Learning Stochastic Rainbow Networks

Vivian White
Department of Computer Science
Western Washington University
Bellingham, WA, USA

Muawiz Chaudhary
Department of Computer Science
Mila, Concordia University
Montréal, QC, Canada

Guy Wolf
Department of Mathematics and Statistics
Mila, Université de Montréal
Montréal, QC, Canada

Guillaume Lajoie
Department of Mathematics and Statistics
Mila, Université de Montréal
Montréal, QC, Canada

Kameron Decker Harris
Department of Computer Science
Western Washington University
Bellingham, WA, USA
kameron.harris@wwu.edu

Abstract

Random feature models are a popular approach for studying network learning that can capture important behaviors while remaining simpler than traditional training. Guth et al. [2024] introduced "rainbow" networks which model the distribution of trained weights as correlated random features conditioned on previous layer activity. Sampling new weights from distributions fit to learned networks led to similar performance in entirely untrained networks, and the observed weight covariance were found to be low rank. This provided evidence that random feature models could be extended to some networks away from initialization. However, White et al. [2024] failed to replicate their results in the deeper ResNet18 architecture. Here we ask whether the rainbow formulation can succeed in deeper networks by directly training a stochastic ensemble of random features, which we call *stochastic rainbow networks*. At every gradient descent iteration, new weights are sampled for all intermediate layers and features aligned layer-wise. We find: (1) this approach scales to deeper models, which outperform shallow networks at large widths; (2) ensembling multiple samples from the stochastic model is better than retraining the classifier head; and (3) low-rank parameterization of the learnable weight covariances can approach the accuracy of full-rank networks. This offers more evidence for rainbow and other structured random feature networks as reduced models of deep learning.

1 Introduction

Random feature networks (RFNs) are a mathematically tractable method of understanding learning mechanisms in "black box" neural networks. They comprise hidden layers with fixed random weights and a trainable readout for classification or regression. In wide regimes, these networks approximate functions in a reproducing kernel Hilbert space [Neal, 1996, Williams, 1997]. While this is a drastic simplification over traditional end-to-end training by backpropagation [Lee et al., 2020], RFNs capture important behaviors. These include double descent [Belkin et al., 2019], benign overfitting

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

[Zhang et al., 2021], architectural implicit bias [Xiao, 2021], and "lazy" learning [Chizat et al., 2018, Jacot et al., 2018].

Structured random feature (SRF) networks are RFNs with nontrivial weight covariance that capture some structure of traditionally trained networks. Gaussian SRFs can achieve better accuracy and learn from fewer examples than RFNs with unstructured covariances [Pandey et al., 2022]. A *rainbow network* [Guth et al., 2024] is a multi-layer SRF with different structured—colored covariance—random features across layers. Importantly, the features of a given layer are aligned under different draws of the random weights by solving an orthogonal Procrustes problem. Rainbow networks were used to model the joint probability distribution of trained neural network weights, and drawing new random weights could successfully approximate the performance of trained 7-layer networks with fixed wavelet filters without additional training. White et al. [2024] introduced learnable SRF networks which allowed the weight covariance itself to be learned. These learnable networks achieve close to the accuracy of traditionally trained deep models with many times fewer parameters using principled factorization of the covariance. However, sampling new weights and aligning as in [Guth et al., 2024] led to a drastic decrease in performance in deep architectures. These results cast doubt on whether or not the rainbow model is a good model for deep learning.

To further test deep rainbow networks, we propose a method of training where new weights are sampled at every gradient descent iteration. Information only propagates through a single layer of reference and sample networks before the two representations are aligned. By training wide rainbow networks in the stochastic ensemble, we approach the performance of traditional deep networks in image classification tasks (ResNets on CIFAR-10 are shown in the main text; Appendix B contains additional architectures and datasets). These networks are enhanced by ensembling and perform well with low-rank covariances. Our work redeems the rainbow model and highlights the usefulness of structured random feature networks as reduced, tractable models for neural network learning.

2 Methods

We investigate a training procedure (outlined in Fig. 1A) where the weights change at every gradient descent iteration. Intermediate layer activations are aligned to the corresponding layer of a reference network [Guth et al., 2024]. We use the SRF networks introduced in White et al. [2024]. The final linear readout layer is a standard, single-pathway linear layer. At each intermediate layer there are two pathways: a reference and sampled pathway. Each computes a convolution with filter weights $W_{\text{ref}}, W_{\text{samp}}$, where $W_{\text{ref}} = G_{\text{ref}} C^{1/2}$ and $W_{\text{samp}} = G_{\text{samp}} C^{1/2}$. Matrix G_{ref} has i.i.d. Gaussian entries $\sim \mathcal{N}(0, 1)$ and is kept fixed during training, G_{samp} is drawn from the same distribution at each minibatch, and $C^{1/2}$ is a square root of a positive semi-definite covariance matrix. We parameterize C factorized into a spatial part C_{space} and channel part C_{channel}. Each factor's square root is represented by an upper-triangular matrix R with non-negative diagonal so that $C = R^T R$.

Layer activations $Z_{\text{ref}} = \texttt{ReLU}(W_{\text{ref}} * X)$, $Z_{\text{samp}} = \texttt{ReLU}(W_{\text{samp}} * X)$ are computed for a minibatch of input X. From these, we compute an alignment matrix $A = UV^T$, where $\texttt{SVD}(Z_{\text{samp}}^T Z_{\text{ref}}) = USV^T$ is the singular decomposition of the channel-wise cross-covariance between sample and reference networks. This aligns Z_{samp} to Z_{ref}, defining the input to the next layer as $Z' = Z_{\text{samp}} A$. Information propagates primarily through the sampled pathway; the reference pathway's only influence is via the alignment. Traditional ResNet layers are defined as Conv-BatchNorm-ReLU [He et al., 2015]; we use BatchNorm-Conv-ReLU before alignment.

