# Neural networks for signal processing reinvent (and improve) the wheel


**Carlos Fernandez-Granda**

www.cims.nyu.edu/~cfgranda


11/22/2019

Data-driven estimation of sinusoid frequencies

Blind denoising of natural images

Bias-free CNNs

Wiener filtering

CNNs learn adaptive filters

CNNs learn unions of subspaces

Data-driven estimation of sinusoid frequencies

Blind denoising of natural images

Bias-free CNNs

Wiener filtering

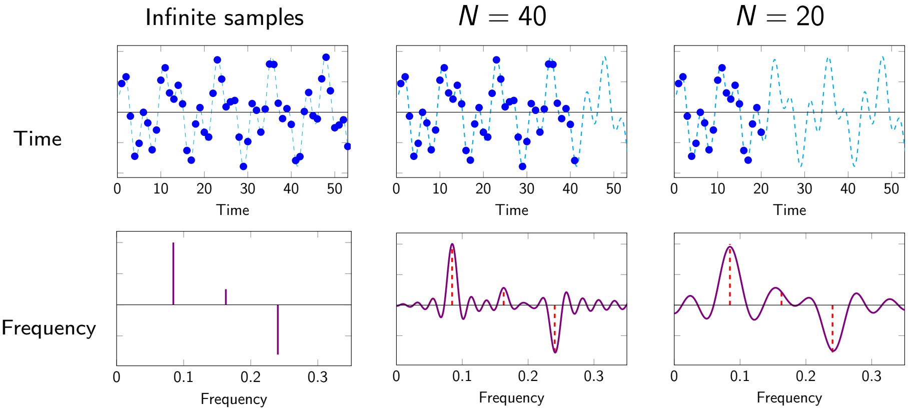
CNNs learn adaptive filters

CNNs learn unions of subspaces

# Data-driven estimation of sinusoid frequencies

Joint work with Brett Bernstein, Gautier Izacard, and Sreyas Mohan

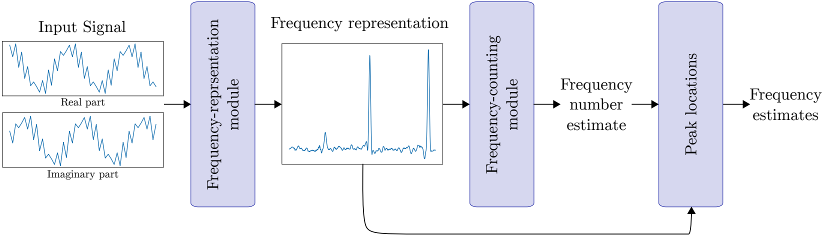
# Frequency estimation (aka super-resolution of line spectra)



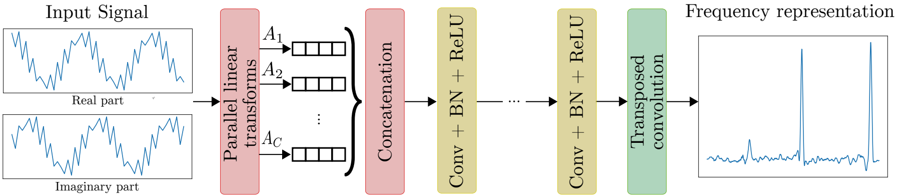
# Traditional methodology

- Linear estimation (periodogram)
- Parametric methods based on eigendecomposition of sample covariance matrix (MUSIC, ESPRIT, matrix pencil)
- Sparsity-based methods

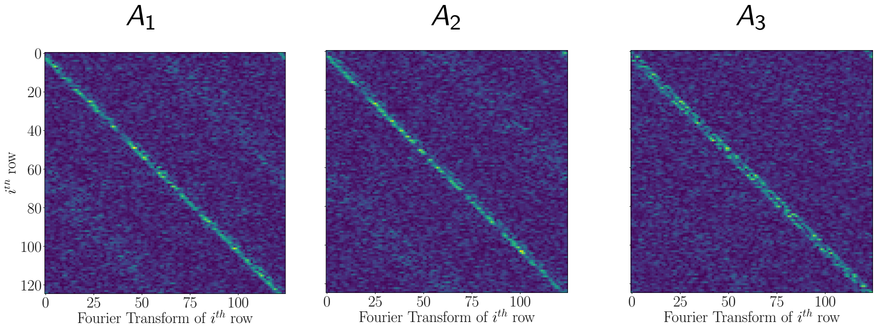
# Learning-based approach



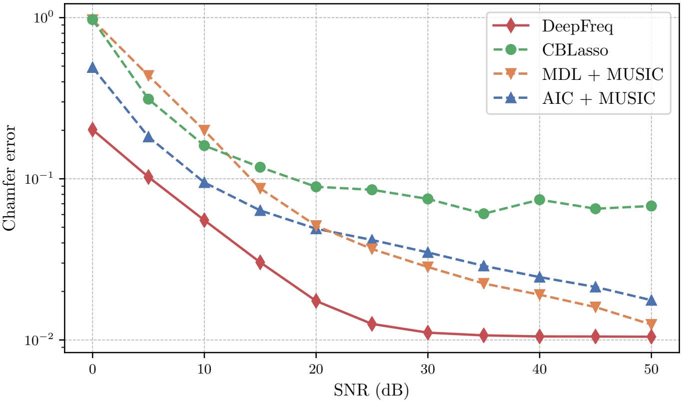
# Frequency-representation module



# Fourier transform of learned transformations



# Comparison to state of the art



# For more information

**A Learning-Based Framework for Line-Spectra Super-resolution**. G. Izacard, B. Bernstein, C. Fernandez-Granda. ICASSP 2019

