Expressivity of Neural Networks with Random Weights and Learned Biases

Ezekiel Williams * EZEKIEL.WILLIAMS@MILA.QUEBEC
Mila - Québec AI Institute, Université de Montréal

Avery Hee-Woon Ryoo † HEE-WOON.RYOO@MILA.QUEBEC
Mila - Québec AI Institute, Université de Montréal

Thomas Jiralerspong † THOMAS.JIRALERSPONG@MILA.QUEBEC
Mila - Québec AI Institute, Université de Montréal

Alexandre Payeur ALEXANDRE.PAYEUR@MILA.QUEBEC
Mila - Québec AI Institute, Université de Montréal

Matthew G. Perich MATTHEW.PERICH@MILA.QUEBEC
Mila - Québec AI Institute, Université de Montréal

Luca Mazzucato ⋆ LMAZZUCA@UOREGON.EDU
University of Oregon

Guillaume Lajoie ⋆ GUILLAUME.LAJOIE@MILA.QUEBEC
Mila - Québec AI Institute, Université de Montréal

Abstract

Landmark universal function approximation results for neural networks with trained weights and biases provided impetus for their ubiquitous use as learning models in Artificial Intelligence (AI) and neuroscience. Recent work has pushed the bounds of universal approximation by showing that arbitrary functions can similarly be learned merely by tuning smaller subsets of parameters of otherwise random networks, for example the output weights. Motivated by the fact that biases can be interpreted as biologically plausible mechanisms for adjusting unit outputs in neural networks, such as tonic inputs or activation thresholds, we investigate the expressivity of neural networks with random weights where only biases are optimized. We provide theoretical and numerical evidence demonstrating that feedforward neural networks with fixed random weights can be trained to perform multiple tasks by learning biases only. We further show that an equivalent result holds for recurrent neural networks predicting dynamical system trajectories. Our results are relevant to neuroscience, where they demonstrate the potential for behaviourally-relevant changes in dynamics without modifying synaptic weights, as well as for AI, where they illuminate and generalize multi-task methods such as bias fine-tuning and network gating/masking and other non-parametric learning mechanisms.

1. Introduction

The universal approximation theorems [6, 11, 13] of the late 1900s highlighted the *expressivity* of neural network models–their ability to approximate or *express* a broad class of functions through tuning of weights and biases, heralding the central role neural networks play in Machine Learning

* Corresponding author; † denotes equal contribution; ⋆ denotes co-senior authors

(ML) and neuroscience today. Since these foundational studies, a rich literature has explored the limits of the expressivity of neural networks by finding smaller parameter subsets whose tuning still results in the approximation of wide classes of functions or dynamics. Prior work has explored approximation capabilities of feed-forward (FFNs) and Recurrent Neural Networks (RNNs) where only the output weights are trained [8, 9, 17, 20], and deep FFNs where only normalization parameters [2, 7], or binary masks–either over units or parameters–are trained, leading to the lottery ticket hypothesis [15] and gated neural networks [24]. Recently, a study has also explored the approximation abilities of transformers where only context is tuned [18]. Here, motivated by recent insights from neuroscience, we initiate the investigation of expressivity in both FFNs and RNNs where only biases are trained, an avenue that remains largely unexplored.

Learned biases are of fundamental relevance to neuroscience and ML. In the latter domain, recent work highlights the optimization or the careful selection of bias vectors: fine-tuning of biases for multi-task learning [29], time-encoding as a bias in score matching [23], and the context tokens used for in-context learning [25] can all be viewed as methods of carefully setting the bias in a model where other connectivity parameters are fixed in order to perform a task.

In neuroscience, single neurons employ diverse mechanisms to adjust their response to inputs, beyond synaptic plasticity. Importantly, some of these mechanisms can be construed as manipulating the firing onset of neurons which, in a firing rate model, is represented by the bias. Among these mechanisms, we have shunting inhibition [10], threshold adaptation [1], and a host of other mechanisms that participate in shaping the input-output transfer function of neurons (reviewed in [5], see also [16]). Experimental evidence also suggests that bias-related signals play a role in learning [22, 30] but, despite this, most work modelling learning in neuroscience has focused on synapses.

If tuning the biases of a neural network will only span a reduced set of functions, or output dynamics, then this would solidify the role of synaptic plasticity as *the* critical component in biological and artificial learning. Conversely, if one can sufficiently express arbitrary dynamics solely by changing biases, this would motivate deeper investigation of when and how non-synaptic plasticity mechanisms might shoulder some of the effort of learning. In this paper we address the question of the expressivity of bias learning. In a regime where all weight parameters are randomly initialized and frozen, and only hidden layer biases are optimized, we provide theoretical guarantees demonstrating that *(1) feed-forward neural networks with wide hidden layers are universal function approximators with high probability* and *(2) rate-style Recurrent Neural Networks (RNNs) with wide hidden layers can arbitrarily approximate finite-time trajectories from smooth dynamical systems with high probability*.

We provide empirical support for, and a deeper interrogation of, these theoretical findings with an array of numerical experiments.

2. Theory

Our theory builds off classic results on universal function approximation [14]. In the interest of space we state theorems without proof. We begin with a theorem for single-hidden layer feed-forward neural networks. Let ϕ be a ReLU or step-function activation, $\alpha > 0$, and p_α be a uniform distribution on the interval $[-(\alpha+1), \alpha+1]$.

Theorem 1. *Consider a single hidden layer, feed-forward, neural network with each individual weight parameter sampled from p_α. We can find a hidden layer width and bias vector such that,*

with a probability that is arbitrarily close to 1*, the random-weight neural network approximates any continuous function on compact support with any strictly positive degree of accuracy on* $(0, 1)$.

We now turn to RNNs, where we study the problem of approximating the (partially observable) dynamical system $z_{n+1} = F(z_n, x_n),\ \ y_n = Qz_n,\ \ z_0 \in U$ where z_n, x_n, and y_n are real vectors of finite dimension, Q is a readout matrix, and U is compact. It is further assumed that F is continuous and arbitrary except for the constraint that for any input x (in a set of interest), and any $z \in \tilde{U}$, $F(z, x) \in \tilde{U}$, where $\tilde{U} \subset U$ is compact such that for any $z \in \tilde{U}$ we have $z + c_0 \in U$ for $||c_0|| < c$. In words, $\tilde{U}$ is an invariant set of F that is also compact and contained in U with some minimal margin c between the edge of $\tilde{U}$ and that of U.

Theorem 2. *Consider a single hidden layer RNN with activations ϕ, defined above, with each element in the input, recurrent, and output weight matrices sampled from p_α. We can find a hidden layer width, a bias vector, and a hidden-state initial condition for the RNN such that, with a probability that is arbitrarily close to* 1*, the RNN approximates finite trajectories from the above dynamical system with any strictly positive degree of accuracy on* $(0, c)$.

