theorem A.2 is that the modulation of σ is controlled by the eigenvalues of the Hessian of the $\mathcal{L}$. Specifically, the projection of σ along the directions corresponding to positive (negative) eigenvalues decreases (increases). Consequently, the ρ_i^2 values corresponding to positive (negative) eigenvalues decrease (increase). Combining this result with theorem A.1, we can infer that $\Delta\mathcal{L}(\mu, \sigma)$ decreases after a step of gradient descent. Over the course of optimization, the parameters of SGB will converge to regions in the (non-convex) loss landscape such that $\Delta\mathcal{L}(\mu, \sigma) < 0$, i.e. the mean loss of the sampled networks is less than the loss of the mean network. Assuming a monotonic relationship between the in-distribution performance performance and training loss (whereby higher performance corresponds to lower loss), we have the desired result. Thus, our empirical observation that the average sampled network outperforms the mean network is analytically grounded, as long as the loss is non-convex.

A.3 Effect of sampling on the Loss

Theorem A.3. *The difference between the mean loss of sampled networks, denoted as $\mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right]$ and the loss of the mean network (i.e. network with parameters, $\theta = \mu$), denoted as $\mathcal{L}(\mu)$ is as follows.*

$$\Delta\mathcal{L}(\mu, \sigma) := \mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right] - \mathcal{L}(\mu) = \frac{1}{2}\sum_i \lambda_i \mathbb{E}_\epsilon\left[\rho_i^2\right] \tag{8}$$

where (λ_i, ν_i)'s denote the eigenvalue and eigenvector pairs of $\nabla^2\mathcal{L}$ and $\rho_i = (\sigma \odot \epsilon)^T \nu_i$.

Proof. We will start by using Taylor series expansion of the loss function. Let us write the loss function for a sampled network with parameters $w = (\mu + \sigma \odot \epsilon)$ using a Taylor Series expansion around $w = \mu$:

$$\mathcal{L}(w) = \mathcal{L}(\mu + \sigma \odot \epsilon) = \mathcal{L}(\mu) + \nabla\mathcal{L}(\mu)^T(\sigma \odot \epsilon) + \frac{1}{2}(\sigma \odot \epsilon)^T \nabla^2\mathcal{L}(\mu)(\sigma \odot \epsilon) + \mathcal{O}(\|\sigma \odot \epsilon\|^3) \tag{9}$$

Assuming that 3rd (and higher) order terms are significantly smaller, we can drop the terms in red. Next, taking an expectation over different ϵ:

$$\mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right] = \mathcal{L}(\mu) + \nabla\mathcal{L}(\mu)^T(\mathbb{E}_\epsilon\left[\sigma \odot \epsilon\right]) + \frac{1}{2}\mathbb{E}_\epsilon\left[(\sigma \odot \epsilon)^T \nabla^2\mathcal{L}(\mu)(\sigma \odot \epsilon)\right] \tag{10}$$

Note that $\mathbb{E}_\epsilon\left[\epsilon\right] = 0$ and consequently $\mathbb{E}_\epsilon\left[\sigma \odot \epsilon\right] = 0$. Therefore, the first order term disappears. To simplify the second order term, we can assume (without loss of generality) that $\nabla^2\mathcal{L}(\mu)$ has eigenpairs $\{(\lambda_1, \nu_1), (\lambda_2, \nu_2) \dots (\lambda_P, \nu_P)\}$ such that $\{\nu_1, \nu_2 \dots \nu_P\}$ are a set of orthonormal basis vectors in the P-dimensional parameter space. Note that all (λ, ν) pairs are functions of μ, but we have dropped the explicit notation for sake of clarity. We can use this orthonormal basis set to write $(\sigma \odot \epsilon)$ as a linear combination of ν_i's. Specifically,

$$\sigma \odot \epsilon = \sum_i \rho_i \nu_i \quad , \text{where} \quad \rho_i = (\sigma \odot \epsilon)^T \nu_i \tag{11}$$

