Supplementary Materials for The Neural Tangent Link Between CNN Denoisers and Non-Local Filters

Julián Tachella, Junqi Tang and Mike Davies*
School of Engineering
University of Edinburgh
{julian.tachella, j.tang, mike.davies}@ed.ac.uk

This manuscript contains the supplementary materials supporting the paper entitled "The Neural Tangent Link Between CNN Denoisers and Non-Local Filters". It provides a more detailed derivation of the formulas in the main paper. We repeat some results associated with neural tangent kernel (NTK) that already exist in the literature [5, 1, 7] for the sake of clarity and to make the document as self-contained as possible. The supplementary note is organized as follows: Section A states the main assumptions made for the derivations, discussing their implications. Section B reviews the dynamics of signal propagation in wide deep neural networks. Section C extends the analysis to backward signal propagation during backpropagation training. Section D presents a detailed derivation of the neural tangent kernel and the filtering process. Section E extends the results for multiple input and output channels and Section F discusses the effect of (linear) downsampling operations of the autoencoder and U-Net architectures. Section G explains the proposed Nyström method for efficient filtering with the neural tangent kernel. Section H details the network architectures evaluated in the experiments. Finally, additional denoising results are presented in Section I.

A Assumptions and other observations

1. We have omitted the use of biases to simplify the presentation. In the case of relu non-linearities, the presence of biases would add an additional constant term to the V and V' maps in eqs. (13) and (24) [1]. We also found that the denoising performance did not vary significantly with or without them (for relu non-linearities). Moreover, it has been recently shown that bias-free denoisers generalize better for different noise levels [8].

2. We focus on the case where all hidden layers have the same number of channels c. Our analysis can be easily extended for different number of channels per layer, as long as they all grow at the same rate when taking $c \to \infty$ [6].

3. Despite we assume that the output z has a single channel for the main derivations, the theory applies to a variable number of channels c_L, as long as they are significantly smaller than the ones of the hidden layers c. The extension to multiple channels is provided in section E.

4. We drop the dependence of the pre-activations a^ℓ on the input x to lighten notations. In this document we assume that the NTK is fixed throughout training, and hence drop the iteration superscript to lighten the notation.

*The code associated with this work is available at https://gitlab.com/Tachella/neural_tangent_denoiser

5. For ease of presentation, we focus on the case where all layers have the same image size $d_\ell = d$. Section F extends the results for downsampling and upsampling layers of U-Net and autoencoder architectures.

6. It is worth noting that some architectures proposed in the deep image prior paper [11] have a number of input channels of order $\mathcal{O}(c)$. However, we noticed that reducing the number of channels does not impact significantly the performance.

7. To the best of our knowledge, the theory presented here cannot not be straightforwardly applied to networks with batch normalization and max pooling. However, we noted that they do not affect significantly the denoising performance of the networks.

B Forward signal propagation

In this section we study the statistics of the signal as it propagates through the neural network. As $c \to \infty$, the preactivations at each layer a_i^ℓ can be well described by a multivariate Gaussian distribution due to the central limit theorem [9]. Hence, computing the mean and covariance is enough to fully characterize their distribution. For the first hidden layer we have, for each channel $i = 1, \dots, c$, mean

$$\mu_{a^1} = \mathbb{E}\{W_{i,1}^1\}\mathbb{E}\{x\} \tag{1}$$
$$= 0 \tag{2}$$

and covariance

$$\Sigma_{a^1} = \mathbb{E}\{W_{i,1}^1 x x^\top (W_{i,1}^1)^\top\} \tag{3}$$

where the independence of weights across different filters was used to simplify the sum. Note that we have dropped the dependence of the mean and covariance on the specific channel i, as all channels share the same mean and covariance. The expression in eq. (3) consists of pairwise expectations