Proof Intuition: both proofs work in two steps. First, the function (dynamical system trajectory) is approximated with a single hidden layer neural network (RNN), $\mathcal{N}$, with weights and biases chosen according to universal approximation theory. Second, we randomly initialize parameters according to p_α in a *very* wide neural network and show that, if the network is wide enough, the Strong Lottery Ticket Hypothesis at the level of nodes holds so that we can find a sub-network, $\hat{\mathcal{N}}$, in the given feed-forward net (RNN) that approximates $\mathcal{N}$. We then tune the biases in $\hat{\mathcal{N}}$ to match the biases of the corresponding nodes in $\mathcal{N}$, and set the biases to be very negative for all nodes in the second network *not* in $\hat{\mathcal{N}}$, thus shutting off their outputs.

Remark: a key element of our proofs is that the random network hidden layers are much larger than those of neural networks which have fully-tuned weights.

3. Numerical Results

3.1. Multi-task Learning with Bias-learning Feed-forward Networks

Before tackling the multi-task setting, we first examined whether a single-hidden-layer, bias-learning FFN could learn to perform digit recognition on the MNIST dataset [4]. Initial weights were sampled from a uniform distribution on $[-0.1, 0.1]$ and were then frozen. In connection with our theory, we investigated the effect of hidden-layer width on performance by training networks with increasing widths and evaluated the validation accuracy. The network did learn the task and validation accuracy increased with layer width, with quickly diminishing returns above 5000 units (Fig. 1A).

We then investigated the flexibility of bias-learning FFNs to learn multiple tasks. We used the same setup as above with 32000 hidden units to train on 7 different tasks: MNIST [4], KMNIST [3], Fashion MNIST [27], Ethiopic-MNIST, Vai-MNIST, and Osmanya-MNIST from Afro-MNIST [26], and Kannada-MNIST [19]. All tasks involved classifying 28×28 grayscale images into 10 classes. The random weights were fixed across tasks while different biases were learned. We compared bias learning against a fully-trained neural network with the same size and architecture (Fig. 1B). We found that the bias-only network achieved similar performance to the fully-trained network on most tasks (only significantly worse on KMNIST). A crucial difference here is that the networks had matching size and architecture, so that the number of trainable parameters in the bias-only network

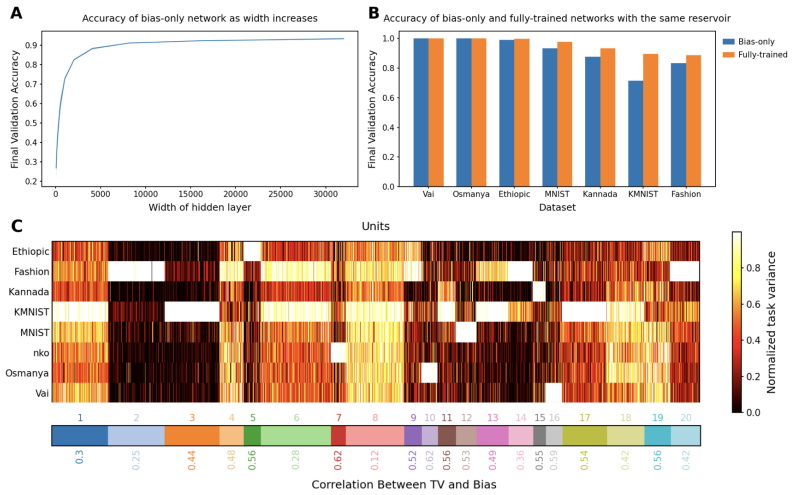

Figure 1: **A.** Validation accuracy on MNIST vs layer width. **B.** Validation accuracy on multiple image classification tasks for bias-only (blue) and fully-trained (orange) networks. In panels A and B, training was on 20 epochs and error bars on 5 runs are omitted because the standard errors are of order 10^{-3}. **C.** Top: K-mean clustering of task-variance (TV) reveals task-specific clusters. Bottom: Pearson correlation between TV and bias vectors (mean across neurons in each cluster). In this and all others figures Adam optimization was used.

(32 000 parameters) was several orders of magnitude smaller than in the fully-trained case (25 440 010 parameters). Notably, a different set of biases was learned for each task. We conclude that bias-only learning in FFNs is a viable avenue to perform multi-tasking with randomly initialized and fixed weights.

We finally investigated the relationship between the bias of a hidden unit and its task variance (TV). We hypothesized that only groups of hidden-layer units would be useful for each task, resulting in a pattern where subsets of units are 'active', with high variance for a given task, while others are quiet. This hypothesis was confirmed experimentally as shown in Fig. 1C. In the appendix, we clustered the biases of all hidden units in the same way as the TVs; the results are shown in Fig. H. We saw qualitatively similar clusters as those for TVs.

3.2. Bias-learning of Dynamical Systems with Recurrent Neural Networks

Finally, we explored the capabilities of an RNN trained on a non-autonomous dynamical system, namely a single dimension of the Lorenz system where the other dimensions are unobserved (Fig. 2). As in the feedforward case, only the biases of the input layer were trained and all of the weights were initialized randomly and frozen. The objective was formulated as follows: given a window of size w from time-points t to $t + w$, predict the output of the system at a future window of $s + \tau$ for $s = t..x.t + w$ observing only the x-coordinate from t to s. This offset value ($\tau = 27$) was chosen to be the half-width at the half-max of the auto-correlation function of the single observed dimension of the Lorenz system, taking into account the numerical integration time step $\Delta t = 0.01$). Each

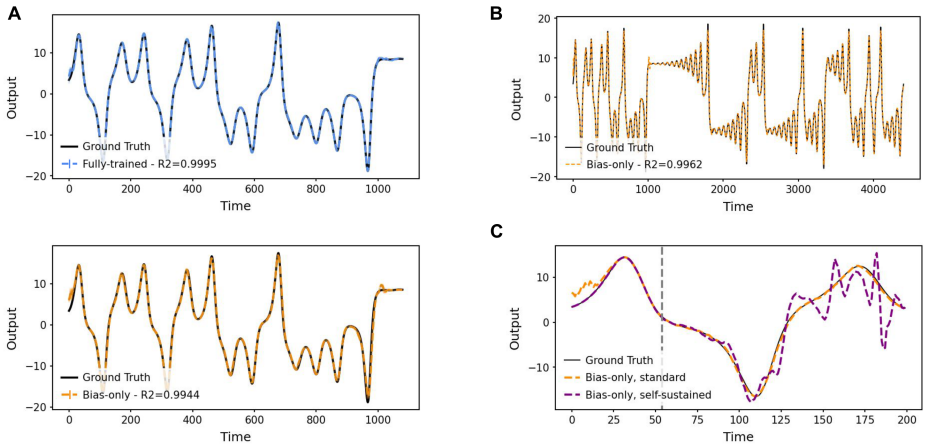

Figure 2: **Learning non-autonomous dynamical systems. A.** The outputs of the fully-trained (top) and bias-only (bottom) networks on a window of a Lorenz system unseen during training ($\dot{x} = \sigma(y - x), \dot{y} = x(\rho - z) - y, \dot{z} = xy - \beta z$). The system was generated using Euler's method with a step size Δt of 0.01 from an initialization at (0,1,0), with $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$. Standard deviation error bars were computed over 5 seeds, but are not visible. **B.** Generalization to sequences longer (4320 time-points) than those trained on. **C.** Output of the bias-only network diverges from the ground truth signal when using its own outputs as context (starting from the grey line).