It is worth noting that: $\mathbb{E}_\epsilon[\rho_i] = \mathbb{E}_\epsilon\left[(\sigma \odot \epsilon)\right]^T \nu_i = 0$ We can now use this linear combination to simplify the quadratic term:

$$\begin{aligned}
(\sigma \odot \epsilon)^T \nabla^2 \mathcal{L}(\mu)(\sigma \odot \epsilon) &= \left(\sum_i \rho_i \nu_i\right)^T \nabla^2 \mathcal{L}(\mu) \left(\sum_j \rho_j \nu_j\right) \\
&= \left(\sum_i \rho_i \nu_i\right)^T \left(\sum_j \rho_j \nabla^2 \mathcal{L}(\mu) \nu_j\right) \\
&= \left(\sum_i \rho_i \nu_i\right)^T \left(\sum_j \rho_j \lambda_j \nu_j\right) \\
&= \sum_{i,j} \rho_i \rho_j \lambda_j \nu_i^T \nu_j = \sum_{i,j} \rho_i \rho_j \lambda_j \delta_{ij} \qquad (12) \\
&= \sum_i \lambda_i \rho_i^2
\end{aligned}$$

Above, in eq. (12), we have used the fact that $\nu_i^T \nu_j = \delta_{ij}$, where δ_{ij} indicates the Dirac-delta function, i.e. it is 1 when $i = j$ and 0 otherwise. This fact is a direct consequence of (ν_i, ν_j) being part of the orthonormal basis set.

Plugging these results about the first and second order terms back in eq. (10):

$$\mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right] = \mathcal{L}(\mu) + \frac{1}{2}\sum_i \lambda_i \mathbb{E}_\epsilon\left[\rho_i^2\right]$$

$$\implies \mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma \odot \epsilon)\right] - \mathcal{L}(\mu) = \frac{1}{2}\sum_i \lambda_i \mathbb{E}_\epsilon\left[\rho_i^2\right]$$

□

Key takeaways: The above equation describes the difference between the mean loss of the population and the loss of the mean of the population. Note that $\mathbb{E}_\epsilon\left[\rho_i^2\right] \geq 0$.

1. In a strictly convex setting, i.e. $\lambda_i \geq 0 \; \forall i$, the right hand side is strictly non-negative. Thus, the mean loss (performance) of the population will be more (less) than the loss (performance) of the population mean network.

2. In a non-convex setting, i.e. $\lambda_i < 0$ for some i, it is possible to have the right hand side of the equation to be negative. Thus, it is possible to have a certain variance structure in the network parameter space (σ) such that the mean loss (performance) of the population will be less (more) than the loss (performance) of the population mean network.

A.4 Effect of gradient descent on Bayesian network parameters

A.4.1 Background: variational inference with Bayesian neural networks

Notation: Let us introduce a few more notations to denote the dataset as $\mathcal{D}$ which consists of a set of inputs and corresponding labels: $\{(x_1, y_1), (x_2, y_2) \ldots (x_i, y_i) \ldots (x_n, y_n)\}$. We present a supervised learning problem here, but this notation can be extended to unsupervised, self-supervised as well as reinforcement learning setups. The task is to learn network parameters w such that the network's predictions, denoted by $\hat{y}(x_i) = f(x_i, w)$ are close to y_i. We can formulate this objective as a reducing the negative log likelihood of the observed y_i to be drawn from the distribution $p(\hat{y}|x_i, w)$ which depends on the network's parameters.

Variational Inference perspective: We can use a variational inference perspective to solve the above task, wherein the problem is to learn a distribution of network parameters, denoted by $q_{(\mu,\sigma)}(w)$, which best matches the posterior distribution of the network parameters given the observed data, $p(w|\mathcal{D})$. For the Bayesian network perspective, we can assume that $q_{(\mu,\sigma)}(w)$ has the following Gaussian form:

$$q_{(\mu,\sigma)}(w) = \mathcal{N}(w|\mu, diag(\sigma)) = -\frac{1}{2}(w-\mu)^T(diag(\sigma))^{-2}(w-\mu)$$

where $diag(\sigma)$ denotes a diagonal matrix with elements of σ in the diagonal.

