Correlated Weights in Infinite Limits of Deep Convolutional Neural Networks

Adrià Garriga-Alonso[1] **Mark van der Wilk**[2]

[1]Department of Engineering, University of Cambridge, UK
[2]Department of Computer Science, Imperial College London, UK

Abstract

Infinite width limits of deep neural networks often have tractable forms. They have been used to analyse the behaviour of finite networks, as well as being useful methods in their own right. When investigating infinitely wide convolutional neural networks (CNNs), it was observed that the correlations arising from spatial weight sharing disappear in the infinite limit. This is undesirable, as spatial correlation is the main motivation behind CNNs. We show that the loss of this property is not a consequence of the infinite limit, but rather of choosing an independent weight prior. Correlating the weights maintains the correlations in the activations. Varying the amount of correlation interpolates between independent-weight limits and mean-pooling. Empirical evaluation of the infinitely wide network shows that optimal performance is achieved between the extremes, indicating that correlations can be useful.

1 INTRODUCTION

Analysing infinitely wide limits of neural networks has long been used to provide insight into the properties of neural networks. Neal [1996] first noted such a relationship, through showing that infinitely wide Bayesian neural networks converge in distribution to Gaussian processes (GPs). The success of GPs raised the question of whether such a comparatively simple model could replace a complex neural network. MacKay [1998] noted that taking the infinite limit resulted in a fixed feature representation, a key desirable property of neural networks. Since this property is lost due to the infinite limit, MacKay inquired: "have we thrown the baby out with the bath water?"

In this work, we follow the recent interest in infinitely wide convolutional neural networks [Garriga-Alonso et al., 2019, Novak et al., 2019], to investigate another property that is lost when taking the infinite limit: correlation in the activations of patches in different parts of the image. Given that convolutions were developed to introduce these correlations, and that they improve performance [Arora et al., 2019], it seems undesirable that they are lost when more filters are added. Currently, the only way of reintroducing spatial correlations is to change the model architecture by introducing mean-pooling [Novak et al., 2019]. This raises two questions:

1) Is the loss of patchwise correlations a necessary consequence of the infinite limit?

2) Is an architectural change the only way of reintroducing patchwise correlations?

We show that the answer to both these questions is "no". Correlations between patches can also be maintained in the limit without pooling by introducing correlations between the weights in the prior. The amount of correlation can be controlled, which allows us to interpolate between the existing approaches of full independence and mean-pooling. Our approach allows the discrete architectural choice of mean-pooling to be replaced with a more flexible continuous amount of correlation.

We empirically show that modest performance improvements can be obtained by replacing mean-pooling at the final layer with an intermediate amount of correlation. In addition, we show that in layers before the final one, the discrete architectural choice of mean-pooling can be replaced by an intermediate amount of correlation, without degrading performance. Avoiding discrete design decisions makes architecture search easier, by allowing continuous optimisation. We speculate that these results from infinite networks could be useful for adapting priors or initialisations in finite networks, leading to better performance, or easier design.

Overall, our work illustrates that non-standard choices in the weight prior can significantly influence properties in the infinite limit, and that good choices can lead to improved

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

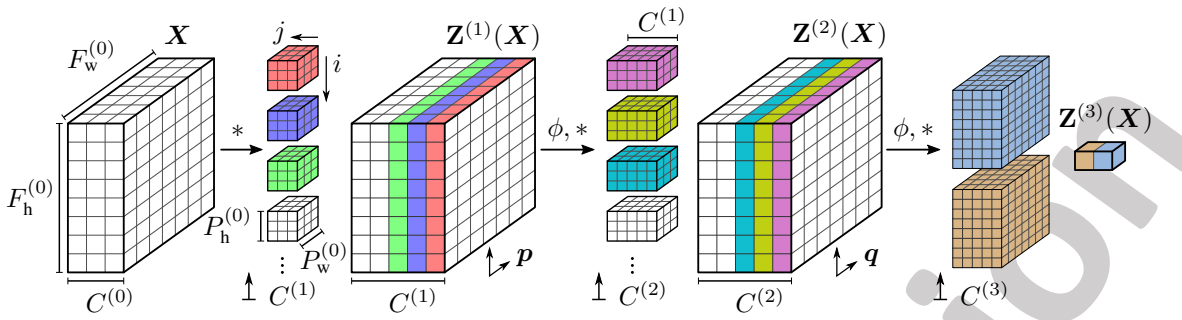

Figure 1: A deep convolutional neural network following our notation. Infinite limits are taken over the number of convolutional filters $C^{(\ell)}$ (vertical), which equals the number of channels in the following layer (horizontal). The network has $L = 3$ layers and $D = 2$ spatial dimensions. The output is not spatially extended ($\boldsymbol{F}^{(3)} = \mathbf{1}$) because $\boldsymbol{P}^{(3)} = \boldsymbol{F}^{(2)}$.

performance. We hope that this work inspires investigation into correlated weights in finite neural networks, as well as more non-standard priors or initialisations.

2 SPATIAL CORRELATIONS IN SINGLE HIDDEN LAYER NETWORKS

To begin, we will analyse the infinite limit of a single hidden layer convolutional neural network (CNN). This illustrates the choices that lead to the disappearance of spatial correlation in the activations. We extend Garriga-Alonso et al. [2019] and Novak et al. [2019] by considering weight priors with correlations. By adjusting the correlation, we can interpolate between existing independent weight limits and mean-pooling, which previously had to be introduced as a discrete architectural choice. We also discuss how existing convolutional Gaussian processes [van der Wilk et al., 2017, Dutordoir et al., 2020] can be obtained from limits of correlated weight priors.

Consider a CNN with $L = 2$ layers. Figure 1 provides a graphical representation of the notation. The input $\boldsymbol{X}$ is a real-valued tensor of shape $C^{(0)} \times \boldsymbol{F}^{(0)}$, where $C^{(0)} \in \mathbb{N}$ is the number of channels and $\boldsymbol{F}^{(0)} \in \mathbb{N}^D$ the spatial size of the input. Superscripts denote the layer index. For images, usually $C^{(0)} = 3$ (one per colour), and the number of spatial input dimensions is $D = 2$, so $\boldsymbol{F}^{(0)} = (F_{\mathrm{h}}^{(0)}, F_{\mathrm{w}}^{(0)})$. The convolution operation at layer $\ell \in [L]$ divides its input into patches of size $\boldsymbol{P}^{(\ell)} \preceq \boldsymbol{F}^{(\ell-1)}$. For a given spatial location of the next activation $\boldsymbol{q} \in [\boldsymbol{F}^{(\ell)}]$,[1] the patch function $\tilde{\boldsymbol{q}}(\cdot) : [\boldsymbol{P}^{(\ell)}] \to [\boldsymbol{F}^{(\ell-1)}]$ iterates over the elements of the patch (eq. 3). Weights are applied by taking an inner product with all patches, which we do $C^{(\ell)}$ times to give multiple channels in the next layer. By collecting all weights in the tensor $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{C^{(\ell)} \times C^{(\ell-1)} \times \boldsymbol{P}^{(\ell)}}$ the pre- and post-nonlinearity activations are respectively, for $\ell \in [L]$,

$$Z^{(\ell)}_{i,\boldsymbol{q}}(\boldsymbol{X}) = \sum_{j=1}^{C^{(\ell-1)}} \sum_{\boldsymbol{p}=\mathbf{1}}^{\boldsymbol{P}^{(\ell)}} W^{(\ell)}_{i,j,\boldsymbol{p}} \, A^{(\ell-1)}_{j,\tilde{\boldsymbol{q}}(\boldsymbol{p})}(\boldsymbol{X}) \,, \tag{1}$$

$$A^{(0)}_{i,\boldsymbol{p}}(\boldsymbol{X}) = X_{i,\boldsymbol{p}}, \quad A^{(\ell)}_{i,\boldsymbol{p}}(\boldsymbol{X}) = \phi\Big(Z^{(\ell)}_{i,\boldsymbol{p}}(\boldsymbol{X})\Big) \,. \tag{2}$$