$$\mathbb{E}\{[W_{i,j}^\ell x]_\mu [W_{i,j}^\ell x]_v\} = \frac{1}{r^2} \sum_{\mu', v'} x_{\mu'} x_{v'} \tag{4}$$

where μ' and v' are the indices of pixels within patches of size $r \times r$ centered at μ and v respectively. It can be written in a more compact form as

$$\Sigma_{a^\ell} = \mathcal{A}\left(x x^\top\right) \tag{5}$$

where the convolution map $\mathcal{A} : \mathrm{PSD}_n \mapsto \mathrm{PSD}_n$ is defined as [13]

$$[\mathcal{A}(\Sigma)]_{\mu,v} = \frac{1}{r^2} \sum_{\mu', v'} [\Sigma]_{\mu', v'} \tag{6}$$

For the following layers we also have zero mean, i.e.,

$$\mu_{a^\ell} = \sum_{j=1}^{c} \mathbb{E}\{W_{i,j}^\ell\}\mathbb{E}\{\phi\left(a_j^{\ell-1}\right)\} \tag{7}$$
$$= 0 \tag{8}$$

and a covariance is given by

$$\Sigma_{a^\ell} = \sum_{j=1}^{c} \mathbb{E}\{W_{i,j}^{\ell-1} \phi\left(a_j^{\ell-1}\right) \phi\left(a_j^{\ell-1}\right)^\top \left(W_{i,j}^{\ell-1}\right)^\top\} \tag{9}$$

where the first term of the right hand side is given by

$$\mathbb{E}\{[W_{i,j}^{\ell} \phi\left(a_j^{\ell-1}\right)]_\mu [W_{i,j}^{\ell} \phi\left(a_j^{\ell-1}\right)]_v\} = \sum_{\mu',v'} \mathbb{E}\{\phi\left(a_{j,\mu'}^{\ell-1}\right) \phi\left(a_{j,v'}^{\ell-1}\right)\} \tag{10}$$

The expression can be written in compact form as

$$\Sigma_{a^\ell} = \mathcal{A}\left(V\left(\Sigma_{a^{\ell-1}}\right)\right) \tag{11}$$

where the map $V : \mathrm{PSD}_n \mapsto \mathrm{PSD}_n$ linked to a non-linearity $\phi(x)$ is defined as

$$V(\Sigma) = \sigma_w^2 \mathbb{E}_{h\sim\mathcal{N}(0,\Sigma)}\{\phi(h)\,\phi\left(h^\top\right)\} \tag{12}$$

The V-map consists of two-dimensional integrals that are available in closed-form for many activation functions. In the case of relu non-linearities, we have [3]

$$[V(\Sigma)]_{\mu,v} = \frac{\sqrt{\Sigma_{\mu,\mu}\Sigma_{v,v}}}{\pi}(\sin(\varphi) + (\pi - \varphi)\cos(\varphi)) \tag{13}$$

where $\varphi = \arccos(\Sigma_{\mu,v}/\sqrt{\Sigma_{\mu,\mu}\Sigma_{v,v}})$. As discussed in [13], ℓ repeated applications of the operator given by eq. (13) quickly converge to a matrix of the form

$$[\Sigma]_{\mu,v} = \begin{cases} 1 & \text{if } \mu = v \\ \kappa_\ell & \text{otherwise} \end{cases} \tag{14}$$

where k_ℓ decreases to zero exponentially fast with depth. Note that the matrix in eq. (14) is invariant to the $\mathcal{A}$ map, as the diagonal elements are averaged with other diagonal elements, whereas the off-diagonal entries are averaged with other off-diagonal ones.

The output z is also characterized by a multivariate Gaussian distribution with

$$\Sigma_z = \mathcal{A}\left(V\left(\Sigma_{a^{L-1}}\right)\right). \tag{15}$$

The main difference between the fully connected and convolutional architectures lies in the covariance Σ_{a^ℓ}. In the fully connected case, $\mathcal{A}$ boils down to the identity operator, and Σ_{a^ℓ} has an isotropic structure for all layers, whereas the convolutional network presents rich covariances within the pixels of each channel in eq. (9), as $\mathcal{A}$ cross-correlates different patches of the image.