**Data-driven Estimation of Sinusoid Frequencies**. G. Izacard, S. Mohan, C. Fernandez-Granda. NeurIPS 2019

Data-driven estimation of sinusoid frequencies

Blind denoising of natural images

Bias-free CNNs

Wiener filtering

CNNs learn adaptive filters

CNNs learn unions of subspaces

# Acknowledgements

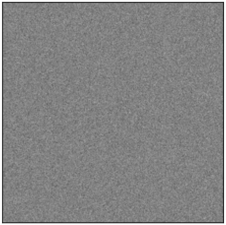
Joint work with Zahra Kadkhodaie, Sreyas Mohan, and Eero Simoncelli

# Image denoising

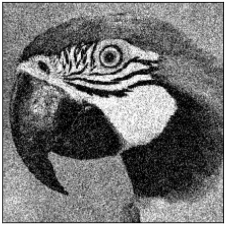
Goal: Estimate image from noisy data

Popular (yet somewhat *unrealistic*) model: Additive Gaussian noise

$+$



$=$

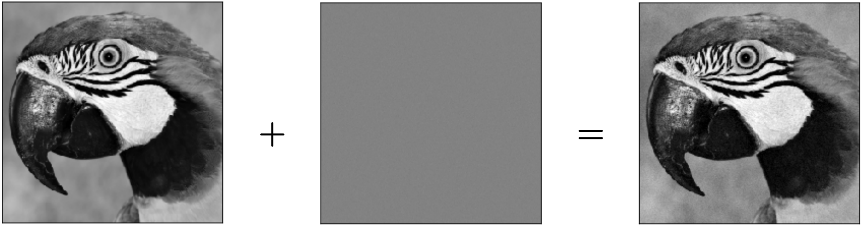


Blind denoising: Noise level is unknown

# Image denoising

Goal: Estimate image from noisy data

Popular (yet somewhat *unrealistic*) model: Additive Gaussian noise

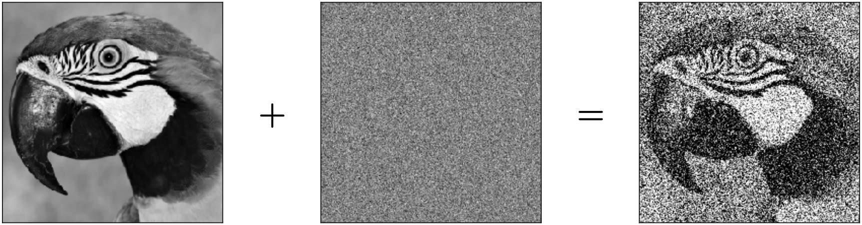


Blind denoising: Noise level is unknown

# Image denoising

Goal: Estimate image from noisy data

Popular (yet somewhat *unrealistic*) model: Additive Gaussian noise



Blind denoising: Noise level is unknown

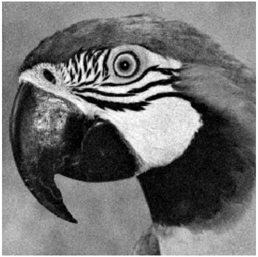
# Deep learning for blind image denoising

- Gather dataset of natural images
- Add noise from a range of noise levels
- Train CNN to estimate clean image minimizing mean squared error
- Works very well for additive Gaussian noise (state of the art)

# Generalization across noise levels

What if we test on noise level not seen during training?

Training data
(low noise)

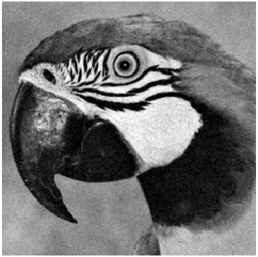


Test image
(high noise)

# Generalization across noise levels

What if we test on noise level not seen during training?

Training data
(low noise)



Test image
(high noise)

CNN

# First-order Taylor expansion

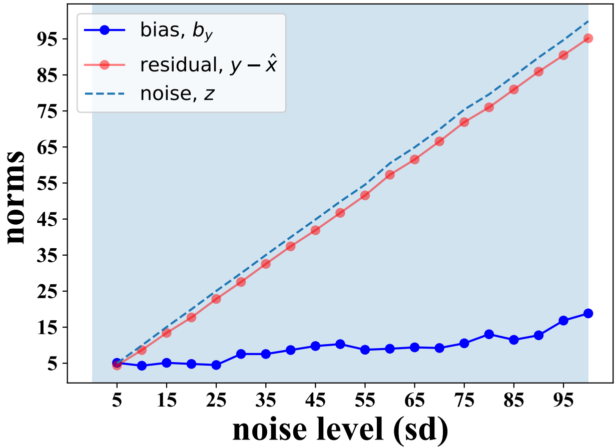
Let $f$ be the function learned by a CNN trained for denoising

