Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes

Roman Novak [⋆], **Lechao Xiao** [⋆⋆], **Jaehoon Lee** [⋆†], **Yasaman Bahri** [⋆†], **Greg Yang**[◦], **Daniel A. Abolafia**, **Jeffrey Pennington**, **Jascha Sohl-Dickstein**

Google Brain, [◦]Microsoft Research AI

{romann, xlc, jaehlee, yasamanb, gregyang@microsoft.com, danabo, jpennin, jaschasd}@google.com

1 Introduction

There is a previously identified equivalence between wide fully connected neural networks (FCNs) and Gaussian processes (GPs) [29, 25, 27, 26] (see §A.2 for details on related work). This equivalence enables, for instance, test set predictions that would have resulted from a fully Bayesian, infinitely wide trained FCN to be computed without ever instantiating the FCN, but by instead evaluating the corresponding GP. In this work, we derive an analogous equivalence for multi-layer convolutional neural networks (CNNs) both with and without pooling layers (§2, §A.6), and achieve state of the art results on CIFAR10 for GPs without trainable kernels (Table 2). We also introduce a Monte Carlo method to estimate the GP corresponding to a given neural network architecture, even in cases where the analytic form has too many terms to be computationally feasible (§A.5).

Surprisingly, in the absence of pooling layers, the GPs corresponding to CNNs with and without weight sharing are identical (§3.1). As a consequence, translation equivariance in finite-channel CNNs trained with stochastic gradient descent (SGD) has no corresponding property in the Bayesian treatment of the infinite channel limit – a qualitative difference between the two regimes that is not present in the FCN case. We confirm experimentally, that while in some scenarios the performance of SGD-trained finite CNNs approaches that of the respective GPs as the channel count increases, with careful tuning SGD-trained CNNs can significantly outperform their corresponding GPs (§3.3), suggesting advantages from SGD training compared to fully Bayesian parameter estimation[1].

2 Many-channel Bayesian CNNs are Gaussian processes

2.1 Notation. Consider a series of L convolutional hidden layers, $l = 0, ..., L-1$. The parameters of the network are the convolutional filters and biases, $\omega^l_{ij,\beta}$ and b^l_i, respectively, with outgoing (incoming) channel index i (j) and filter relative spatial location $\beta = -k, ..., k$. Assume a Gaussian prior on both the filter weights and biases, $\omega^l_{ij,\beta} \sim \mathcal{N}\left(0, \sigma^2_\omega / \left[n^l (2k+1)\right]\right), b^l_i \sim \mathcal{N}\left(0, \sigma^2_b\right)$. The weight and bias variances are $\sigma^2_\omega, \sigma^2_b$, respectively. n^l is the number of channels in layer l, $2k+1$ is the filter size. Let $\mathcal{X}$ denote a set of input images (training set or validation set or both). The network has activations $y^l(x)$ and pre-activations $z^l(x)$ for each input image $x \in \mathcal{X} \subset \mathbb{R}^{n^0 \times d}$, with input channel count n^0, number of pixels d (we consider 1-d convolution without loss of generality), where

$$y^l_{i,\alpha}(x) \equiv \begin{cases} x_{i,\alpha} & l = 0 \\ \phi\left(z^{l-1}_{i,\alpha}(x)\right) & l > 0 \end{cases}, \qquad z^l_{i,\alpha}(x) \equiv \sum_{j=1}^{n^l} \sum_{\beta=-k}^{k} \omega^l_{ij,\beta} y^l_{j,\alpha+\beta}(x) + b^l_i. \tag{1}$$

Finally, we define the empirical covariance matrix K^l of the activations y^l as

$$\left[K^l\right]_{\alpha,\alpha'}(x, x') \equiv \frac{1}{n^l} \sum_{i=1}^{n^l} y^l_{i,\alpha}(x) y^l_{i,\alpha'}(x'). \tag{2}$$

[*]Google AI Residents (g.co/airesidency). [⋆], [†] Equal contribution.

[1]This work is an extended abstract of Novak et al. [32], which contains a more detailed presentation.

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

Please refer to §A.9, §A.6, and Figure 5 for more details on notation.

2.2 Derivation. As can be seen in Equation 1, the pre-activations z^l are an affine transformation of the multivariate Gaussian $\left\{\omega^l, b^l\right\}$, specified by the previous layer's activations y^l. An affine transformation of a multivariate Gaussian is itself a Gaussian with a covariance matrix that can be derived straightforwardly. Specifically,

$$\left(z^l|y^l\right) \sim \left(z^l|K^l\right) \sim \mathcal{N}\left(0, \mathcal{A}\left(K^l\right) \otimes I_{n^{l+1}}\right), \tag{3}$$

where $I_{n^{l+1}}$ is an $n^{l+1} \times n^{l+1}$ identity matrix, and $\mathcal{A}\left(K^l\right)$ is the covariance of the pre-activations z_i^l and is derived in Xiao et al. [46] as follows: $\left[\mathcal{A}\left(K\right)\right]_{\alpha,\alpha'}(x,x') \equiv \sigma_b^2 + \sigma_\omega^2 / \left(2k+1\right) \sum_\beta \left[K\right]_{\alpha+\beta,\alpha'+\beta}(x,x')$. It follows from Equation 3 that the summands in Equation 2 are i.i.d. conditioned on K^{l-1}. Subject to weak restrictions on the nonlinearity ϕ, we can apply the weak law of large numbers and conclude that

$$\forall K^{l-1} \quad \left(K^l|K^{l-1}\right) \xrightarrow[n^l \to \infty]{P} \left(\mathcal{C} \circ \mathcal{A}\right)\left(K^{l-1}\right) \quad \text{in probability, where,} \tag{4}$$

$$\left[\mathcal{C}\left(K\right)\right]_{\alpha,\alpha'}(x,x') \equiv \mathbb{E}_{z\sim\mathcal{N}(0,K)}\left[\phi\left(z_\alpha(x)\right)\phi\left(z_{\alpha'}(x')\right)\right]. \tag{5}$$

For nonlinearities such as ReLU [28] and the error function (erf) $\mathcal{C}$ can be computed in closed form as derived in Cho & Saul [10] and Williams [41] respectively.

A less obvious result is that, under slightly stronger assumptions on ϕ, the top-layer activation covariance K^L becomes unconditionally deterministic as channels in all hidden layers grow to infinity simultaneously (Theorem A.4). This lets us treat z^L as a sequence of Gaussians with a random covariance matrix converging to a constant, which allows for the following result proven in §A.8.4.

Result. If $\phi : \mathbb{R} \to \mathbb{R}$ has an exponentially-bounded derivative, i.e. $\exists\, a, b \in \mathbb{R} : \forall x \in \mathbb{R} \;\; \left|\phi'\left(x\right)\right| \leq a \exp\left(bx\right)$ a.e. (almost everywhere), then the following convergence in distribution holds:

$$\left(z^L|y^0\right) \xrightarrow[\min\{n^1,\ldots,n^L\}\to\infty]{\mathcal{D}} \mathcal{N}\left(0, \mathcal{A}\left(K_\infty^L\right) \otimes I_{n^{L+1}}\right), \quad \text{where} \quad K_\infty^L \equiv \left(\mathcal{C} \circ \mathcal{A}\right)^L\left(K^0\right). \tag{6}$$

2.3 Class predictions. We have shown in Equation 6 that in the infinite channel limit a deep CNN is a GP indexed by input samples and spatial locations of the top layer. We now remark that transformations to obtain class predictions that are common in CNN classifiers can be represented as either vectorization or projection, preserving the GP equivalence. We compare the presented below strategies in Figure 1.

2.3.1 Vectorization. A common readout strategy is to vectorize (flatten) the output of the last convolutional layer into a vector and stack a fully connected layer on top: $z_i^{L+1}(x) = \sum_{j=1}^{n^{L+1}d} \omega_{ij}^{L+1} \phi\left(\text{vec}\left[z^L(x)\right]\right)_j + b_i^{L+1}$. One can easily verify that in the infinite channel limit this architecture results in the following per-class covariance: $\mathcal{K}_\infty^{\text{vec}} \equiv \sigma_\omega^2/d \sum_\alpha \left[K_\infty^{L+1}\right]_{\alpha,\alpha} + \sigma_b^2$. As observed in Xiao et al. [46], to compute diagonal terms of $\left[K_\infty^{l+1}\right](x,x')$ one needs only the corresponding diagonal terms of $\left[K_\infty^l\right](x,x')$. Consequently the memory cost is only $\mathcal{O}\left(|\mathcal{X}|^2 d\right)$. Note that this approach ignores pixel-pixel covariances and produces a GP corresponding to a locally-connected network (see §3).

2.3.2 Projection. Another approach is a projection collapsing spatial dimensions. Let $h \in \mathbb{R}^d$, and define the output to be $z_i^{L+1}(x) = \sum_{j=1}^{n^{L+1}} \omega_{ij}^{L+1} \left(\phi\left(z^L(x)\right) h\right)_j + b_i^{L+1}$. This leads to the output per-class covariance $\mathcal{K}_\infty^h \equiv \sigma_\omega^2 \sum_{\alpha,\alpha'} h_\alpha h_{\alpha'} \left[K_\infty^{L+1}\right]_{\alpha,\alpha'} + \sigma_b^2$. Examples of this approach include

2.3.2.1 Global average pooling: take $h = \frac{1}{d}\mathbf{1_d}$. Then $\mathcal{K}_\infty^{\text{pool}} \equiv \sigma_\omega^2/d^2 \sum_{\alpha,\alpha'} \left[K_\infty^{L+1}\right]_{\alpha,\alpha'} + \sigma_b^2$. This corresponds to applying global average pooling right after the last convolutional layer. It takes all pixel-pixel covariances into consideration and makes the kernel translation invariant. However, it requires $\mathcal{O}\left(|\mathcal{X}|^2 d^2\right)$ memory. It is impractical to use this method to analytically evaluate the GP, and we evaluate it using Monte Carlo sampling (see §A.5).

2.3.2.2 Subsampling one particular pixel: take $h = e_\alpha$, leading to $\mathcal{K}_\infty^{e_\alpha} \equiv \sigma_\omega^2 \left[K_\infty^{L+1}\right]_{\alpha,\alpha} + \sigma_b^2$. This approach makes use of only one pixel-pixel covariance, and requires the same amount of memory as $\mathcal{K}_\infty^{\text{vec}}$ to compute.

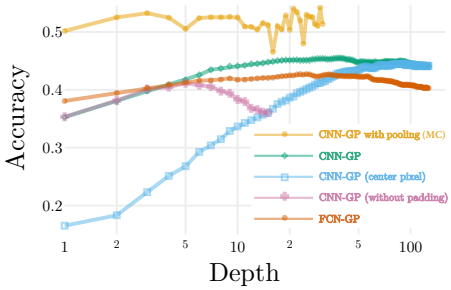

Figure 1: Different dimensionality collapsing strategies described in §2.3. Validation accuracy of a **CNN-GP with pooling** (§2.3.2.1, Monte Carlo estimate, see §A.5) is better than other models due to translation invariance. **CNN-GP with zero padding** (§2.3.1) outperforms an analogous **CNN-GP without padding** as depth increases. At depth 15 output without padding is reduced spatially to 1×1, making the **CNN-GP without padding** equivalent to the **center pixel selection strategy** (§2.3.2.2), also performing worse than the **CNN-GP** (we conjecture, due to overfitting to centrally-located features) but approaches the latter in the limit of large depth, as information becomes more uniformly spatially distributed [46]. **CNN-GPs** generally outperform **FCN-GP** (right) due to the local connectivity prior, but can fail to capture nonlinear interactions between spatially-distant pixels at shallow depths (left). Values are reported on a 2K/4K train/validation subsets of CIFAR10. See §A.10.3 for experimental details.

3 Discussion

3.1 Bayesian CNNs with many channels are identical to locally connected networks, in the absence of pooling. Locally Connected Networks (LCNs) [13, 23] are CNNs without weight sharing between spatial locations. LCNs preserve the connectivity pattern, and thus topology, of a CNN. However, they do not possess the equivariance property of a CNN – if an input is translated, the latent representation in an LCN will be completely different, rather than also being translated. The CNN-GP predictions without spatial pooling in §2.3.1 and §2.3.2.2 depend only on sample-sample covariances, and do not depend on pixel-pixel covariances. LCNs destroy pixel-pixel covariances: $\left[K_\infty^L\right]^{\text{LCN}}_{\alpha,\alpha'}(x,x') = 0$, for $\alpha \neq \alpha'$. However, LCNs preserve the covariances between input examples at every pixel: $\left[K_\infty^L\right]^{\text{LCN}}_{\alpha,\alpha}(x,x') = \left[K_\infty^L\right]^{\text{CNN}}_{\alpha,\alpha}(x,x')$. As a result, in the absence of pooling, LCN-GPs and CNN-GPs are identical. Moreover, LCN-GPs with pooling are identical to CNN-GPs with vectorization of the top layer (under suitable scaling of x^{L+1}). We confirm these findings experimentally in Figures 2 (b), 3, 4, and Table 1.

3.2 Pooling leverages equivariance to provide invariance. The only kernel leveraging pixel-pixel covariances is that of the CNN-GP with pooling. This enables the predictions of this GP and the corresponding CNN to be invariant to translations (modulo edge effects) – a beneficial quality for an image classifier (a well known in literature observation [7]). We observe strong experimental evidence supporting the benefits of invariance throughout this work (Figures 1, 2 (b), 3; Tables 1, 2), in both CNNs and CNN-GPs.

3.3 Finite-channel SGD-trained CNNs can outperform infinite-channel Bayesian CNNs, in the absence of pooling. In the absence of pooling, the benefits of equivariance and weight sharing are more challenging to explain in terms of Bayesian priors on class predictions. Indeed, in this work we find that the performance of finite-width SGD-trained CNNs often approaches that of their CNN-GP counterpart (Figure 2, b, c)[3], suggesting that in those cases equivariance does not play a beneficial role in SGD-trained networks. However, as can be seen in Tables 1, 2, and Figure 2 (c), the best CNN *overall* outperforms the best CNN-GP by a significant margin – an observation specific to CNNs and not FCNs or LCNs. We observe this gap in performance especially in the case of ReLU networks trained with a large learning rate. In Table 1 we demonstrate this large gap in performance by evaluating different models with equivalent architecure and hyperparameter settings, chosen for good SGD-trained CNN performance.