After re-sampling the changing pathway, the network may be adapted to the data. When un-adapted, we immediately evaluate the network on the test set without retraining the final linear layer or BatchNorms. Adaptation refers to a secondary training phase, after weights are resampled and aligned, where BatchNorm layers modify their statistics and we tune the final classifier layer [Vianna et al., 2024]. The alignment matrix is computed by averaging the feature cross-covariances across the entire training set; a single SVD is performed on this matrix, and this alignment is used for all batches at test time ("trainset" alignment). We use trainset alignment on all unadapted and adapted samples and when we ensemble multiple unadapted samples of the stochastic network. In Appendix B.1, we explore another type of alignment where the alignment matrix is calculated per batch on the test set and an SVD is performed for each minibatch as during training ("minibatch" alignment). Additional network training details are in Appendix A.1.

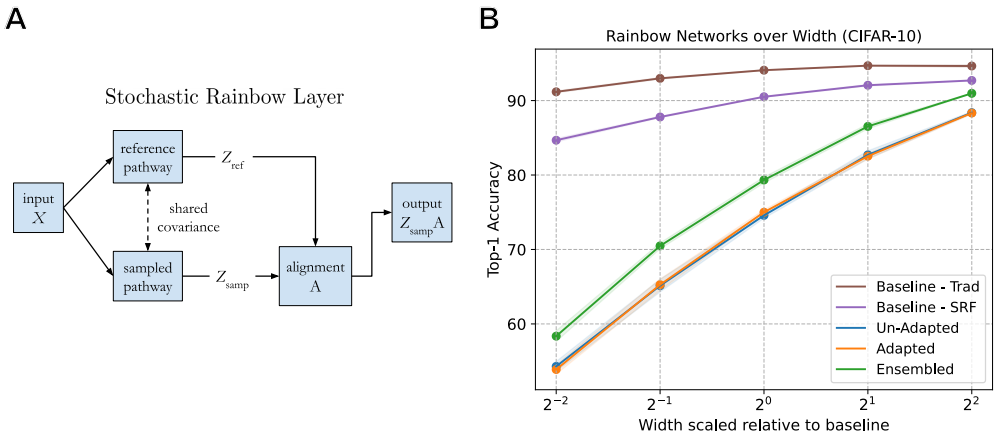

Figure 1: **(A)** Stochastic rainbow network layer block. Input propagates through the sampled pathway which is aligned to the activations of the reference pathway. These sampled and reference blocks correspond to BatchNorm-Conv-ReLU blocks in ResNet architectures. **(B)** Performance of rainbow networks over widths compared to traditional ResNet18 (Baseline - Trad) and SRF (Baseline - SRF) networks, averaged over five seeds. Ensembled rainbow samples are within 2% of SRF accuracy at the highest width. Adapting the BatchNorm statistics and classifier heads has little to no impact, while ensembling multiple network samples offers large performance gains.

3 Results

Fig. 1B shows the performance of a traditional ResNet18 contrasted with an SRF and our stochastic rainbow model with un-adapted, adapted, and ensembled networks. Our un-adapted samples averaged over 5 seeds reach 88.6% accuracy at the largest width (scale 2^2 or 2048 features in the widest layer). Fig. 4 in Appendix B.2 shows that our training procedure can successfully outperform the shallower 7-layer scattering networks used in Guth et al. [2024], Andreux et al. [2020]. Adapting the BatchNorm statistics and classifier of a sampled model, as in Guth et al. [2024], does not have a noticeable impact with our trainset method of alignment compared to unadapted samples.

We consider generating ensembles of network output by resampling the changing pathways while the reference pathways stay fixed (Fig. 1A) without any adaptation of the classifier or BatchNorm layers. Ensembles of 5 samples significantly increased performance at all widths and outperform adaptation. Ensembled width-4 networks achieve 90.97% accuracy, a difference of just 1.73% from the SRF network at the same batch size and optimization parameters.

Overall, our results show that ensembling multiple draws of the sampled network pathway is a reliable way to boost stochastic rainbow network performance.

In Fig. 2A, we visualize the eigenvalue spectra of the channel covariances C_{channel} in our stochastic rainbow networks across all layers. The channel covariance is generally much higher-dimensional than the spatial covariance since ResNet uses small filters. We do not show the spectra of the first layer nor skip-connection layers. Generally, the eigenvalues decay quickly with a slower decay in deeper layers. All layers in the stochastic networks are effectively low-rank. Comparing with Fig. 5 in Guth et al. [2024], for scattering and ResNet models trained on ImageNet, shows that our stochastic rainbow model captures the behavior of traditional models trained on similar tasks.

Motivated by findings of effective low-rank structure, we directly trained low-rank stochastic rainbow models. Fig. 2B shows the results of sweeping the maximum rank compared with SRF networks at widths 1, 2, and 4. At higher widths, rainbow network performance is closer to the SRFs at all ranks. Across ranks and widths, ensembling consistently outperforms adaptation. The low-rank stochastic curves flatten out with higher ranks, indicating that strong compression of the parameters can be achieved with marginal loss of accuracy. Appendix A.2 shows that these low-rank formulations also find success in Wide ResNet and ResNeXt SRF architectures.