The first-order Taylor expansion for a fixed input $y$ is exact

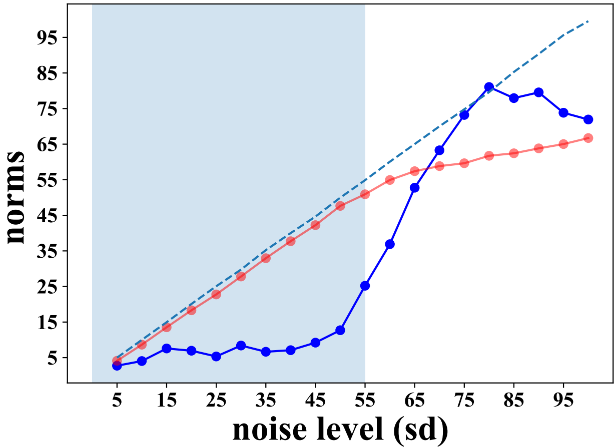
$$\begin{aligned}\hat{x} = f(y) &= W_L R(\ldots W_2 R(W_1 y + b_1) + b_2 \ldots) + b_L \\ &= A_y y + b_y\end{aligned}$$

$W_1$, $W_2$, ..., $W_L$ are weight matrices
$b_1$, $b_2$, ..., $b_L$ are bias vectors

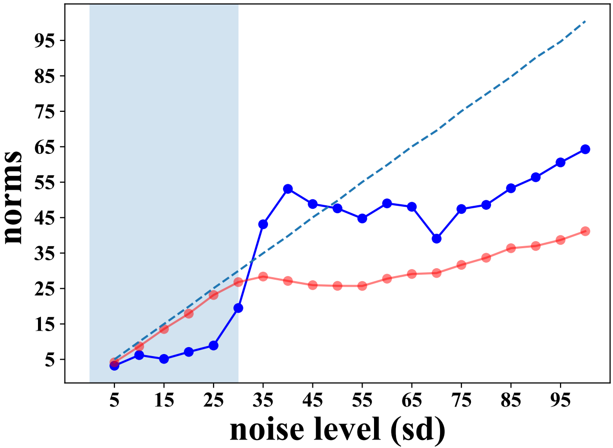
# Residual and net bias



# Residual and net bias



# Residual and net bias



# Bias-free networks

Within training range, learned net bias is small

Out of the range, it explodes, coinciding with dramatic performance loss

Net bias seems to overfit trained noise levels

# Bias-free networks

Within training range, learned net bias is small

Out of the range, it explodes, coinciding with dramatic performance loss

Net bias seems to overfit trained noise levels

This motivates *removing all additive constants*

$$f(y) = W_L R(\dots W_2 R(W_1 y + b_1) + b_2 \dots) + b_L$$

# Bias-free networks

Within training range, learned net bias is small

Out of the range, it explodes, coinciding with dramatic performance loss

Net bias seems to overfit trained noise levels

This motivates *removing all additive constants*

$$f(y) = W_L R(\dots W_2 R(W_1 y + \cancel{b_1}) + \cancel{b_2} \dots) + \cancel{b_L}$$

# It works

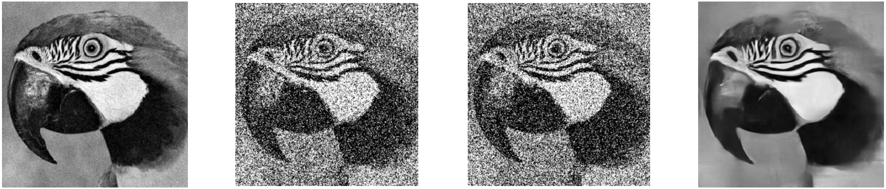
Training data (low noise)

Test image (high noise)

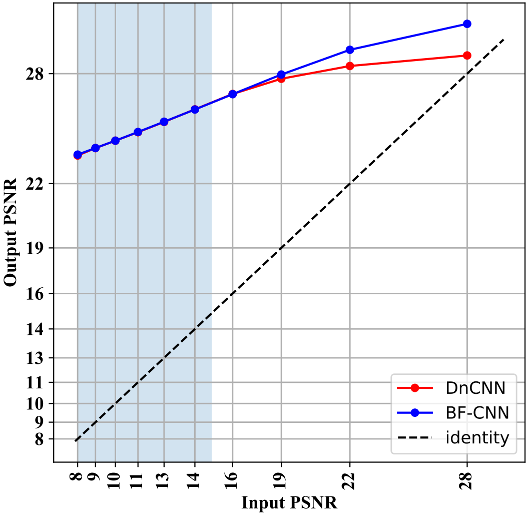
CNN

# It works

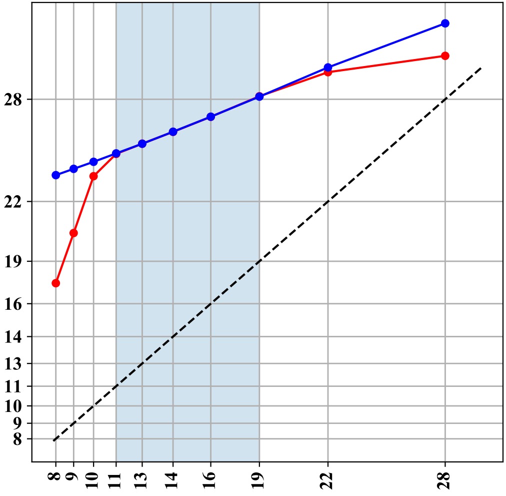
Training data (low noise) | Test image (high noise) | CNN | Bias-free CNN