model had a hidden-layer width of 1024 units and was trained on different windows of 1080 (i.e., 40τ) time-points.

We found that both the fully-trained and bias-only networks accurately predicted future points of the system, evidenced by a consistent R^2 metric of >0.99 (n=5) on a window of the generated Lorenz attractor held out from training (Fig. 2A). Furthermore, the models showed stability when predicting windows that were several times larger than the ones used during training, continuing to reconstruct the system without a notable change in accuracy (Fig. 2B). However, when the networks were fed their own previous predictions as input, their prediction accuracy decreased, demonstrating the devastating effect of small compounding deviations propagated through time (Fig. 2C).

4. Discussion

In this paper, we presented theoretical results demonstrating that feed-forward and recurrent neural networks with fixed random weights but learnable biases can approximate arbitrary functions. We showcased the expressitivity of bias-learned networks in numerical experiments where we perform auto-regressive modelling, multi-task learning, dynamic pattern generation, and dynamical system forecasting. Finally, we showed how functional specialization emerges from bias learning and elucidated signatures and advantages of bias-only learning, notably in multi-task learning.

While we showcase the expressivity of learning biases in random networks, bias learning would likely be much more efficient if the connectivity weights were not random, but rather we pre-trained or meta-learned for a corpus of tasks. Indeed, brain networks–where analogous input-driven learning

might takes place–are not random, but have instead evolved or developed to be optimally useful in a variety of conditions. Future work into bias learning with non-random networks might yield important advances for meta-learning and amortized learning techniques where context-dependent biases can quickly "reconfigure" generalist networks for specific tasks. Bias learning may also advance the field of In Context Learning (ICL) in modern Transformer architectures, where attention heads that operate over sequences of tokens (i.e. the input context or latent tokens) provide attention scores to a MLP that outputs to subsequent layers. When learning in-context, the input tokens provide essentially a set of constant inputs to an MLP which are equivalent to our learned biases. A better understanding of the functions produced by this procedure through bias learning could help advance our understanding of ICL (see also [18]).

References

[1] Rony Azouz and Charles M Gray. Dynamic spike threshold reveals a mechanism for synaptic coincidence detection in cortical neurons in vivo. *Proceedings of the National Academy of Sciences*, 97(14):8110–8115, 2000.

[2] Rebekka Burkholz. Batch normalization is sufficient for universal function approximation in cnns. In *The Twelfth International Conference on Learning Representations*, 2023.

[3] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature, 2018.

[4] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[5] Katie A Ferguson and Jessica A Cardin. Mechanisms underlying gain modulation in the cortex. *Nature Reviews Neuroscience*, 21(2):80–92, 2020.

[6] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[7] Angeliki Giannou, Shashank Rajput, and Dimitris Papailiopoulos. The expressive power of tuning only the normalization layers. *arXiv preprint arXiv:2302.07937*, 2023.

[8] Lukas Gonon, Lyudmila Grigoryeva, and Juan-Pablo Ortega. Approximation bounds for random neural networks and reservoir systems. *The Annals of Applied Probability*, 33(1):28–69, 2023.

[9] Allen G Hart, James L Hook, and Jonathan HP Dawes. Echo state networks trained by tikhonov least squares are l2 (μ) approximators of ergodic dynamical systems. *Physica D: Nonlinear Phenomena*, 421:132882, 2021.

[10] Gary R Holt and Christof Koch. Shunting inhibition does not have a divisive effect on firing rates. *Neural computation*, 9(5):1001–1013, 1997.

[11] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[12] Kurt Hornik. Some new results on neural network approximation. *Neural networks*, 6(8): 1069–1072, 1993.

[13] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[14] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[15] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

[16] Luca Mazzucato, Giancarlo La Camera, and Alfredo Fontanini. Expectation-induced modulation of metastable activity underlies faster coding of sensory stimuli. *Nature neuroscience*, 22 (5):787–796, 2019.

[17] Ariel Neufeld and Philipp Schmocker. Universal approximation property of random neural networks. *arXiv preprint arXiv:2312.08410*, 2023.

[18] Aleksandar Petrov, Philip HS Torr, and Adel Bibi. Prompting a pretrained transformer can be a universal approximator. *arXiv preprint arXiv:2402.14753*, 2024.

[19] Vinay Uday Prabhu. Kannada-mnist: A new handwritten digits dataset for the kannada language. *arXiv preprint arXiv:1908.01242*, 2019.

[20] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21, 2008.

[21] Anton Maximilian Schäfer and Hans Georg Zimmermann. Recurrent neural networks are universal approximators. In *Artificial Neural Networks–ICANN 2006: 16th International Conference, Athens, Greece, September 10-14, 2006. Proceedings, Part I 16*, pages 632–640. Springer, 2006.

[22] Megha Sehgal, Chenghui Song, Vanessa L Ehlers, and James R Moyer Jr. Learning to learn–intrinsic plasticity as a metaplasticity mechanism for memory formation. *Neurobiology of learning and memory*, 105:186–199, 2013.

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[24] Joel Veness, Tor Lattimore, David Budden, Avishkar Bhoopchand, Christopher Mattern, Agnieszka Grabska-Barwinska, Eren Sezener, Jianan Wang, Peter Toth, Simon Schmitt, et al. Gated linear networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10015–10023, 2021.

[25] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

[26] Daniel J Wu, Andrew C Yang, and Vinay U Prabhu. Afro-mnist: Synthetic generation of mnist-style datasets for low-resource languages, 2020.

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[28] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[29] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

[30] Wei Zhang and David J Linden. The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, 4(11):885–900, 2003.

Throughout the appendix the proofs are restated for ease of reference. We will always take $||\cdot||$ to be the 1-norm, unless otherwise stated.