B.1 Gaussian process interpretation

We can use the distribution of an infinite neural network at initialization to define a prior $p(z) = \mathcal{N}(0, \Sigma_z)$ for images, following a Bayesian inference viewpoint [9], a strategy named the Bayesian deep image prior in [2]. In the case of standard Gaussian noise $z = y + n$ we have

$$y|z \sim \mathcal{N}(z, \sigma_n^2 I) \tag{16}$$

$$z \sim \mathcal{N}(0, \Sigma_z) \tag{17}$$

where the posterior distribution is available in closed form

$$z|y \sim \mathcal{N}\left((I + \sigma_n^2 \Sigma_z^{-1})^{-1} z, (I\sigma_n^{-2} + \Sigma_z^{-1})^{-1}\right) \tag{18}$$

Note that, if iid noise is placed at the input of the network, Σ_z does not depend on the noise image z in any way. Moreover, for a relu network, this covariance is given by eq. (14). Figure 1 shows that the off-diagonal elements κ_L tend to 1 as the network becomes larger. This prior just promotes constant images.

C Backward signal propagation

A similar analysis can be made for the propagation of gradients through the network in backwards direction. This is especially useful to study the behaviour of backpropagation training and avoid vanishing or exploding gradients in deep networks. Computing gradients with respect to the weights of the ℓth layer can be done using the chain rule:

$$\frac{\delta \mathcal{L}}{\delta w^\ell} = \frac{\delta \mathcal{L}}{\delta z} \frac{\delta z}{\delta a^{L-1}} \cdots \frac{\delta a^{\ell+1}}{\delta a^\ell} \frac{\delta a^\ell}{\delta w^\ell} \tag{19}$$

We define the gradient as:

$$\delta_i^\ell \stackrel{\text{def}}{=} \frac{\delta \mathcal{L}}{\delta z} \frac{\delta z}{\delta a^{L-1}} \cdots \frac{\delta a^\ell}{\delta a_i^{\ell-1}} \in \mathbb{R}^d \tag{20}$$

with $\delta^L \stackrel{\text{def}}{=} \frac{\delta \mathcal{L}}{\delta z}$. For a squared loss, the gradient at the last layer is

$$\delta^L = z - y. \tag{21}$$

Assuming that independence between gradients and preactivations [13][1], we have for each channel $i = 1, \ldots, c$ of layer $L-1$

$$\delta_i^{L-1} = \operatorname{diag}\left(\phi'\left(a_i^{L-1}\right)\right) (W_{1,i}^L)^\top \delta^L \tag{22}$$

which has zero mean and covariance given by

$$\Sigma_{\delta^{L-1}} = \frac{1}{c} V'\left(\Sigma_{a^{L-1}}\right) \circ \mathcal{A}\left(\Sigma_{\delta^L}\right) \tag{23}$$

where the map $V' : \mathrm{PSD}_n \mapsto \mathrm{PSD}_n$ is defined as

$$V'(\Sigma) = \sigma_w^2 \mathbb{E}_{h \sim \mathcal{N}(0,\Sigma)}\{\phi'(h)\, \phi'\left(h^\top\right)\} \tag{24}$$

The expected values are available in closed form for many non-linearities. We can use the following recursive formula to compute the rest of the layers $\ell = L-2, \ldots, 1$

$$\delta_i^\ell = \sum_{j=1}^{C^\ell} \operatorname{diag}\left(\phi'\left(a_i^{\ell-1}\right)\right) (W_{j,i}^\ell)^\top \delta_j^{\ell+1} \tag{25}$$

Computing the propagation recursively in backwards direction, we have $\mu_{\delta^\ell} = 0$ and covariance

$$\Sigma_{\delta_i^\ell} = \mathcal{A}\left(\Sigma_{\delta^{\ell+1}}\right) \circ V'\left(\Sigma_{a^\ell}\right) \tag{26}$$

[1] This assumption is formally justified in a recent work [14].