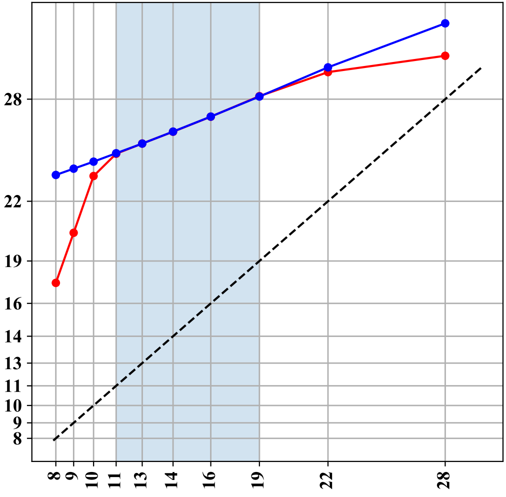
# DnCNN [Zhang *et al* 2016] vs bias-free DnCNN



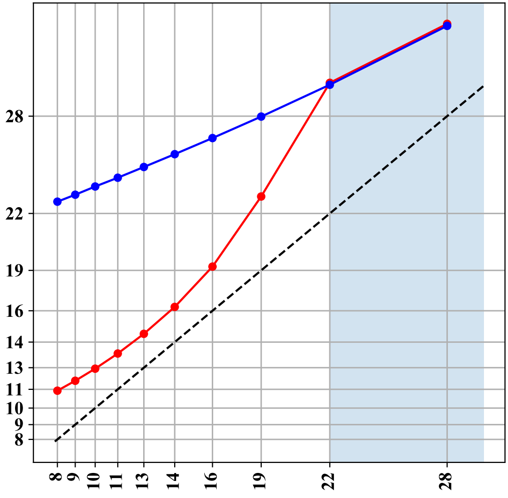
# DnCNN [Zhang *et al* 2016] vs bias-free DnCNN



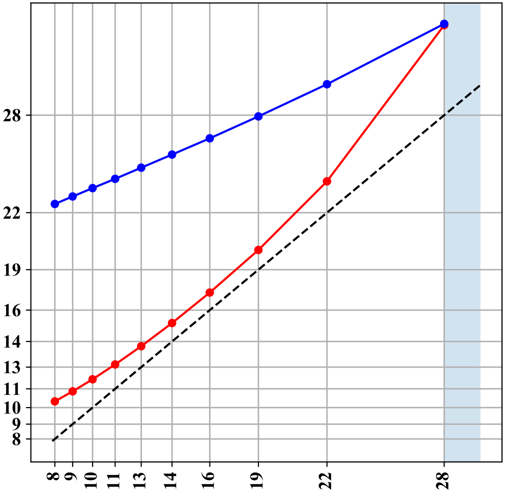
# DnCNN [Zhang *et al* 2016] vs bias-free DnCNN



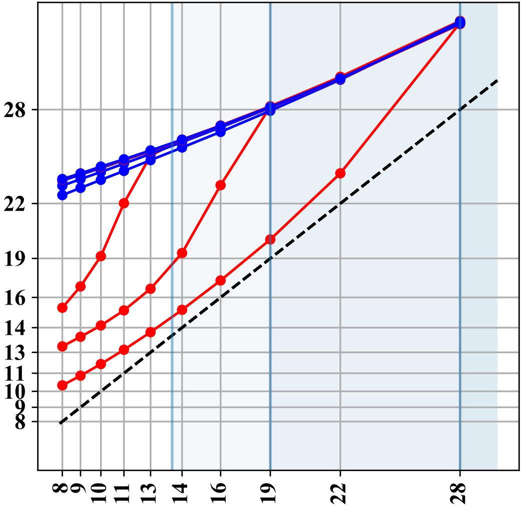
# DnCNN [Zhang *et al* 2016] vs bias-free DnCNN



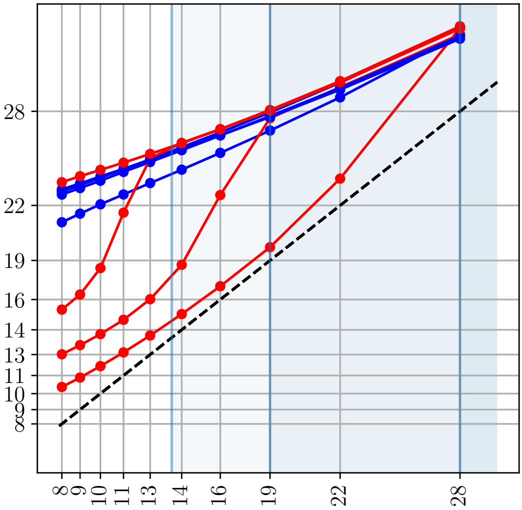
# DnCNN [Zhang *et al* 2016] vs bias-free DnCNN