Appendix A. Parameter-Bounding Activations

A suitable activation ϕ is referred to as a γ-*parameter bounding activation* if it allows for universal approximation even when each individual parameter, e.g. an element of a weight matrix or bias vector, is bounded. This is a property that we use in the main proofs. We show that this property holds for ReLU and Heaviside activations, below.

Proposition 1. *The ReLU and the Heaviside step function are γ-parameter bounding activations for any $\gamma > 0$.*

Proof We prove this solely for the ReLU, as the logic for the Heaviside is effectively the same. Let ϕ thus be a ReLU. First, observe the following useful property: for all $\alpha > 0$ we have $\alpha\phi(x) = \phi(\alpha x)$. From this, consider the neural network of hidden layer width n with ReLU activations, $y_n(\theta)$, and observe:

$$y_n(\theta) = \alpha^2 \sum_{i=1}^{n} \frac{A_{:i}}{\alpha} \phi\left(\frac{B_{i:}}{\alpha} x + \frac{b_i}{\alpha}\right) = \alpha^2 y_n\left(\frac{\theta}{\alpha}\right). \tag{A.1}$$

Moreover, if $\alpha \in \mathbb{N}$ we have

$$y_n(\theta) = \sum_{i=1}^{\alpha^2 n} \tilde{A}_{:i} \phi\big(\tilde{B}_{i:} x + \tilde{b}\big) = y_{\alpha^2 n}(\tilde{\theta}), \tag{A.2}$$

where $\tilde{\theta} = [\frac{\theta_1}{\alpha}, \ldots, \frac{\theta_n}{\alpha}, \ldots \frac{\theta_1}{\alpha}, \ldots, \frac{\theta_n}{\alpha}]$ so that each element is simply a re-scaled and repeated version of the original parameters (we have α^2 repeats).

Now, given an arbitrary compact set $U \in \mathbb{R}^d$, continuous function $h : \mathbb{R}^d \to \mathbb{R}^l$, and $\varepsilon > 0$, by the universal approximator theory (see e.g. [12, 14]) we can find n such that

$$\int_U ||h(x) - y_n(x, \theta)|| \leq \epsilon \tag{A.3}$$

holds. Because n is finite we can bound every individual (scalar) parameter by M, for some sufficiently large M. Suppose we want the parameters to be bounded instead by γ with $M > \gamma > 0$. If we select $\alpha \in \mathbb{N}$ s.t. $\alpha > \frac{M}{\gamma}$ then we can find $y_{\alpha^2 n}(x, \tilde{\theta})$ such that $y_{\alpha^2 n}(x, \tilde{\theta}) = y_n(x, \theta)$. Thus we have found a parameter-bounding ReLU neural network satisfying Eq.A.3, completing the proof.

Remark 1: The intuition behind this result, for the ReLU, is credited to this stack exchange answer.

Appendix B. γ-Bias-Learning Activations

If ϕ is a γ-parameter bounding activation *and* if $\exists \tau \in \mathbb{R}$ such that for $x < \tau$ $\phi(x) = 0$ then we say that ϕ is a γ-*bias-learning activation*.

Appendix C. Random Neural Network Formulation

The proofs of this section revolve around masked, random, neural networks:

$$\tilde{r}_m^{\mathcal{M}} = -\alpha r_0 + \bar{\mathcal{M}} \odot \beta\phi(Wr_0 + Bx + b), \quad \tilde{y}_m^{\mathcal{M}} = A\tilde{r}_m^{\mathcal{M}}, \tag{C.1}$$

where $0 \leq \alpha < \infty$, $0 < \beta < \infty$, $m \in \mathbb{N}$, r_0, $\tilde{r}_m^{\mathcal{M}} \in \mathbb{R}^m$, $x \in \mathbb{R}^d$, $\tilde{y}_m^{\mathcal{M}} \in \mathbb{R}^l$, $\mathcal{M} \in \{0,1\}^m$, and all other matrices and vectors have real elements with the dimensions required by the above definitions. We assume that ϕ is γ-parameter bounding and that each individual (scalar) parameter, be it weight or bias, is sampled randomly–before masking–from a uniform distribution on $[-(\gamma+1), \gamma+1]$. In this way the parameters are random variables with compact support. If $\mathcal{M} = \mathbf{1}$ then we drop the superscript. To account for feed-forward neural networks we simply assume that W is the zero matrix.

W.l.o.g. assume there are n non-zero elements in $\mathcal{M}$. We construct $W^{\mathcal{M}} \in \mathbb{R}^{n\times n}$–the recurrent matrix restricted to participating (non-masked) hidden units–by beginning with W and deleting the i^{th} row and i^{th} column of the matrix if $\mathcal{M}_i = 0$. We construct $B^{\mathcal{M}} \in \mathbb{R}^{n\times d}$, $A^{\mathcal{M}} \in \mathbb{R}^{l\times n}$, and $b^{\mathcal{M}} \in \mathbb{R}^n$ by deleting the i^{th} row of B, A, and i^{th} element of b if $\mathcal{M}_i = 0$.

Consider the case where the i^{th} element of r_0 is 0 whenever $\mathcal{M}_i = 0$. Then, regardless of whether Eq.C.1 represents a feed-forward network or the transition function for an RNN, the masked units will always be zero. We can thus simply track the n units that correspond with 1's in $\mathcal{M}$ as the outputs, $y^{\mathcal{M}}$ will depend solely on these. We observe that the behaviour of these units can be described by the following network:

$$r_m^{\mathcal{M}} = -\alpha r_0 + \beta\phi(W^{\mathcal{M}} r_0 + B^{\mathcal{M}} x + b^{\mathcal{M}}), \quad y_m^{\mathcal{M}} = A^{\mathcal{M}} r_m^{\mathcal{M}}. \tag{C.2}$$

It is networks of the form of Eq.C.2 that will be the primary subject of study in what follows. Note that the '$\sim$', over the r, is dropped to denote the fact that r is a different vector on account of dropping the zero units. In the feed-forward case we use subscripts, as we have done above, to denote hidden layer width. Whenever we discuss RNNs or dynamical systems we will instead use the subscript to denote time.

Lastly, we define the class of dynamical systems that we will approximate by learning only biases:

$$z_{t+1} = F(z_t, x_t), \quad y_t = Qz_t, \quad z_0 \in U_z, \tag{C.3}$$

where t and x_t are as defined for the RNN, $F : U_z \times U_x \to \mathbb{R}^s$ is continuous, and $Q \in \mathbb{R}^{l\times s}$. Because we build from the classic universal approximation results, we must be working with functions operating on compact sets. To guarantee that this will be the case we must make several more assumptions about the dynamical system. First, $U_z \subset \mathbb{R}^s$ is assumed to be a compact invariant set of the dynamical system: if the system is in U_z it remains there for all t and for all inputs, x_t in U_x. Second, we will assume that the dynamical system is well-defined on a slightly larger compact set, $\tilde{U}_z = \{z_0 + c_0 : z_0 \in U_z,\ ||c_0|| < c\}$ for some $c > 0$, containing $U_z \times U_x$.