We conjecture that equivariance, a property lacking in LCNs and the Bayesian treatment of the infinite channel CNN limit, improves the performance of SGD-trained finite-channel CNNs with the correct settings of hyperparameters. We hope our work stimulates future research into disentangling the contributions of the two qualities (Bayesian treatment and infinite width) to the performance gap observed.

[3]This observation is conditioned on the respective NN fitting the training set to 100%. Underfitting breaks the correspondance to an NN-GP, since train set predictions of such a network no longer correspond to the true training labels. Properly tuned underfitting often also leads to better generalization (Table 2).

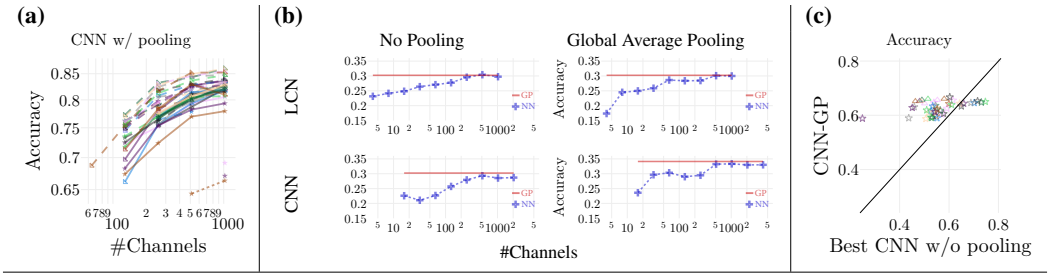

Figure 2: **(a): SGD-trained CNNs often perform better with increasing number of channels.** Each line corresponds to a particular choice of architecture and initialization hyperparameters, with best learning rate and weight decay selected independently for each number of channels (x-axis). **(b): SGD-trained CNNs often approach the performance of their corresponding CNN-GP with increasing number of channels.** All models have the same architecture except for pooling and weight sharing, as well as training-related hyperparameters such as learning rate, weight decay and batch size, which are selected for each number of channels (x-axis) to maximize validation performance (y-axis) of a neural network. As the number of channels grows, best validation accuracy increases and approaches accuracy of the respective GP (solid horizontal line). **(c): However, the best-performing SGD-trained CNNs can outperform their corresponding CNN-GPs.** Each point corresponds to the validation accuracy of: (y-axis) a specific CNN-GP; (x-axis) the best CNN with the same architectural hyper-parameters selected among the 100%-accurate models on the full training CIFAR10 dataset with different learning rates, weight decay and number of channels. While CNN-GP appears competitive against 100%-accurate CNNs (above the diagonal), the best CNNs *overall* outperform CNN-GPs by a significant margin (below the diagonal, right). **Experimental details:** all networks have reached 100% training accuracy on CIFAR10. Values in (b) are reported on an 0.5K/4K train/validation subset downsampled to 8×8 for computational reasons. See §A.10.5 and §A.10.1 for full experimental details of (a, c) and (b) plots respectively.

Quality:	Compositionality		Local connectivity		Equivariance	Invariance
Model:	FCN	FCN-GP	LCN (w/ pooling)	CNN-GP	CNN	CNN w/ pooling
Error:	46.26	41.45	36.52 (36.23)	36.71	19.93	**16.54**

Table 1: **Disentangling the role of network topology, equivariance, and invariance on test performance, for SGD-trained and infinite width Bayesian networks.** Test error (%) on CIFAR10 of different models of the same depth, nonlinearity, and weight and bias variances. LCN and CNN-GP have a hierarchical local topology, beneficial for image recognition tasks and outperform fully connected models (FCN and FCN-GP). As predicted in §3.1: (i) weight sharing has no effect in the Bayesian treatment of an infinite width CNN (CNN-GP performs similarly to an LCN, a CNN without weight sharing) , and (ii) pooling has no effect on generalization of an LCN model (LCN and LCN with pooling perform nearly identically) . Local connectivity combined with equivariance (CNN) is enabled by weight sharing in an SGD-trained finite model, allowing for a significant improvement. Finally, invariance enabled by weight sharing and pooling (CNN w/ pooling) allows for the best performance. Values are reported for 8-layer ReLU models. See §A.10.6 for experimental details and Table 2 for more model comparisons.

Acknowledgments

We thank Sam Schoenholz, Vinay Rao, Daniel Freeman, and Qiang Zeng for frequent discussion and feedback on preliminary results. We thank AnonReviewer3 for an insightful and detailed openreview of the respective ICLR2019 submission.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Anonymous. Deep convolutional gaussian process. In *Submitted to International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyeUPi09Y7. under review.

[3] Patrick Billingsley. *Convergence of probability measures*. John Wiley & Sons, 1999.

[4] Kenneth Blomqvist, Samuel Kaski, and Markus Heinonen. Deep convolutional gaussian processes. *arXiv preprint arXiv:1810.03052*, 2018.

[5] Anastasia Borovykh. A gaussian process perspective on convolutional neural networks. *arXiv preprint arXiv:1810.10798*, 2018.

[6] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

[7] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[8] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[9] Minmin Chen, Jeffrey Pennington, and Samuel Schoenholz. Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 873–882, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/chen18i.html.

[10] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pp. 342–350, 2009.

[11] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[12] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pp. 2253–2261, 2016.

[13] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

[14] Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow Gaussian processes. *arXiv preprint arXiv:1808.05587*, aug 2018. URL https://arxiv.org/abs/1808.05587.

[15] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1487–1495. ACM, 2017.

[16] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *arXiv preprint arXiv:1803.01719*, 2018.

[17] Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*, 2015.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*, 2015.

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[20] Vinayak Kumar, Vaibhav Singh, PK Srijith, and Andreas Damianou. Deep gaussian processes with convolutional kernels. *arXiv preprint arXiv:1806.01655*, 2018.

[21] Neil D Lawrence and Andrew J Moore. Hierarchical gaussian process latent variable models. In *Proceedings of the 24th international conference on Machine learning*, pp. 481–488. ACM, 2007.

[22] Nicolas Le Roux and Yoshua Bengio. Continuous neural networks. In *Artificial Intelligence and Statistics*, pp. 404–411, 2007.

[23] Yann Lecun. Generalization and network design strategies. In *Connectionism in perspective*. Elsevier, 1989.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Jaehoon Lee, Yasaman Bahri, Roman Novak, Sam Schoenholz, Jeffrey Pennington, and Jascha Sohl-dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1EA-M-0Z.

[26] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 9 2018.

[27] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 4 2018. URL https://openreview.net/forum?id=H1-nGgWC-.

[28] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[29] Radford M. Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

[30] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *Proceeding of the international Conference on Learning Representations workshop track*, abs/1412.6614, 2015.

[31] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJC2SzZCW.

[32] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018.

[33] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances In Neural Information Processing Systems*, pp. 3360–3368, 2016.

[34] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[35] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. In *In Neural Infomration Processing Systems*, 2007.

[36] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[37] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *ICLR*, 2017.

[38] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

[39] Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional gaussian processes. In *Advances in Neural Information Processing Systems 30*, pp. 2849–2858, 2017.

[40] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[41] Christopher KI Williams. Computing with infinite networks. In *Advances in neural information processing systems*, pp. 295–301, 1997.

[42] Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pp. 2586–2594, 2016.

[43] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378, 2016.

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

[45] Lechao Xiao, Yasaman Bahri, Sam Schoenholz, and Jeffrey Pennington. Training ultra-deep cnns with critical initialization. In *NIPS Workshop*, 2017.

[46] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5393–5402, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/xiao18a.html.

[47] Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In *Advances in neural information processing systems*, pp. 7103–7114, 2017.

A Appendix

A.1 Further model comparison

Model	CIFAR10	MNIST	Fashion-MNIST
CNN with pooling	**14.85 (15.65)**	–	–
CNN with ReLU and large learning rate	24.76 (17.64)	–	–
CNN-GP	32.86	0.88	**7.40**
CNN with small learning rate	33.31(22.89)	–	–
CNN with erf (any learning rate)	33.31(22.17)	–	–
Convolutional GP [39]	35.40	1.17	–
ResNet GP [14]	–	**0.84**	–
Residual CNN-GP [14]	–	0.96	–
CNN-GP [14]	–	1.03	–
FCN-GP	41.06	1.22	8.22
FCN-GP [25]	44.34	1.21	–
FCN	45.52 (44.73)	–	–

Table 2: **Aspects of architecture and inference influencing test performance.** Test error (%) for best model within each model family, maximizing validation accuracy over depth, width, and training and initialization hyperpameters. Except where indicated by parentheses, all models achieve 100% training accuracy. For SGD-trained CNNs, numbers in parentheses correspond to the same model family, but without restriction on training accuracy. CNN-GP achieves state of the art results on CIFAR10 for GPs without trainable kernels and outperforms SGD models optimized with a small learning rate to 100% train accuracy. When SGD optimization is allowed to underfit the training set, there is a significant improvement in generalization. Further, when ReLU nonlinearities are paired with large learning rates, the performance of SGD-trained models again improves relative to CNN-GPs, suggesting a beneficial interplay between ReLUs and fast SGD training. These differences in performance between CNNs and CNN-GPs are not observed between FCNs and FCN-GPs, or between LCNs and LCN-GPs (Table 1), suggesting that *equivariance* is the underlying factor responsible for the improved performance of finite SGD-trained CNNs relative to infinite Bayesian CNNs without pooling. See §A.10.5 for experimental details.

A.2 Related work and summary of contributions

In early work on neural network priors, Neal [29] demonstrated that, in a fully connected network with a single hidden layer, certain natural priors over network *parameters* give rise to a Gaussian process prior over *functions* when the number of hidden units is taken to be infinite. Follow-up work extended the conditions under which this correspondence applied [41, 22, 17]. An exactly analogous correspondence for infinite width, finite depth *deep* fully connected networks was developed recently in Lee et al. [25], Matthews et al. [27], with Matthews et al. [26] extending the convergence guarantees to a more general class of nonlinearities and width growth rates.

The line of work examining signal propagation in random deep networks [33, 37, 47, 16, 9] is related to the construction of the GPs we consider. They apply a mean field approximation in which the pre-activation signal is replaced with a Gaussian, and the derivation of the covariance function with depth is the same as for the kernel function of a corresponding GP. Recently, Xiao et al. [45, 46] extended this to convolutional architectures without pooling. Xiao et al. [46] also analyzed properties of the convolutional kernel at large depths to construct a *phase diagram* which will be relevant to NN-GP performance, as discussed in §A.4.

Compositional kernels coming from wide convolutional and fully connected layers also appeared outside of the GP context. Cho & Saul [10] derived closed-form compositional kernels for rectified polynomial activations (including ReLU). Daniely et al. [12] proved approximation guarantees between a network and its corresponding kernel, and show that empirical kernels will converge as the number of channels increases.

There is a line of work considering stacking of GPs, such as *deep GP*s [21, 11]. These no longer correspond to GPs, though they can describe a rich class of probabilistic models beyond GPs. Alternatively, *deep kernel learning* [43, 42, 6] utilizes GPs with base kernels which take in features produced by a deep neural network (often a CNN), and train the resulting model end-to-end. Finally, van der Wilk et al. [39] incorporates convolutional structure into GP kernels, with follow-up work stacking multiple such GPs [20, 4, 2] to produce a deep convolutional GP (which is no longer a GP). Our work differs from all of these in that our GP corresponds exactly to a fully Bayesian CNN in the infinite channel limit, when all layers are taken to be of infinite size. We remark that while alternative models, such as deep GPs, do include infinite-sized layers in their construction, they do not treat all layers in this way – for instance, through insertion of bottleneck layers which are kept finite. While it remains to be seen exactly which limit is applicable for understanding realistic CNN architectures in practice, the limit we consider is natural for a large class of CNNs, namely those for which all layers sizes are large and rather comparable in size. The deep GP approach, on the other hand, has better modelling capabilities at the expense of higher inference cost.

Borovykh [5] analyzes the convergence of CNN outputs at different spatial locations (or different timepoints for a temporal CNN) to a GP for a single input example. Thus, while they also consider a GP limit (and perform an experiment comparing posterior GP predictions to an SGD-trained CNN), they do not address the dependence of network outputs on multiple input examples, and thus their model is unable to generate predictions on a test set consisting of new input examples.

In concurrent work, Garriga-Alonso et al. [14] derive an NN-GP kernel equivalent to one of the kernels considered in our work. In addition to explicitly specifying kernels corresponding to pooling and vectorizing, we also compare the NN-GP performance to finite width SGD-trained CNNs and analyze the differences between the two models.

The specific novel contributions of the present work are:

1. We show analytically that CNNs with many channels, trained in a fully Bayesian fashion, correspond to an NN-GP (§2). We show this for CNNs both with and without pooling, with arbitrary convolutional striding, and with both same and valid padding. We prove convergence as the number of channels in hidden layers approach infinity simultaneously (i.e. $\min\left\{n^1, \ldots, n^L\right\} \to \infty$, see §A.8.4 for details), strengthening and extending the result of [26] under mild conditions on the nonlinearity derivative.
2. We show that in the absence of pooling, the NN-GP for a CNN and a Locally Connected Network (LCN) are identical (§3.1). An LCN has the same local connectivity pattern as a CNN, but without weight sharing or translation equivariance.
3. We experimentally compare trained CNNs and LCNs and find that under certain conditions both perform similarly to the respective NN-GP (Figure 2, b, c). Moreover, both architectures tend to perform better with increased channel count, suggesting that similarly to FCNs [30, 31] CNNs benefit from overparameterization (Figure 2, a, b), corroborating a similar trend observed in Canziani et al. [8, Figure 2]. However, we also show that careful tuning of hyperparameters allows finite CNNs trained with SGD to outperform their corresponding NN-GP by a significant margin. We experimentally disentangle and quantify the contributions stemming from local connectivity, equivariance, and invariance in a convolutional model in one such setting (Table 1).
4. We introduce a Monte Carlo method to compute NN-GP kernels for situations (such as CNNs with pooling) where evaluating the NN-GP is otherwise computationally infeasible (§A.5).