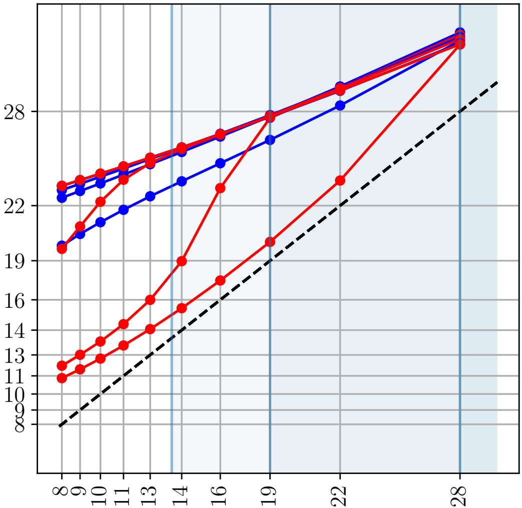
# DnCNN [Zhang *et al* 2016] vs bias-free DnCNN



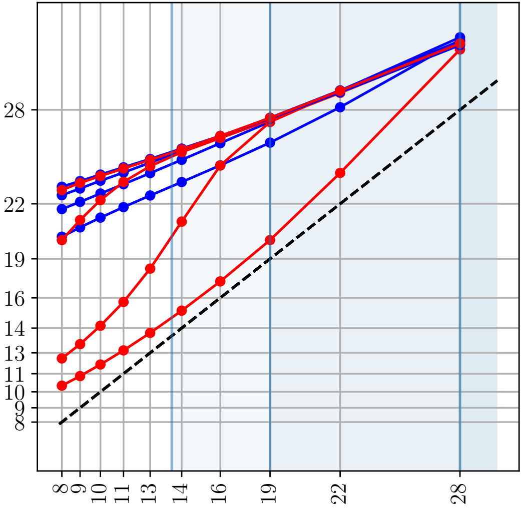
# DenseNet [Huang *et al* 2017] vs bias-free DenseNet



# UNet [Ronneberger *et al* 2015] vs bias-free UNet



# Recurrent CNN [Zhang *et al* 2018] vs bias-free recurrent CNN



Data-driven estimation of sinusoid frequencies

Blind denoising of natural images

Bias-free CNNs

Wiener filtering

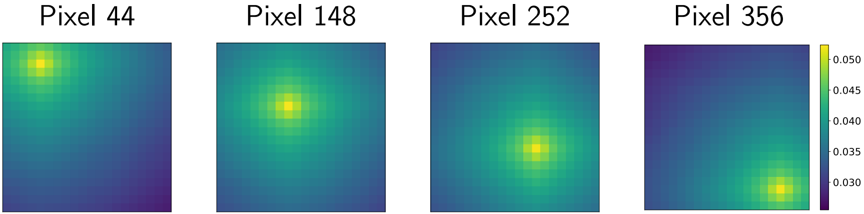
CNNs learn adaptive filters

CNNs learn unions of subspaces

# Linear estimation

Linear regression from pixels to pixels is intractable ($10^4 \times 10^4$ matrix!)

No need: covariance between pixels is translation invariant




Linear estimator can be parameterized by a convolutional filter

# Wiener filter [Wiener 1950]

Filter $w$ that achieves optimal mean squared error

Random vectors: $x$ (image), $z$ (noise), $y := x + z$ (data)

Fourier transform is an orthogonal transformation so

$$\mathrm{E}\left(||x - w * y||_2^2\right) = \mathrm{E}\left(||\hat{x} - \hat{w} \circ \hat{y}||_2^2\right)$$

# Wiener filter [Wiener 1950]

Filter $w$ that achieves optimal mean squared error

Random vectors: $x$ (image), $z$ (noise), $y := x + z$ (data)

Fourier transform is an orthogonal transformation so

$$\begin{aligned} \mathrm{E}\left(||x - w * y||_2^2\right) &= \mathrm{E}\left(||\hat{x} - \hat{w} \circ \hat{y}||_2^2\right) \\ &= \sum_k \mathrm{E}\left((\hat{x}_k - \hat{w}_k \hat{y}_k)^2\right) \end{aligned}$$