Appendix D. Proofs for the Feedforward and Recurrent Networks

For function approximation we study a single-layer feedforward network whose output is given by

$$y_n(x,\theta) = \sum_{i=1}^{n} A_{:i}\phi(B_{i:}x + b), \tag{D.1}$$

with $A \in \mathbb{R}^{l\times n}$, $B \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$, and $\theta = \{B, A, b\}$. For the dynamical system approximation we study a vanilla recurrent network architecture in discrete time, given by

$$r_t = -\alpha r_{t-1} + \beta\phi(Wr_{t-1} + Bx_{t-1} + b), \quad y_t = Cr_t, \tag{D.2}$$

where $r_t \in \mathbb{R}^m$ for all $0 \le t \le T$ for some $T \in \mathbb{N}$, α and β control the time scale of the dynamics, $W \in \mathbb{R}^{m\times m}$, $C \in \mathbb{R}^{l\times m}$, and B and b are as in the previous section. The parameters are now $\theta = \{W, B, A, b\}$ and the input is a function of time: $\mathbf{x} = \{x_t\}_{t\ge1}$, s.t. $x_t \in \mathbb{R}^d$. Throughout, we will assume $x_t \in U_x \subset \mathbb{R}^d$, with U_x being compact for all t. When $\alpha = 0$, $\beta = 1$ one gets the standard vanilla RNN formulation; alternatively α and β can be set to approximate continuous time dynamics using Euler's method.

Lemma 1. *Let $h : U \to \mathbb{R}^l$ be a continuous function on compact support $U \subset \mathbb{R}^d$. Then for any ϵ, δ both in $(0,1)$, we can find a layer width $m \in \mathbb{N}$ such that with probability at least $1-\delta$ $\exists\mathcal{M} \in \{0,1\}^m$ satisfying the following:*

$$\int_U ||h(x) - y_m^{\mathcal{M}}(x)||dx \le \epsilon. \tag{D.3}$$

Proof

First, we find a neural network with parameters that approximate the desired function h. Given the assumptions on ϕ, we can use Proposition 2 to find n and parameters $\theta^* = \{A^*, B^*, b^*\}$ such that

$$\int_U ||h(x) - y_n(x,\theta^*)||dx \le \frac{\epsilon}{2}, \tag{D.4}$$

because U is compact and h is continuous.

We make a brief comment about the domain of a given activation function in y_n. ϕ will be operating on a compact domain $\{ux + b : u \in [-\gamma,\gamma]^d, x \in U, b \in [-\gamma,\gamma]\}$, as a consequence of the compactness of the support of the parameters, and of the assumed compactness of U. By its continuity, ϕ is Lipschitz and bounded on this domain. We label Lipschitz constant and bound K_ϕ and M_ϕ respectively. We further define M_x to be the bound for x on U, and $|U|$ to be the value $\int_U dx$, which is finite by the boundedness of the domain.

Next, we construct a masked random network that approximates y_n with high probability. By Lemma 3, we can find a random feed-forward neural network of hidden layer width m such that a mask, $\mathcal{M}$, exists satisfying $|\theta_i^* - \theta_i^{\mathcal{M}}| < \varepsilon$ for some arbitrarily ε on $(0,1)$. In particular, we can choose ε as:

$$|\theta_i^* - \theta_i^{\mathcal{M}}| < \varepsilon = \frac{\epsilon}{2|U|\max\left(nl(K_\phi\gamma[1+M_x]+M_\phi), 1\right)} \tag{D.5}$$

for all i with probability at least $1-\delta$. If we are in the regime of probability $1-\delta$ where the mask satisfying the above error bound exists then we get

$$||y_m^{\mathcal{M}}(x) - y_n(x)|| \leq nl[K_\phi \gamma(1 + M_x) + M_\phi]\varepsilon \leq \frac{\epsilon}{2|U|}, \tag{D.6}$$

where, in addition to Eq.D.5, we use the assumptions on ϕ and U stated before the start of the proof and repeated applications of the triangle inequality.

Integrating the error over the domain and using Eq.D.4 and Eq.D.6 gives

$$\int_U ||h(x) - y_m^{\mathcal{M}}(x)||dx \leq \int_U ||h(x) - y_n(x)||dx + \int_U ||y_m^{\mathcal{M}}(x) - y_n(x)||dx \leq \epsilon, \tag{D.7}$$

with probability $1 - \delta$.

Theorem 3. *Assume that ϕ is γ-bias learning and, for compact $U \subset \mathbb{R}^d$, $h : U \to \mathbb{R}^l$ is continuous. Then for any degree of accuracy $\epsilon \in (0, 1)$, there exists $m \in \mathbb{N}$ and $b \in \mathbb{R}^m$ such that a neural network of the form shown in Eq.D.1 with hidden layer width m, each individual weight sampled from p_R, and bias vector b approximates h with error less than ϵ.*

Proof Observe that, once we have choosen an m satisfying the desiderata of Lemma 1, because ϕ is assumed to be γ-bias-learning, m is some finite value and all variables that make up the input of ϕ are bounded, we can implement the mask by setting b_i to be very negative for every i such that $\mathcal{M}_i = 0$. For every b_i such that $\mathcal{M} = 1$ we simply leave b_i at its original randomly chosen value.

Corollary 1. *Assume $d = l$, that is, the output and input spaces are the same. Then the results of Lemma 1 and Theorem 3 also hold for res-nets; that is, networks whose output is of the form $x + y_m^{\mathcal{M}}(x)$.*

Proof This follows by observing that $h(x) + x$ is also a continuous function and then replacing $h(x)$ with $h(x) + x$ in Eq.D.3 and rearranging.