Equation 1 is a channel-wise sum of D-dimensional convolutions, and ϕ denotes the elementwise nonlinearity.

For layer $\ell \in [L]$, stride s, dilation h, the patch function is $\tilde{\boldsymbol{q}}(\boldsymbol{p}) \triangleq (\tilde{q}_1(p_1), \ldots, \tilde{q}_D(p_D))$, where

$$\tilde{q}_d(p_d) = sq_d - h(p_d - \lceil P_d/2 \rceil) \,. \tag{3}$$

Using eq. 3, it is possible to verify that eq. 1 is the usual deep learning convolution (appendix A).

In a single hidden layer CNN, these activations are followed by a fully-connected layer with weights $\boldsymbol{W}^{(2)} \in \mathbb{R}^{C^{(1)} \times \boldsymbol{P}^{(2)}}$, where $\boldsymbol{P}^{(2)} = \boldsymbol{F}^{(1)}$. Our final output is again given by a summation over the activations

$$f(\boldsymbol{X}) = \sum_{j=1}^{C^{(1)}} \sum_{\boldsymbol{p}=1}^{\boldsymbol{F}^{(1)}} W^{(2)}_{j,\boldsymbol{p}} \, A^{(1)}_{j,\boldsymbol{p}}(\boldsymbol{X}) = \sum_{\boldsymbol{p}=\mathbf{1}}^{\boldsymbol{F}^{(1)}} Z^{(f)}_{\boldsymbol{p}}(\boldsymbol{X}) \,, \tag{4}$$

where $Z^{(f)}_{\boldsymbol{p}}(\boldsymbol{X})$ denotes the result before the summation over spatial locations $\boldsymbol{p}$.

We analyse the distribution on function outputs $f(\boldsymbol{X})$ for some Gaussian prior $p(\mathcal{W})$ on the weights of all layers $\mathcal{W}$. In all the cases we consider, we take the prior to be independent over layers and channels. Here we extend earlier work by allowing spatial correlation in the final layer's weights (we will consider all layers later) through the covariance tensor $\boldsymbol{\Sigma}^{(\ell)} \in \mathbb{R}^{\boldsymbol{F}^{(\ell)} \times \boldsymbol{F}^{(\ell)}}$. This gives the prior

$$p(\boldsymbol{W}^{(1)}) = \prod_{i=1}^{C^{(1)}} \prod_{j=1}^{C^{(0)}} \mathcal{N}\Big(\boldsymbol{W}^{(1)}_{i,j} \,;\, 0, \boldsymbol{I}\Big) \,, \tag{5}$$

$$p(\boldsymbol{W}^{(2)}) = \prod_{i=1}^{C^{(1)}} \mathcal{N}\left(\boldsymbol{W}^{(2)}_{i} \,;\, 0, \frac{1}{C^{(1)}} \boldsymbol{\Sigma}^{(2)}\right) , \tag{6}$$

[1] For some number $P \in \mathbb{N}$, the expression $[P]$ is the set $\{1, \ldots, P\}$. For a tuple $\boldsymbol{P} \in \mathbb{N}^D$, $[\boldsymbol{P}] = [P_1] \times \cdots \times [P_D]$.

with independence between different layers' weights. Here, a tensor-valued covariance $\boldsymbol{\Sigma}^{(2)}$ expresses arbitrary covariance over the spatial dimensions of the tensor $\boldsymbol{W}_i^{(2)}$: $\mathbb{C}\left[W_{i,\boldsymbol{p}}^{(2)}, W_{i,\boldsymbol{p}'}^{(2)}\right] = \Sigma_{\boldsymbol{p},\boldsymbol{p}'}^{(2)}/C^{(\ell-1)}$.

Since $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}$ are i.i.d. over channels $i \in C^{(1)}$, the random variables $\sum_{\boldsymbol{p}=\boldsymbol{1}}^{\boldsymbol{F}^{(1)}} W_{i,\boldsymbol{p}}^{(2)}\, A_{i,\boldsymbol{p}}^{(1)}(\boldsymbol{X})$ are identically distributed and independent for each $i \in [C^{(1)}]$. This allows us to apply the central limit theorem (CLT) to their sum $f(\boldsymbol{X})$, showing that $f(\boldsymbol{X})$ converges in distribution to a Gaussian process as $C^{(1)} \to \infty$ [Neal, 1996].

The covariance between the final-layer activations for two inputs $\boldsymbol{X}, \boldsymbol{X}'$ becomes

$$\begin{aligned}
&\underset{\boldsymbol{W}}{\mathbb{C}}\left[Z_{\boldsymbol{p}}^{(f)}(\boldsymbol{X}), Z_{\boldsymbol{p}'}^{(f)}(\boldsymbol{X}')\right] = \\
&= \underset{\boldsymbol{W}}{\mathbb{E}}\left[\sum_{j=1}^{C^{(1)}}\sum_{j'=1}^{C^{(1)}} W_{j,\boldsymbol{p}}^{(2)} A_{j,\boldsymbol{p}}^{(1)}(\boldsymbol{X})\; W_{j',\boldsymbol{p}'}^{(2)} A_{j',\boldsymbol{p}'}^{(1)}(\boldsymbol{X}')\right],
\end{aligned}$$

use independences to split the expectations, and substitute the weight covariance,

$$\begin{aligned}
&= \sum_{j=1}^{C^{(1)}}\sum_{j'=1}^{C^{(1)}} \underset{\boldsymbol{W}^{(1)}}{\mathbb{E}}\left[A_{j,\boldsymbol{p}}^{(1)}(\boldsymbol{X})\, A_{j',\boldsymbol{p}'}^{(1)}(\boldsymbol{X}')\right] \underset{\boldsymbol{W}^{(2)}}{\mathbb{E}}\left[W_{j,\boldsymbol{p}}^{(2)} W_{j',\boldsymbol{p}'}^{(2)}\right] \\
&= \sum_{j=1}^{C^{(1)}}\sum_{j'=1}^{C^{(1)}} \underset{\boldsymbol{W}^{(1)}}{\mathbb{E}}\left[A_{j,\boldsymbol{p}}^{(1)}(\boldsymbol{X})\, A_{j',\boldsymbol{p}'}^{(1)}(\boldsymbol{X}')\right]\; \delta_{j,j'}\frac{\Sigma_{\boldsymbol{p},\boldsymbol{p}'}^{(2)}}{C^{(1)}},
\end{aligned}$$