We can estimate each Fourier coefficient separately

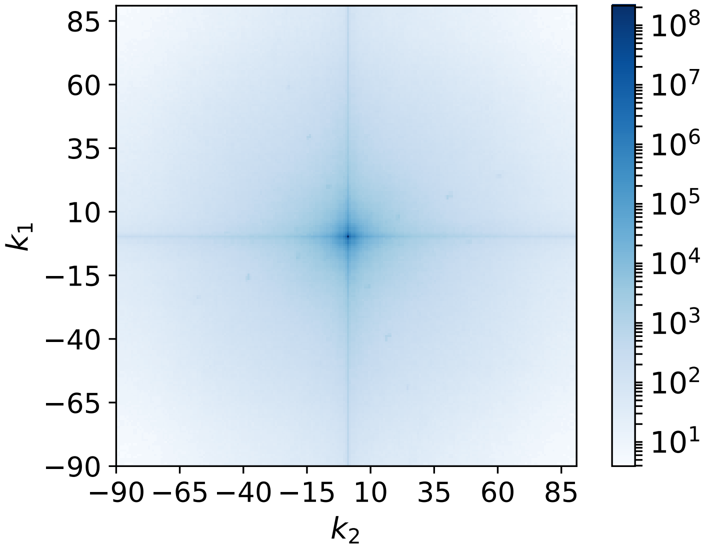
# Wiener filter

If $x$ and $z$ are independent, and $z$ is i.i.d. with variance $\sigma^2$

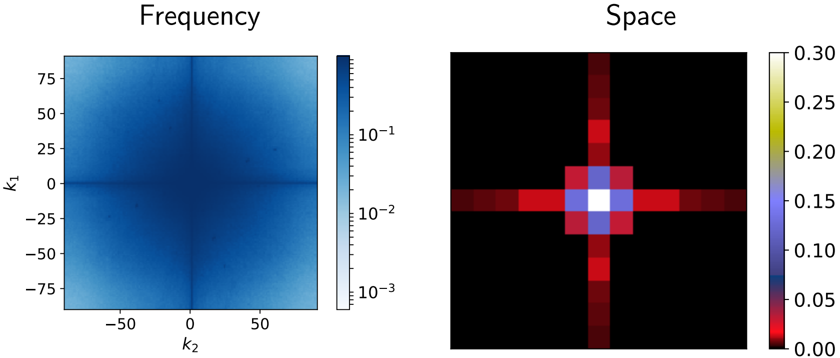
$$\hat{w}_k^{\text{opt}} := \arg\min_{\hat{w}} \mathrm{E}\left((\hat{x}_k - \hat{w}_k \hat{y}_k)^2\right)$$
$$= \frac{\mathrm{E}\left(|\hat{x}_k|^2\right)}{\mathrm{E}\left(|\hat{x}_k|^2\right) + n\sigma^2}$$

Depends on spectral statistics of natural images and on noise level $\sigma^2$ ($n$ is the number of pixels)

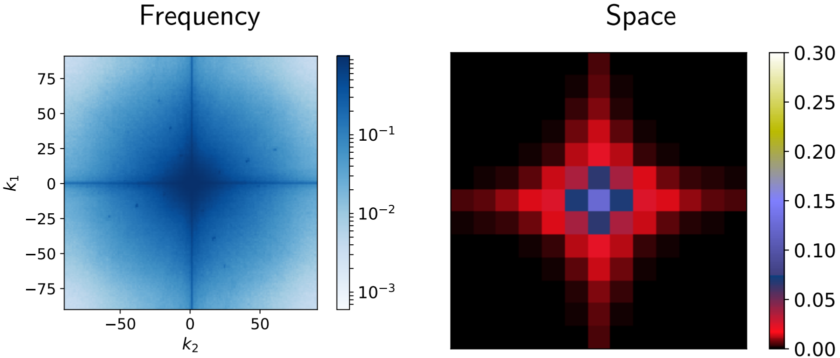
# Image data: Mean square of Fourier coefficients



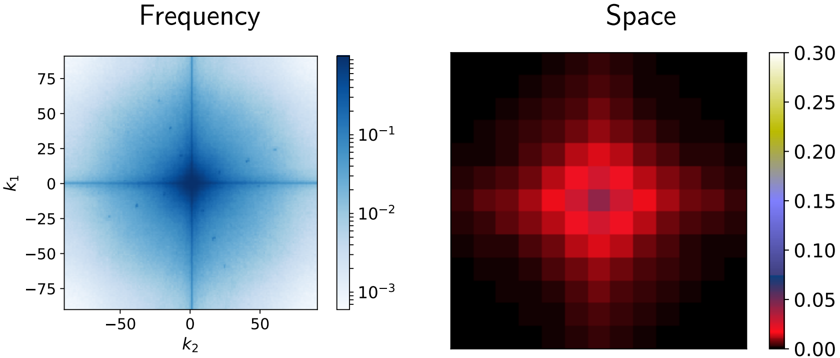
# Wiener filter: $\sigma = 0.04$



# Wiener filter: $\sigma = 0.1$



# Wiener filter: $\sigma = 0.2$



# Wiener filter

Two perspectives:

1. *Image domain*: Weighted average of nearby pixels
2. *Frequency domain*: Weighted projection onto low-pass 2D sinusoids

Data-driven estimation of sinusoid frequencies

Blind denoising of natural images

Bias-free CNNs

Wiener filtering

CNNs learn adaptive filters

CNNs learn unions of subspaces

# Wiener filter

*Image domain*: Weighted average of nearby pixels

Problem: Same average for each pixel

Blurs edges and other features

Previous solution:
Adapt filter locally (e.g. bilateral filter [Tomasi and Manduchi 1998])