Lemma 2. *Consider a discrete time, partially observed dynamical system of the form of, and satisfying the same conditions as, the one in Eq.C.3. Let $0 < T < \infty$, initial condition $z_0 \in U_z$, input $\mathbf{x} \in U_x^T$, $\epsilon \in (0, c)$ and $\delta \in (0, 1)$. Then we can find an RNN initial condition r_0 and a layer width $m \in \mathbb{N}$ such that with probability at least $1 - \delta$ $\exists \mathcal{M} \in \{0, 1\}^m$ satisfying the following:*

$$\sum_{t=1}^{T} ||y_t - y_t^{\mathcal{M}}|| < \epsilon. \tag{D.8}$$

Proof

It is well known that we can arbitrarily approximate this dynamical system with an RNN ; we provide a simple proof of this in Proposition 3. In particular, for arbitrary $\epsilon \in (0, 1)$ we can find an RNN of the form in Eq.D.2, with hidden layer width $n \in \mathbb{N}$ and output $\hat{y}$, satisfying:

$$\sum_{t=1}^{T} ||\hat{y}_t - y_t|| < \frac{\epsilon}{2}, \tag{D.9}$$

for any initial condition of the dynamical system selected within the invariant set U_z. We note that this is a point-wise convergence. It can be shown (see Prop.3) that the hidden states of this

RNN–not including its initial conditions–remain on a compact set when approximating finite time trajectories of the original dynamical system with initial conditions in U_z. We will name the compact set containing RNN hidden states U_r. Given this compactness, we can then show that the following set

$$\tilde{U} = \{ur + vx + b : u \in [-\gamma, \gamma]^n, v \in [-\gamma, \gamma]^d, r \in U_r, x \in U, b \in [-\gamma, \gamma]\}, \tag{D.10}$$

is itself compact, by the compactness of the sets from which it is formed. By its own compactness and the assumptions on ϕ we observe that ϕ has Lipschitz constant K_ϕ. By the compactness of U and U_r, we can bound x and r on these sets, to get bounds R_x and R_r respectively.

Let the parameters of the above-defined approximating RNN be given by $\theta^* = \{A^*, W^*, B^*, b^*\}$. Then by Lemma 3 we can find a random RNN of hidden width m and with parameters θ such that a mask, $\mathcal{M}$, exists satisfying

$$|\theta_i^* - \theta_i^{\mathcal{M}}| < \varepsilon = \frac{\epsilon}{2nT \max\left(Rn\beta K_\phi \tilde{R} \sum_{t=0}^{T-1} (\alpha + R\beta K_\phi n)^t + R_r, 1\right)}, \tag{D.11}$$

for all i with probability at least $1 - \delta$, where $\tilde{R} = R_x + R_r + 1$. One can show quite simply using induction that if $|\theta_i^* - \theta_i^{\mathcal{M}}| < \varepsilon$ for all i we get

$$\sum_{t=1}^{T} ||y_t^{\mathcal{M}} - \hat{y}_t|| < nT\big[Rn\beta K_\phi \tilde{R} \sum_{t=0}^{T-1} (\alpha + R\beta K_\phi n)^t + R_r\big]\varepsilon. \tag{D.12}$$

The triangle inequality on $||y_t^{\mathcal{M}} - \hat{y}_t + \hat{y}_t - y_t||$, along with Equations D.9, D.11, and D.12 completes the proof.

Theorem 4. *Consider the RNN in Eq.D.2 with ϕ a γ-bias learning activation, and input, output, and recurrent weight parameters for each hidden unit sampled from p_R. We can find a hidden layer width, a bias vector, and a hidden-state initial condition for the RNN such that, with a probability that is arbitrarily close to 1, the RNN approximates finite trajectories from the dynamical system defined in Eq.C.3 to any positive error on the interval $(0, c)$, where c is defined above.*

Proof This follows directly from Theorem 2, by observing that one can replace the mask by simply setting biases to some sufficiently low value.

Appendix E. Supplementary Lemmas

The following result is well known in the literature; see e.g. Proposition 1 of [14].

Proposition 2. *For any $\epsilon > 0$ $\exists n \in \mathbb{N}$ s.t.*

$$\int_U ||h(x) - y_n(x, \theta^*)|| dx \leq \epsilon \tag{E.1}$$

Corollary 2. *The above result holds if we restrict the output weight matrix of the neural network to have rank equal to the output dimension.*

Proof This is because the set of full rank matrices is dense in $\mathbb{R}^{m\times n}$ for $m, n \in \mathbb{N}$.

Consider matrices $W^* \in \mathbb{R}^{n\times n}$, $B^* \in \mathbb{R}^{n\times d}$, $A^* \in \mathbb{R}^{l\times n}$, and vector $b^* \in \mathbb{R}^n$. We can vectorize and concatenate their elements into the single long vector $\theta \in \mathbb{R}^\pi$, where $\pi = n(n+d+l+1)$. Assume that $|\theta_i^*| < \gamma$ for all i.

Next, construct $W \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{m\times d}$, $A \in \mathbb{R}^{l\times m}$, and vector $b \in \mathbb{R}^m$, by sampling each element randomly from a uniform distribution on $[-(\gamma+1), (\gamma+1)]$. We analogously group these into a single vector, $\theta \in \mathbb{R}^{m(m+d+l+1)}$ Observe that for each $\mathcal{M} \in \{0,1\}^m$, we can construct sub-matrices of W, B, A, and sub-vector of b by deleting column and row pairs in W, rows in B, columns in A, and elements of b whose indices correspond to $i \in \{1,\dots,m\}$ such that $\mathcal{M}_i = 0$. For a given $\mathcal{M}$, we define $\theta^{\mathcal{M}}$ to be the vector constructed by flattening and concatenating these sub-matrices and vector. We then have the following lemma:

Lemma 3. *For θ^* defined above (and otherwise arbitrary), and arbitrary ϵ, δ both on $(0,1)$, we can find $m > n$ such that with probability at least $1-\delta$ $\exists \mathcal{M} \in \{0,1\}^m$ with only n non-zero elements such that $|\theta_i^* - \theta_i^{\mathcal{M}}| < \epsilon$ for all $i \in \{1,\dots,n\}$. In particular, any $m \geq \frac{n\log\delta}{\log[1-(\frac{\epsilon}{\gamma+1})^\pi]}$ will satisfy the result.*

Proof We will refer to the event that the desiderata of the lemma are satisfied as A_1, that is: $\exists \mathcal{M} \in \{0,1\}^m$ with only n non-zero elements such that $|\theta_i^* - \theta_i^{\mathcal{M}}| < \epsilon$ for all $i \in \{1,\dots,n\}$. The event that the desiderata are not satisfied is A_1^c.