For relu non-linearities the V' map is computed as

$$[V'(\Sigma)]_{\mu,v} = 1 - \frac{1}{\pi}\arccos\frac{\Sigma_{\mu,v}}{\sqrt{\Sigma_{\mu,\mu}\Sigma_{v,v}}} \tag{27}$$

which as with the V counterpart[2], repeated applications of this map converge exponentially fast to the simple matrix structure in eq. (14).

D Neural Tangent Kernel

In this section, we will denote all the trainable network parameters at iteration t as w^t. Consider training a network via gradient descent[3], that is

$$w^{t+1} = w^t - \eta\frac{\delta\mathcal{L}}{\delta w}(w^t) \tag{28}$$

We can study the evolution of the function defined by the weights $z^t \overset{\text{def}}{=} z(w^t)$, using a first order Taylor expansion, i.e.,

$$z^{t+1} \approx z(w^t) + \frac{\delta z}{\delta w}(w^{t+1} - w^t) \tag{29}$$

$$\approx z^t - \eta\frac{\delta z}{\delta w}\frac{\delta\mathcal{L}}{\delta w} \tag{30}$$

$$\approx z^t - \eta\frac{\delta z}{\delta w}(\frac{\delta z}{\delta w})^\top\frac{\delta\mathcal{L}}{\delta z} \tag{31}$$

where we have used eq. (28) in the second line and the chain rule in the third line. The neural tangent kernel (NTK) is given by

$$\Theta_L = \frac{\delta z}{\delta w}(\frac{\delta z}{\delta w})^\top \tag{32}$$

$$= \sum_{\ell,i,j,\alpha}\frac{\delta z}{\delta w^\ell_{i,j,\alpha}}(\frac{\delta z}{\delta w^\ell_{i,j,\alpha}})^\top \tag{33}$$

We can start with the base case,

$$\Theta_2 = cV\left(\mathcal{A}\left(xx^\top\right)\right) \tag{34}$$

and notice the following recursive formulation

$$\Theta_\ell = \frac{\delta a^\ell_i}{\delta w^\ell}(\frac{\delta a^\ell_i}{\delta w^\ell})^\top + \frac{\delta a^\ell_i}{\delta a^{\ell-1}}\Theta_{\ell-1}(\frac{\delta a^\ell_i}{\delta a^{\ell-1}})^\top \tag{35}$$

$$= \sum_{j=1}^{c}\mathcal{A}\left(\phi\left(a^{\ell-1}_j\right)\phi\left(a^{\ell-1}_j\right)^\top\right) + W^\ell_{i,j}\text{diag}\left(\phi'\left(a^{\ell-1}_j\right)\right)\Theta_{\ell-1}\text{diag}\left(\phi'\left(a^{\ell-1}_j\right)\right)\left(W^\ell_{i,j}\right)^\top \tag{36}$$

where w^ℓ denotes the weights corresponding to layer ℓ. The learning rate η is chosen of order $\mathcal{O}(c^{-1})$, in order to converge to global minimum [6]. Without loss of generality, we use $\eta = \gamma c^{-1}$ for

[2]Note that the discontinuity of the relu function at 0 is unimportant here due to the expectation operator.
[3]A very similar analysis can be done for gradient flow and stochastic gradient descent [7]

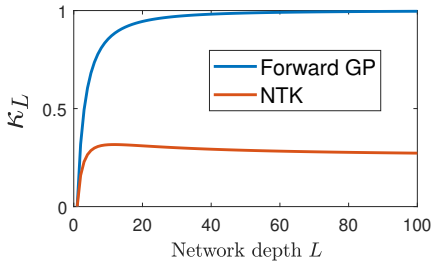

Figure 1: Off-diagonal elements of the filtering matrix associated with the Gaussian process at initialization and the neural tangent kernel with iid noise input.