# Bias-free CNN is locally linear

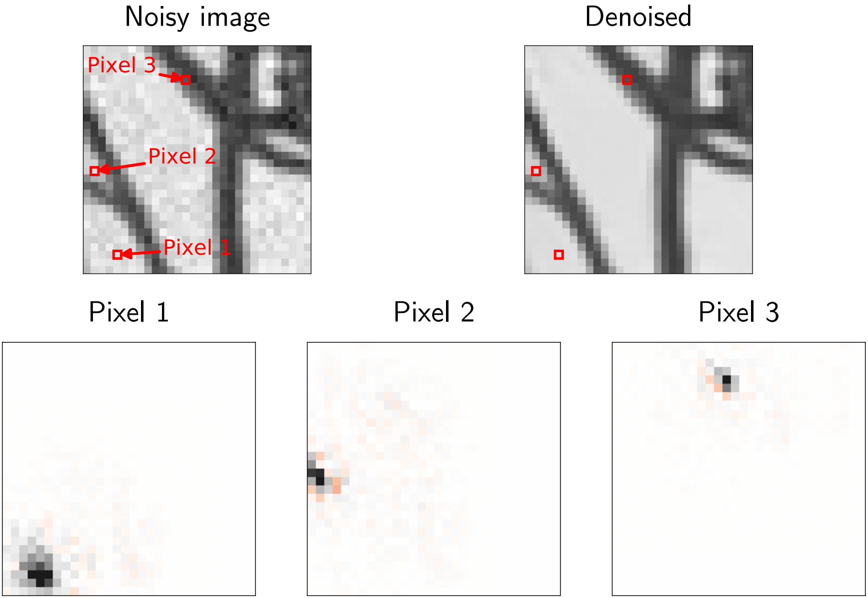
$$f(y) = W_L \, R \, W_{L-1} ... R \, W_1 y = A_y y$$

# Rows interpreted as filters

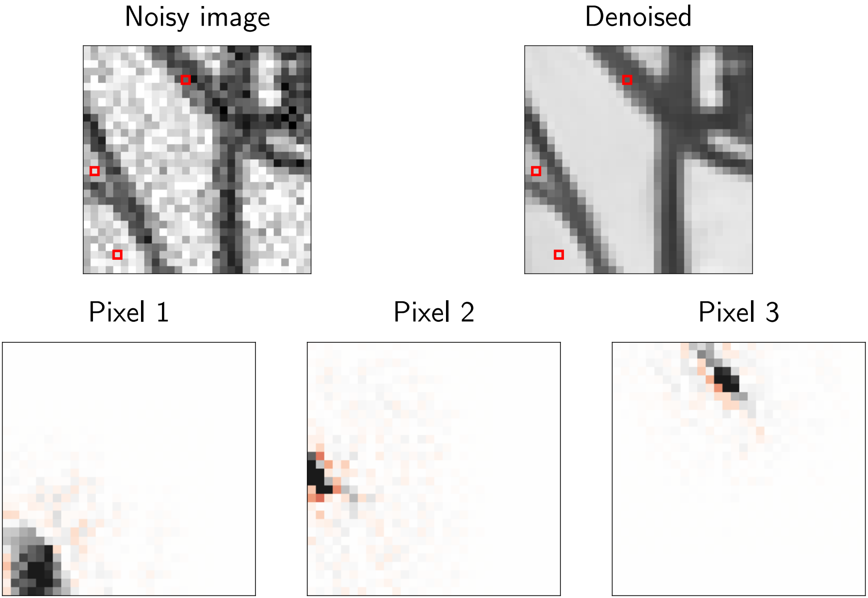
Estimate at pixel $i$:

$$f_{\text{BF}}(y)(i) = (A_y y)(i) = < i\text{th row of } A_y, y >$$

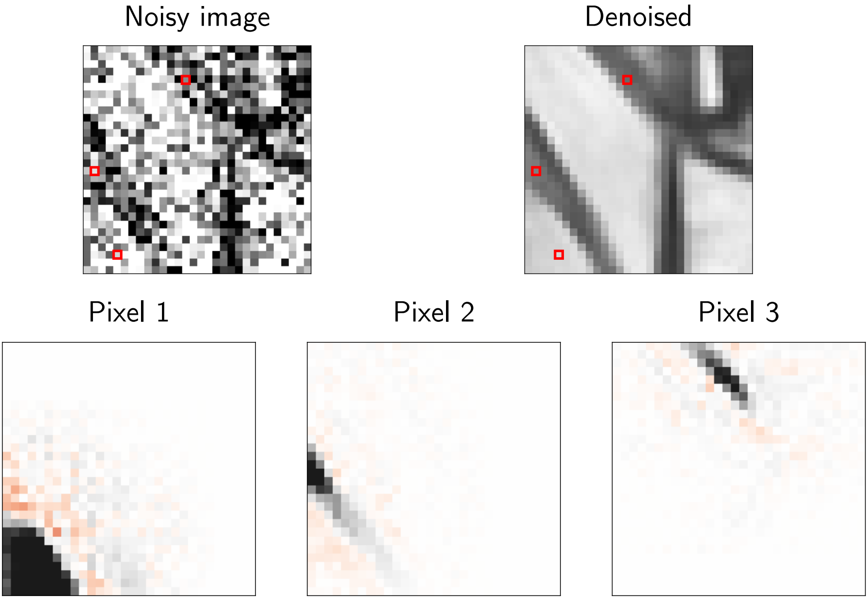
# Low noise


Noisy image
Denoised
Pixel 3
Pixel 2
Pixel 1
Pixel 1
Pixel 2
Pixel 3

# Medium noise



# High noise



# Conclusion

BF-CNN implicitly learns filters *adapted to image structure and noise*!