eliminate one of the sums over j using $\delta_{j,j'}$, and rearrange

$$\begin{aligned}
&= \underset{\boldsymbol{W}^{(1)}}{\mathbb{E}}\left[\frac{1}{C^{(1)}}\sum_{j=1}^{C^{(1)}} A_{j,\boldsymbol{p}}^{(1)}(\boldsymbol{X})\, A_{j,\boldsymbol{p}'}^{(1)}(\boldsymbol{X}')\right]\Sigma_{\boldsymbol{p},\boldsymbol{p}'}^{(2)} \\
&= V_{\boldsymbol{p},\boldsymbol{p}'}^{(1)}(\boldsymbol{X}, \boldsymbol{X}')\, \Sigma_{\boldsymbol{p},\boldsymbol{p}'}^{(2)}\,. \qquad (7)
\end{aligned}$$

The limit of the sum of the final expectation over $\boldsymbol{W}^{(1)}$ can be found in closed form for many activations (see section 3.2) and is denoted $V_{\boldsymbol{p},\boldsymbol{p}'}^{(1)}(\boldsymbol{X}, \boldsymbol{X}')$. Note in eq. 1 that the activations for some location $\boldsymbol{p} \in [\boldsymbol{F}^{(1)}]$ only depend on the input patch at $\boldsymbol{p}$, that is, on the elements of $\boldsymbol{X}$ that are in the image $\mathrm{im}(\tilde{\boldsymbol{p}})$ of the patch function $\tilde{\boldsymbol{p}}(\cdot)$. Thus, the kernel acts locally on patches: $V_{\boldsymbol{p},\boldsymbol{p}'}^{(1)}(\boldsymbol{X}, \boldsymbol{X}') = k^{(1)}\left(\boldsymbol{X}_{:,\,\mathrm{im}(\tilde{\boldsymbol{p}})}, \boldsymbol{X}'_{:,\,\mathrm{im}(\tilde{\boldsymbol{p}}')}\right)$.

We find the final kernel for the GP by taking the covariance between function values $f(\boldsymbol{X})$ and $f(\boldsymbol{X}')$ and performing the final sum in eq. 4:

$$\begin{aligned}
K(\boldsymbol{X}, \boldsymbol{X}') &= \mathbb{C}\left[f(\boldsymbol{X}), f(\boldsymbol{X}')\right] \\
&= \sum_{\boldsymbol{p},\boldsymbol{p}'} k^{(1)}\left(\boldsymbol{X}_{:,\,\mathrm{im}(\tilde{\boldsymbol{p}})}, \boldsymbol{X}'_{:,\,\mathrm{im}(\tilde{\boldsymbol{p}}')}\right)\Sigma_{\boldsymbol{p},\boldsymbol{p}'}^{(2)}\,. \qquad (8)
\end{aligned}$$

We can now see how different choices for $\boldsymbol{\Sigma}^{(2)}$ give different forms of spatial correlation.

Independence. Garriga-Alonso et al. [2019] and Novak et al. [2019] consider $\Sigma_{\boldsymbol{p},\boldsymbol{p}'}^{(2)} = \delta_{\boldsymbol{p},\boldsymbol{p}'}\sigma_{\mathrm{w}}^2$, i.e. the case where all weights are independent. The resulting kernel simply sums components over patches, which implies an *additive model* [Stone, 1985], where a *different* function is applied to each patch, after which they are all summed together: $f(\boldsymbol{X}) = \sum_{\boldsymbol{p}} f_{\boldsymbol{p}}(\boldsymbol{X}_{:,\mathrm{im}(\tilde{\boldsymbol{p}})})$. This structure has commonly been applied to improve GP performance in high-dimensional settings [e.g. Duvenaud et al., 2011, Durrande et al., 2012]. Novak et al. [2019] point out that the same kernel can be obtained by taking an infinite limit of a *locally connected network* (LCN) [LeCun, 1989] where connectivity is the same as in a CNN, but without weight sharing, indicating that a key desirable feature of CNNs is lost.

Mean-pooling. By taking $\Sigma_{\boldsymbol{p},\boldsymbol{p}'}^{(2)} = 1/\left|\boldsymbol{F}^{(2)}\right|^2$ we make the weights fully correlated over all locations[2], leading to identical weights for all $\boldsymbol{p}$, i.e. $W_{i,\boldsymbol{p}}^{(2)} = W_i^{(2)}$. This is equivalent to taking the mean response over all spatial locations (see eq. 4), or global average pooling. As Novak et al. [2019] discuss, this reintroduces the spatial correlation that is the intended result of weight sharing. The "translation invariant" convolutional GP of van der Wilk et al. [2017] can be obtained by this single-layer limit using Gaussian activation functions [van der Wilk, 2019]. Since this mean-pooling was shown to be too restrictive in this single-layer case, Van der Wilk et al. [2017] considered pooling with constant weights $\alpha_{\boldsymbol{p}}$ (i.e. without a prior on them). In this framework, this is equivalent to placing a rank 1 prior on the final-layer weights by taking $\Sigma_{\boldsymbol{p}\boldsymbol{p}'}^{(2)} = \alpha_{\boldsymbol{p}}\alpha_{\boldsymbol{p}'}$. This maintains the spatial correlations, but requires the $\alpha_{\boldsymbol{p}}$ parameters to be learned by maximum *marginal* likelihood (ML-II, empirical Bayes).

Spatially correlated weights. In the pooling examples above, the spatial covariance of weights is taken to be a rank-1 matrix. We can add more flexibility to the model by varying the strength of correlation between weights based on their distance in the image. We consider an exponential decay depending on the distance between two patches: $\Sigma_{\boldsymbol{p}\boldsymbol{p}'}^{(2)} = \exp\left(-d(\boldsymbol{p}, \boldsymbol{p}')/l\right)$. We recover full independence by taking $l \to 0$, and mean-pooling with $l \to \infty$. Intermediate values of l allow the rigid assumption of complete weight sharing to be relaxed, while still retaining spatial correlations between similar patches. This construction gives the same kernel as investigated by Mairal et al. [2014] and Dutordoir et al. [2020], who named this property "translation insensitivity", as opposed to the stricter invariance that mean-pooling gives. The additional flexibility improved performance without needing to add many parameters that are learned in a non-Bayesian fashion.

Our construction shows that spatial correlation can be retained in infinite limits without needing to resort to architec-

[2]For a size $\boldsymbol{F} \in \mathbb{N}^D$, its number of elements is $|\boldsymbol{F}| \triangleq \prod_{d=1}^{D} F_d$.

tural changes. A simple change to the prior on the weights is all that is needed. This property is retained in wide limits of deep networks, which we investigate next.