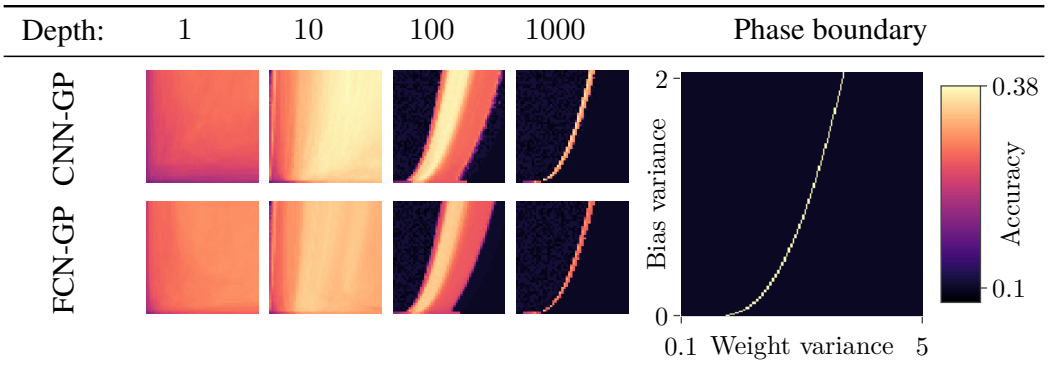

Table 3: **Validation accuracy** of CNN- and FCN- GPs as a function of weight (σ_ω^2, horizontal axis) and bias (σ_b^2, vertical axis) variances. As predicted in §A.4, the regions of good performance concentrate around the critical line (phase boundary, right) as the depth increases (left to right). All plots share common axes ranges and employ the erf nonlinearity. See §A.10.2 for experimental details.

A.3 Review of exact Bayesian regression with GPs

Our discussion in the paper has focused on model *priors*. A crucial benefit we derive by mapping to a GP is that Bayesian inference is straightforward to implement and can be done *exactly* for regression [36, chapter 2], requiring only simple linear algebra. Let $\mathcal{X}$ denote training inputs $x_1, ..., x_{|\mathcal{X}|}$, $\mathbf{t}^T = (t_1, ..., t_{|\mathcal{X}|})$ training targets, and collectively $\mathcal{D}$ for the training set. The integral over the posterior can be evaluated analytically to give a posterior predictive distribution on a test point y_* which is Normal, $(z^*|\mathcal{D}, y^*) \sim \mathcal{N}\left(\mu_*, \sigma_*^2\right)$, with

$$\mu_* = \mathcal{K}(x^*, \mathcal{X})(\mathcal{K}(\mathcal{X}, \mathcal{X}) + \sigma_\varepsilon^2 \mathbb{I}_{|\mathcal{X}|})^{-1}\mathbf{t}, \tag{7}$$

$$\sigma_*^2 = \mathcal{K}(x^*, x^*) - \mathcal{K}(x^*, \mathcal{X})(\mathcal{K}(\mathcal{X}, \mathcal{X}) + \sigma_\varepsilon^2 \mathbb{I}_{|\mathcal{X}|})^{-1}\mathcal{K}(\mathcal{X}, x^*). \tag{8}$$

We use the shorthand $\mathcal{K}(\mathcal{X}, \mathcal{X})$ to denote the $|\mathcal{X}| \times |\mathcal{X}|$ matrix formed by evaluating the GP covariance on the training inputs, and likewise $\mathcal{K}(x^*, \mathcal{X})$ is a $|\mathcal{X}|$-length vector formed from the covariance between the test input and training inputs. Computationally, the costly step in GP posterior predictions comes from the matrix inversion, which in all experiments were carried out exactly, and typically scales as $\mathcal{O}(|\mathcal{X}|^3)$ (though algorithms scaling as $\mathcal{O}(|\mathcal{X}|^{2.4})$ exist for sufficiently large matrices). Nonetheless, there is a broad literature on approximate Bayesian inference with GPs which can be utilized for efficient implementation [36, chapter 8]; [34, 38].

A.4 Relationship to Deep Signal Propagation

The recurrence relation linking the GP kernel at layer $l+1$ to that of layer l following from Equation 6 (i.e. $K_\infty^{l+1} = (\mathcal{C} \circ \mathcal{A})\left(K_\infty^l\right)$) is precisely the *covariance map* examined in a series of related papers on signal propagation [46, 33, 37, 25] (modulo notational differences; denoted as $F, \mathcal{C}$ or e.g. $\mathcal{A} \star \mathcal{C}$ in Xiao et al. [46]). In those works, the action of this map on hidden-state covariance matrices was interpreted as defining a dynamical system whose large-depth behavior informs aspects of trainability. In particular, as $l \to \infty$, $K_\infty^{l+1} = (\mathcal{C} \circ \mathcal{A})\left(K_\infty^l\right) \approx K_\infty^l \equiv K_\infty^*$, i.e. the covariance approaches a fixed point K_∞^*. The convergence to a fixed point is problematic for learning because the hidden states no longer contain information that can distinguish different pairs of inputs. It is similarly problematic for GPs, as the kernel becomes pathological as it approaches a fixed point. Precisely, in the chaotic regime outputs of the GP become asymptotically decorrelated and therefore independent, while in the ordered regime they approach perfect correlation of 1. Either of these scenarios captures no information about the training data in the kernel and makes learning infeasible.

This problem can be ameliorated by judicious hyperparameter selection, which can reduce the rate of exponential convergence to the fixed point. For hyperpameters chosen on a critical line separating two untrainable phases, the convergence rates slow to polynomial, and very deep networks can be trained, and inference with deep NN-GP kernels can be performed – see Table 3.

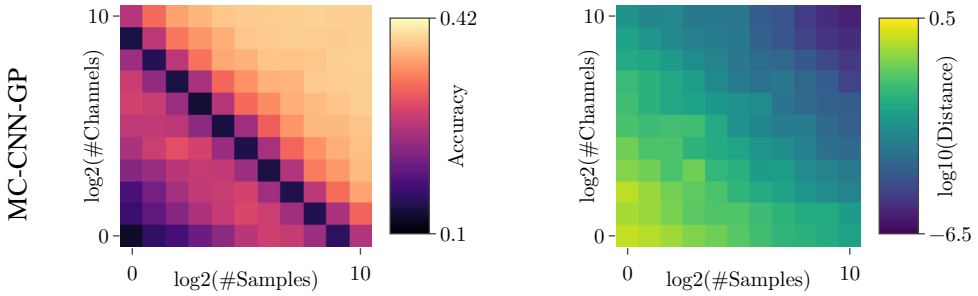

Figure 3: **Validation accuracy** (left) of an MC-CNN-GP increases with $n \times M$ (i.e. channel count times number of samples) and approaches that of the exact CNN-GP (not shown), while the **distance** (right) to the exact kernel decreases. The dark band in the left plot corresponds to ill-conditioning of $K^{L+1}_{n,M}$ when the number of outer products contributing to $K^{L+1}_{n,M}$ approximately equals its rank. Values reported are for a 3-layer model applied to a 2K/4K train/validation subset of CIFAR10 downsampled to 8×8. See Figure 4 for similar results with other architectures and §A.10.2 for experimental details.

A.5 Monte Carlo evaluation of intractable GP kernels

We introduce a Monte Carlo estimation method for NN-GP kernels which are computationally impractical to compute analytically, or for which we do not know the analytic form. Similar in spirit to traditional random feature methods [35], the core idea is to instantiate many random *finite* width networks and use the empirical uncentered covariances of activations to estimate the Monte Carlo-GP (MC-GP) kernel,

$$\left[K^l_{n,M}\right]_{\alpha,\alpha'}(x,x') \equiv \frac{1}{Mn}\sum_{m=1}^{M}\sum_{c=1}^{n} x^l_{c\alpha}\left(x;\theta_m\right) x^l_{c\alpha'}\left(x';\theta_m\right) \tag{9}$$

where θ consists of M draws of the weights and biases from their prior distribution, $\theta_m \sim p\left(\theta\right)$, and n is the width or number of channels in hidden layers. The MC-GP kernel converges to the analytic kernel with increasing width, $\lim_{n\to\infty} K^l_{n,M} = K^l_\infty$ in probability.

For finite width networks, the uncertainty in $K^l_{n,M}$ is $\mathrm{Var}[K^l_{n,M}] = \mathrm{Var}_\theta\left[K^l_n\left(\theta\right)\right]/M$. From Daniely et al. [12], we know that $\mathrm{Var}_\theta\left[K^l_n\left(\theta\right)\right] \propto \frac{1}{n}$, which leads to $\mathrm{Var}_\theta[K^l_{n,M}] \propto \frac{1}{Mn}$. For finite n, $K^l_{n,M}$ is also a biased estimate of K^l_∞, where the bias depends solely on network width. We do not currently have an analytic form for this bias, but we can see in Figures 3 and 4 that for the hyperparameters we probe it is small relative to the variance. In particular, $\left\|K^l_{n,M}\left(\theta\right) - K^L_\infty\right\|^2_{\mathrm{F}}$ is nearly constant for constant Mn. We thus treat Mn as the effective sample size for the Monte Carlo kernel estimate. Increasing M and reducing n can reduce memory cost, though potentially at the expense of increased compute time and bias.

In a non-distributed setting, the MC-GP reduces the memory requirements to compute $\mathcal{GP}^{\text{pool}}$ from $\mathcal{O}\left(\left|\mathcal{X}\right|^2 d^2\right)$ to $\mathcal{O}\left(\left|\mathcal{X}\right|^2 + n^2 + nd\right)$, making the evaluation of CNN-GPs with pooling practical.

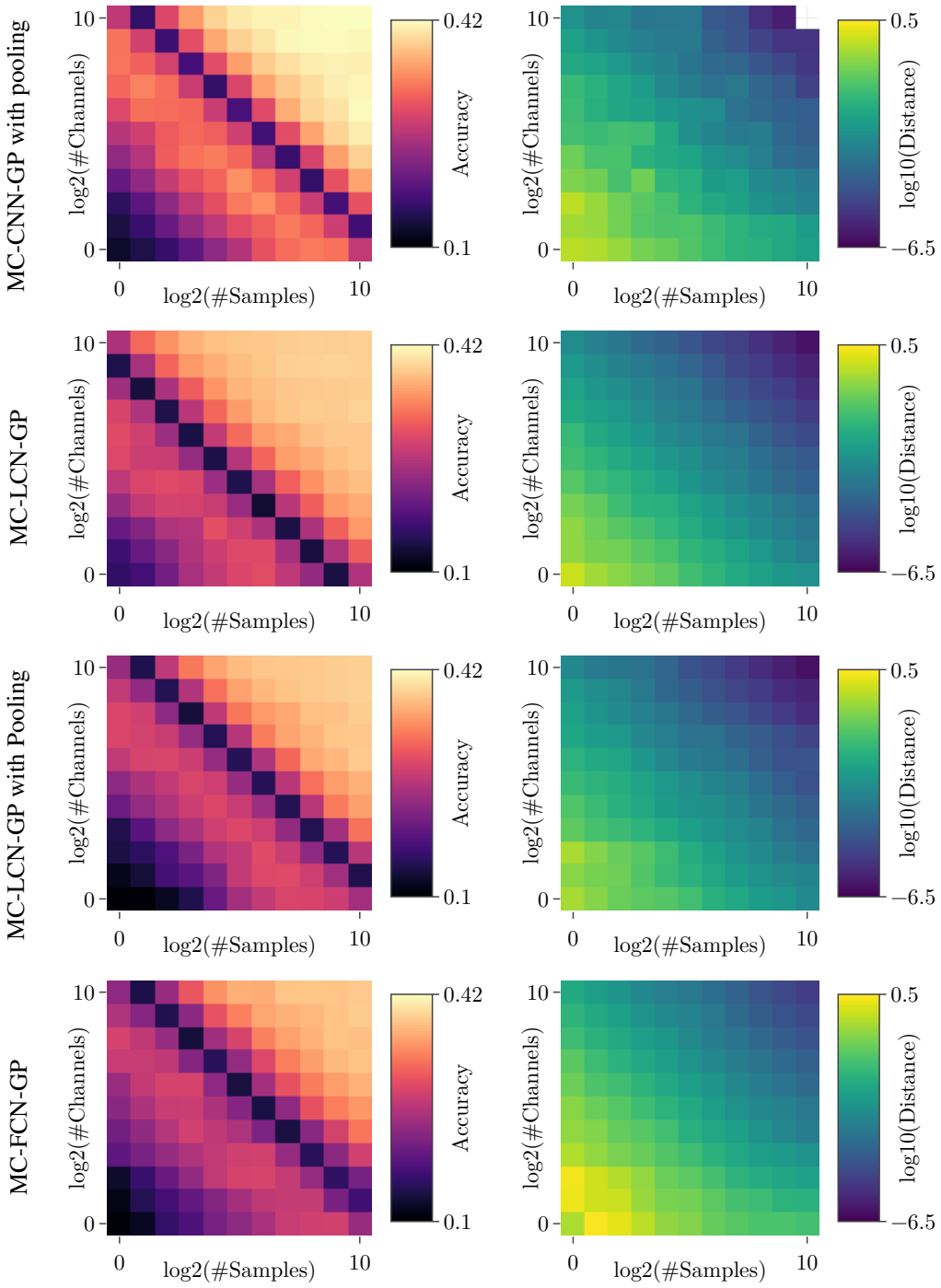

Figure 4: As in Figure 3, **validation accuracy** (left) of MC-GPs increases with $n \times M$ (i.e. width times number of samples), while the **distance** (right) to the the respective exact GP kernel (or the best available estimate in the case of CNN-GP with pooling, top row) decreases. We remark that when using shared weights, convergence is slower as smaller number of independent random parameters are being used. For example a single-layer MC-LCN-GP kernel is expected to converge approximately $\mathrm{Var}[K_{\mathrm{CNN}}]/\mathrm{Var}[K_{\mathrm{LCN}}] \sim \sqrt{\#\text{ LCN params}/\#\text{ CNN params}} = \sqrt{\text{spatial size of the output layer}}$ times faster than MC-CNN-GP, which is in agreement with our results obtained in the second row and Figure 3. I.e. the geometric mean of the ratios of the kernel distance from (3-layer) MC-CNN-GP and MC-LCN-GP to the respective CNN-GP is $\approx 2.2 > \sqrt{\text{spatial size of the output layer}} = \sqrt{2 \times 2} = 2$). See §A.10.2 for experimental details.