the following derivations, where γ is $\mathcal{O}(1)$ and chosen such that the neural tangent kernel has its eigenvalues bounded by 1. As shown in [14], for an infinite number of channels $c \to \infty$, due to the law of large numbers we have

$$\eta\Theta_\ell = \Sigma_{a^\ell} + \mathcal{A}\left(V'\left(\Sigma_{a^\ell}\right) \circ \eta\Theta_{\ell-1}\right) \tag{37}$$

which is a fixed (deterministic) matrix. As a function of the input image (or noise) patches, the NTK defines a kernel acting on pairs of input patches x_1 and x_2, i.e., $k(x_1, x_2) : \mathbb{R}^{d_0} \times \mathbb{R}^{d_0} \mapsto \mathbb{R}_+$. As discussed in the main paper, if iid noise is placed at the input, the resulting Gram matrix is given by eq. (14) with κ_L as shown in Figure 1.

For a squared loss $\mathcal{L} = \frac{1}{2}\|z - y\|_2^2$, the dynamics of eq. (29) can be written as

$$z^{t+1} = z^t + \eta\Theta_L\left(y - z^t\right) \tag{38}$$

$$= (I - \eta\Theta)^{t+1} z^0 + \sum_{k=1}^{t} (\eta\Theta_L)^k y \tag{39}$$

with initial condition z^0 given by the Gaussian process initialization described in Section B. The expression for z^t can be simplified further by noting that the learning rate has to be chosen such that $\eta\Theta$ has its eigenvalues bounded from above by 1 (to avoid a diverging gradient descent). Hence, as $I - \eta\Theta$ is invertible, we can apply the geometric series formula

$$z^t = (I - \eta\Theta_L)^t z^0 + (I - \eta\Theta_L)^{-1}\left(I - (\eta\Theta_L)^t\right) y \tag{40}$$

Note that the only random component of this equation is the Gaussian process initialization z^0. As z^t is an affine transformation of a Gaussian process, it is also itself a Gaussian process for every iteration t. Hence, we have

$$z^t \sim \mathcal{N}\left((I - \eta\Theta)^{-1}\left(I - (\eta\Theta)^t\right) y, (I - \eta\Theta_L)^t \Sigma_z (I - \eta\Theta)^t\right) \tag{41}$$

It is easy to see that z^t converges at an exponential rate towards a singular distribution centered at y as $t \to \infty$.

E Multiple input and output channels

The theory applies for any number of input and output channels, as long as they are much smaller than the number of hidden channels c. A multi-channel input modifies the computation in the first

layer eq. (3). In this case, first multiplying the patches channel-wise and then summing the result, that is

$$\Sigma_{a^1} = \sum_{j=0}^{c_0} \mathbb{E}\{W^1_{i,j} x_j x_j^\top (W^1_{i,j})^\top\} \tag{42}$$

where x_j denotes the jth channel of the input, and the corresponding infinite-width operator is computed as

$$\Sigma_{a^1} = \frac{1}{c_0} \sum_{j=0}^{c_0} \mathcal{A}\left(x_j x_j^\top\right) \tag{43}$$

Hence, the pixel affinity function is now defined for a receptive field $d_0 \leq d$, and patches x_1 and x_2 of c_0 channels as

$$k(x_1, x_2) : \mathbb{R}^{c_0 d_0} \times \mathbb{R}^{c_0 d_0} \mapsto \mathbb{R}_+. \tag{44}$$

Multiple output channels are computed separately using the same filtering matrix, i.e.,

$$z_i^{t+1} = z_i^t + \eta \Theta_L (y - z_i^t) \tag{45}$$

for $i = 1, \ldots, c_L$. Note that both the color versions of NLM and BM3D do a similar procedure, computing the filtering matrix with luminance (i.e., a linear combination of the RGB channels), and apply the filtering process to each channel separately.