3 SPATIAL CORRELATIONS IN DEEP NETWORKS

Here, we provide an informal extension of the previous section's results to deep networks. In deep networks, correlated weights also retain spatial correlation in the activations. Appendix B provides a formal justification for this section, using the framework by Yang [2019].

The procedure for computing the kernel has a recursive form similar to existing analyses [Garriga-Alonso et al., 2019, Novak et al., 2019]. Negligible additional computation is needed to consider arbitrary correlations, compared to only considering mean-pooling [Novak et al., 2019, Arora et al., 2019]. The main bottleneck is the need for computing covariances for all pairs of patches in the image, as in eq. 8. For a D-dimensional convolutional layer, the corresponding kernel computation is a convolution of the activations' second moment with the $2D$-dimensional covariance tensor of the weights.

The setup for the case of a deep neural network follows that of section 2, but with the number of layers $L > 2$. The outputs of the network are simply the pre-nonlinearity activations of the Lth layer, $Z^{(L)}_{i,\boldsymbol{p}}(\boldsymbol{X})$. If we need several outputs, for example in K-class classification, we may set $C^{(L)} = K$. If the output of the network should not be spatially extended, we set the spatial size to $\boldsymbol{F}^{(L)} = \boldsymbol{1}$. This can be achieved by making the weights $\boldsymbol{W}^{(L)}$ (and their corresponding convolutional patch) have the same size as $\boldsymbol{F}^{(L-1)}$ (see fig. 1).

As pointed out by Matthews et al. [2018], a straightforward application of the central limit theorem is not possible for deep networks. Fortunately, Yang [2019] developed a general framework for expressing neural network architectures and finding their corresponding Gaussian process infinite limits. The resulting kernel is given by the recursion that can be derived from a more informal argument which takes the infinite width limit in a sequential layer-by-layer fashion, as was used in Garriga-Alonso et al. [2019]. We follow this informal derivation, as this more naturally illustrates the procedure for computing the kernel. A formal justification can be found in appendix B.

3.1 RECURSIVE COMPUTATION OF THE KERNEL

In our weight prior, we correlate weights *within* a convolutional filter. The weights remain independent over layers and channels. For each $\ell \in [L]$,

$$p\Big(\boldsymbol{W}^{(\ell)}\Big) = \prod_{i=1}^{C^{(\ell)}} \prod_{j=1}^{C^{(\ell-1)}} \mathcal{N}\Big(\boldsymbol{W}^{(\ell)}_{i,j}; 0, \frac{1}{C^{(\ell-1)}}\boldsymbol{\Sigma}^{(\ell)}\Big). \quad (9)$$

As in section 2, $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{C^{(\ell)} \times C^{(\ell-1)} \times \boldsymbol{F}^{(\ell)}}$, and the covariance tensor $\boldsymbol{\Sigma}^{(\ell)} \in \mathbb{R}^{\boldsymbol{F}^{(\ell)} \times \boldsymbol{F}^{(\ell)}}$ is positive semi-definite. Our derivation is general for any weight covariance, so layers with correlated weights can be interspersed with the usual layers.

A Gaussian process is determined by the mean and covariance of function values for pairs of inputs $\boldsymbol{X}, \boldsymbol{X}'$. The mean is zero. Using the recursion in eq. 1, we can find the covariance between any two pre-nonlinearity activations from a pair of inputs $\boldsymbol{X}, \boldsymbol{X}'$ and the covariance of the previous layer. For $i, i' \in [C^{(\ell)}]$ and $\boldsymbol{q}, \boldsymbol{q}' \in [\boldsymbol{F}^{(\ell)}]$,

$$\begin{aligned}
&\underset{\mathcal{W}}{\mathbb{C}}\left[Z^{(\ell)}_{i,\boldsymbol{q}}(\boldsymbol{X}), Z^{(\ell)}_{i',\boldsymbol{q}'}(\boldsymbol{X}')\right] = \\
&= \sum_{j,j',\,\boldsymbol{p},\boldsymbol{p}'} \underset{\mathcal{W}}{\mathbb{E}}\left[W^{(\ell)}_{i,j,\boldsymbol{p}} A^{(\ell-1)}_{j,\tilde{\boldsymbol{q}}(\boldsymbol{p})}(\boldsymbol{X})\, W^{(\ell)}_{i',j',\boldsymbol{p}'} A^{(\ell-1)}_{j',\tilde{\boldsymbol{q}}'(\boldsymbol{p}')}(\boldsymbol{X}')\right],
\end{aligned}$$

substituting the expression for the weight covariance,

$$\begin{aligned}
&= \delta_{i,i'} \sum_{\boldsymbol{p},\boldsymbol{p}'} \Sigma^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'} \underset{\mathcal{W}}{\mathbb{E}}\Bigg[\\
&\qquad \frac{1}{C^{(\ell-1)}} \sum_{j=1}^{C^{(\ell-1)}} A^{(\ell-1)}_{j,\tilde{\boldsymbol{q}}(\boldsymbol{p})}(\boldsymbol{X})\, A^{(\ell-1)}_{j,\tilde{\boldsymbol{q}}'(\boldsymbol{p}')}(\boldsymbol{X}')\Bigg] \\
&= \delta_{i,i'} K^{(\ell)}_{\boldsymbol{q},\boldsymbol{q}'}(\boldsymbol{X}, \boldsymbol{X}')\,. \qquad (10)
\end{aligned}$$

We can see that the covariance of activations in different channels ($i \neq i'$) is zero. Otherwise, to calculate $K^{(\ell)}(\boldsymbol{X}, \boldsymbol{X}')$, we need to calculate the expectation over $\mathcal{W}$, which we term $V^{(\ell-1)}(\boldsymbol{X}, \boldsymbol{X}')$. The resulting kernel expression is

$$K^{(\ell)}_{\boldsymbol{q},\boldsymbol{q}'}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{\boldsymbol{p}=\boldsymbol{1}}^{\boldsymbol{P}^{(\ell)}} \sum_{\boldsymbol{p}'=\boldsymbol{1}}^{\boldsymbol{P}^{(\ell)}} \Sigma^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'} V^{(\ell-1)}_{\tilde{\boldsymbol{q}}(\boldsymbol{p}),\tilde{\boldsymbol{q}}'(\boldsymbol{p}')}(\boldsymbol{X}, \boldsymbol{X}')\,, \quad (11)$$

which, because the concatenation of patch functions is a patch function (remark A.3), is equivalent to a $2D$-dimensional convolution. This kernel does not correspond to a locally-connected network, because it uses off-diagonal elements of the previous layer's kernel.

3.2 EXPECTATION OF THE NONLINEARITIES

For $\ell = 0$, the activations in the previous layer are the image inputs, i.e. $\boldsymbol{A}^{(0)}(\boldsymbol{X}) = \boldsymbol{X}$ (eq. 2), making $V^{(0)}_{\boldsymbol{q},\boldsymbol{q}'}(\boldsymbol{X}, \boldsymbol{X}')$ an inner product between image patches.