A.6 Strided convolutions, average pooling in intermediate layers, higher dimensions

Our analysis in the main text can easily be extended to cover average pooling and strided convolutions (applied before the pointwise nonlinearity). Recall that conditioned on K^l the pre-activation $z_j^l(x) \in \mathbb{R}^{d_1}$ is a zero-mean multivariate Gaussian. Let $B \in \mathbb{R}^{d_2 \times d_1}$ denote a linear operator. Then $Bz_j^l(x) \in \mathbb{R}^{d_2}$ is a zero-mean Gaussian, and the covariance is

$$\mathbb{E}_{\{\omega^l, b^l\}}\left[\left(Bz_j^l(x)\right)\left(Bz_j^l(x')\right)^T \middle| K^l\right] = B\mathbb{E}_{\{\omega^l, b^l\}}\left[z_j^l(x)\, z_j^l(x')^T \middle| K^l\right] B^T. \tag{10}$$

One can easily see that $\left\{Bz_j^l \middle| K^l\right\}_j$ are i.i.d. multivariate Gaussian as well.

Strided convolution. Strided convolution is equivalent to a non-strided convolution composed with subsampling. Let $s \in \mathbb{N}$ denote size of the stride. Then the strided convolution is equivalent to choosing B as follows: $B_{ij} = \delta(is - j)$ for $i \in \{0, 1, \dots (d_2 - 1)\}$.

Average pooling. Average pooling with stride s and window size ws is equivalent to choosing $B_{ij} = 1/ws$ for $i = 0, 1, \dots (d_2 - 1)$ and $j = is, \dots, (is + ws - 1)$.

ND convolutions. Note that our analysis in the main text (1D) easily extends to higher-dimensional convolutions by replacing integer pixel indices and sizes d, α, β with tuples (see also Figure 5). In Equation 1 β values would have to span the hypercube $[\pm k]^N = \{-k, \dots, k\}^N$ in the pre-activation definition. Similarly, in §2.3 the normalizing factor d (d^2) should be the product (squared) of its entries, and summations over α, β should span the $[d_0] \times \cdots \times [d_N]$ hypercube as well. The definition of the kernel propagation operator $\mathcal{A}$ in §2.2 will remain exactly the same, so long as β is summed over the hypercube, and the variance weights remain respectively normalized $\sum_\beta v_\beta = 1$.

A.7 Additional Figures

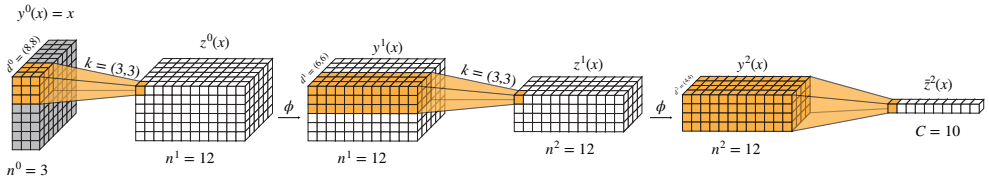

Figure 5: A sample CNN classifier annotated according to notation in §2.1, §2.3.1. The network transforms $n^0 \times d^0 = 3 \times 8 \times 8$-dimensional inputs $y^0(x) = x \in \mathcal{X}$ into $C = 10$-dimensional logits $\bar{z}^2(x)$. Model has two convolutional layers with $k = (3, 3)$-shaped filters, nonlinearity ϕ, and a fully-connected layer at the top ($y^2 \to \bar{z}^2(x)$, §2.3.1). Hidden (pre-)activations have $n^1 = n^2 = 12$ filters. As $\min\left\{n^1, n^2\right\} \to \infty$, the prior of this CNN will approach that of a GP indexed by inputs x and target class indices from 1 to $C = 10$. The covariance of such GP can be computed as $\left[\frac{\sigma_\omega^2}{6\times 6}\sum_{\alpha=(1,1)}^{d^2=(6,6)}\left[(\mathcal{C}\circ\mathcal{A})^2\left(K^0\right)\right]_{\alpha,\alpha} + \sigma_b^2\right] \otimes I_C$, where the sum is over the $\{1, \dots, 6\}^2$ hypercube (see §2.2, §2.3.1, §A.6). Presented is a CNN with stride 1 and no ("valid") padding, i.e. the spatial shape of the input shrinks as it propagates through it $\left(d^0 = (8, 8) \to d^1 = (6, 6) \to d^2 = (4, 4)\right)$. Note that for notational simplicity 1D CNN and circular padding with $d^0 = d^1 = d^2 = d$ is assumed in the text, yet our formalism easily extends to the model displayed (§A.6). Further, while displayed (pre-)activations have 3D shapes, in the text we treat them as 1D vectors (§A.9).

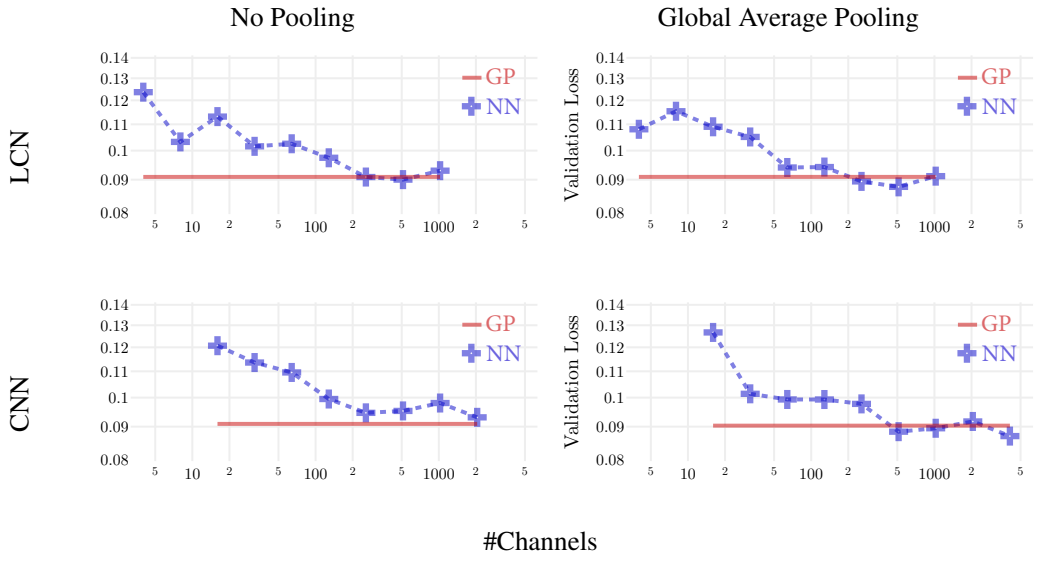

Figure 6: Best validation loss (vertical axis) of **trained neural networks** (dashed line) as the number of channels increases (horizontal axis) approaches that of a respective (Monte Carlo) **CNN-GP** (solid horizontal line). See Figure 2 (b) for validation accuracy, Figure 7 for training loss and §A.10.1 for experimental details.

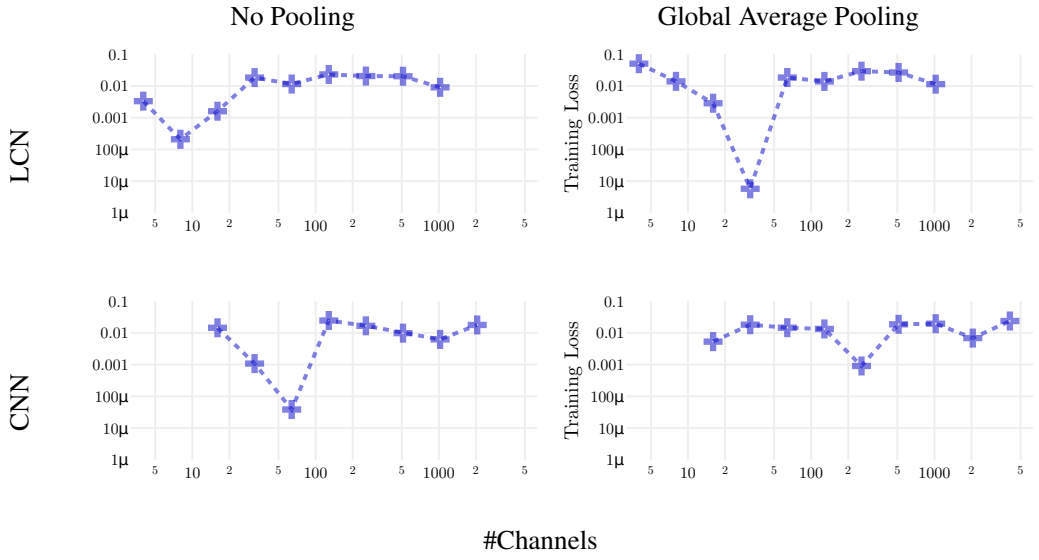

Figure 7: **Training loss** (vertical axis) of best (in terms of validation loss) neural networks as the number of channels increases (horizontal axis). While perfect 0 loss is not achieved (but 100% accuracy is), we observe no consistent improvement when increasing the capacity of the network (left to right). This eliminates underfitting as a possible explanation for why small models perform worse in Figure 2 (b). See Figure 6 for validation loss and §A.10.1 for experimental details.

A.8 Equivalence between randomly initialized NNs and GPs

In this section, we present two different approaches, the *sequential limit* (§A.8.3) and *simultaneous limit* (§A.8.4), to illustrate the relationship between many-channels Bayesian CNNs and GPs.

Sequential limit (§A.8.3) involves taking the infinite channel limit in hidden layers in a sequence, starting from bottom (closest to inputs) layers and going upwards (to the outputs), i.e. $n^1 \to \infty, \ldots, n^L \to \infty$. Note that this approach in fact *constructs* a GP using a NN architecture to define its covariance, and does not provide guarantees on actual convergence of large but finite Bayesian CNNs to GPs, which is of most practical interest. However, it has the following benefits:

1. Weak assumptions on the NN activation function ϕ and on the distribution of the NN parameters.
2. The arguments can be easily extended to more complicated network architectures, e.g. architectures with max pooling, dropout, etc.
3. This approach provides a straightforward and intuitive way to compute the covariance of the Gaussian process without diving into mathematical details.

Simultaneous limit (§A.8.4) considers growing the number of channels in hidden layers uniformly, i.e. $\min\left\{n^1, \ldots, n^L\right\} \to \infty$. This approach establishes convergence of finite channel Bayesian CNNs to GPs and is thus a more practically relevant result. However, it makes stronger assumptions, and the proof is more involved.

We highlight that the GPs obtained by the two approaches are identical.

In both sections, we only provide the arguments for CNNs. It is straightforward (and in fact simpler) to extend them to LCNs and FCNs. Indeed, an FCN is a particular case of a CNN where the inputs and filters have singular spatial dimensions ($d = 1$, $k = 0$). For LCNs, the proof goes through in an identical fashion if we replace $\mathcal{A}$ with $\mathcal{A}^{\text{LCN}}$ defined as $\left[\mathcal{A}^{\text{LCN}}(K)\right]_{\alpha,\alpha'}(x, x') \equiv \delta_{\alpha,\alpha'}\left[\mathcal{A}(K)\right]_{\alpha,\alpha'}(x, x')$.

A.8.1 Setup

Probability space. Let $\mathscr{P}$ be a collection of countably many mutually independent random variables (R.V.s) defined on some probability space $(\Omega, \mathcal{F}, P)$, where $\mathcal{F}$ is a σ-algebra defined on Ω and P is a probability measure. Here $\mathscr{P} \equiv \mathscr{W} \cup \mathscr{B} \cup \mathscr{H}$ is the collection of parameters used to define neural networks:

1. **Weights.** $\mathscr{W} = \bigcup_{l\in\mathbb{N}} \mathscr{W}^l$ and $\mathscr{W}^l = \left\{\omega^l_{ij,\beta} : i, j \in \mathbb{N}, \beta \in [\pm k]\right\}$, where $[\pm k] \equiv [-k, k] \cap \mathbb{Z}$. We assume $\omega^l_{ij,\beta}$ are i.i.d. R.V.s with mean zero and finite variance $0 < \sigma^2_\omega < \infty$. When $l = 0$, we further assume they are Gaussian distributed.
2. **Biases**. $\mathscr{B} = \bigcup_{l\in\mathbb{N}} \mathscr{B}^l$ and $\mathscr{B}^l = \left\{b^l_j : j \in \mathbb{N}\right\}$. We assume b^l_j are i.i.d. Gaussian with mean zero and variance $0 \le \sigma^2_b < \infty$.
3. **Place-holder.** $\mathscr{H}$ is a place-holder to store extra (if needed) R.V.s , e.g. parameters coming from the final dense layer.

Inputs. We will consider a fixed $\mathcal{X} \subseteq \mathbb{R}^{n^0 \times d}$ to denote the inputs, with input channel count n^0, number of pixels d. Assume $x \neq 0$ for all $x \in \mathcal{X}$ and $|\mathcal{X}|$, the cardinality of the inputs, is finite.

However, the *simultaneous limit* (§A.8.4) result can be extended to a countably-infinite input indexing spaces $\mathcal{X}$ for certain topologies via an argument presented in Matthews et al. [27, section 2.2], allowing to infer weak convergence on $\mathcal{X}$ from convergence on any finite subset (which is the case we consider in this text; see also Billingsley [3, page 19] for details).

Notation, shapes, and indexing. We adopt the notation, shape, and indexing convention similar to §2.1 and §A.9, which the reader is encouraged to review. We emphasize that whenever an index is omitted, the variable is assumed to contain all possible entries along the respective dimension (e.g. whole $\mathcal{X}$ if x is omitted, or all n^l channels if the channel i is omitted).