Data-driven estimation of sinusoid frequencies

Blind denoising of natural images

Bias-free CNNs

Wiener filtering

CNNs learn adaptive filters

CNNs learn unions of subspaces

# Wiener filter

*Frequency domain*: Approximate projection onto low-pass 2D sinusoids

Problem: Same projection for each image

Blurs edges and other features

# Projection onto union of subspaces

Previous methodology [*too many works to cite...*]:

1. Learn/design overcomplete dictionary of basis functions
2. Select sparse subset for each image/patch through thresholding/optimization
3. Project on span of sparse subset

Projection onto union of low-dimensional subspaces

# Bias-free CNN is locally linear

$$f(y) = W_L\, R\, W_{L-1} ... R\, W_1 y = A_y y$$

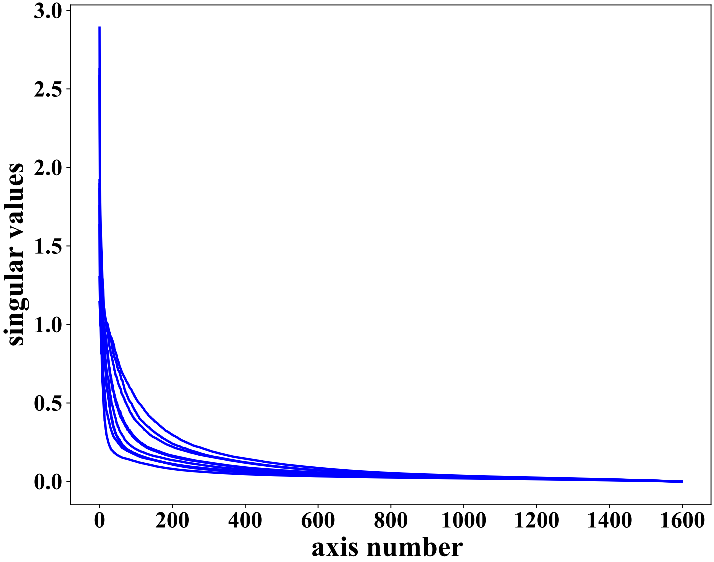
# SVD analysis

$$A_y = U\, S\, V^T$$

Empirical observations:

- Matrix is approximately symmetric $U \approx V$
- Matrix is approximately low-rank

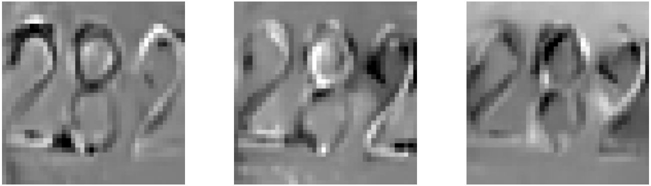
# Singular values



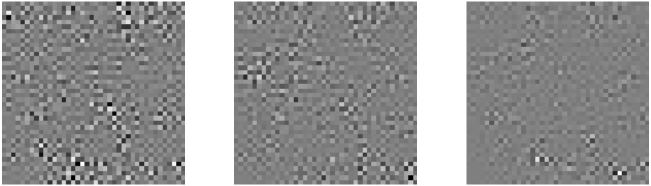
# Singular vectors computed from noisy image

Clean image

Large singular values

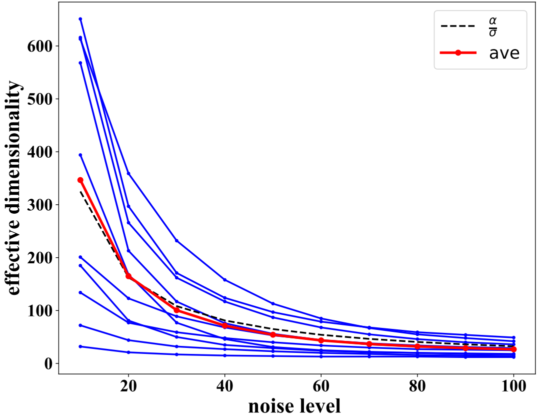


Small singular values



# Dimensionality of learned subspace

Approximate dimensionality = sum of squared singular values




Subspaces are approximately nested

# Conclusion

BF-CNN implicitly learns to project onto union of subspaces adapted to image features and noise!

# For more information

**Robust and interpretable blind image denoising via bias-free convolutional neural networks**
S. Mohan, Z. Kadkhodaie, E. Simoncelli, C. Fernandez-Granda

# Directions for future research

Properties of the learned representation in frequency estimation

Why does bias hinder generalization across noise levels?

Linear-algebraic analysis is completely empirical and very local

How are these adaptive filters / unions of subspaces learned?

How do the learned mechanisms vary as we change the input?