The true objective is to reduce the KL-divergence between $q_{(\mu,\sigma)}(w)$ and $p(w|\mathcal{D})$.

$$\begin{aligned} d_{KL}\left[q_{(\mu,\sigma)}(w)\|p(w|\mathcal{D})\right] &= \mathbb{E}_{w\sim q_{(\mu,\sigma)}}\left[log(q_{(\mu,\sigma)}(w)) - log(p(w|\mathcal{D})\right] \\ &= \mathbb{E}_{w\sim q_{(\mu,\sigma)}}\left[log(q_{(\mu,\sigma)}(w)) - log(p(w,\mathcal{D}))\right] + log(p(\mathcal{D})) \\ &= \mathbb{E}_{w\sim q_{(\mu,\sigma)}}\left[log(q_{(\mu,\sigma)}(w)) - log(p(w)) - log(p(\mathcal{D}|w))\right] + log(p(\mathcal{D})) \end{aligned}$$

Here, $p(w)$ is the prior over network parameters. We can also assume this to be have a Gaussian form with mean 0:

$$\begin{aligned} p(w) &= \mathcal{N}\left(w|0, \frac{1}{\sqrt{\hat{\lambda}}}\mathbb{I}\right) \\ \implies log(p(w)) &= -\frac{1}{2}\hat{\lambda}w^T w \end{aligned}$$

where $\hat{\lambda}$ can be thought of as the weight decay hyperparameter, or how strongly we want to rely on the chosen prior of a mean 0 distribution.

Based on these expressions, we can define the following loss:

$$\mathcal{L}_{VI}(w) = \mathbb{E}_{w\sim q_{(\mu,\sigma)}}\left[log(q_{(\mu,\sigma)}(w)) + \frac{1}{2}\hat{\lambda}w^T w - log(p(\mathcal{D}|w))\right] \tag{13}$$

Minimizing this loss will lead to minimizing the overall objective of reducing the KL-divergence. However, evaluating and minimizing this loss requires us to sample from the current estimate of the posterior distribution and then backpropagate gradients through the distribution parameters, which if often intractable.

Reparameterization trick: To make the problem tractable, we will use the reparameterization trick, wherein we will write $w = \mu + \sigma \odot \epsilon$, wherein elements of ϵ are drawn from a standard Gaussian. In doing so, the $log(q_{(\mu,\sigma)}(w))$ becomes a constant. Note that this is exactly the Bayesian neural network framework, as described above! Now, we can write a tractable version of eq. (13):

$$\begin{aligned} \hat{\mathcal{L}}(\mu,\sigma) &= \mathbb{E}_\epsilon\left[-log(p(\mathcal{D}|w = \mu + \sigma\odot\epsilon)) + \frac{1}{2}\hat{\lambda}(\mu + \sigma\odot\epsilon)^T(\mu + \sigma\odot\epsilon)\right] \\ &= \mathbb{E}_\epsilon\left[-log(p(\mathcal{D}|w = \mu + \sigma\odot\epsilon)) + \frac{1}{2}\hat{\lambda}\left(\mu^T\mu + 2\mu^T\sigma\odot\epsilon + (\sigma\odot\epsilon)^T(\sigma\odot\epsilon)\right)\right] \end{aligned}$$

Note that the first term is the negative log likelihood of the data, given the current network parameters. Therefore, we can represent it as the task loss, $\mathcal{L}(w = \mu + \sigma \odot \epsilon)$. Thus,

$$\hat{\mathcal{L}}(\mu,\sigma) = \mathbb{E}_\epsilon\left[\mathcal{L}(\mu + \sigma\odot\epsilon) + \frac{1}{2}\hat{\lambda}\left(\mu^T\mu + 2\mu^T\sigma\odot\epsilon + (\sigma\odot\epsilon)^T(\sigma\odot\epsilon)\right)\right] \tag{14}$$

A.4.2 Impact of gradient descent

We can now introduce the theorem from the main text and present its proof.