A.8.2 Preliminary

We will use the following well-known theorem.

Theorem A.1. *Let $X, \{X_n\}_{n\in\mathbb{N}}$ be R.V.s in $\mathbb{R}^m$. The following are equivalent:*

1. *$X_n \xrightarrow{\mathcal{D}} X$ (converges in distribution / converges weakly),*
2. *(Portmanteau Theorem) For all bounded continuous function $f : \mathbb{R}^m \to \mathbb{R}$,*
$$\lim_{n\to\infty} \mathbb{E}\left[f(X_n)\right] = \mathbb{E}\left[f(X)\right], \tag{11}$$
3. *(Lévy's Continuity Theorem) The characteristic functions of X_n, i.e. $\mathbb{E}\left[e^{\mathbb{i}t^T X_n}\right]$ converge to that of X pointwisely, i.e. for all $t \in \mathbb{R}^m$,*
$$\lim_{n\to\infty} \mathbb{E}\left[e^{\mathbb{i}t^T X_n}\right] = \mathbb{E}\left[e^{\mathbb{i}t^T X}\right], \tag{12}$$
where $\mathbb{i}$ denotes the imaginary unit.

Using the equivalence between (i) and (iii), it is straightforward to show that

$$X_n \xrightarrow{\mathcal{D}} X \quad \iff \quad a^T X_n \xrightarrow{\mathcal{D}} a^T X \quad \text{for all } a \in \mathbb{R}^m. \tag{13}$$

In particular,

$$X_n \xrightarrow{\mathcal{D}} \mathcal{N}(0,\Sigma) \quad \iff \quad a^T X_n \xrightarrow{\mathcal{D}} \mathcal{N}(0, a^T\Sigma a) \quad \text{for all } a \in \mathbb{R}^m. \tag{14}$$

A.8.3 Sequential limit

In this section, we construct a Gaussian process using an infinite channel CNN with the limits taking sequentially. We will show that pre-activation functions of different channels are i.i.d. Gaussian with mean zero and covariance that can be computed recursively.

Let Ψ_1 denote the following space of functions:
Uniformly square-integrable: for every $R \geq 1$,

$$C_2\left(R, \phi\right) \equiv \sup_{1/R\leq r\leq R} \mathbb{E}_{x\sim\mathcal{N}(0,r)} \left|\phi(x)\right|^2 < \infty. \tag{15}$$

Let $\mathbb{N}^* = \mathbb{N}\backslash\{0\}$. We construct Gaussian processes recursively using an infinite width neural network architecture with the following procedure:

1. If $l > 0$, assume $\{z_i^{l-1,\infty}\}_{i\in\mathbb{N}^*}$ are i.i.d. Gaussian with mean zero and variance K_∞^{l-1} and independent from future events (i.e. independent from the σ-alegbra generated by all R.V.s with layer index greater than $(l-1)$). For $n \in \mathbb{N}^*$, let
$$y_{i,\alpha}^l(x) \equiv \begin{cases} x_{i,\alpha} & l = 0 \\ \phi\left(z_{i,\alpha}^{l-1,\infty}(x)\right) & l > 0 \end{cases}, \tag{16}$$
$$z_{i,\alpha}^{l,n}(x) \equiv \begin{cases} \frac{1}{\sqrt{n^0}} \sum_{j\in[n^0]} \frac{1}{\sqrt{2k+1}} \sum_{\beta\in[\pm k]} \omega_{ij,\beta}^l y_{j,\alpha+\beta}^l(x) + b_i^l & l = 0 \\ \frac{1}{\sqrt{n}} \sum_{j\in[n]} \frac{1}{\sqrt{2k+1}} \sum_{\beta\in[\pm k]} \omega_{ij,\beta}^l y_{j,\alpha+\beta}^l(x) + b_i^l & l > 0 \end{cases}, \tag{17}$$
where
$$[n] \equiv \{1,\dots,n\} \quad \text{and} \quad [\pm k] \equiv \{-k,\dots,0,\dots k\}. \tag{18}$$
2. Prove that for any $m \geq 1$, $\left[z_i^{l,n}\right]_{i\in[m]} \subseteq \mathbb{R}^{|\mathcal{X}|d}$ converges in distribution to a multivariate normal with mean zero and covariance $\mathcal{A}\left(K_\infty^l\right) \otimes I_m$, where K_∞^l is defined identically to Equation 6.

3. Define random variable $\left[z_i^{l,\infty}\right]_{i\in[m]}$ in $(\Omega, \mathcal{F}, P)$ with the same distribution as the limit in (ii) so that they are also independent from any future events, i.e. independent from the σ-algebra generated by all R.V.s with layer index greater than l.

In (iii), we implicitly assume the probability space $(\Omega, \mathcal{F}, P)$ has enough capacity to fit in all R.V.s generated by the above procedure, if not we will extend the sample space Ω using product measures. In what follows, we use the central limit theorem to prove (ii).

Theorem A.2. *If $\phi \in \Psi_1$, then for every $l \geq 0$ and every $m \geq 1$, $\left[z_i^{l,n}\right]_{i\in[m]} \subseteq \mathbb{R}^{|\mathcal{X}|d}$ converges in distribution to a multivariate normal with mean zero and covariance $\mathcal{A}\left(K_\infty^l\right) \otimes I_m$.*

Proof. We proceed by induction. This is obvious in the base case $l = 0$, since the weights and biases are assumed to be independent Gaussian. Now we assume the theorem holds for $l-1$. This implies $\{y_i^l\}_{i\in\mathbb{N}^*} = \left\{\phi(z_i^{l-1,\infty})\right\}_{i\in\mathbb{N}^*}$ are i.i.d. random variables.

Choose a vector $a \equiv [a_i(x)]^T_{i\in[m],\, x\in\mathcal{X}} \in \mathbb{R}^{m|\mathcal{X}|}$. Then

$$\sum_{\substack{i\in[m]\\x\in\mathcal{X}}} a_i(x) z_i^{l,n}(x) = \sum_{\substack{i\in[m]\\x\in\mathcal{X}}} a_i(x) \left(\frac{1}{\sqrt{n}} \sum_{j\in[n]} \frac{1}{\sqrt{2k+1}} \sum_{\beta\in[\pm k]} \omega_{ij,\beta}^l y_{j,\alpha+\beta}^l(x) + b_i^l \right) \tag{19}$$

$$= \frac{1}{\sqrt{n}} \sum_{j\in[n]} \sum_{\substack{i\in[m]\\x\in\mathcal{X}}} a_i(x) \frac{\sum_{\beta\in[\pm k]} \omega_{ij,\beta}^l y_{j,\alpha+\beta}^l(x)}{\sqrt{2k+1}} + \sum_{\substack{i\in[m]\\x\in\mathcal{X}}} a_i(x) b_i^l \tag{20}$$

$$\equiv \frac{1}{\sqrt{n}} \sum_{j\in[n]} u_j + v. \tag{21}$$

It is not difficult to see that u_j are i.i.d. and v is Gaussian. Then we can use the central limit theorem to conclude that the above converges in distribution to a Gaussian, once we verify the second moment of u_j is finite. Using the fact that $\left\{\omega_{ij,\beta}^l\right\}_{i,\beta}$ is a collection of independent R.V.s and integrating over these R.V.s first, we get

$$\mathbb{E}u_j^2 = \frac{1}{2k+1} \mathbb{E}\left[\sum_{i\in[m]\,,\, x\in\mathcal{X}} a_i(x) \sum_{\beta\in[\pm k]} \omega_{ij,\beta}^l y_{j,\alpha+\beta}^l(x) \right]^2 \tag{22}$$

$$= \frac{\sigma_\omega^2}{2k+1} \sum_{i\in[m]} \sum_{\beta\in[\pm k]} \mathbb{E}\left[\sum_{x\in\mathcal{X}} a_i(x) y_{j,\alpha+\beta}^l(x) \right]^2 \tag{23}$$

$$= \sum_{i\in[m]} a_i^T \bar{\mathcal{A}}\left(K_\infty^l\right) a_i \tag{24}$$

where by $\bar{\mathcal{A}}$ we denote the linear transformation part of $\mathcal{A}$ from §2.2, i.e. $\mathcal{A}$ without the translation term σ_b^2:

$$\left[\bar{\mathcal{A}}(K)\right]_{\alpha,\alpha'}(x,x') \equiv \frac{\sigma_\omega^2}{2k+1} \sum_{\beta} [K]_{\alpha+\beta,\alpha'+\beta}(x,x'). \tag{25}$$

To prove finiteness of the second moment in Equation 24, it is sufficient to show that all the diagonal terms of K_∞^l are finite. This easily follows from the assumption $\phi \in \Psi_1$ and the definition of $K_\infty^l = (\mathcal{C}\circ\mathcal{A})^l\left(K^0\right)$. Together with the distribution of v (whose covariance is straightforward to compute), the joint distribution of $\left[z_i^{l,n}\right]_{i\in[m]}$ converges weakly to a mean zero Gaussian with covaraince matrix $\mathcal{A}\left(K_\infty^l\right) \otimes I_m$ by Theorem A.1 (Equation 14). □

Remark A.1. *The results of Theorem A.2 can be strengthened / extended in many directions.*

1. *Same analysis carries over (and the covariance matrix can be computed without much extra effort) if we stack a channel-wise deterministic affine transform after the convolution operator. Note that average pooling (not max pooling, which is not affine), global average pooling and the convolutional striding are particular examples of such affine tranforms. Moreover, valid padding (i.e. no padding) convolution can be regarded as subsampling (a linear projection) operator composed with the regular circular padding.*

2. *Same analysis applies to max pooling, but computing the covariance may require non-trivial effort. Let $\mathfrak{m}$ denote the max pooling operator and assume it is applied right after the activation function. The assumption $\left\{y_i^l\right\}_{i\in[n]} = \left\{\phi(z_i^{l-1,\infty})\right\}_{i\in[n]}$ are i.i.d. implies $\left\{\mathfrak{m}\left(y_i^l\right)\right\}_{i\in[n]}$ are also i.i.d. Then we can proceed exactly as above except for verifying the finiteness of second moment of $\mathfrak{m}(y_i^l)$ with the following trivial estimate:*

$$\mathbb{E}\max\left\{y_{i,\alpha}^l\right\}_{\alpha\in[s]}^2 \leq \mathbb{E}\sum_{\alpha\in[s]}\left|y_{i,\alpha}^l\right|^2 \tag{26}$$

where s is the window size of the max pooling.

In general, one can stack a channel-wise deterministic operator $\mathfrak{op}$ on $\{y_i^l\}$ so long as the second moment of $\{\mathfrak{op}(y_i^l)\}$ is finite. One can also stack a stochastic operator (e.g. dropout), so long as the outputs are still channel-wisely i.i.d. and have finite second moments.

A.8.4 Simultaneous limit

In this section, we present a sufficient condition on the activation function ϕ so that the neural networks converge to a Gaussian process as all the widths approach to infinity simultaneously. Precisely, let $t \in \mathbb{N}$ and for each $l \geq 0$, let $n^l : \mathbb{N} \to \mathbb{N}$ be the width function at layer l (by convention $n^0(t) = n^0$). We are interested in the simultaneous limit $n^l = n^l(t) \to \infty$ as $t \to \infty$, i.e., for any fixed $L \geq 1$

$$\min\left\{n^1(t), \ldots, n^L(t)\right\} \xrightarrow[t\to\infty]{} \infty. \tag{27}$$

Define a sequence of finite channel CNNs as follows:

$$y_{i,\alpha}^{l,t}(x) \equiv \begin{cases} x_{i,\alpha} & l = 0 \\ \phi\left(z_{i,\alpha}^{l-1,t}(x)\right) & l > 0 \end{cases}, \tag{28}$$

(29)

$$z_{i,\alpha}^{l,t}(x) \equiv \begin{cases} \frac{1}{\sqrt{n^0}}\sum_{j\in[n^0]}\frac{1}{\sqrt{2k+1}}\sum_{\beta\in[\pm k]}\omega_{ij,\beta}^l y_{j,\alpha+\beta}^{l,t}(x) + b_i^l\,, & l = 0 \\ \frac{1}{\sqrt{n^l(t)}}\sum_{j\in[n^l(t)]}\frac{1}{\sqrt{2k+1}}\sum_{\beta\in[\pm k]}\omega_{ij,\beta}^l y_{j,\alpha+\beta}^{l,t}(x) + b_i^l\,, & l > 0 \end{cases}. \tag{30}$$

This network induces a sequence of covariances matrices K_t^l (which are R.V.s): for $l \geq 0$ and $t \geq 0$, for $x, x' \in \mathcal{X}$

$$\left[K_t^l\right]_{\alpha,\alpha'}(x,x') \equiv \frac{1}{n^l(t)}\sum_{i=1}^{n^l(t)} y_{i,\alpha}^{l,t}(x) y_{i,\alpha'}^{l,t}(x'). \tag{31}$$

We make an extra assumption on the parameters.

Assumption: all R.V.s in $\mathscr{W}$ are Gaussian distributed.