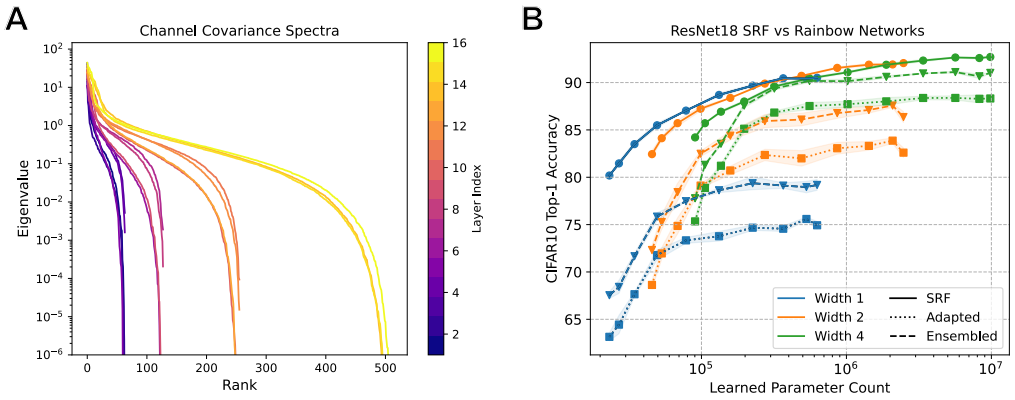

Figure 2: **(A)** Spectra of stochastic rainbow network channel covariance matrices across layers. The eigenvalues decay rapidly in the early layers and more slowly in the later layers. All layers are effectively low-rank. **(B)** Low-rank stochastic networks compared to low-rank ResNet18 SRF networks with relative widths 1, 2, and 4. At higher widths, the stochastic networks are closer to the SRF performance. Ensembling 5 network samples (dashed lines) consistently outperforms adapting the BatchNorm and linear classifier head for 5 epochs (dotted lines). Results are averaged over 3 seeds.

Some additional results follow in Appendix B. We break down the difference in adapting BatchNorm or linear readout separately for both single-shot or ensembled networks. Aligning across the training set does marginally better than aligning per minibatch on ResNets, and adapting the BatchNorms has slight gains over tuning the linear layer. We give direct comparison with the Guth et al. [2024] results, show results for varying depth up to 50 layers, explore additional datasets (CIFAR-100, SVHN, MNIST, KMNIST, FMNIST) and architectures (MLP, sigmoid nonlinearities), and visualize some of the learned spatial covariances. We also tested the SiLU [Hendrycks and Gimpel, 2023] and Mish [Misra, 2020] nonlinearities in the ResNet, finding similar results as ReLU (not shown).

4 Discussion

We successfully build deep stochastic rainbow networks, random networks with structure that capture much of the performance of traditional deep networks. Our training procedure scales the rainbow formulation to networks which outperform the shallow rainbow networks in Guth et al. [2024]. We explore the effects of adapting the BatchNorm layers and classifier heads of rainbow sampled networks versus ensembling multiple stochastic network samples. Adaptation provides little difference compared to un-adapted samples, but the ensemble method that we introduce reliably increases performance. Finally, we find that using low-rank covariances in these networks allows for major savings in learned parameter counts at little cost to accuracy. Our results hold over depth, datasets, non-linearities, and architectures.

The key differences between our model and the original rainbow work are: (1) we train the network end-to-end with changing weights, and (2) information only propagates through one layer before sample and reference paths are aligned (Fig. 1A). Guth et al. [2024] took traditional networks and fit their weights as samples from an SRF. Their only gradient-based training was an adaptation phase on the classifier layer, and they align two entirely different networks where information propagates *through all upstream layers* before that layer is aligned. White et al. [2024] showed that activation alignment began to break down approximately 7 layers into the network. We believe our method succeeds in deeper networks because it only has to compensate for one layer's misalignment at a time. This change is important, yet it seems that our model remains theoretically close to the rainbow model: alignment only depends on previous layer activations, and entirely new network samples maintain their accuracy without adaptation.

Some limitations to our work include our focus on small-scale image classification and a limited set of architectures; replacing the fully-connected blocks of transformers with stochastic rainbow

layers is an intriguing possibility. We would like to understand better how the channel covariances compare across reference pathway draws; rainbow theory assumes that there is an "eigennetwork" that all samples can be aligned to for a given architecture and task. The ensemble results have connections to Bayesian neural networks and could induce robustness. Some of our deepest and widest networks show reduced performance which may be due to overfitting, suggesting we should explore regularization of the covariance or alignment. It would be interesting to study loss basins from the rainbow lens in light of work by Ainsworth et al. [2023]. In future work, we plan to investigate these issues.

Many in the community believe that random networks are important for understanding deep learning, and we study a general and flexible family of random networks. Our results indicate that the rainbow model remains a viable model for deep learning. By decoupling the randomness of features from their structure, it offers an avenue for deciphering the role that randomness (initialization, SGD, etc.) plays and how it interacts with the features that are learned. Structured random feature networks shed light on biological neural principles where evolution may encode random circuit statistics that are then leveraged by plastic readouts [Litwin-Kumar et al., 2017, Harris, 2019, Xie et al., 2022]. The principles underlying biological and artificial networks are shared, and our work makes progress towards more fundamental understanding of distributed processing and learning.

Acknowledgments and Disclosure of Funding

We are grateful to Florentin Guth for discussions and sharing of numerical results from the rainbow paper. Thank you to Olexa Bilaniuk for discussions and support with the Mila cluster. Thank you to Oliver Richardson and the anonymous reviewers for their helpful comments on the initial submission which have improved the manuscript.

V.W. was the lead on the low-rank experiments, tested alignment and adaptation variations, ran network depth experiments, and drafted the majority of the paper. M.C. conceived and initially implemented this formulation of the stochastic rainbow layer, led its implementation including the variants of alignment and adaptation, validated results on additional datasets, non-linearties, and architectures, investigated the impact of increasing number of samples in the rainbow ensemble, drafted the covariance results, and edited the paper. V.W. and M.C. collaboratively discussed and worked on experiment design and methodology and analyzed results. G.W., G.L., and K.D.H. helped conceptualize the project, provided mentorship and resources, and edited the paper.

V.W. was supported by a fellowship from the International Network for Bio-Inspired Computing (NSF AccelNet award 2019976) and travel grants from the WWU CS department, WWU Graduate School, and ASWWU Student Enhancement Fund.