Theorem A.4. *Let $\hat{\mathcal{L}}$ denote the variational inference loss, computed with reparameterization trick, that is used to train the Bayesian neural network parameters μ, σ. The updates to the parameters following a gradient descent step on $\hat{\mathcal{L}}$ are as follows.*

$$\Delta\mu = -\eta\mathbb{E}_\epsilon\left[\nabla_\mu\hat{\mathcal{L}}(\mu+\sigma\odot\epsilon)\right] = -\eta\nabla\mathcal{L}(\mu) - \eta\hat{\lambda}\mu \tag{15}$$

$$\Delta\sigma = -\eta\mathbb{E}_\epsilon\left[\nabla_\sigma\hat{\mathcal{L}}(\mu+\sigma\odot\epsilon)\right] = -\eta\nabla^2\mathcal{L}(\mu)\sigma \tag{16}$$

where η is the learning rate and $\hat{\lambda}$ is the weight for the KL divergence term in the variational inference (ELBO) loss, i.e. the weight for the prior.

Proof. Let us write the gradients with respect to the mean and variance parameters of our distribution. Before we dive in to deriving the expressions for the parameter updates, we present some results that are consequences on the Chain rule and will be useful.

$$\nabla_\mu\mathcal{L}(\mu+\sigma\odot\epsilon) = \nabla_\mu(\mu+\sigma\odot\epsilon)\nabla_{(\mu+\sigma\odot\epsilon)}\mathcal{L}(\mu+\sigma\odot\epsilon) = \nabla\mathcal{L}(\mu+\sigma\odot\epsilon) \tag{17}$$

$$\nabla_\sigma\mathcal{L}(\mu+\sigma\odot\epsilon) = \nabla_\sigma(\mu+\sigma\odot\epsilon)\nabla_{(\mu+\sigma\odot\epsilon)}\mathcal{L}(\mu+\sigma\odot\epsilon) = \nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\odot\epsilon \tag{18}$$

Focusing on the expression for mean, μ:

$$\begin{aligned}
\nabla_\mu\hat{\mathcal{L}}(\mu,\sigma) &= \mathbb{E}_\epsilon\left[\nabla_\mu\mathcal{L}(\mu+\sigma\odot\epsilon) + \hat{\lambda}\,(\mu+\sigma\odot\epsilon)\right] \\
&= \mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\right] + \hat{\lambda}\mu + \hat{\lambda}\sigma\odot\mathbb{E}_\epsilon\left[\epsilon\right] \\
&= \mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\right] + \hat{\lambda}\mu \\
\implies \Delta\mu = -\eta\nabla_\mu\hat{\mathcal{L}}(\mu,\sigma) &= -\eta\mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\right] - \eta\hat{\lambda}\mu
\end{aligned}$$

where η is the learning rate. Again, we can use the Taylor Series expansion for $\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)$ around $w=\mu$:

$$\nabla\mathcal{L}(\mu+\sigma\odot\epsilon) = \nabla\mathcal{L}(\mu) + \nabla^2\mathcal{L}(\mu)^T(\sigma\odot\epsilon) + \mathcal{O}(\|\epsilon\|^2) \tag{19}$$

$$\begin{aligned}
\implies \Delta\mu &= -\eta\nabla\mathcal{L}(\mu) - \eta\nabla^2\mathcal{L}(\mu)^T\mathbb{E}_\epsilon[\sigma\odot\epsilon] - \mathbb{E}_\epsilon[\mathcal{O}(\eta\|\epsilon\|^2)] - \eta\hat{\lambda}\mu \\
&= -\eta\nabla\mathcal{L}(\mu) - \eta\hat{\lambda}\mu - \mathbb{E}_\epsilon[\mathcal{O}(\eta\|\epsilon\|^2)]
\end{aligned}$$

Ignoring the third (and higher) order terms of η and $\|\epsilon\|$,

$$\Delta\mu = -\eta\nabla\mathcal{L}(\mu) - \eta\hat{\lambda}\mu \tag{20}$$

Now, we can repeat the same process for variance, σ.