Assume that $m^\star = kn$ for some $k \in \mathbb{N}^+$. Consider the *'block' mask* $\mathcal{M}^{k_1}$ s.t. $\mathcal{M}_i^{k_1} = 1$ only for $i \in \{(k_1-1)n+1,\dots,k_1 n\}$, with $0 < k_1 \leq k$. Note that the n elements selected by these block masks are non-overlapping for two different k_1. Let event A_2 be the event that there is a block mask that occurs satisfying the desiderata of the lemma. Clearly $A_2 \subset A_1 \implies A_1^c \subset A_2^c \implies P(A_1^c) \leq P(A_2^c)$. A_2^c is the probability that there is no block mask satisfying the desiderata of the lemma. Observe that

$$P(A_2^c) = P\left[\bigcap_{k_1=1}^{k} \{k_1^{th} \text{ block mask doesn't work}\}\right] = \prod_{k_1=1}^{k} P(\{k_1^{th} \text{ block mask doesn't work}\})$$
$$= \prod_{k_1=1}^{k} 1 - P(\{k_1^{th} \text{ block mask works}\}) = \left[1 - \left(\frac{\epsilon}{\gamma+1}\right)^\pi\right]^{\frac{m^\star}{n}}, \tag{E.2}$$

which follows from the fact that the elements of the matrices are independently sampled and the elements corresponding to sub-matrices selected by a given block mask are independent of those associated with another block mask. By making $m^\star$ *very* large we can make $P(A_2^c)$ arbitrarily small. Because $P(A_1^c) \leq P(A_2^c)$–and the desiderata of the lemma are not satisfied solely on A_1^c–the result follows by selecting $m^\star = m$ such that $P(A_2^c) \leq \delta$, and taking $\mathcal{M}$ to be a block mask which, from the above, we can do with probability greater than or equal to $1-\delta$.

Remark: We note that, in Eq.E.2, n will likely also depend implicitly on γ. If γ is very small then we will need to stack many ReLUs on top of each other to approximate the desired function, leading to a larger number of units. Conversely, if γ is very large we will need to sample a large number of units to before we get a unit appropriately close to the correct parameter value. This suggests the existence of some sweet spot in the value γ, which we leave for future work to explore.

For the following proposition we consider the discrete time dynamical system that we wish to approximate to be as in Eq.C.3.

Proposition 3. *For any initial condition $z_0 \in U_z$, input $\mathbf{x} \in U_x^T$, $\epsilon \in (0, c)$ and any $\infty > \alpha, \beta > 0$ we can find an RNN of the style of Eq. D.2 of hidden width $n \in \mathbb{N}$ and an initial value r_0 for the RNN such that:*

$$\sum_{t=1}^{T} ||\hat{y}_t - y_t|| < \epsilon \tag{E.3}$$

where $\hat{y}_t$ is the output of the RNN and y_t is that of the dynamical system.

This result is well known, see e.g. [21]. For completeness, we provide one such proof that works from the case of universal approximation with feed-forward neural networks.

Proof We want to approximate the dynamical system:

$$z_{t+1} = F(z_t, x_t), \quad y_t = Cz_t, \quad z_0 \in U_z, \tag{E.4}$$

defined on set $\tilde{U}_z = \{z_0 + c_0 : z_0 \in U_z,\ ||c_0|| < c\}$, where U_z is an invariant set (see §**??**).

We define the set:

$$U_{zx} = \{[z + c_0 \ \ x] : z \in U_z, x \in U, ||c_0|| < c\}. \tag{E.5}$$

Importantly, this set is compact given the compactness assumptions on U and U_z. Also note that, since F is assumed continuous, it will be K_F-Lipschitz on this compact set for some constant K_F. We can thus use the corollary to Proposition 2 to find a neural network of hidden dimension $n \in \mathbb{N}$ that approximates F with a maximum-rank output matrix, A. We write this neural network:

$$\hat{z} = -\alpha z + \beta A \phi(Wz + Bx + b) = \hat{F}(z, x), \tag{E.6}$$

assuming $z \in U_z$ and $x \in U$, with $A \in \mathbb{R}^{s \times n}$, $W \in \mathbb{R}^{n \times s}$, $B \in \mathbb{R}^{n \times d}$, and $b \in \mathbb{R}^n$. In particular, we can find arbitrary ϵ with $0 < \epsilon < c$ such that:

$$||\hat{F}(z, x) - F(z, x)|| < \varepsilon = \frac{\epsilon}{T \max(R_C \sum_{t=0}^{T-1} K_F^t, 1)}, \tag{E.7}$$

where $R_C = ||C||$. Fix $T \geq 1$. To prove that we can approximate the underlying dynamical system, we use induction starting at time $t = 1$. The base case will be

$$||\hat{z}_1 - z_1|| = ||\hat{F}(z_0, x_0) - F(z_0, x_0)|| \leq \varepsilon, \tag{E.8}$$

by our choice of n and initial condition, and that $[z_0, x_0] \in U_{zx}$. Importantly, this implies also that $||\hat{z}_1 - z_1|| < \varepsilon$. Because $\varepsilon < c$ this means that $[\hat{z}_1 \ \ x_1]^\top \in U_{zx}$.

For $t = 1$, $\varepsilon = \sum_{t'=0}^{t-1} K_F^{t'} \varepsilon$. We thus make the induction hypothesis that $||\hat{z}_t - z_t|| < \sum_{t'=0}^{t-1} K_F^{t'} \varepsilon$ and that $[\hat{z}_t \ \ x_t]^\top \in U_{zx}$. If $T = 1$ we are finished. If $T > 1$ we assume $1 < t < T$ and use this hypothesis to prove the induction step:

$$||\hat{z}_{t+1} - z_{t+1}|| \leq ||\hat{F}(\hat{z}_t, x_t) - F(\hat{z}_t, x_t)|| + ||F(\hat{z}_t, x_t) - F(z_t, x_t)|| \tag{E.9}$$

$$\leq \varepsilon + K_F ||\hat{z}_t - z_t|| = \varepsilon \sum_{t'=0}^{t} K_F^{t'} < \frac{c}{T}. \tag{E.10}$$

Because $\frac{c}{T} \leq c$, $[\hat{z}_{t+1} \;\; x_{t+1}]^\top \in U_{zx}$. Then

$$\sum_{t=1}^{T} ||\hat{y}_t - y_t|| \leq R_C T \varepsilon \sum_{t=0}^{T} K_F^t \leq \epsilon. \tag{E.11}$$

While we have approximated the dynamical system it is not yet in the standard rate-style RNN form. However, we can obtain the rate form by changing from tracking $\hat{z}$ to a different dynamical variable: $r_t \in \mathbb{R}^n$. Because A is assumed to be maximum rank, we can find $r_0 \in \mathbb{R}^n$ such that $z_0 = Ar_0$. Take this as the initial value for an RNN with dynamics:

$$r_{t+1} = -\alpha r_t + \phi(WAr_t + Bx_t + b) \tag{E.12}$$

It is easy to see that $\hat{z}_t = Ar_t$ for all $0 \leq t \leq T$. It follows that $y_t = CAr_t \; \forall t$. Thus, this RNN approximates the original partially observed dynamical system.

Lastly, we note that for the choice of r_0 that we have made all r_t that we encounter will be contained within a compact set. This is because all $Ar_t = \hat{z}_t$ (not including r_0). We will define this set to be U_r, for use in Lemma 2.