F Downsampling and upsampling layers

Downsampling can be achieved either via 2-strided convolutional layers or directly with linear downsampling operations, such as bilinear or nearest neighbor downsampling. Strided convolutions are a straightforward extension of the $\mathcal{A}$ operator defined in eq. (6), summing over strided patches instead of contiguous ones. Linear downsampling operations can be expressed as a matrix vector product applied channel-wise, i.e., $a_i^{\ell+1} = D a_i^\ell$ where $D \in \mathbb{R}^{d \times d/2}$ is a fixed matrix given by downsampler (bilinear, nearest neighbor, etc.). The covariance of $a_i^{\ell+1}$ is then

$$\Sigma_{a^{\ell+1}} = D \Sigma_{a^\ell} D^\top. \tag{46}$$

Upsampling is generally performed with bilinear or nearest neighbor layers, as transposed convolutions provide worse results [11]. These are analogous to the downsampling case, but with an upsampling matrix $U \in \mathbb{R}^{d/2 \times d}$, that is

$$\Sigma_{a^{\ell+1}} = U \Sigma_{a^\ell} U^\top. \tag{47}$$

G Nyström denoising

The Nystrom method approximates the first m eigenvectors of the NTK matrix by computing only a subset of $m \ll d$ columns [12], i.e., the sub-matrix

$$\Theta_{d,m} = \begin{bmatrix} \Theta_{m,m} \\ \Theta_{d-m,m} \end{bmatrix} \tag{48}$$

Module	Function	Infinite-channel forward operator
input	3 channel RGB image	
conv1	11×11 pixel convolution	$\mathcal{A}$ with $r = 11$
relu1	relu activation $\max(x, 0)$	V
conv2	1×1 pixel convolution	$\mathcal{A}$ with $r = 1$
output	3 channel RGB image	

Table 1: Vanilla configuration with a single-hidden layer.

We first perform a singular value decomposition of the small sub-matrix $\Theta_{m,m} = \sum_{i=1}^{m} \tilde{\lambda}_i \tilde{v}_i \tilde{v}_i^\top$, and then approximate the eigenvectors and eigenvalues of the full matrix as

$$v_i = \sqrt{\frac{m}{d}} \frac{1}{\tilde{\lambda}_i} \Theta_{d,m} \tilde{v}_i \tag{49}$$

$$\lambda_i = \frac{d}{m} \tilde{\lambda}_i \tag{50}$$

We fix $m = 0.02d$, which allows us to compute most of the md pixel affinities in parallel on the GPU. The selection of columns is done similarly to global image denoising [10], choosing a random selection of pixels uniformly distributed in space. Before applying the denoising procedure, we scale the eigenvalues, such that the maximum eigenvalue is 1.

H Architectures

H.1 Vanilla CNN

Table 1 shows the configuration used for the vanilla CNN results with $c = 512$ channels per hidden layer. The network has a total of 187,392 trainable weights.

H.2 Autoencoder

Table 2 shows the configuration used for the autoencoder results with $c = 128$ channels per hidden layer. The network has a total of 1,036,032 trainable weights.

H.3 U-Net

The U-Net considered in this paper shares the same architecture and number of weights than the autoencoder, adding skip connections at each level.

I Additional results

In all the denoising experiments, we normalize the corrupted images by subtracting 0.5 from all pixels, such that they defined in the centered interval $[-0.5, 5]$. Before computing the PSNR, we denormalize the images by summing 0.5 to all pixels and clipping, such that all pixels are in the interval $[0, 1]$.