$$\begin{aligned}
\nabla_\sigma\hat{\mathcal{L}}(\mu,\sigma) &= \mathbb{E}_\epsilon\left[\nabla_\sigma\mathcal{L}(\mu+\sigma\odot\epsilon) + \hat{\lambda}\,(\mu\odot\epsilon+\sigma\odot\epsilon)\right] \\
&= \mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\odot\epsilon + \hat{\lambda}\,(\mu\odot\epsilon+\sigma\odot\epsilon)\right] \\
&= \mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\odot\epsilon\right] + \hat{\lambda}\,(\mu+\sigma)\odot\mathbb{E}_\epsilon\left[\epsilon\right] \\
&= \mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\odot\epsilon\right] \\
\implies \Delta\sigma = -\eta\nabla_\sigma\hat{\mathcal{L}}(\mu,\sigma) &= -\eta\mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)\odot\epsilon\right]
\end{aligned}$$

Using the Taylor Series expansion for $\nabla\mathcal{L}(\mu+\sigma\odot\epsilon)$ around $w=\mu$ from eq. (19):

$$\begin{aligned}
\Delta\sigma &= -\eta\mathbb{E}_\epsilon\left[\left(\nabla\mathcal{L}(\mu) + \nabla^2\mathcal{L}(\mu)^T(\sigma\odot\epsilon) + \mathcal{O}(\|\epsilon\|^2)\right)\odot\epsilon\right] \\
&= -\eta\mathbb{E}_\epsilon\left[\nabla\mathcal{L}(\mu)\odot\epsilon + \nabla^2\mathcal{L}(\mu)^T(\sigma\odot\epsilon\odot\epsilon) + \mathcal{O}(\|\epsilon\|^3)\right] \\
&= -\eta\nabla\mathcal{L}(\mu)\odot\mathbb{E}_\epsilon\left[\epsilon\right] - \eta\nabla^2\mathcal{L}(\mu)^T(\sigma\odot\mathbb{E}_\epsilon\left[\epsilon\odot\epsilon\right]) \\
\implies \Delta\sigma &= -\eta\nabla^2\mathcal{L}(\mu)^T\sigma
\end{aligned} \tag{21}$$

Without loss of generality, we can assume that the Hessian is symmetric (the order of derivatives can be interchanged). So,

$$\Delta\sigma = -\eta \nabla^2 \mathcal{L}(\mu)\sigma \tag{22}$$

□

A.5 Intuitive explanation of the consequences of above Theorems

Let us write σ using the basis set of $(\nu_1, \nu_2 \dots \nu_P)$, i.e. the eigenvectors of $\nabla^2\mathcal{L}(\mu)$, to further simplify the expression of $\Delta\sigma$ in theorem A.4.

$$\sigma = \sum_i z_i \nu_i \quad \text{, where} \quad z_i = \sigma^T \nu_i \tag{23}$$

Note that ν_i's are dependent on μ and can be considered to be constant while updating σ. So,

$$\begin{aligned}
\Delta\sigma = \sum_i \Delta z_i \nu_i = -\eta\nabla^2\mathcal{L}(\mu)\sigma = -\eta\nabla^2\mathcal{L}(\mu)\sum_i z_i\nu_i = -\eta\sum_i z_i\lambda_i\nu_i \\
\implies \sum_i \Delta z_i \nu_i = -\eta \sum_i z_i \lambda_i \nu_i \\
\implies \Delta z_i = -\eta\lambda_i z_i \quad \forall i \in \{1, 2 \dots P\}
\end{aligned} \tag{24}$$

This equation indicates that directions corresponding to $\lambda_i > 0$ would correspond to $\frac{\Delta z_i}{z_i} < 0$ and consequently lead to pushing z_i closer to 0, i.e. shrink variance in those directions. The contrary will be true in directions corresponding to $\lambda_i < 0$, i.e. variance will expand in those directions. Thus, gradient descent leads to a *reduction* of the term in theorem A.3 that quantifies the difference between the mean loss of the population and the loss of the population mean.