References

Florentin Guth, Brice Ménard, Gaspar Rochette, and Stéphane Mallat. A rainbow in deep network black boxes. *Journal of Machine Learning Research*, 25:1–59, 2024.

Vivian White, Muawiz Sajjad Chaudhary, Guy Wolf, Guillaume Lajoie, and Kameron Decker Harris. Learning and aligning structured random feature networks. In *ICLR 2024 Workshop on Representational Alignment*, 2024. URL `https://openreview.net/forum?id=vWhUQXQoFF`.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer New York, New York, NY, 1996. ISBN 978-0-387-94724-2 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0. URL `http://link.springer.com/10.1007/978-1-4612-0745-0`. ISSN: 0930-0325.

Christopher K. I. Williams. Computing with Infinite Networks. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 295–301. MIT Press, 1997. URL `http://papers.nips.cc/paper/1197-computing-with-infinite-networks.pdf`.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite Versus Infinite Neural Networks: an Empirical Study.

In *Advances in Neural Information Processing Systems*, volume 33, pages 15156–15172. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/ad086f59924fffe0773f8d0ca22ea712-Abstract.html.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, August 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1903070116. URL https://www.pnas.org/content/116/32/15849.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115, March 2021. ISSN 0001-0782, 1557-7317. doi: 10.1145/3446776. URL https://dl.acm.org/doi/10.1145/3446776.

Lechao Xiao. Eigenspace Restructuring: a Principle of Space and Frequency in Neural Networks, December 2021. URL http://arxiv.org/abs/2112.05611. arXiv:2112.05611 [cs, stat].

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. *arXiv:1812.07956 [cs, math]*, December 2018. URL http://arxiv.org/abs/1812.07956. arXiv: 1812.07956.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper/2018/hash/5a4be1fa34e62bb8a6ec6b91d2462f5a-Abstract.html.

Biraj Pandey, Marius Pachitariu, Bingni W. Brunton, and Kameron Decker Harris. Structured random receptive fields enable informative sensory encodings. *PLOS Computational Biology*, 18(10), 10 2022. doi: 10.1371/journal.pcbi.1010484. URL https://doi.org/10.1371/journal.pcbi.1010484.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Pedro Vianna, Muawiz Chaudhary, Paria Mehrbod, An Tang, Guy Cloutier, Guy Wolf, Michael Eickenberg, and Eugene Belilovsky. Channel-selective normalization for label-shift robust test-time adaptation, 2024. URL https://arxiv.org/abs/2402.04958.

Mathieu Andreux, Tomas Angles, Georgios Exarchakis, Roberto Leonarduzzi, Gaspar Rochette, Louis Thiry, John Zarka, Stephane Mallat, Joakim Anden, Eugene Belilovsky, Joan Bruna, Vincent Lostanlen, Muawiz Chaudhary, Matthew J. Hirn, Edouard Oyallon, Sixin Zhang, Carmine Cella, and Michael Eickenberg. Kymatio: Scattering transforms in python. *Journal of Machine Learning Research*, 21(60):1–6, 2020. URL http://jmlr.org/papers/v21/19-047.html.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Diganta Misra. Mish: A self regularized non-monotonic activation function, 2020. URL https://arxiv.org/abs/1908.08681.

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries, 2023. URL https://arxiv.org/abs/2209.04836.

Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and L. F. Abbott. Optimal Degrees of Synaptic Connectivity. *Neuron*, 93(5):1153–1164.e7, March 2017. ISSN 0896-6273. doi: 10.1016/j.neuron.2017.01.030. URL http://www.sciencedirect.com/science/article/pii/S0896627317300545.

Kameron Decker Harris. Additive function approximation in the brain. *NeurIPS, Real Neurons Hidden Units workshop*, arXiv:1909.02603, 2019.

Marjorie Xie, Samuel Muscinelli, Kameron Decker Harris, and Ashok Litwin-Kumar. Task-dependent optimal representations for cerebellar learning. *bioRxiv*, 2022.08.15.504040, August 2022. doi: 10.1101/2022.08.15.504040. URL http://biorxiv.org/lookup/doi/10.1101/2022.08.15.504040.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL `http://arxiv.org/abs/1912.01703`.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. URL `http://arxiv.org/abs/1611.05431`.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL `http://arxiv.org/abs/1605.07146`.

Tien Ho-Phuoc. Cifar10 to compare visual recognition performance between deep neural networks and humans. *arXiv preprint arXiv:1811.07270*, 2018.

Supplemental Material

A Additional Methods

A.1 Training Details

All models used in the main text are ResNet18 (width 1 corresponding to 512 features) trained for 200 epochs on the CIFAR-10 dataset with a batch size of 1024. We use the SGD optimizer, a learning rate of 0.1, and cosine annealing learning rate scheduler. Our alignment block is implemented as a Pytorch module [Paszke et al., 2019] which uses a regularized SVD for numerical stability of the gradients. Our code is at `https://github.com/glomerulus-lab/fact-conv`.

A.2 Low-Rank Covariance Parameterization

In learnable SRF networks [White et al., 2024], the weights at layer l in a network are $G_l R_l$, where G_l is a fixed i.i.d. Gaussian matrix and $R_l = R_1 \otimes R_2$ is an upper triangular matrix factored as the product of the channel covariance R_1 and the spatial covariance R_2. In these factored layers, the upper-triangular R_1 and R_2 matrices are the Cholesky square root of the covariance and the only learnable weight parameters. To ensure positive semi-definiteness, we apply the absolute value function[1] to the diagonals of R_1 and R_2, which are initialized as the identity matrix.

We learn a rank-r covariance by only learning the first r rows of R_1 or R_2. Since channel covariances are higher-dimensional than spatial covariances [White et al., 2024, Guth et al., 2024], we keep the spatial covariances full while learning low-rank channel covariances. Visualizing the full-rank channel covariances (not shown) indicates strong learned diagonals, so we experiment with learning low-ranks plus the full main diagonal. All low-rank results in the main body of the paper refer to low-rank plus the full main diagonal.