Notation. Let PSD_m denote the set of $m \times m$ positive semi-definite matrices and for $R \geq 1$, define

$$\text{PSD}_m(R) \equiv \left\{\Sigma \in \text{PSD}_m : 1/R \leq \Sigma_{\alpha,\alpha} \leq R \quad \text{for} \quad 1 \leq \alpha \leq m\right\}. \tag{32}$$

Further let $\mathcal{T}_\infty : \text{PSD}_2 \to \mathbb{R}$ be a function given by

$$\mathcal{T}_\infty(\Sigma) \equiv \mathbb{E}_{(x,y)\sim\mathcal{N}(0,\Sigma)}\left[\phi(x)\phi(y)\right], \tag{33}$$

and $C_k(\phi, R)$ (may equal to ∞) denotes the uniform upper bound for the k-th moment

$$C_k(\phi, R) \equiv \sup_{1/R \le r \le R} \mathbb{E}_{x \sim \mathcal{N}(0,r)} |\phi(x)|^k . \tag{34}$$

Let Ψ denotes the space of measurable functions with the following properties:

1. **Uniformly square-integrable:** $C_2(\phi, R) < \infty$.
2. **Lipschitz continuity:** for every $R \ge 1$, there exists $\beta = \beta(\phi, R) > 0$ such that for all $\Sigma, \Sigma' \in \mathrm{PSD}_2(R)$,
$$|\mathcal{T}_\infty(\Sigma) - \mathcal{T}_\infty(\Sigma')| \le \beta \|\Sigma - \Sigma'\|_\infty ; \tag{35}$$
3. **Uniform convergence in probability:** for every $R \ge 1$ and every $\varepsilon > 0$ there exists a positive sequence $\rho_n(\phi, \varepsilon, R)$ with $\rho_n(\phi, \varepsilon, R) \to 0$ as $n \to \infty$ such that for every $\Sigma \in \mathrm{PSD}_2(R)$ and any $\{(x_i, y_i)\}_{i=1}^n$ i.i.d. $\sim \mathcal{N}(0, \Sigma)$
$$P\left(\left| \frac{1}{n} \sum_{i=1}^n \phi(x_i)\phi(y_i) - \mathcal{T}_\infty(\Sigma) \right| > \varepsilon \right) \le \rho_n(\phi, \varepsilon, R). \tag{36}$$

We will also use Ψ_1, Ψ_2 and Ψ_3 to denote the spaces of functions satisfying property 1, property 2 and property 3, respectively. It is not difficult to see that for every i, Ψ_i is a vector space, and so is $\Psi = \cap_i \Psi_i$.

Finally, we say that a function $f : \mathbb{R} \to \mathbb{R}$ is exponentially bounded if there exist $a, b > 0$ such that

$$|f(x)| \le a e^{b|x|} \quad \text{a.e. (almost everywhere)} \tag{37}$$

We now prove our main result presented in §2.2 through the following three theorems.

Theorem A.3. *If ϕ' is exponentially bounded then $\phi \in \Psi$.*

Theorem A.4. *If $\phi \in \Psi$, then for $l \ge 0$, $K_t^l \xrightarrow{P} K_\infty^l$.*

Theorem A.5. *If for $l \ge 0$, $K_t^l \xrightarrow{P} K_\infty^l$ and $m \ge 1$, the joint distribution of $\left[z_j^{l,t}\right]_{j \in [m]}$ converges in distribution to a multivariate normal distribution with mean zero and covariance $\mathcal{A}\left(K_\infty^l\right) \otimes I_m$.*

The proofs of Theorems A.3, A.4, and A.5 can be found in §A.8.7, §A.8.6, and §A.8.5 respectively.

A.8.5 Proof of Theorem A.5

It suffices to prove that for any vector $[a_i(x)]_{i \in [m],\, x \in \mathcal{X}} \in \mathbb{R}^{m|\mathcal{X}|}$,

$$\sum_{\substack{i \in [m] \\ x \in \mathcal{X}}} a_i(x) z_i^{l,t}(x) \xrightarrow{\mathcal{D}} \mathcal{N}\left(0, \sum_{i \in [m],\, x \in \mathcal{X}} a_i(x)^T \mathcal{A}\left(K_\infty^l\right) a_i(x)\right). \tag{38}$$

Indeed, the characteristic function

$$\mathbb{E} \exp\left(\mathbb{i} \sum_{i \in [m],\, x \in \mathcal{X}} a_i(x) z_i^{l,t}(x) \right) \tag{39}$$

$$= \mathbb{E} \exp\left(\mathbb{i} \sum_{\substack{i \in [m] \\ x \in \mathcal{X}}} a_i(x) \left(\frac{1}{\sqrt{n^l(t)}} \sum_{j \in [n^l(t)]} \frac{1}{\sqrt{2k+1}} \sum_{\beta \in [\pm k]} \omega_{ij,\beta}^l y_{j,\alpha+\beta}^{l,t}(x) + b_i^l \right) \right). \tag{40}$$

Note that conditioned on $\left\{y_j^{l,t}\right\}_{j \in [n^l(t)]}$, the exponent in the above expression is just a linear combination of independent Gaussian R.V.s $\left\{\omega_{ij,\beta}^l,\, b_i^l\right\}$, which is also a Gaussian. Integrating

out these R.V.s using the formula of the characteristic function of Gaussian distribution yields

$$\mathbb{E}\exp\left(\mathbb{i}\sum_{i\in[m],\,x\in\mathcal{X}} a_i(x)z_i^{l,t}(x)\right) \tag{41}$$

$$=\mathbb{E}\exp\left(-\frac{1}{2}\sum_{i\in[m]} a_i^T\left(\mathcal{A}\left(K_t^l\right)\right)a_i\right) \tag{42}$$

$$\longrightarrow\exp\left(-\frac{1}{2}\sum_{i\in[m]} a_i^T\left(\mathcal{A}\left(K_\infty^l\right)\right)a_i\right)\quad\text{as}\quad t\to\infty\,, \tag{43}$$

where have used $K_t^l \xrightarrow{P} K_\infty^l$ in the last step. Therefore, Equation 38 is true by Theorem A.1 (iii).

□

Remark A.2. *We briefly comment how to handle the cases when stacking an average pooling, a subsampling or a dense layer after flattening the activations in the last layer.*

1. ***Global Average pooling / subsampling.*** *Let $B\in\mathbb{R}^{1\times d}$ be any deterministic linear functional defined on $\mathbb{R}^d$. The fact that*

$$K_t^l \xrightarrow{P} K_\infty^l \tag{44}$$

implies that the empirical covariance of $\left\{By_j^{l,t}\right\}$

$$\frac{1}{n^l(t)}\sum_{j\in n^l(t)} B^{\otimes|\mathcal{X}|}y_j^{l,t}\left(B^{\otimes|\mathcal{X}|}y_j^{l,t}\right)^T \xrightarrow{P} B^{\otimes|\mathcal{X}|}K_\infty^l\left(B^{\otimes|\mathcal{X}|}\right)^T \tag{45}$$

where $B^{\otimes|\mathcal{X}|}\in\mathbb{R}^{1\times d|\mathcal{X}|}$, $|\mathcal{X}|$ copies of B. Invoking the same "characteristic function" arguments as above, it is not difficult to show that stacking a dense layer (assuming the weights and biases are drawn from i.i.d. Gaussian with mean zero and variances σ_ω^2 and σ_b^2, and are properly normalized) on top of $\left\{By_j^{l,t}\right\}$ the outputs are i.i.d. Gaussian with mean zero and covariance $\sigma_\omega^2 B^{\otimes|\mathcal{X}|}K_\infty^l\left(B^{\otimes|\mathcal{X}|}\right)^T+\sigma_b^2$.

Taking $B=\left(\frac{1}{d},\ldots,\frac{1}{d}\right)\in\mathbb{R}^{1\times d}$ $\left(\text{or } B=e_\alpha\in\mathbb{R}^{1\times d}\right)$ implies the result of global average pooling (§2.3.2.1), or subsampling (§2.3.2.2).

2. ***Vectorization and a dense layer.*** *Let $\left\{\omega_{ij,\alpha}^l\right\}_{i\in[m],j\in[n^l(t)],\alpha\in[d]}$ be the weights of the dense layer, $\omega_{ij,\alpha}^l$ represents the weight connecting the α-th pixel of the j-channel to the i-th output. Note that the range of α is $[d]$ not $[\pm k]$ because there is no weight sharing. Define the outputs to be*

$$f_i(x)=\frac{1}{\sqrt{dn^l(t)}}\sum_{\alpha\in[d]}\sum_{j\in[n^l(t)]}\omega_{ij,\alpha}^l y_{j,\alpha}^{l,t}(x)+b_i^l \tag{46}$$

Now let $[a_i(x)]_{i\in[m],x\in\mathcal{X}}\in\mathbb{R}^{m|\mathcal{X}|}$ and compute the characteristic function of

$$\sum_{i,x} a_i(x)f_i(x). \tag{47}$$

Using the fact $\mathbb{E}\omega_{ij,\alpha}\omega_{i'j',\alpha'} = 0$ unless $(ij,\alpha) = (i'j',\alpha')$ and integrating out the R.V.s of the dense layer, the characteristic function is equal to

$$\mathbb{E}\exp\left(-\frac{1}{2}\sum_{i\in[m]}\sum_{x,x'\in\mathcal{X}} a_i(x)a_i(x')\left(\frac{1}{dn^l(t)}\sum_{\substack{j\in[n^l(t)]\\ \alpha\in[d]}}\sigma_\omega^2 y_{j,\alpha}^{l,t}(x)y_{j,\alpha}^{l,t}(x') + \sigma_b^2\right)\right) \tag{48}$$

$$= \mathbb{E}\exp\left(-\frac{1}{2}\sum_{i\in[m]}\sum_{x,x'\in\mathcal{X}} a_i(x)a_i(x')\bar{tr}\left(\sigma_\omega^2 K_t^l(x,x') + \sigma_b^2\right)\right) \tag{49}$$

$$\longrightarrow \exp\left(-\frac{1}{2}\sum_{i\in[m]}\sum_{x,x'\in\mathcal{X}} a_i(x)a_i(x')\bar{tr}\left(\sigma_\omega^2 K_\infty^l(x,x') + \sigma_b^2\right)\right), \tag{50}$$

where $\bar{tr}$ denotes the mean trace operator acting on the pixel by pixel matrix, i.e. the functional computing the mean of the diagonal terms of the pixel by pixel matrix. Therefore $[f_i]_{i\in[m]}$ converges weakly to a mean zero Gaussian with covariance $\left[\sigma_\omega^2\bar{tr}\left(K_\infty^l(x,x')\right) + \sigma_b^2\right]_{x,x'\in\mathcal{X}} \otimes I_m$ in the case of vectorization (§2.3.1).

A.8.6 Proof of Theorem A.4

Proof. We recall K_t^L and K_∞^L to be random matrices in $\mathbb{R}^{d|\mathcal{X}|\times d|\mathcal{X}|}$, and we will prove convergence $K_t^L \xrightarrow[t\to\infty]{P} K_\infty^L$ with respect to $\|\cdot\|_\infty$, the pointwise ℓ^∞-norm (i.e. $\|K\|_\infty = \max_{x,x',\alpha,\alpha'}|K_{\alpha,\alpha'}(x,x')|$). Note that due to finite dimensionality of K^L, convergence w.r.t. all other norms follows.

We first note that the affine transform $\mathcal{A}$ is σ_ω^2-Lipschitz and property 2 of Ψ implies that the $\mathcal{C}$ operator is β-Lipschitz. Indeed, if we consider

$$\Sigma \equiv \begin{pmatrix}[K]_{\alpha,\alpha}(x,x) & [K]_{\alpha,\alpha'}(x,x')\\ [K]_{\alpha',\alpha}(x',x) & [K]_{\alpha',\alpha'}(x',x')\end{pmatrix}, \tag{51}$$

then $[\mathcal{C}(K)]_{\alpha,\alpha'}(x,x') = \mathcal{T}_\infty(\Sigma)$. Thus $\mathcal{C}\circ\mathcal{A}$ is $\sigma_\omega^2\beta$-Lipschitz.

We now prove the theorem by induction. Assume $K_t^l \xrightarrow{P} K_\infty^l$ as $t\to\infty$ (obvious for $l=0$).

Let $\varepsilon > 0$ be sufficiently small so that the $\frac{\varepsilon}{2\beta}$-neighborhood of $\mathcal{A}(K_\infty^l)$ is contained in $\mathrm{PSD}_{|\mathcal{X}|d}(R)$, where we take R to be large enough for K_∞^l to be an interior point of $\mathrm{PSD}_{|\mathcal{X}|d}(R)$.

Since

$$\left\|K_\infty^{l+1} - K_t^{l+1}\right\|_\infty \le \left\|K_\infty^{l+1} - \mathcal{C}\circ\mathcal{A}\left(K_t^l\right)\right\|_\infty + \left\|\mathcal{C}\circ\mathcal{A}\left(K^l\right) - K_t^{l+1}\right\|_\infty \tag{52}$$

$$= \left\|\mathcal{C}\circ\mathcal{A}\left(K_\infty^l\right) - \mathcal{C}\circ\mathcal{A}\left(K_t^l\right)\right\|_\infty + \left\|\mathcal{C}\circ\mathcal{A}\left(K_t^l\right) - K_t^{l+1}\right\|_\infty, \tag{53}$$

to prove $K_t^{l+1} \xrightarrow{P} K_\infty^{l+1}$, it suffices to show that for every $\delta > 0$, there is a t^* such that for all $t > t^*$,

$$P\left(\left\|\mathcal{C}\circ\mathcal{A}\left(K_\infty^l\right) - \mathcal{C}\circ\mathcal{A}\left(K_t^l\right)\right\|_\infty > \frac{\varepsilon}{2}\right) + P\left(\left\|\mathcal{C}\circ\mathcal{A}\left(K_t^l\right) - K_t^{l+1}\right\|_\infty > \frac{\varepsilon}{2}\right) < \delta. \tag{54}$$

By our induction assumption, there is a t^l such that for all $t > t^l$

$$P\left(\left\|K_\infty^l - K_t^l\right\|_\infty > \frac{\varepsilon}{2\sigma_\omega^2\beta}\right) < \frac{\delta}{3}. \tag{55}$$

Since $\mathcal{C}\circ\mathcal{A}$ is $\sigma_\omega^2\beta$-Lipschitz, then

$$P\left(\left\|\mathcal{C}\circ\mathcal{A}\left(K_\infty^l\right) - \mathcal{C}\circ\mathcal{A}\left(K_t^l\right)\right\|_\infty > \frac{\varepsilon}{2}\right) < \frac{\delta}{3}. \tag{56}$$

To bound the second term in Equation 54, let $U(t)$ denotes the event

$$U(t) \equiv \left\{\mathcal{A}\left(K_t^l\right) \in \mathrm{PSD}_{|\mathcal{X}|d}(R)\right\} \tag{57}$$

and $U(t)^c$ its complement. For all $t > t^l$ it's probability is

$$\begin{aligned} P(U(t)^c) &< P\left(\left\|\mathcal{A}\left(K_\infty^l\right) - \mathcal{A}\left(K_t^l\right)\right\|_\infty > \frac{\varepsilon}{2\beta}\right) && \text{[assumption on small } \varepsilon\text{]} && (58)\\ &< P\left(\sigma_\omega^2 \left\|K_\infty^l - K_t^l\right\|_\infty > \frac{\varepsilon}{2\beta}\right) && \left[\mathcal{A} \text{ is } \sigma_\omega^2\text{-Lipshitz}\right] && (59)\\ &= P\left(\left\|K_\infty^l - K_t^l\right\|_\infty > \frac{\varepsilon}{2\sigma_\omega^2\beta}\right) < \frac{\delta}{3}. && \text{[Equation 55]} && (60)\end{aligned}$$

Finally, denote

$$[V(t)]_{\alpha,\alpha'}(x,x') \equiv \left\{\left|\left[\mathcal{C}\circ\mathcal{A}\left(K_t^l\right)\right]_{\alpha,\alpha'}(x,x') - \left[K_t^{l+1}\right]_{\alpha,\alpha'}(x,x')\right| > \frac{\varepsilon}{2}\right\}. \tag{61}$$

The fact

$$\left\{\left\|\mathcal{C}\circ\mathcal{A}\left(K_t^l\right) - K_t^{l+1}\right\|_\infty > \frac{\varepsilon}{2}\right\} \subseteq U(t)^c \bigcup \left(\bigcup_{x,x',\alpha,\alpha'} [V(t)]_{\alpha,\alpha'}(x,x') \bigcap U(t)\right)$$

implies

$$P\left(\left\{\left\|\mathcal{C}\circ\mathcal{A}\left(K_t^l\right) - K^{l+1}(t)\right\|_\infty > \frac{\varepsilon}{2}\right\}\right) \leq \frac{\delta}{3} + |\mathcal{X}|^2 d^2 \max_{x,x',\alpha,\alpha'} P\left([V(t)]_{\alpha,\alpha'}(x,x') \cap U(t)\right), \tag{62}$$

Where the maximum is taken over all $(x,x',\alpha,\alpha') \in \mathcal{X}^2 \times [d]$.