Key takeaways:

1. The changes in the mean are like doing gradient descent on a neural network with the parameters equal to the mean.
2. The changes in the variance are such that the variance along strictly convex directions are reduced and variance along strictly concave directions are increased.

Taken together, we have presented theoretical justification that leverages the effect of gradient descent and sampling in Bayesian neural networks to explain why the mean performance of the population can be better than the performance of the population mean network.

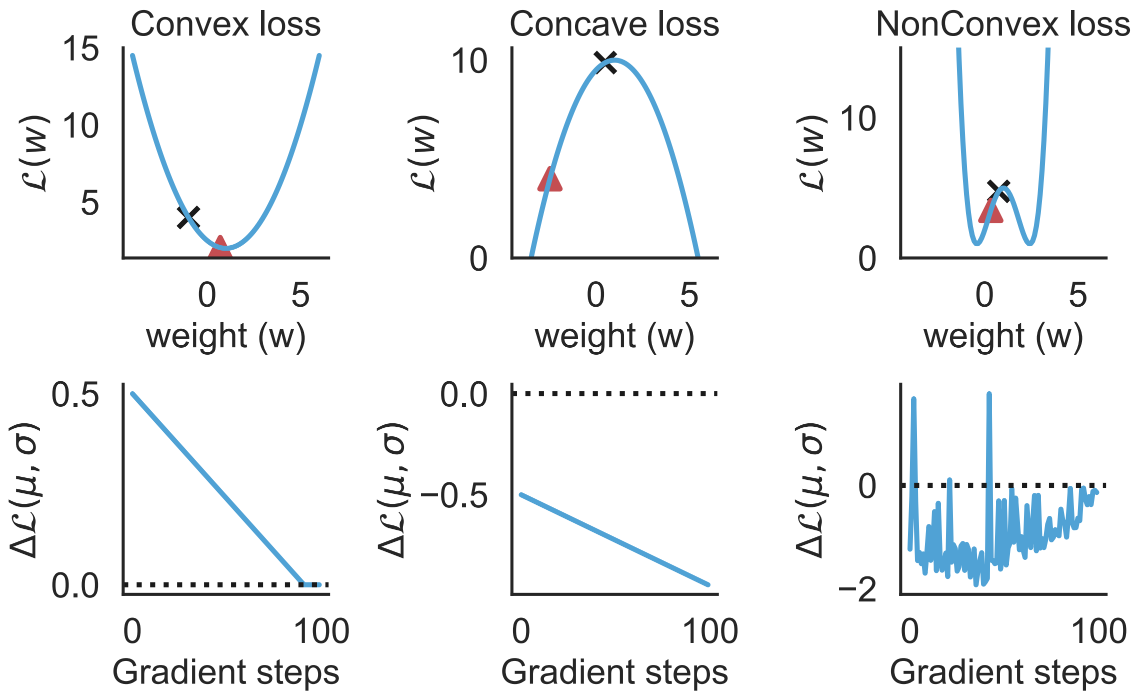

Figure 6: Empirical verification of subsection A.5 in toy 1D settings with Convex (left), Concave (middle) and NonConvex (right) loss functions. The black cross and the red triangle indicate the initial and final (after 100 gradient steps) parameter values of the mean network, μ. Note that $\Delta\mathcal{L}(\mu, \sigma)$ is always ≥ 0 for the Convex loss landscape and ≤ 0 for the Concave loss landscape. For the NonConvex loss landscape, $\Delta\mathcal{L}(\mu, \sigma)$ depends on the location of μ in the parameter space.

B Real Environment Interfaces for RL tasks

In this section, we provide a detailed view of the real environment interfaces used for the reinforcement learning (RL) tasks discussed in the main text. Specifically, we focused on the motor control tasks involving the Ant, HalfCheetah, and Anymal environments, which are part of the Brax framework. The Ant and HalfCheetah tasks require the agent to learn efficient locomotion strategies, balancing speed and stability, while navigating diverse terrains. The Anymal environment, a simulated quadruped robot, introduces additional complexity with varied terrains, making it an excellent testbed for assessing the ability of RL models to generalize across different physical conditions.