Module	Function	Infinite-channel forward operator
input	3 channel RGB image	
convd1	3×3 convolution	$\mathcal{A}$ with $r = 3$
relu1	relu activation $\max(x, 0)$	V
down1	Bilinear downsampling	D
convd2	3×3 convolution	$\mathcal{A}$ with $r = 3$
relu2	relu activation $\max(x, 0)$	V
down2	Bilinear downsampling	D
convd3	3×3 convolution	$\mathcal{A}$ with $r = 3$
relu3	relu activation $\max(x, 0)$	V
down3	Bilinear downsampling	D
convd4	3×3 convolution	$\mathcal{A}$ with $r = 3$
relu4	relu activation $\max(x, 0)$	V
conv4	3×3 convolution	$\mathcal{A}$ with $r = 3$
up1	Bilinear upsampling	U
convu1	3×3 convolution	$\mathcal{A}$ with $r = 3$
relu5	relu activation $\max(x, 0)$	V
up2	Bilinear upsampling	U
convu2	3×3 convolution	$\mathcal{A}$ with $r = 3$
relu6	relu activation $\max(x, 0)$	V
up3	Bilinear upsampling	U
convu3	3×3 convolution	$\mathcal{A}$ with $r = 3$
relu7	relu activation $\max(x, 0)$	V
convu4	1×1 convolution	$\mathcal{A}$ with $r = 1$
output	3 channel RGB image	

Table 2: Autoencoder configuration with bilinear downsampling and upsampling layers.

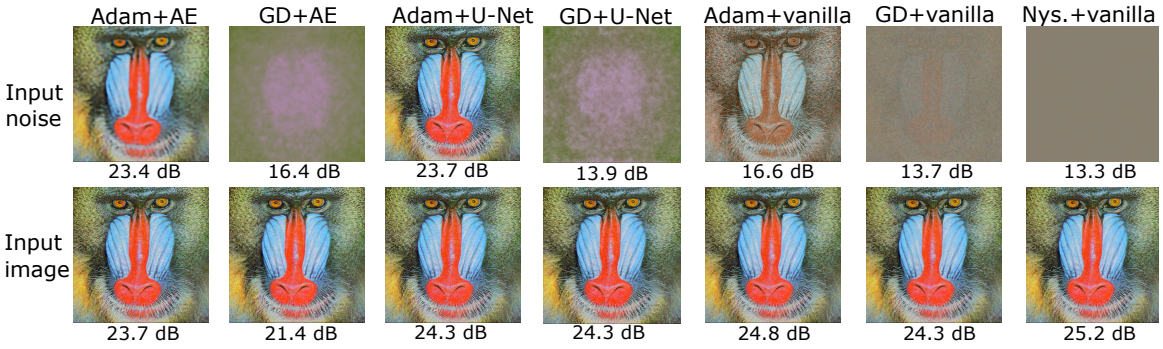

Figure 2: Results for the 'baboon' image. PSNR values are reported below each restored image. The best results are obtained by the Nyström approximation of a vanilla CNN filter.

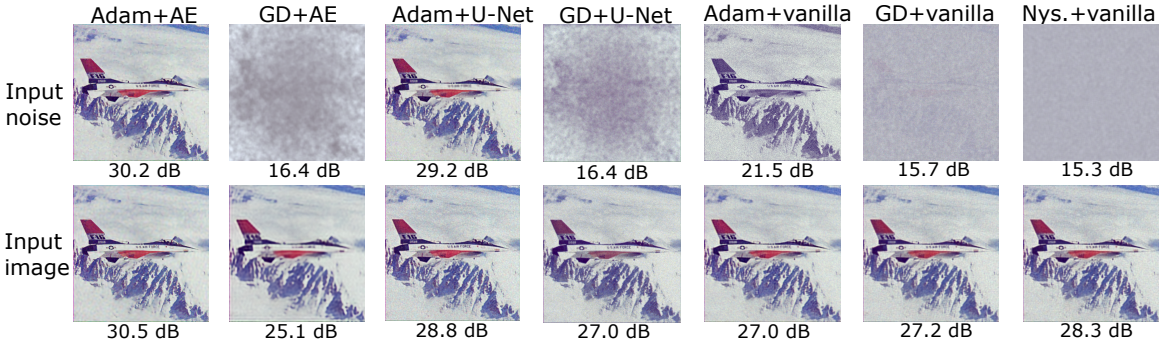

Figure 3: Results for the 'F16' image. PSNR values are reported below each restored image. The best results are obtained by an autoencoder trained with Adam, which is able to provide smoother estimates while preserving sharp edges.