Consider a fixed $\kappa \in \mathrm{PSD}_{|\mathcal{X}|d}$, and define

$$\Sigma(\kappa,\alpha,\alpha',x,x') \equiv \begin{pmatrix} [\mathcal{A}(\kappa)]_{\alpha,\alpha}(x,x) & [\mathcal{A}(\kappa)]_{\alpha,\alpha'}(x,x') \\ [\mathcal{A}(\kappa)]_{\alpha',\alpha}(x',x) & [\mathcal{A}(\kappa)]_{\alpha',\alpha'}(x',x') \end{pmatrix}, \tag{63}$$

a deterministic matrix in PSD_2. Then

$$[\mathcal{C}\circ\mathcal{A}(\kappa)]_{\alpha,\alpha'}(x,x') = \mathcal{T}_\infty(\Sigma(\kappa,\alpha,\alpha',x,x')), \tag{64}$$

and, conditioned on K_t^l,

$$\left[K_t^{l+1}|K_t^l = \kappa\right]_{\alpha,\alpha'}(x,x') = \frac{1}{n^{l+1}(t)} \sum_{i=1}^{n^{l+1}(t)} \phi\left(z_{i,\alpha}^{l,t}(x)\right)\phi\left(z_{i,\alpha'}^{l,t}(x')\right), \tag{65}$$

where $\left\{\left(z_{i,\alpha}^{l,t}(x), z_{i,\alpha'}^{l,t}(x')\right)|K_t^l = \kappa\right\}_{i\in[n^{l+1}(t)]}$ are i.i.d. $\sim \mathcal{N}(0,\Sigma(\kappa,\alpha,\alpha',x,x'))$.

Then if $\Sigma(\kappa,\alpha,\alpha',x,x') \in \mathrm{PSD}_2(R)$ we can apply property 3 of Ψ to conclude that:

$$P\left(\left([V(t)]_{\alpha,\alpha'}(x,x') \cap U(t)\right)\middle|K_t^l = \kappa\right) < \rho_{n^{l+1}(t)}\left(\phi, R, \frac{\varepsilon}{2}\right). \tag{66}$$

However, if $\Sigma(\kappa,\alpha,\alpha',x,x') \notin \mathrm{PSD}_2(R)$, then necessarily $\mathcal{A}(\kappa) \notin \mathrm{PSD}_{|\mathcal{X}|d}(R)$ (since $\Sigma(\kappa,\alpha,\alpha',x,x') \in \mathrm{PSD}_2$), ensuring that $P\left(U(t)|K_t^l = \kappa\right) = 0$. Therefore Equation 66 holds for any $\kappa \in \mathrm{PSD}_{|\mathcal{X}|d}$, and for any $(x,x',\alpha,\alpha') \in \mathcal{X}^2 \times [d]^2$.

We further remark that $\rho_{n^{l+1}(t)}\left(\phi, R, \frac{\varepsilon}{2}\right)$ is deterministic and does not depend on $(\kappa,x,x',\alpha,\alpha')$. Marginalizing out K_t^l and maximizing over (x,x',α,α') in Equation 66 we conclude that

$$\max_{x,x',\alpha,\alpha'} P\left([V(t)]_{\alpha,\alpha'}(x,x') \cap U(t)\right) < \rho_{n^{l+1}(t)}\left(\phi, R, \frac{\varepsilon}{2}\right). \tag{67}$$

Since $\rho_n\left(\phi, R, \frac{\varepsilon}{2}\right) \to 0$ as $n \to \infty$, there exists n such that for any $n^{l+1}(t) \geq n$,

$$\max_{x,x',\alpha,\alpha'} P\left([V(t)]_{\alpha,\alpha'}(x,x') \cap U(t)\right) < \rho_{n^{l+1}(t)}\left(\phi, R, \frac{\varepsilon}{2}\right) \leq \frac{\delta}{3|\mathcal{X}|^2 d^2}, \tag{68}$$

and, substituting this bound in Equation 62,

$$P\left(\left\{\left\|\mathcal{C} \circ \mathcal{A}\left(K_t^l\right) - K_t^{l+1}\right\|_\infty > \frac{\varepsilon}{2}\right\} \cap U(t)\right) < \frac{2\delta}{3}. \tag{69}$$

Therefore we just need to choose $t^{l+1} > t^l$ so that $n^{l+1}(t) \geq n$ for all $t > t^{l+1}$.

□

Remark A.3. *We list some directions to strengthen / extent the results of Theorem A.4 (and thus Theorem A.5) using the above framework.*

1. *Consider stacking a deterministic channel-wise linear operator right after the convolutional layer. Again, strided convolution, convolution with no ("valid") padding and (non-global) average pooling are particular examples of this category. Let $B \in \mathbb{R}^{d' \times d}$ denote a linear operator. Then the recurrent formula between two consecutive layers is*

$$K_\infty^{l+1} = \mathcal{C} \circ \mathcal{B} \circ \mathcal{A}(K_\infty^l) \tag{70}$$

   where $\mathcal{B}$ is the linear operator on the covariance matrix induced by B. Conditioned on K_t^l, since the outputs after applying the linear operator B are still i.i.d. Gaussian (and the property 3 is applicable), the analysis in the above proof can carry over with $\mathcal{A}$ replaced by $\mathcal{B} \circ \mathcal{A}$.

2. *More generally, one may consider inserting an operator $\mathfrak{op}$ (e.g. max-pooling, dropout and more interestingly, normalization) in some hidden layer.*

3. *It is possible to drop the Gaussianity assumption on the weight parameters by incorporating the proof of the central limit theorem using characteristic function into the above arguments.*

A.8.7 Proof of Theorem A.3

Note that exponentially bounded functions contain all polynomials, and are closed under multiplication and integration in the sense that for any constant C the function

$$\int_0^x \phi(t)dt + C \tag{71}$$

is also exponentially bounded. Theorem A.3 is a consequence of the following lemma.

Lemma A.6. *The following is true:*

1. *for $k \geq 1$, $C_k(\phi, R) < \infty$ if ϕ is exponentially bounded.*
2. *$\phi \in \Psi_2$ if ϕ' exists a.e. and is exponentially bounded.*
3. *$\phi \in \Psi_3$ if $C_4(\phi, R) < \infty$.*

Indeed, if ϕ' is exponentially bounded, then ϕ is also exponentially bounded. By the above lemma, $\phi \in \Psi$.

Proof of Lemma A.6. **1.** We prove the first statement. Assume $|\phi(x)| \leq ae^{b|x|}$.

$$\mathbb{E}_{x\sim\mathcal{N}(0,r)}|\phi(x)|^k = \mathbb{E}_{x\sim\mathcal{N}(0,1)}\left|\phi\left(\sqrt{r}x\right)\right|^k \leq \mathbb{E}_{x\sim\mathcal{N}(0,1)}\left|ae^{\sqrt{r}b|x|}\right|^k \leq 2a^k e^{k^2b^2r/2}. \tag{72}$$

Thus

$$C_k(\phi, R) = \sup_{1/R\leq r\leq R} \mathbb{E}_{x\sim\mathcal{N}(0,r)}|\phi(x)|^k \leq 2a^k e^{k^2b^2R/2}. \tag{73}$$

2. To prove the second statement, let $\Sigma, \Sigma' \in \mathrm{PSD}_2(R)$ and define A (similarly for A'):

$$A \equiv \begin{pmatrix} \sqrt{\Sigma_{11}} & 0 \\ \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}}} & \sqrt{\frac{\Sigma_{22}\Sigma_{11}-\Sigma_{12}^2}{\Sigma_{11}}} \end{pmatrix}. \tag{74}$$

Then $AA^T = \Sigma$ (and $A'A'^T = \Sigma'$). Let

$$A(t) \equiv (1-t)A + tA', \quad t \in [0,1] \tag{75}$$

and

$$f(w) \equiv \phi(x)\phi(y) \quad \text{where} \quad w \equiv (x,y)^T. \tag{76}$$

Since ϕ' is exponentially bounded, ϕ is also exponentially bounded. In addition, $p\left(\|w\|_2\right) \|\nabla f(w)\|_2$ is exponentially bounded for any polynomial $p\left(\|w\|_2\right)$.

Applying the Mean Value Theorem (we use the notation $\lesssim$ to hide the dependence on R and other absolute constants)

$$\begin{aligned}
|\mathcal{T}_\infty(\Sigma) - \mathcal{T}_\infty(\Sigma')| =& \frac{1}{2\pi}\left|\int \left(f\left(Aw\right) - f\left(A'w\right)\right) \exp\left(-\|w\|_2^2/2\right) dw\right| && (77)\\
=& \frac{1}{2\pi}\left|\int\int_{[0,1]} \left(\nabla f\left(A(t)w\right)\right)\left(\left(A'-A\right)w\right) \exp\left(-\|w\|_2^2/2\right) dt dw\right| && (78)\\
\lesssim& \int_{[0,1]} \int \left\|\left(A'-A\right)w\right\|_2 \left\|\nabla f\left(A(t)w\right)\right\|_2 \exp\left(-\|w\|_2^2/2\right) dw dt && (79)\\
\leq& \int_{[0,1]} \int \left\|A'-A\right\|_{\mathrm{op}} \|w\|_2 \left\|\nabla f\left(A(t)w\right)\right\|_2 \exp\left(-\|w\|_2^2/2\right) dw dt. && (80)
\end{aligned}$$

Note that the operator norm is bounded by the infinity norm (up to a multiplicity constant) and $\|w\|_2 \left\|\nabla f\left(A(t)w\right)\right\|_2$ is exponentially bounded. There is a constant a (hidden in $\lesssim$) and b such that the above is bounded by

$$\begin{aligned}
&\int_{[0,1]} \int \left\|A'-A\right\|_\infty \exp\left(b\left\|A(t)\right\|_\infty \|w\|_2\right) \exp\left(-\|w\|_2^2/2\right) dw dt && (81)\\
\lesssim& \left\|A'-A\right\|_\infty \int_{[0,1]} \int \exp\left(b\sqrt{R}\|w\|_2 - \|w\|_2^2/2\right) dw dt && (82)\\
\lesssim& \left\|A'-A\right\|_\infty && (83)\\
\lesssim& \left\|\Sigma'-\Sigma\right\|_\infty. && (84)
\end{aligned}$$

Here we have applied the facts

$$\left\|A'-A\right\|_\infty \lesssim \left\|\Sigma-\Sigma'\right\|_\infty \quad \text{and} \quad \left\|A(t)\right\|_\infty \leq \sqrt{R}. \tag{85}$$

3. Chebyshev's inequality implies

$$\begin{aligned}
&P\left(\left|\frac{1}{n}\sum_{i=1}^n \phi\left(x_i\right)\phi\left(y_i\right) - \mathcal{T}_\infty\left(\Sigma\right)\right| > \varepsilon\right) && (86)\\
\leq& \frac{1}{n\epsilon^2}\mathrm{Var}\left(\phi\left(x_i\right)\phi\left(y_i\right)\right) \leq \frac{1}{n\epsilon^2}\mathbb{E}\left|\phi\left(x_i\right)\phi\left(y_i\right)\right|^2 && (87)\\
\leq& \frac{1}{n\epsilon^2} C_4\left(\phi, R\right) \to 0 \quad \text{as} \quad n \to \infty. && (88)
\end{aligned}$$

□

Remark A.4. *In practice the $1/n$ decay bound obtained by Chebyshev's inequality in Equation 88 is often too weak to be useful. However, if ϕ is linearly bounded, then one can obtain an exponential decay bound via the following concentration inequality:*

Lemma A.7. *If $|\phi(x)| \leq a + b|x|$ a.e., then there is an absolute constant $c > 0$ and a constant $\kappa = \kappa(a, b, R) > 0$ such that property 3 (Equation 36) holds with*

$$\rho_n(\phi, \epsilon, R) = 2 \exp\left(-c \min\left\{\frac{n^2\varepsilon^2}{(2\kappa)^2}, \frac{n\varepsilon}{2\kappa}\right\}\right). \tag{89}$$

Proof. We postpone the proof of the following claim.