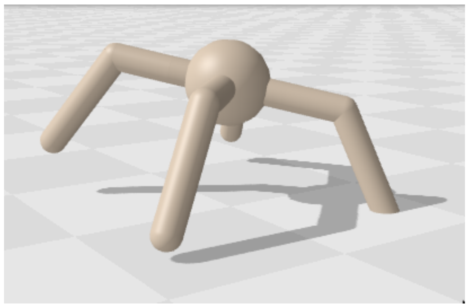

Figure 7: Ant environment in the Brax simulator.

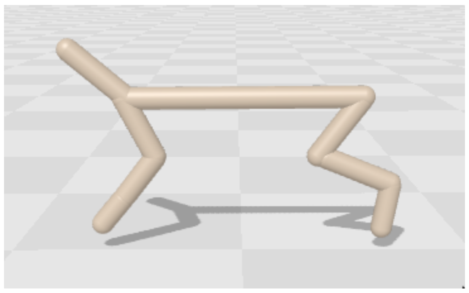

Figure 8: Cheetah environment in the Brax simulator.

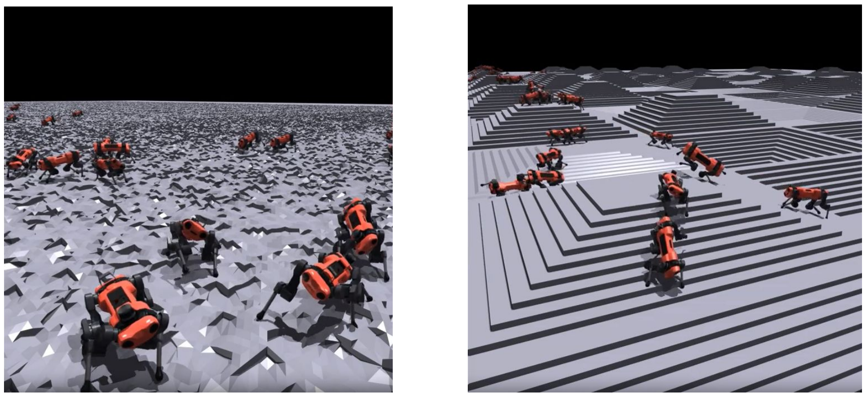

Figure 9: Multiple instances of the Anymal robot in the Brax simulator. The left image shows robots on rough terrain, while the right image shows them on a stair terrain. Different agents represent separate simulations with no interaction between agents.

References

Mark V Albert, Adam Schnabel, and David J Field. Innate visual learning through spontaneous activity patterns. *PLoS Computational Biology*, 4(8):e1000137, 2008.

Mehdi Azabou, Michael Mendelson, Nauman Ahad, Maks Sorokin, Shantanu Thakoor, Carolina Urzay, and Eva Dyer. Relax, it doesn't matter how you get there: A new self-supervised approach for multi-timescale behavior analysis. *Advances in Neural Information Processing Systems*, 36, 2023. Adapted from Figure 3.

Sara Ballouz, Risa Karakida Kawaguchi, Maria T Pena, Stephan Fischer, Megan Crow, Leon French, Frank M Knight, Linda B Adams, and Jesse Gillis. The transcriptional legacy of developmental stochasticity. *Nature Communications*, 14(1):7226, 2023.

Dániel L Barabási, Taliesin Beynon, Ádám Katona, and Nicolas Perez-Nieves. Complex computation from developmental priors. *Nature Communications*, 14(1):2226, 2023.

Dániel L Barabási, Gregor FP Schuhknecht, and Florian Engert. Functional neuronal circuits emerge in the absence of developmental activity. *Nature Communications*, 15(1):364, 2024.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Feng Chen, Daniel Kunin, Atsushi Yamamura, and Surya Ganguli. Stochastic collapse: How gradient noise attracts sgd dynamics towards simpler subnetworks. *Advances in Neural Information Processing Systems*, 36, 2024.