I.1 Denoising examples

The deep image prior setting (autoencoder, noise input and Adam optimizer), performs very well in images with large piece-wise smooth patches, such as the 'house' image shown in the main paper or the 'F16' image in Figure 3, but does not provide good reconstructions in images with noise-like textures, such as the 'baboon' shown in Figure 2. The best performing denoiser for this image is the closed form filter associated with a vanilla CNN, approximated with Nyström.

I.2 Additional noise levels

We evaluate the best-performing denoisers (autoencoder with noise or image input trained using Adam and Nyström approximation of a vanilla CNN) for iid Gaussian noise with standard deviations of $\sigma = 5$ (low noise) and $\sigma = 100$ (high noise). Table 3 shows the results for the dataset of 9 color images [4]. Inputting the image when using Adam achieves an improvement of 1.8 dB in the low-noise case, whereas it provides slightly worse (0.3 dB) results in the high noise case.

	AE/Adam/noise	AE/Adam/image	Vanilla/Nyström/image
$\sigma = 5$	33.5	**35.3**	34.5
$\sigma = 100$	**24.4**	24.1	22.3

Table 3: Average PSNR [dB] obtained by the best-performing algorithms for different noise levels.

I.3 Epoch count

Table 4 shows the average epoch-count of all methods for the 9 color image dataset. Inputting the image instead of noise reduces the number of iterations when optimizing with Adam, as the induced filtering matrix is better conditioned. Gradient descent requires many more iterations than Adam as it does not uses any momentum. As discussed in the main paper, the filtering matrix associated with a vanilla CNN and noise input is so ill-conditioned that gradient descent does not converge even after 10^6 iterations.

	Vanilla CNN		U-Net		Autoencoder	
	Noise	Image	Noise	Image	Noise	Image
Adam	145340	64	7692	74	10248	5088
Gradient descent	$> 10^6$	69526	50054	5506	50355	286042
Nyström	368	504				

Table 4: Average epoch-count by different combinations of network architecture, input and optimizer on the dataset of 9 color images [4].

References

[1] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems 32*, pages 8141–8150. Curran Associates, Inc., 2019. 1

[2] Z. Cheng, M. Gadelha, S. Maji, and D. Sheldon. A Bayesian perspective on the deep image prior. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5438–5446, 2019. 3

[3] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009. 3

[4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 10, 11

[5] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31*, pages 8571–8580. Curran Associates, Inc., 2018. 1

[6] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. In *Prof. of Mach. Learning Research*, volume 89, pages 1032–1041. PMLR, 16–18 Apr 2019. 1, 5

[7] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models

under gradient descent. In *Advances in Neural Information Processing Systems 32*, pages 8572–8583. Curran Associates, Inc., 2019. 1, 5

[8] Sreyas Mohan, Zahra Kadkhodaie, Eero P. Simoncelli, and Carlos Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. In *International Conference on Learning Representations*, 2020. 1

[9] Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995. 2, 3

[10] H. Talebi and P. Milanfar. Global image denoising. *IEEE Transactions on Image Processing*, 23(2):755–768, 2014. 8

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018. 2, 7

[12] Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001. 7

[13] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. *arXiv preprint arXiv:1806.05393*, 2018. 2, 3, 4

[14] Greg Yang. Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation. *arXiv e-prints*, page arXiv:1902.04760, Feb. 2019. 4, 6