We apply our learnable low-rank covariances to stochastic rainbow networks as well as traditionally-trained 18-layer ResNet [He et al., 2015], ResNeXt [Xie et al., 2016], and Wide ResNet [Zagoruyko and Komodakis, 2016] SRF (non-rainbow) architectures. We train these models with maximum channel ranks ranging from $r = 1$ to full rank in powers of two on the CIFAR-10 dataset. These low-rank SRF models have a similar training set-up to the models used in the main text, but use a batch size of 128 instead of 1024.

Fig. 3A shows low-rank ResNet18 SRF models compared to low-rank-plus-full-main-diagonal ResNet18 SRFs. Learning the full main diagonal with low-rank models improves performance while keeping the number of learned parameters relatively low. Fig. 3B shows low-rank SRF performance across different architectures. At half their maximum rank, all SRF models are within three percent accuracy of their traditional counterparts while reducing the number of learnable parameters. Low-rank models can achieve decent accuracy with significant parameter compression across many vision architectures. We can use low-rank SRFs to reduce the number of learnable parameters needed for human-level accuracy $\approx 94\%$ [Ho-Phuoc, 2018].

B Supplementary Experiments

B.1 Training versus Minibatch Alignment and Adaptation

We test different forms of alignment: "trainset", where we align to the average of the feature cross-covariances across the training set, or "minibatch", where we align the features per each minibatch, on un-adapted and adapted rainbow samples. Table 1 shows results on single network samples and Table 2 ensembles results over 5 un-adapted and adapted samples (not 10 as in the main text), where we align multiple sampled pathways to a single reference. Generally, trainset alignment slightly outperforms minibatch accuracy, and tuning the BatchNorms has more of an impact than the linear layer in the ensembled networks.

[1] We find the use of absolute value on the diagonal elements of the upper-triangular matrices improves performance compared to the exponential function used in White et al. [2024]

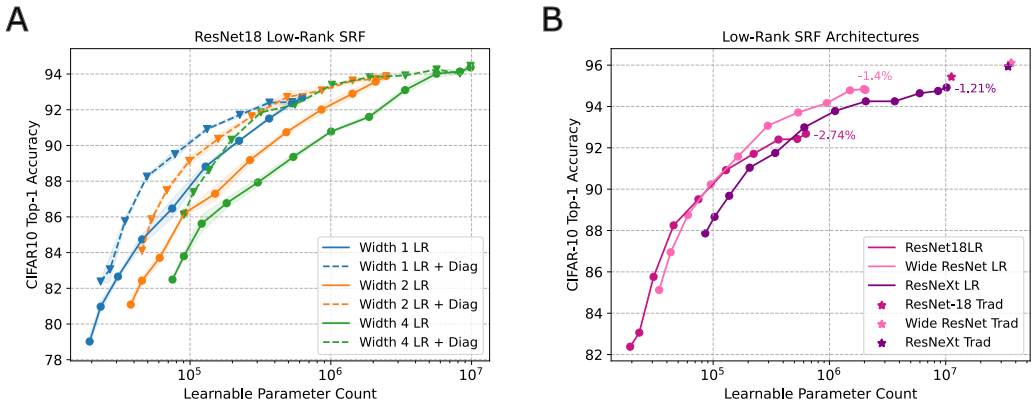

Figure 3: (**A**) Performance of Low-Rank (LR) and Low-Rank plus the full main diagonal (LR + Diag) ResNet18 SRF models at scaled widths 1, 2, and 4 relative to baseline. All low-rank-plus-diag models achieve higher accuracies than low-ranks at little cost to the numbers of learned parameters. (**B**) Performance of low-rank SRF ResNet18, Wide ResNet, and ResNeXt models compared to traditionally trained models. All factored SRF models come close to the traditional models at full rank, and further parameter reductions only smoothly decrease the error.

Table 1: Alignment adaptation for single network samples

	Width 1		Width 2		Width 4	
Adaptation and alignment modes	Train	Test	Train	Test	Train	Test
Un-Adapted Minibatch	75.02	73.51	84.17	79.85	93.20	87.84
Un-Adapted Trainset	77.41	76.94	85.35	82.66	94.21	88.10
Adapted Minibatch (BN)	77.46	76.97	85.23	81.72	93.84	88.53
Adapted Minibatch (LL)	77.56	76.79	85.16	81.76	93.99	88.23
Adapted Minibatch (BN+LL)	77.71	76.78	85.42	81.64	93.94	88.71
Adapted Trainset (BN)	77.38	76.90	85.46	82.62	94.10	88.40
Adapted Trainset (LL)	77.60	77.05	85.11	82.32	94.04	88.32
Adapted Trainset (BN + LL)	77.54	77.27	85.10	82.80	93.93	88.68

Table 2: Alignment adaptation for ensembled networks

	Width 1		Width 2		Width 4	
Adaptation and alignment modes	Train	Test	Train	Test	Train	Test
Un-Adapted Minibatch	80.53	79.96	89.12	84.31	93.04	90.98
Un-Adapted Trainset	80.15	80.08	88.30	85.59	91.64	90.75
Adapted Minibatch (BN)	80.93	80.23	89.30	83.89	93.06	91.16
Adapted Minibatch (LL)	79.58	79.05	88.66	83.65	92.78	90.76
Adapted Minibatch (BN+LL)	80.98	79.68	89.24	84.24	93.02	90.89
Adapted Trainset (BN)	80.92	80.06	88.95	85.94	92.07	90.71
Adapted Trainset (LL)	79.88	79.14	88.64	85.34	91.79	90.52
Adapted Trainset (BN + LL)	82.80	79.93	89.06	85.78	92.08	91.05

B.2 Comparison with Guth et al. [2024]

Fig. 4 compares our unadapted and ensembled stochastic rainbow networks to the unadapted and adapted scattering rainbow networks from Guth et al. [2024] (results were provided in personal communication). At the largest width scale, our unadapted stochastic rainbow networks outperform unadapted scattering rainbow networks and match the adapted ones, and our ensembled networks outperform adapted networks.