Claudia Clopath, Lars Büsing, Eleni Vasilaki, and Wulfram Gerstner. Connectivity reflects coding: a model of voltage-based stdp with homeostasis. *Nature neuroscience*, 13(3):344–352, 2010.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553): 452–459, May 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14541. URL https://www.nature.com/articles/nature14541.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Geoffrey J Goodhill. Contributions of theoretical modeling to the understanding of neural map development. *Neuron*, 56(2):301–311, 2007.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 38–44. IEEE, 2016.

Akira Imamura, Yoshiro Morimoto, Shinji Ono, Naohiro Kurotaki, Shinji Kanegae, Naoki Yamamoto, Hirohisa Kinoshita, Takahiro Tsujita, Yuji Okazaki, and Hiroki Ozawa. Genetic and environmental factors of schizophrenia and autism spectrum disorder: insights from twin studies. *Journal of Neural Transmission*, 127:1501–1515, 2020.

Johan Ludwig William Valdemar Jensen. On convex functions and inequalities between mean values. *Acta mathematica*, 30(1):175–193, 1906.

Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1711.03113*, 2017.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision?, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alexei Koulakov, Sergey Shuvaev, Divyansha Lachi, and Anthony Zador. Encoding innate ability through a genomic bottleneck. *BiorXiv*, pp. 2021–03, 2021.

Alexei A Koulakov and Dmitry N Tsigankov. A stochastic model for retinocollicular map development. *BMC neuroscience*, 5:1–17, 2004.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, 7(1):1–14, 2017.

Gerit Arne Linneweber, Maheva Andriatsilavo, Suchetana Bias Dutta, Mercedes Bengochea, Liz Hellbruegge, Guangda Liu, Radoslaw K Ejsmont, Andrew D Straw, Mathias Wernet, Peter Robin Hiesinger, and Bassem A Hassan. A neurodevelopmental origin of behavioral individuality in the drosophila visual system. *Science*, 367(6482):1112–1119, 2020.

Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pp. 2218–2227, 2017.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Kenneth D Miller and David JC MacKay. The role of constraints in hebbian learning. *Neural computation*, 6(1):100–126, 1994.

Kenneth D Miller, Joseph B Keller, and Michael P Stryker. Ocular dominance column development: analysis and simulation. *Science*, 245(4918):605–615, 1989.

Kevin J Mitchell. *Innate*. Princeton University Press Princeton, NJ, 2018.

Kevin J Mitchell. Variability in neural circuit formation. *Cold Spring Harbor Perspectives in Biology*, 16(3):a041504, 2024.

Kevin J Mitchell and Nick Cheney. The genomic code: The genome instantiates a generative model of the organism. *arXiv preprint arXiv:2407.15908*, 2024.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Christoph Stöckl, Dominik Lang, and Wolfgang Maass. Probabilistic skeletons endow brain-like neural networks with innate computing capabilities. *bioRxiv*, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Dmitry Tsigankov and Alex Koulakov. Optimal axonal and dendritic branching strategies during the development of neural circuitry. *Frontiers in neural circuits*, 3:690, 2009.

Alan Mathison Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, Aug 1952. doi: 10.1098/rstb.1952.0012.

Günter Vogt. Stochastic developmental variation, an epigenetic source of phenotypic diversity with far-reaching biological consequences. *Journal of Biosciences*, 40:159–204, 2015.

Jesse N Weber, Brant K Peterson, and Hopi E Hoekstra. Discrete genetic modules are responsible for complex burrow evolution in peromyscus mice. *Nature*, 493(7432):402–405, 2013.

Yi Wei, Dmitry Tsigankov, and Alexei Koulakov. The molecular basis for the development of neural maps. *Annals of the New York Academy of Sciences*, 1305(1):44–60, 2013.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization, 2022.

Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):3770, 2019.