For $\ell \geq 1$, the expression inside the expectation in eq. 10 is a random variable, an average over $j \in [C^{(\ell-1)}]$. From

eq. 1 we see that all its terms have the same expectation, i.e.

$$V^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'}(\boldsymbol{X},\boldsymbol{X}') = \mathop{\mathbb{E}}_{\mathcal{W}}\left[\frac{1}{C^{(\ell)}}\sum_{j=1}^{C^{(\ell)}} A^{(\ell)}_{j,\boldsymbol{p}}(\boldsymbol{X})A^{(\ell)}_{j,\boldsymbol{p}'}(\boldsymbol{X}')\right]$$
$$= \mathop{\mathbb{E}}_{\mathbf{Z}^{(\ell)}(\boldsymbol{X}),\mathbf{Z}^{(\ell)}(\boldsymbol{X}')}\left[\phi\Big(Z^{(\ell)}_{1,\boldsymbol{p}}(\boldsymbol{X})\Big)\phi\Big(Z^{(\ell)}_{1,\boldsymbol{p}'}(\boldsymbol{X}')\Big)\right]. \quad (12)$$

For the purposes of eq. 12, in the infinite width limit, the pre-nonlinearities $\mathbf{Z}^{(\ell)}(\boldsymbol{X}), \mathbf{Z}^{(\ell)}(\boldsymbol{X}')$ converge in distribution to a joint Gaussian (theorem B.7). Accordingly, the value of the expectation above depends only on the entries of their 2×2 covariance matrix and the form of ϕ. Here we represent this dependence through the function $F_\phi(\Sigma_{\mathrm{x}}, \Sigma_{\mathrm{y}}, \Sigma_{\mathrm{xy}})$,

$$V^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'}(\boldsymbol{X},\boldsymbol{X}') = F_\phi\big(K^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'}(\boldsymbol{X},\boldsymbol{X}),\, K^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'}(\boldsymbol{X}',\boldsymbol{X}'), K^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'}(\boldsymbol{X},\boldsymbol{X}')\big). \quad (13)$$

Combining eq. 11, 13 and the input inner product provides us with a recursive procedure to compute the covariances all the way up to the final layer.

For the balanced ReLU nonlinearity ($\phi(x) = \sqrt{2}\max(0,x)$), which we use in all the experiments in this paper, we can use the expression by Cho and Saul [2009]:

$$F_\phi(\Sigma_{\mathrm{x}},\Sigma_{\mathrm{y}},\Sigma_{\mathrm{xy}}) = \frac{1}{\pi}\sqrt{\Sigma_{\mathrm{x}}\Sigma_{\mathrm{y}} - {\Sigma_{\mathrm{xy}}}^2} + \left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{{\Sigma_{\mathrm{xy}}}^2}{\Sigma_{\mathrm{x}}\Sigma_{\mathrm{y}}}\right)\right)\Sigma_{\mathrm{xy}}. \quad (14)$$

This expression implies that $V^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}}(\boldsymbol{X},\boldsymbol{X}) = K^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}}(\boldsymbol{X},\boldsymbol{X})$, for all $\boldsymbol{X}$ and $\boldsymbol{p}$ [Lee et al., 2018, Matthews et al., 2018].

3.3 COMPUTATIONAL COMPLEXITY, DIAGONAL PROPAGATION

To handle the covariance tensor for $K^{(\ell)}(\boldsymbol{X},\boldsymbol{X}')$, we need to compute and represent $|\boldsymbol{F}^{(\ell)}|^2$ entries. This can be considerably more expensive than the forward pass of the corresponding CNN, where the activations have size $|\boldsymbol{F}^{(\ell)}|$. In special cases, the computation or memory costs can be reduced, compared to the $2D$-dimensional convolution in eq. 11, which is a generalisation of previous algorithms. These cases do not include layers with mean-pooling, for which our algorithm is equally expensive to previous ones [Arora et al., 2019].

If weights are independent, only the diagonal of $\boldsymbol{\Sigma}^{(\ell)}$ has nonzero entries, so $\Sigma^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'} = \delta_{\boldsymbol{p},\boldsymbol{p}'}\Sigma^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}'}$. One of the sums in the eq. 11 can then be removed,

$$K^{(\ell)}_{\boldsymbol{q},\boldsymbol{q}'}(\boldsymbol{X},\boldsymbol{X}') = \sum_{\boldsymbol{p}=\boldsymbol{1}}^{\boldsymbol{P}^{(\ell-1)}} \Sigma^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}}\, V^{(\ell-1)}_{\tilde{\boldsymbol{q}}(\boldsymbol{p}),\tilde{\boldsymbol{q}}'(\boldsymbol{p})}(\boldsymbol{X},\boldsymbol{X}'). \quad (15)$$

The patch functions that access $V^{(\ell-1)}(\boldsymbol{X},\boldsymbol{X}')$ are still different ($\tilde{\boldsymbol{q}}(\cdot)$ and $\tilde{\boldsymbol{q}}'(\cdot)$), but their argument $\boldsymbol{p}$ is the same.

Patch functions (definition A.2) subtract their argument multiplied by the dilation. Consequently, the difference of two patch functions with the same argument is constant: $\tilde{\boldsymbol{q}}(\boldsymbol{p}) - \tilde{\boldsymbol{q}}'(\boldsymbol{p}) = s\cdot(\boldsymbol{q}-\boldsymbol{q}')$. This means that the terms of the sum are on the same diagonal. Thus, to calculate the covariance for a given location pair $\boldsymbol{q},\boldsymbol{q}'$, we need to do a single sum *over a diagonal* of the second moment tensors $V^{(\ell-1)}(\boldsymbol{X},\boldsymbol{X}')$ and $\boldsymbol{\Sigma}^{(\ell)}$.

This results in exact same algorithm as Arora et al. [2019], which convolves over the diagonals, for layers with independent weights. Its memory cost is still $O(|\boldsymbol{F}^{(\ell)}|^2)$, but the computational cost is reduced to $O(|\boldsymbol{F}^{(\ell)}|^2|\boldsymbol{P}^{(\ell)}|)$, compared to $O(|\boldsymbol{F}^{(\ell)}|^2|\boldsymbol{P}^{(\ell)}|^2)$ for non-diagonal covariance.

Diagonal propagation with independent weights. Exactly *which* diagonal of $V^{(\ell-1)}(\boldsymbol{X},\boldsymbol{X}')$ do we need to sum over? Clearly, it is the one indexed by $s\cdot(\boldsymbol{q}-\boldsymbol{q}')$, i.e. the one that contains the position $(s\boldsymbol{q}, s\boldsymbol{q}')$. Thus, the number of diagonals of $V^{(\ell-1)}(\boldsymbol{X},\boldsymbol{X}')$ that we will need to access is exactly the number of possible values that $\boldsymbol{q}-\boldsymbol{q}'$ can take. That number is determined by the size $\boldsymbol{F}^{(\ell)}$ of layer ℓ, but is completely unrelated to the size $\boldsymbol{F}^{(\ell-1)}$ of layer $\ell-1$.