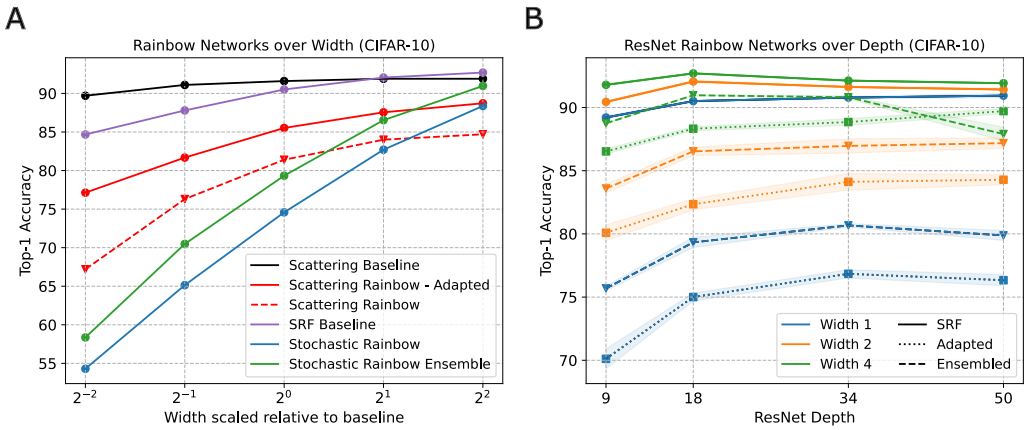

Figure 4: (**A**) We compare the 7-layer scattering rainbow network results from Guth et al. [2024] to our ResNet18 stochastic rainbow networks. At the largest width, our methods outperform the scattering models. (**B**) We implement stochastic rainbow networks using 9-, 18-, 34-, and 50-layer ResNet architectures with width scaling factors of 1, 2, and 4 and plot accuracy versus depth. All models are trained with a batchsize of 1024 except the width 4 adapted and ensembled ResNet50 samples, which use a batchsize of 512. Results are averaged over 5 seeds. Rainbow network performance improves with depth and width except in the ResNet50 case. The deepest and widest networks, rainbow and baseline models, may be overfitting.

B.3 ResNet depth experiments

To support our claim that the stochastic rainbow method succeeds in deep models, we trained ResNet9, ResNet18, ResNet34, and ResNet50 rainbow models [He et al., 2015]. Fig. 4 displays adapted, ensembled, and SRF baselines for each of the ResNet depths at widths 1, 2, and 4. We see an increase in rainbow network performance with depth until depth 50. Width 4 ensembled results decrease in accuracy after depth 18. We believe that these decreases are due to overfitting.

B.4 ResNet SVHN and CIFAR-100 Experiments

To demonstrate that our results are not specific to CIFAR-10, we applied our stochastic methodology to SVHN and CIFAR-100. Evaluation results are averaged across 5 different random seeds for the initialization and minibatch draws.

SVHN results appear in Fig. 5A as accuracy versus width for baseline and stochastic rainbow models. This is qualitatively similar to Fig. 1B, although the difference between trainset adapted and ensembled models is less than for CIFAR-10. We found that we were able to match the performance of the SRF baseline with our rainbow ensemble at the widest width.

On CIFAR-100, shown in Fig. 5B, the differences between the traditional and SRF baselines are much larger, having a 10% difference at the smaller widths. The rainbow approximation tends to get better over width, but the ensemble retains a more significant gap from SRF baseline than with other datasets.

B.5 Ensemble Size

We investigate the impact of ensemble size on model performance. Results are shown averaged across 5 different seeds in Fig. 6A (CIFAR-10) and Fig. 6B (CIFAR-100). In both cases, the performance increases rapidly for the smallest size ensembles and plateaus around 10 samples.

B.6 Spatial Covariances

In Fig. 7 we visualize the spatial covariances across layers of traditional convolutional, SRF, and stochastic rainbow networks at a width 4. Each row shows a specific layer's covariance matrix across

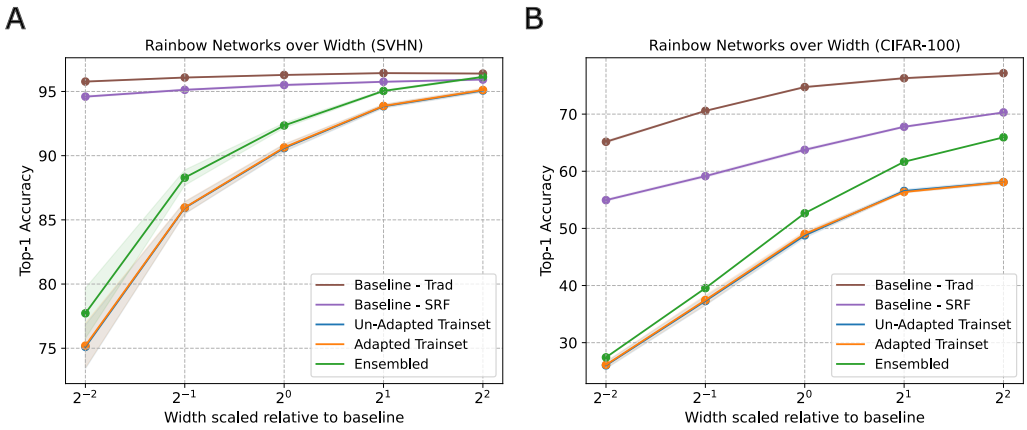

Figure 5: (**A**) Performance of rainbow networks over widths compared to traditional ResNet18 (Baseline - Trad) and SRF (Baseline - SRF) networks on SVHN, averaged over five seeds. Ensembled rainbow samples match performance of SRF baseline at highest width. (**B**) Performance of rainbow networks over widths compared to traditional ResNet18 (Baseline - Trad) and SRF (Baseline - SRF) networks on CIFAR-100, averaged over five seeds.