Claim A.8. *Assume $|\phi(x)| \leq a+b|x|$. Then there is a $\kappa = \kappa(a, b, R)$ such that for all $\Sigma \in \mathrm{PSD}_2(R)$ and all $p \geq 1$,*

$$\left(\mathbb{E}_{(x,y)\sim\mathcal{N}(0,\Sigma)}|\phi(x)\phi(y)|^p\right)^{1/p} \leq \kappa p. \tag{90}$$

Claim A.8 and the triangle inequality imply

$$\left(\mathbb{E}_{(x,y)\sim\mathcal{N}(0,\Sigma)}|\phi(x)\phi(y) - \mathbb{E}\phi(x)\phi(y)|^p\right)^{1/p} \leq 2\kappa p. \tag{91}$$

We can apply Bernstein-type inequality [40, Lemma 5.16] to conclude that there is a $c > 0$ such that for every $\Sigma \in \mathrm{PSD}_2(R)$ and any $\{(x_i, y_i)\}_{i=1}^n$ i.i.d. $\sim \mathcal{N}(0, \Sigma)$

$$P\left(\left|\frac{1}{n}\sum_{i=1}^n \phi(x_i)\,\phi(y_i) - \mathcal{T}_\infty(\Sigma)\right| > \varepsilon\right) \leq 2\exp\left(-c\min\left\{\frac{n^2\varepsilon^2}{(2\kappa)^2}, \frac{n\varepsilon}{2\kappa}\right\}\right). \tag{92}$$

It remains to prove Claim A.8. For $p \geq 1$,

$$\begin{aligned}
\left(\mathbb{E}_{(x,y)\sim\mathcal{N}(0,\Sigma)}|\phi(x)\phi(y)|^p\right)^{1/p} &\leq \left(\mathbb{E}_{x\sim\mathcal{N}(0,\Sigma_{11})}|\phi(x)|^{2p}\right)^{1/2p}\left(\mathbb{E}_{y\sim\mathcal{N}(0,\Sigma_{22})}|\phi(y)|^{2p}\right)^{1/2p} && (93)\\
&\leq \left(a + b\left(\mathbb{E}|x|^{2p}\right)^{1/2p}\right)\left(a + b\left(\mathbb{E}|y|^{2p}\right)^{1/2p}\right) && (94)\\
&\leq \left(a + b\sqrt{R}\left(\mathbb{E}_{u\sim\mathcal{N}(0,1)}|u|^{2p}\right)^{1/2p}\right)^2 && (95)\\
&\leq \left(a + b\sqrt{R}\left(c'^{2p}p^p\right)^{1/2p}\right)^2 && (96)\\
&\leq \left(a + bc'^2\sqrt{R}\right)^2 p && (97)\\
&\equiv \kappa p. && (98)
\end{aligned}$$

We applied Cauchy-Schwarz' inequality in the first inequality, the triangle inequality in the second one, the fact $\Sigma_{11}, \Sigma_{22} \leq R$ in the third one, absolute moments estimate of standard Gaussian in the fourth one, where c' is a constant such that

$$\left(\mathbb{E}_{u\sim\mathcal{N}(0,1)}|u|^p\right)^{1/p} \leq c'\sqrt{p}. \tag{99}$$

□

A.9 Glossary and notation

We use the following shorthands in this work:

1. NN - neural network;
2. CNN - convolutional neural network;
3. LCN - locally-connected network, a.k.a. convolutional network without weight sharing;
4. FCN - fully connected network, a.k.a. multilayer perceptron (MLP);
5. GP - Gaussian process;
6. X-GP - a GP equivalent to a Bayesian infinitely wide neural network of architecture X (§2).
7. MC-(X-)-GP - a Monte Carlo estimate (§A.5) of the X-GP.
8. Width, (number of) filters, (number of) channels represent the same property for CNNs and LCNs.

9. Pooling - referring to architectures as "with" or "without pooling" means having a single global average pooling layer (collapsing the spatial dimensions of the activations x^{L+1}) before the final linear FC layer giving the regression outputs z^{L+1}.
10. Invariance and equivariance are always discussed w.r.t. translations in the spatial dimensions of the inputs.
11. Whenever an index is omitted, the variable is assumed to contain all possible entries along the respective dimension. E.g. y^0 is a vector of size $|\mathcal{X}|\, n^0 d$, $\left[K^l\right]_{\alpha,\alpha'}$ is a matrix of shape $|\mathcal{X}| \times |\mathcal{X}|$, z^l_j is a vector of size $|\mathcal{X}|\, d$, etc.
12. Our work concerns proving that the top-layer pre-activations z^L converge in distribution to an $|\mathcal{X}|\, n^{L+1} d$-variate normal random vector with a particular covariance matrix of shape $\left(|\mathcal{X}|\, n^{L+1} d\right) \times \left(|\mathcal{X}|\, n^{L+1} d\right)$ as $\min\left\{n^1, \ldots, n^L\right\} \to \infty$. We emphasize that only the channels in *hidden* layers are taken to infinity, and n^{L+1}, the number of channels in the top-layer pre-activations z^L, remains fixed. Therefore for convergence proofs, we always consider z^l, y^l, as well as any of their indexed subsets like z^l_j, $y^l_{i,\alpha}$ to be 1D vector random variables, while K^l, as well as any of their indexed subsets (when applicable, e.g. $\left[K^l\right]_{\alpha,\alpha'}$, $\left[K^l\right](x, x')$) to be 2D matrix random variables.

A.10 Experimental Setup

Throughout this work we only consider 3×3 (possibly unshared) convolutional filters with stride 1 and no dilation.

All inputs are normalized to have zero mean and unit variance, i.e. lie on the $d-$dimensional sphere of radius $\sqrt{d}$, where d is the total dimensionality of the input. All labels are treated as regression targets with zero mean. I.e. for a single-class classification problem with C classes targets are $C-$dimensional vectors with $-1/C$ and $(C-1)/C$ entries in incorrect and correct class indices respectively.

If a subset of a full dataset is considered for computational reasons, it is randomly selected in a balanced fashion. No data augmentation is used.

All experiments were implemented in Tensorflow [1] and executed with the help of Vizier [15]. All neural networks are trained using Adam [18] minimizing the mean squared error loss.

A.10.1 Many-channel CNNs and LCNs

Relevant Figures: 2 (b), 6, 7.

We use a training and validation subsets of CIFAR10 of sizes 500 and 4000 respectively. All images are bilinearly downsampled to 8×8 pixels. All models have 3 hidden layers with an erf nonlinearity. No ("valid") padding is used.

Weight and bias variances are set to $\sigma_\omega^2 \approx 1.7562$ and $\sigma_b^2 \approx 0.1841$, corresponding to the pre-activation variance fixed point $q^* = 1$ [33] for the erf nonlinearity.

NN training proceeds for 2^{19} gradient updates, but aborts if no progress on training loss is observed for the last 100 epochs. If the training loss does not reduce by at least 10^{-4} for 20 epochs, the learning rate is divided by 10.

All computations are done with 32-bit precision.

The following NN parameters are considered[4]:

1. Architecture: CNN or LCN.
2. Pooling: no pooling or a single global average pooling (averaging over spatial dimensions) before the final FC layer.
3. Number of channels: 2^k for k from 0 to 12.

[4]Due to time and memory limitations certain large configurations could not be evaluated. We believe this did not impact the results of this work in a qualitative way.

4. Initial learning rate: 10^{-k} for k from 0 to 15.
5. Weight decay: 0 and 10^{-k} for k from 0 to 8.
6. Batch size: 10, 25, 50, 100, 200.

For NNs, all models are filtered to only 100%-accurate ones on the training set and then for each configuration of {architecture, pooling, number of channels} the model with the lowest validation loss is selected among the configurations of {learning rate, weight decay, batch size}.

For GPs, the same CNN-GP is plotted against CNN and LCN networks without pooling. For LCN with pooling, inference was done with an appropriately rescaled CNN-GP kernel, i.e. $\left(\mathcal{K}_{\infty}^{\text{vec}} - \sigma_b^2\right)/d + \sigma_b^2$, where d is the spatial size of the penultimate layer. For CNNs with pooling, a Monte Carlo estimate was computed (see §A.5) with $n = 2^{12}$ filters and $M = 2^6$ samples.

For GP inference, the initial diagonal regularization term applied to the training convariance matrix is 10^{-10}; if the cholesky decompisition fails, the regularization term is increased by a factor of 10 until it either succeeeds or reaches the value of 10^5, at which point the trial is considered to have failed.

A.10.2 Monte Carlo Evaluation of Intractable GP Kernels

Relevant Figures: 3, 4.

We use the same setup as in §A.10.1, but training and validation sets of sizes 2000 and 4000 respectively.

For MC-GPs we consider the number of channels n (width in FCN setting) and number of NN instantiations M to accept values of 2^k for k from 0 to 10.

Kernel distance is computed as $\|\mathcal{K}_{\infty} - K_{n,M}\|_{\text{F}}^2 / \|\mathcal{K}_{\infty}\|_{\text{F}}^2$, where $\mathcal{K}_{\infty}$ is substituted with $K_{2^{10},2^{10}}$ for the CNN-GP pooling case (due to impracticality of computing the exact $\mathcal{K}_{\infty}^{\text{pool}}$). GPs are regularized in the same fashion as in §A.10.1, but the regularization factor starts at 10^{-4} and ends at 10^{10} and is multiplied by the mean of the training covariance diagonal.

A.10.3 Transforming a GP over spatial locations into a GP over classes

Relevant Figure: 1.

We use the same setup as in §A.10.2, but rescale the input images to size of 31×31, so that at depth 15 the spatial dimension collapses to a 1×1 patch if no padding is used (hence the curve of the CNN-GP without padding halting at that depth).

For MC-CNN-GP with pooling, we use samples of networks with $n = 16$ filters. Due to computational complexity we only consider depths up to 31 for this architecture. The number of samples M was selected independently for each depth among $\left\{2^k\right\}$ for k from 0 to 15 to maximize the validation accuracy on a separate 500-points validation set. This allowed us to avoid the poor conditioning of the kernel. GPs are regularized in the same fashion as in §A.10.1, but for MLP-GP the multiplicative factor starts at 10^{-4} and ends at 10^{10}.

A.10.4 Relationship to Deep Signal Propagation

Relevant Table: 3.

We use a training and validation subsets of CIFAR10 of sizes 500 and 1000 respectively.

We use the erf nonlinearity. For CNN-GP, images are zero-padded ("same" padding) to maintain the spatial shape of the activations as they are propagated through the network.

Weight and bias variances (horizontal axis σ_ω^2 and vertical axis σ_b^2 respectively) are sampled from a uniform grid of size 50×50 on the range $[0.1, 5] \times [0, 2]$ including the endpoints.

All computations are done with 64-bit precision. GPs are regularized in the same fashion as in §A.10.1, but the regularization factor is multiplied by the mean of the training covariance diagonal. If the experiment fails due to numerical reasons, 0.1 (random chance) validation accuracy is reported.

A.10.5 CNN-GP on full datasets

Relevant Table: 2, Figure 2 (a, c).

We use full training, validation, and test sets of sizes 50000, 10000, and 10000 respectively for MNIST [24] and Fashion-MNIST [44], 45000, 5000, and 10000 for CIFAR10 [19]. We use validation accuracy to select the best configuration for each model (we do not retrain on valdiation sets).

GPs are computed with 64-bit precision, and NNs are trained with 32-bit precision. GPs are regularized in the same fashion as in §A.10.4.

Zero-padding ("same") is used.

The following parameters are considered:

1. Architecture: CNN or FCN.
2. Nonlinearity: erf or ReLU.
3. Depth: 2^k for k from 0 to 4 (and up to 2^5 for MNIST and Fashion-MNIST datasets).
4. Weight and bias variances. For erf: q^* from $\{0.1, 1, 2, \ldots, 8\}$. For ReLU: a fixed weight variance $\sigma_\omega^2 = 2 + 4e^{-16}$ and bias variance σ_b^2 from $\{0.1, 1, 2, \ldots, 8\}$.

On CIFAR10, we additionally train NNs for 2^{18} gradient updates with a batch size of 128 with corresponding parameters in addition to[5]

1. Pooling: no pooling or a single global average pooling (averaging over spatial dimensions) before the final FC layer (only for CNNs).
2. Number of channels or width: 2^k for k from 1 to 9 (and up to 2^{10} for CNNs with pooling in Figure 2, a).
3. Learning rate: $10^{-k} \times 2^{16} / (\text{width} \times q^*)$ for k from 5 to 9, where width is substituted with the number of channels for CNNs and q^* is substituted with σ_b^2 for ReLU networks. "Small learning rate" in Table 2 refers to $k \in \{8, 9\}$.
4. Weight decay: 0 and 10^{-k} for k from 0 to 5.

For NNs, all models are filtered to only 100%-accurate ones on the training set (expect for values in parentheses in Table 2). The reported values are then reported for models that achieve the best validation accuracy.

A.10.6 Model comparison on CIFAR10

Relevant Table: 1.

We use the complete CIFAR10 dataset as described in §A.10.5 and consider 8-layer ReLU models with weight and bias variances of $\sigma_\omega^2 = 2$ and $\sigma_b^2 = 0.01$. The number of channels / width is set to 2^5, 2^{10} and 2^{12} for LCN, CNN, and FCN respectively.

GPs are computed with 64-bit precision, and NNs are trained with 32-bit precision. No padding ("valid") is used.

NN training proceeds for 2^{18} gradient updates with batch size 64, but aborts if no progress on training loss is observed for the last 10 epochs. If the training loss does not reduce by at least 10^{-4} for 2 epochs, the learning rate is divided by 10. Values for NNs are reported for the best validation accuracy over different learning rates (10^{-k} for k from 2 to 12) and weight decay values (0 and 10^{-k} for k from 2 to 7). For GPs, validation accuracy is maximized over initial diagonal regularization terms applied to the training convariance matrix: $10^{-k} \times$ [mean of the diagonal] for k among 2, 4 and 9 (if the cholesky decompisition fails, the regularization term is increased by a factor of 10 until it succeeeds or k reaches the value of 10).

[5] Due to time and compute limitations certain large configurations could not be evaluated. We believe this did not impact the results of this work in a qualitative way.