Fix some layer $\ell \in [L]$. We can iterate this argument from layer ℓ to layer 1 to show that, for all $m \le \ell$, the number of diagonals of $K^{(m)}(\boldsymbol{X},\boldsymbol{X}')$ that one needs to calculate depends only on $\boldsymbol{F}^{(\ell)}$. This can yield significant computational savings when the stride is $s \ge 2$ for one or more layers.

Last layer not spatially extended. When the last layer is not spatially extended, its size is $\boldsymbol{F}^{(L)} = \boldsymbol{1}$, so it only has one diagonal. If all the weights of the CNN are independent, this implies that we only need to calculate one diagonal of the covariance for every layer. That is:

$$K^{(\ell)}_{\boldsymbol{q},\boldsymbol{q}}(\boldsymbol{X},\boldsymbol{X}') = \sum_{\boldsymbol{p}=\boldsymbol{1}}^{\boldsymbol{P}^{(\ell-1)}} \Sigma^{(\ell)}_{\boldsymbol{p},\boldsymbol{p}}\, V^{(\ell-1)}_{\tilde{\boldsymbol{q}}(\boldsymbol{p}),\tilde{\boldsymbol{q}}(\boldsymbol{p})}(\boldsymbol{X},\boldsymbol{X}'). \quad (16)$$

With this simplification, the convolutions required to calculate the kernel are D-dimensional, bringing the memory cost to $O(|\boldsymbol{F}^{(\ell)}|)$ and computational cost to $O(|\boldsymbol{F}^{(\ell)}||\boldsymbol{P}^{(\ell)}|)$, same as the finite CNN [Garriga-Alonso et al., 2019]. The resulting kernel is equivalent to that of a locally connected network.

3.4 IMPLEMENTATION

We extend the `neural-tangents` [Novak et al., 2020] library with a convolution layer and a fully connected layer, that admit a 4-dimensional covariance tensor for the weights. This allows interoperation with existing layers.

Since 4d convolutions are uncommon in deep learning, our implementation uses a sum over $P_{\mathrm{h}}^{(\ell)}$ 3-d convolutions, where $P_{\mathrm{h}}^{(\ell)} = 3$ is the spatial height of the convolutional filter. While this enables GPU acceleration, computing the kernel is a costly operation. Reproducing our results takes around 10 days using an nVidia RTX 2070 GPU. Access to computational resources limited our experiments to subsets of data on CIFAR-10.

4 EXPERIMENTS

By considering different amounts of correlation, we can interpolate between existing architectures that use independent weights or full mean-pooling. We consider two possible benefits of using this larger, continuously parameterised space of models:

1) Decreased reliance on discrete architectural choices like mean-pooling.

2) Improved performance by finding a better model in the expanded search space.

Discrete choices pose a challenge for architecture search, as a separate network needs to be trained to evaluate the effect of each choice, which is computationally expensive. Continuous choices are preferable, as gradients can often be used to adjust many choices simultaneously. We investigate whether the discrete choice of mean-pooling can instead be replaced by a suitable selection of the continuous correlation parameter in a larger convolutional filter. While searching in this larger space of kernels, we also hope to observe improved performance. We investigate these two questions by performing parameter search in the next two sections.

4.1 EXPERIMENTAL SETUP

We evaluate various models on class-balanced subsets of CIFAR-10 of size $2^i \cdot 10$, following Arora et al. [2020]. As is standard practice in the wide network literature, we reframe classification as regression to one-hot targets $\boldsymbol{Y}$. We subtract $C = 0.1$ from $\boldsymbol{Y}$ to make its mean zero, but we observed that this affects the results very little. The prediction is the class k with highest mean of the posterior Gaussian process

$$\begin{aligned}\text{label}(x_*) &= \text{argmax}_k\, f_k(x_*) \\ &= \text{argmax}_k\, \boldsymbol{K}_{x_* \boldsymbol{X}} \left(\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}\right)^{-1} \boldsymbol{Y}_{:,k}\,,\end{aligned} \tag{17}$$

where σ^2 is a hyperparameter, the variance of the observation noise of the GP regression. We perform cross-validation to find a setting for σ^2. We use the eigendecomposition of $\boldsymbol{K}_{xx}$ to avoid the need to recompute the inverse for each value of σ^2.

In the next two experiments we investigate the cross-validation performance on subsets of CIFAR-10 for a sweep of correlation parameters on two different neural network architectures. We consider two architectures used in the neural network kernel literature, the CNN-GP [Novak et al., 2019, Arora et al., 2019] with 14 layers, and the Myrtle network [Shankar et al., 2020] with 10 layers. The CNNGP-14 architecture $((\texttt{conv}, \texttt{relu}) \times 14, \texttt{pool})$ has a 32×32-sized layer at the end, which is usually transformed into the 1×1 output using global average pooling. The Myrtle10 architecture $(((\texttt{conv}, \texttt{relu}) \times 2, \texttt{pool}_{2\times 2}) \times 3, \texttt{pool})$ has a 8×8 pooling layer at the end.

4.2 CORRELATED WEIGHTS IN THE LAST LAYER

We begin by investigating the addition of correlations in the weights of the final layer, since this is sufficient to prevent the disappearance of spatially correlated activations. Following Dutordoir et al. [2020], the covariance $\boldsymbol{\Sigma}_{pp'}$ of the weights is given by the Matérn-3/2 kernel with lengthscale λ:

$$\Sigma^{(L)}_{\boldsymbol{p},\boldsymbol{p}'} = \left(1 + \frac{\sqrt{3}\|\boldsymbol{p} - \boldsymbol{p}'\|_2}{\lambda}\right) \exp\left(-\frac{\sqrt{3}\|\boldsymbol{p} - \boldsymbol{p}'\|_2}{\lambda}\right). \tag{18}$$

where we see the patch locations $\boldsymbol{p}, \boldsymbol{p}'$ as vectors. The "extremes" of independent weights and mean pooling are represented by $\Sigma^{(L)}_{\boldsymbol{p},\boldsymbol{p}'} = \delta_{\boldsymbol{p},\boldsymbol{p}'}$ and $\Sigma^{(L)}_{\boldsymbol{p},\boldsymbol{p}'} = 1$, respectively. [3]

Figure 2, shows how the 4-fold cross-validation accuracy on the training sets varies with the lengthscale λ of the Matérn-3/2 kernel, which controls the "amount" of spatial correlation in the weights of the last layer. For each data point in each line, we split the data set into 4 folds, and we calculate the test accuracy on 1 fold using the other 3 as training set, for each value of σ that we try. We take the maximum accuracy over σ.

We investigate how the effect above varies with data set size. The results in fig. 2 show that particularly for the CNNGP-14 architecture, correlated weights in the final layer lead to a modest but consistent improvement in performance, with the effect becoming larger with increasing dataset size. We can also see the optimal lengthscale λ converging to a similar value for both architectures, of about $\lambda \approx 17$, which is evidence that the improvement holds for larger data sets. The optimal lengthscale is the same for both networks, so we speculate it may be a property of the CIFAR10 data set.