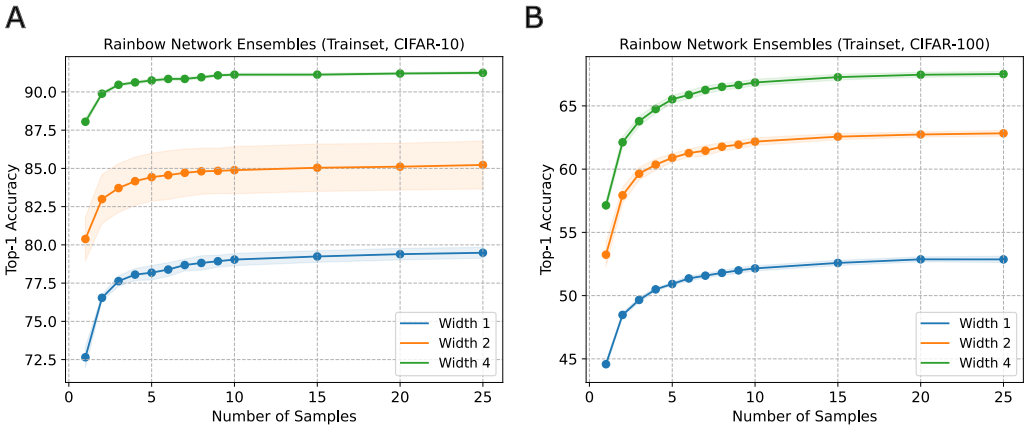

Figure 6: (**A**) Increasing number of samples increases ensemble performance for CIFAR-10. One of the seeds in the width 2 ensemble had outlying low performance (7% less than others) leading to increased variance. (**B**) For CIFAR100, increasing number of samples increases ensemble performance.

5 seeds. In all cases, covariance are similar across seeds, but the rainbow networks show the least obviously similar structure.

B.7 Multilayer Perceptron Architectures

To show that our results are not specific to the previous architectures and nonlinearities, we implemented a multilayer perceptron (MLP), i.e. a fully-connected network, with ReLU or sigmoid nonlinearities. These MLP networks were tested on the MNIST, Fashion-MNIST, and KMNIST datasets. The architecture of the MLP includes 3 hidden stochastic rainbow layers (BatchNorm-linear-nonlinearity) then a standard linear readout. Width 1 corresponds to 512 units in the hidden layers, and we varied this by powers of 2. The network is trained for 100 epochs using cross-entropy loss, batch size 1000, and the ADAM optimizer with default settings. We include results for both minibatch and trainset alignment. All data are averaged across 5 random seeds, and ensembles include 5 samples.