Appendix F. Task Variance Analysis for FFNs in Section 3.1

We investigated the task-specific functional organization of the hidden units by estimating the single-unit task variance (TV) [28] (same setup as in Fig. 1B), defined as the variance of a hidden unit activation across the test set for each task. The TV conveys the overall amount of stimulus-related information a unit encodes during each task. A unit with high TV in a particular task indicates that its responses vary across stimuli, suggesting that the unit is instrumental for that task performance. A unit with high TV in one task and low TV for all other tasks is specialized to one particular task and does not participate in other tasks. We clustered the hidden units TV using K-means clustering (K chosen by cross-validation) on the vectors of TVs for each unit and found that distinct functional clusters of hidden units emerged (Fig. 1C). Most units reflected strong task specialization, i.e., they were only being used for specific tasks (ex: cluster 3 for KMNIST and cluster 10 for Osmanya). Others were used for many or all tasks (ex: clusters 1 and 8), although a smaller fraction of clusters exhibited shared activation across multiple tasks. Overall, we conclude that multi-task bias-learning leads to the emergence of task-specific functional organization.

Appendix G. Relationship Between Bias Learning and Masked Learning in FFNs

As our theory shows that biases can universally approximate simply by turning units off, we wished to test whether bias learning performs similarly to learning masks, and to what extent solutions learned by these approaches are different from each other. We compared training mask to bias learning on networks with the same random input/output weight matrices. For mask-training, we approximated binary masks with 'soft' sigmoid masks and optimized the sigmoid parameters. The sigmoid was steepened over the course of training to approximate the mask, and at test time the slope was effectively binarized. We compared pairs of masks learned in this fashion with learned biases on single-hidden layer ReLU networks with $10,000$ hidden units. We observed a trend of bias-training slightly improving upon mask-training (Fig.G.1.A), an expected increase given the added flexibility of biases over masks. Further research is needed to determine if this trend is reliable

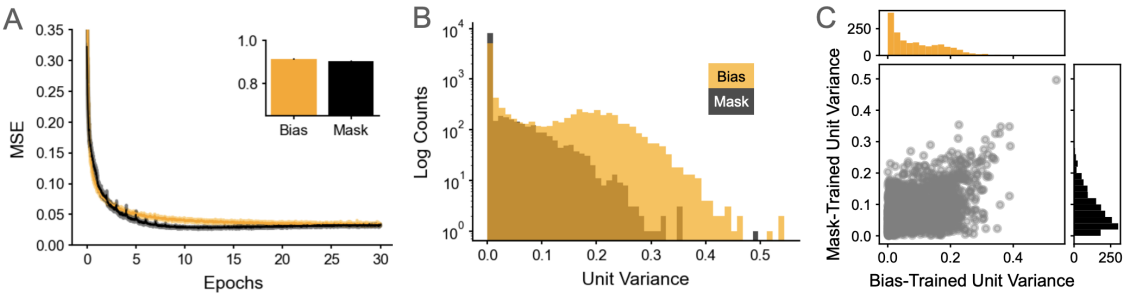

Figure G.1: **Comparing bias and masked learning on Same Weights. A.** Bias-learning achieved ~ 1 percentage points higher accuracy on MNIST over mask-learning (0.915 ± 0.0028SD bias vs. 0.905 ± 0.0028SD mask). **B.** Histograms of hidden unit variances, calculated over $10,000$ MNIST samples, for bias-trained (orange) and mask-trained (black). Histogram counts are log-scaled. **C.** Scatter plot of hidden unit variances from C but taking only units that are non-zero/not approximately-zero in mask/bias-trained networks; bias-trained on x-axis and mask-trained on y-axis. We observe a mean correlation coefficient of 0.461 ± 0.022SD. Plot A is mean$\pm$1sd over 4 mask-trained/bias-trained nets. Plot B, and C are both one representative network trained with biases and with masks. Learned parameters were initialized to uniform on $[-0.1, 0.1]$, weights to uniform on $[-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}]$; all other hyper-parameters can be found in the codebase.

across datasets/different network parameterizations, and whether there might be scenarios where one style of learning works better or worse.

Interestingly, we found that bias and mask learning find different, albeit correlated, solutions. We calculated the variance of each hidden unit across $10,000$ MNIST images, in both bias and mask trained paradigms, as a measure of the hidden layer representation of MNIST. The histogram over hidden units of these variances is plotted for trained masks/biases (black/orange) on the same network weight (Fig.G.1.B). The mask-trained networks find sparser solutions (more 0s, far left histogram bin) and perform the task with lower unit variance values (see middle/right of histogram) compared to the bias-trained networks. We also plotted the correlation (scatter plot; Fig.G.1.C) between hidden unit variances for units in mask and bias-trained networks. The scatter plot and correlation were calculated after removing unit variances that were zero in the mask-trained net and sufficiently close to zero (mean variance divided by 100) in the bias-trained network. Despite the differences observed in the histogram, we do notice a correlation between hidden unit variances, suggesting that there is some overlap in the way that these two methods evaluate on MNIST.

Appendix H. Bias Distribution

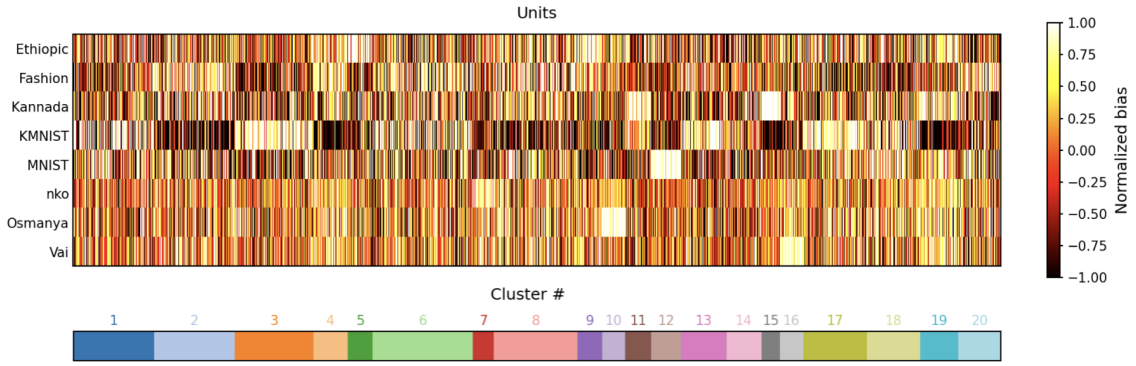

Figure H.1: **Bias distribution of bias-only network with 32000 hidden units**. We observe similar clusters as in Figure 1C, further showing the relationship between task variance and bias