Data partitioning. The largest data set size in each part of the plot was run only once because of computational con-

[3] Strictly speaking, $\boldsymbol{\Sigma}^{(L)} = \boldsymbol{1}\boldsymbol{1}^\mathsf{T}$ corresponds to sum-pooling, but the missing constant $|\boldsymbol{F}|^{-2}$ does not affect the maximum in eq. 17.

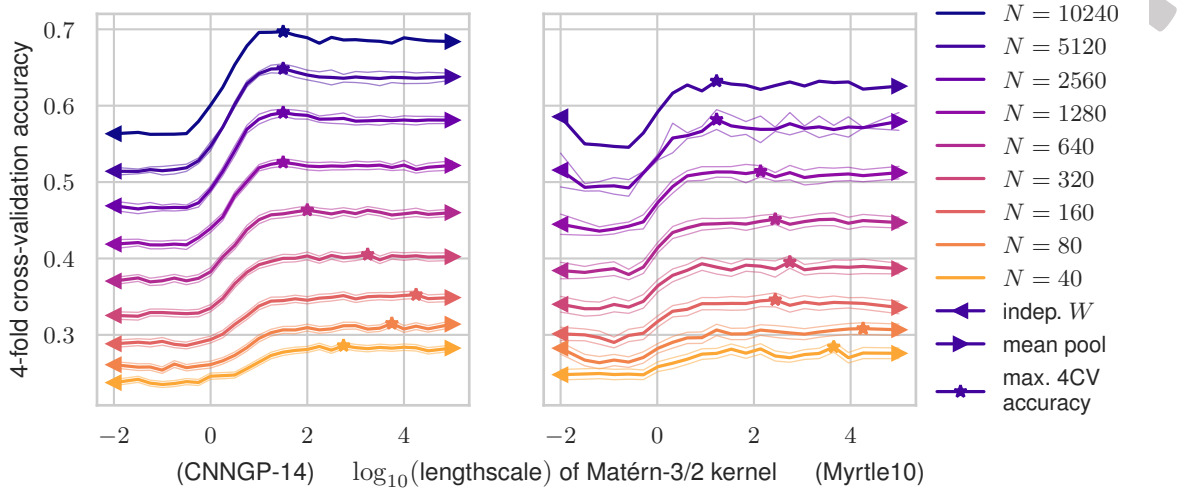

Figure 2: Cross-validation accuracy of the CNNGP-14 and Myrtle10 networks on subsets of CIFAR10, with varying lengthscale of the Matérn-$3/2$ kernel that determines the weight correlation in the last layer. With larger data set sizes N, the improvement is larger, and the optimal lengthscale λ converges to a similar value ($\lambda \approx 17$). For all data sets except the largest, the values are averaged over several runs, and the thin lines represent the $\pm 2\sigma_n$, the estimated standard deviation of the mean. We can improve the performance of the classifier by choosing an intermediate λ.

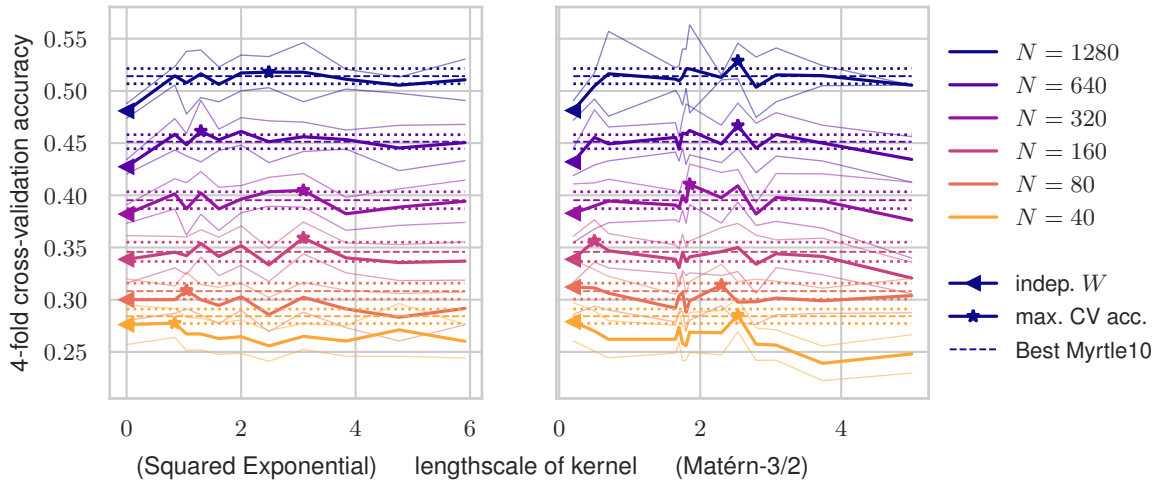

Figure 3: Correlated weights in intermediate layers. We replace pooling layers in the Myrtle10 architecture with larger convolutional filters with correlated weights. The lengthscale, and thus the amount of correlation, is varied along the x-axis. By adding correlations to a convolutional layer, we can recover (but not, in this case, exceed) the performance of the hand-selected architecture with mean-pooling.

straints. We transform one data set of size N into two data sets of size $N/2$ by taking block diagonals of the stored kernel matrix, so we have more runs for the smallest sizes. This is an unbiased Monte Carlo estimate of the true accuracy under the data distribution, since the individual data points are uniformly distributed (but not independent, since they are sampled without replacement). It also has less variance than independent data sets, because the data sets taken are anti-correlated; they have no points in common. Accordingly, the error bars in figs. 2 and 3 are an estimate of the standard error: the square root of an upwards-biased estimator of the variance of the mean.

Implementation. We use the `neural-tangents` [Novak et al., 2020] library to calculate the spatial kernel at the previous-to-last layer, $K^{(L-1)}(\boldsymbol{X}, \boldsymbol{X}')$, once. Since only the lengthscale of the last layer changes, we can cheaply obtain the final layer kernel matrix $\boldsymbol{K}^{(L)}_{\boldsymbol{X}\boldsymbol{X}'}$ for all lengthscales.

4.3 CORRELATED WEIGHTS IN INTERMEDIATE LAYERS

We take the same approach to the experiment in fig. 3. To investigate whether correlated weights can replace mean-pooling, we replace the 2×2 intermediate mean-pooling layer, together with the next 3×3 convolution layer, in the Myrtle10 architecture with correlated weights. We change them to a 6×6 weight-correlated convolution. We vary the lengthscale for the covariance of all the newly correlated layers, setting them to the same value.

We observe that for independent weights (lengthscale is 0) the performance of the network is significantly below the optimum. Correlating the weights improves performance, although after adding small amounts of correlation, performance stays roughly constant. This indicates that for intermediate layers mean-pooling is not a sub-optimal choice, as it is for the last layer. However, the amount of correlation is a continuous parameter, which could lead to avoiding this discrete choice in model architecture.