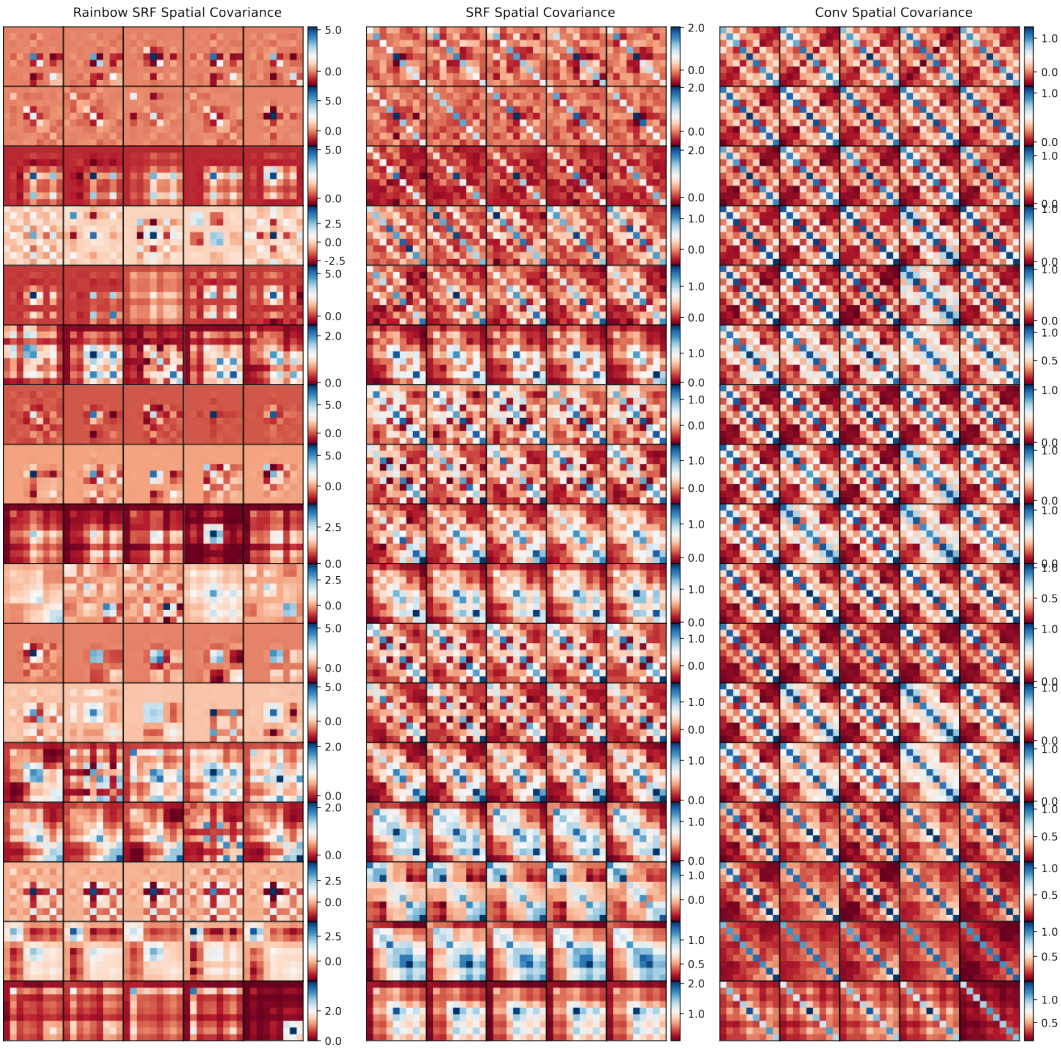

Figure 7: Spatial covariances across 5 seeds of stochastic rainbow (left), SRF (middle), and traditional (right) ResNet18 networks. SRF and Conv networks share more spatial covariance similarities across seeds than rainbow networks.

The results are shown in Fig. 8A-E. The ensemble uses trainset alignment for each sample in the ensemble. In most cases, there is an optimal width for stochastic network performance. Unlike with ResNet models, minibatch alignment—where the matrix is calculated per-batch—sometimes outperforms trainset alignment and ensembling. At the largest widths, there is evidence of overfitting seen with the decrease in test set accuracy. Usually this overfitting is more pronounced in the stochastic rainbow methods than in the traditional or SRF models.

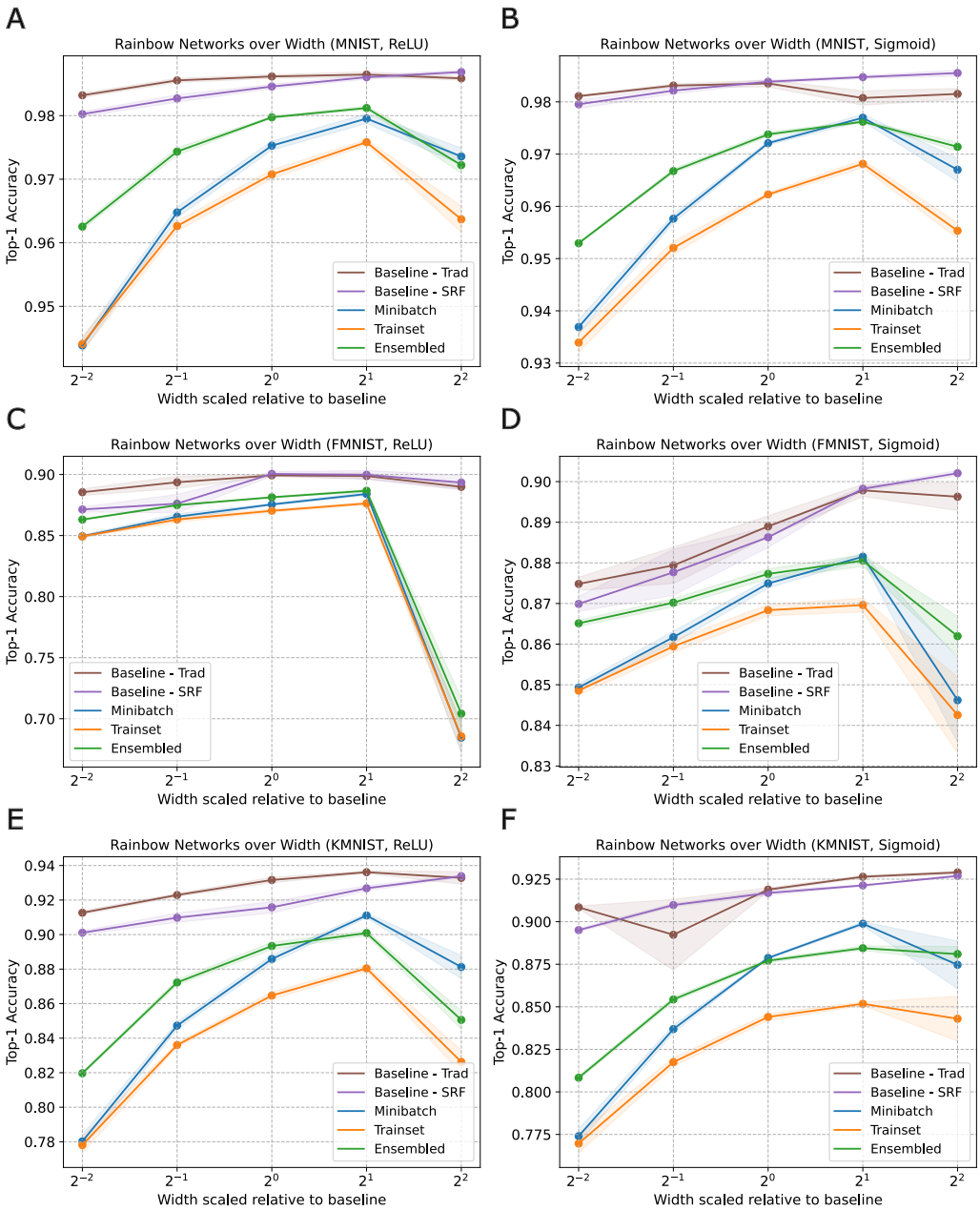

Figure 8: Performance of stochastic rainbow MLP network compared to traditional and SRF baselines. (**A**) ReLU nonlinearity on MNIST. (**B**) Sigmoid nonlinearity on MNIST. (**C**) ReLU nonlinearity on FMNIST. (**D**) Sigmoid nonlinearity on FMNIST. (**E**) ReLU nonlinearity on KMNIST. (**F**) Sigmoid nonlinearity on KMNIST.