Implementation. In this experiment, the lengthscales vary across the whole network, so we need to calculate $K^{(L-1)}(\boldsymbol{X}, \boldsymbol{X}')$ every time. For a given data set size, this makes each point in fig. 3 considerably more expensive. For each data point, we optimise over the lengthscale of the last layer like in fig. 2, picking the one with highest cross-validation accuracy.

5 RELATED WORK

Infinitely wide limits of neural networks are currently an important tool for creating approximations and analyses. Here we provide a background on the different infinite limits that have been developed, together with a brief overview of where they have been applied.

Interest in infinite limits first started with research into properties of Bayesian priors on the weights of neural networks. Neal [1996] noted that prior function draws from a single hidden layer neural network with appropriate Gaussian priors on the weights tended to a Gaussian process as the width grew to infinity. The simplicity of performing Bayesian inference in Gaussian process models led to their widespread adoption soon after [Williams and Rasmussen, 1996, Rasmussen and Williams, 2006]. Over the years, the wide limits of networks with different weight priors and activation functions have been analysed, leading to various *kernels* which specify the properties of the limiting Gaussian processes [Williams, 1997, Cho and Saul, 2009].

With the increasing prominence of deep learning, recursive kernels were introduced in an attempt to obtain similar properties. Cho and Saul [2009], Mairal et al. [2014] investigated such methods for fully-connected and convolutional architectures respectively. Despite similarities between recursive kernels and neural networks, the derivation did not provide clear relationships, or any equivalence in a limit. Hazan and Jaakkola [2015] took initial steps to showing the wide limit equivalence of a neural network beyond the single layer case. Recently, Matthews et al. [2018], Lee et al. [2018] simultaneously provided general results for the convergence of the prior of deep fully-connected networks to a GP.[4] A different class of limiting kernels, the Neural Tangent Kernel (NTK), originated from analysis of the function implied by a neural network during optimisation [Jacot et al., 2018], rather than the prior implied by the weight initialisation. Just like the Bayesian prior limit, this kernel sheds light on certain properties of neural networks, as well as providing a method with predictive capabilities of its own. The two approaches end up with subtly different kernels, which both can be computed as a recursive kernel. Both such infinite limits have recently been used for predicting and analysing training properties of finite neural networks [Poole et al., 2016, Schoenholz et al., 2017, Hayou et al., 2019], as well as for (Bayesian) training of infinitely wide networks.

With the general tools in place, Garriga-Alonso et al. [2019], Novak et al. [2019] derived limits of the prior of convolutional neural networks with infinite filters. These two papers directly motivated this work by noting that spatial correlations disappeared in the infinite limit. Spatial mean pooling at the last layer was suggested as one way to recover correlations, with Novak et al. [2019] providing initial evidence of its importance. Due to computational constraints, they were limited to using a Monte Carlo approximation to the limiting kernel, while Arora et al. [2019] performed the computation

[4] The derivation of the limiting kernel differs between the two papers, with the results being consistent. Matthews et al. [2018] carefully take limits of realisable networks, while Lee et al. [2018] take the infinite limit of each layer sequentially.

with the exact NTK. Very recent preprints provide follow-on work that pushes the performance of limit kernels [Shankar et al., 2020] and demonstrated the utility of limit kernels for small data tasks [Arora et al., 2020]. Extending on the results for convolutional architectures, Yang [2019] showed how infinite limits could be derived for a much wider range of network architectures.

In the kernel and Gaussian process community, kernels with convolutional structure have also been proposed. Notably, these retained spatial correlation in either a fixed [van der Wilk et al., 2017] or adjustable [Mairal et al., 2014, Dutordoir et al., 2020] way. While these methods were not derived using an infinite limit, Van der Wilk [2019] provided an initial construction from an infinitely wide neural network limit. Inspired by these results, we propose limits of deep convolutional neural networks which retain spatial correlation in a similar way.

6 CONCLUSION

The disappearance of spatial correlations in infinitely wide limits of deep convolutional neural networks could be seen as another example of how Gaussian processes lose favourable properties of neural networks. While other work sought to remedy this problem by changing the architecture (mean-pooling), we showed that changing the weight prior could achieve the same effect. Our work has three main consequences:

1. Weight correlation shows that locally connected models (without spatial correlation) and mean-pooling architectures (with spatial correlation) actually exist at ends of a spectrum. This unifies the two views in the neural network domain. We also unify two known convolutional architectures that were introduced from the Gaussian process community.
2. We show empirically that performance improvements can be gained by using weight correlations *between* the extremes of locally connected networks or mean-pooling. We also show that mean-pooling in intermediate layers can be replaced by weight correlation in infinitely wide architectures.
3. Using weight correlation may provide advantages during hyperparameter tuning. Discrete architectural choices need to be searched through simple evaluation, while continuous parameters can use gradient-based optimisation. While we have not taken advantage of this in our current work, this may be a fruitful direction for future research.

Acknowledgements

The authors would like to thank the reviewers for helpful comments. AGA was supported by a UK Engineering and Physical Sciences Research Council studentship [1950008].

References

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 2019.

Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*. 2009.

Nicolas Durrande, David Ginsbourger, and Olivier Roustant. Additive covariance kernels for high-dimensional Gaussian process modeling. *Annales de la Faculté des sciences de Toulouse: Mathématiques*, Ser. 6, 21(3):481–499, 2012.

Vincent Dutordoir, Mark van der Wilk, Artem Artemev, and James Hensman. Bayesian image classification with deep convolutional Gaussian processes. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

David K. Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems 24 (NeurIPS)*. 2011.

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow Gaussian processes. In *7th International Conference on Learning Representations (ICLR)*, 2019.

Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2672–2680. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/hayou19a.html`.

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

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural

networks. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*. 2018.

Yann LeCun. Generalization and network design strategies. *Connectionism in perspective*, 19:143–155, 1989.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. In *6th International Conference on Learning Representation (ICLR)*, 2018.

David J. C. MacKay. Introduction to Gaussian processes. In *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer Academic Press, 1998.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*. 2014.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *6th International Conference on Learning Representations (ICLR)*, 2018.

Radford M. Neal. *Bayesian learning for neural networks*, volume 118. Springer, 1996.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are Gaussian processes. In *7th International Conference on Learning Representations (ICLR)*, 2019.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in Python. In *9th International Conference on Learning Representations (ICLR)*, 2020.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/148510031349642de5ca0c544f31b2ef-Paper.pdf.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning. 2006*. The MIT Press, Cambridge, MA, USA, 2006.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net, 2017. URL https://openreview.net/forum?id=H1W1UN9gg.

Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Jonathan Ragan-Kelley, Ludwig Schmidt, and Benjamin Recht. Neural kernels without tangents. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Charles J. Stone. Additive regression and other nonparametric models. *Annals of Statistics*, 13(2):689–705, 06 1985. doi: 10.1214/aos/1176349548.

Mark van der Wilk. *Sparse Gaussian process approximations and applications*. PhD thesis, University of Cambridge, 2019.

Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*. 2017.

Christopher K. I. Williams. Computing with infinite networks. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9 (NIPS 1996)*. 1997.

Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8 (NeurIPS 1995)*. 1996.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are Gaussian processes. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 2019.