Image Denoising with Neural Networks

GESINE KRÖGER, ADEBANJI ADELOWO, ANTON JABS,
LAURIDS JEPPE

Introduction

- Image denoising: Recovering an image that is contaminated by noise

- Reasons for noise?
  - Physical – camera noise
  - Digital – image compression
  - → often unavoidable!

Well-performing denoising methods are important for image processing!

What is an Image?

- Image consists of pixels with certain color
- Mathematical model for a grayscale image:

$$u \in \mathbb{R}^{m \times n},\ \ 0 \leq u_{ij} \leq 1$$

- Alternatively, model as a function:

$$u:\ \ \Omega \subset \mathbb{R}^2 \rightarrow [0,1]$$

Figure: *Lenna*, a common test image

Mathematical Noise Models

- Noise: An additional component interfering with a pure image due to various physical processes

- Different mathematical types of noise

- Measure of the quality of the reconstruction:

Peak Signal-to-Noise Ratio

$$\mathrm{PSNR}(\tilde{u}, u) = 10 \log_{10} \left(\frac{1}{\|\tilde{u} - u\|_2^2} \right)$$

with the original image u and the noisy image $\tilde{u}$

Gaussian Noise

- A Gaussian distributed value is added randomly to each pixel

- Noisy image: $\tilde{u} = u + w$
  where u is the original image and w the noise

- Probability density function:

$$w \sim \mathcal{N}_{0,\sigma},\ p_w(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Figure: Example of Gaussian noise

Salt-and-Pepper Noise

- Randomly chosen pixels are set to either maximum or minimum value
- Mathematical model:

$$
\begin{aligned}
P(\tilde{u} = 0) &= \quad p/2 \\
P(\tilde{u} = 1) &= \quad p/2 \\
P(\tilde{u} = u) &= 1 - p
\end{aligned}
$$

with the probability of alteration p

Figure: Example of salt-and-pepper noise

Classical Methods

- There are a lot of possibilities to denoise an image without machine learning, e.g.:

a) Linear filtering
b) Variational methods
c) Wavelet thresholding
d) Anisotropic diffusion (Perona-Malik)

Linear Filtering

- As before: $\tilde{u} = w + u$
- Noise w is assumed to be Gaussian → equally distributed around zero
- Idea to remove the noise:

  Convolute the image with an appropriate function h

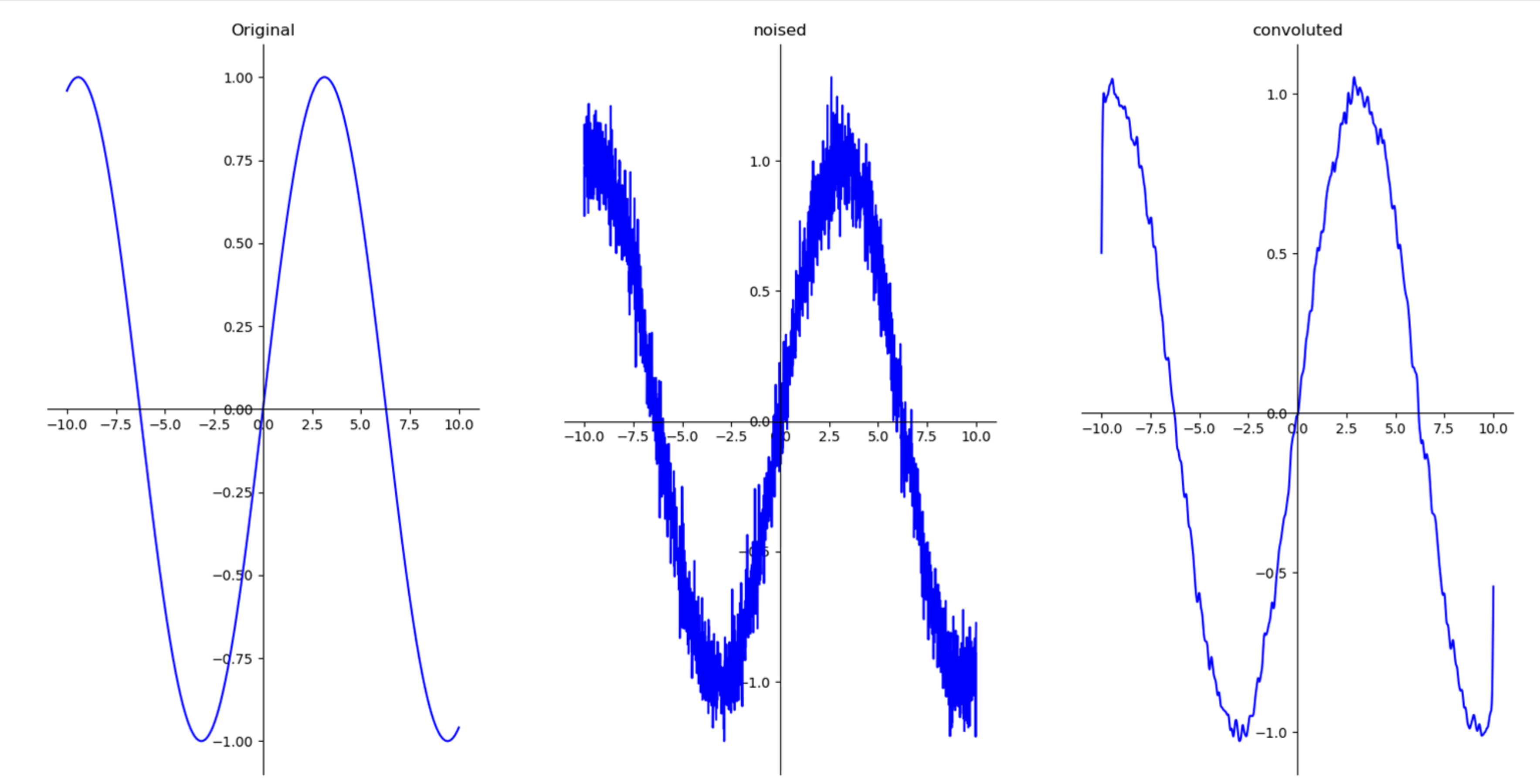
$$\tilde{u} * h = (w + u) * h = w * h + u * h \approx u$$

- A good choice for the function h is a Gaussian:

$$h(x) = ae^{\frac{-\|x\|^2}{2s^2}}, \qquad a \text{ such that } \int_{\mathbb{R}^2} h(x)dx = 1$$

Linear Filtering

- Example for 1D:



$s = 0.1$

Linear Filtering

- Example for 2D:

Linear Filtering

- Example for 2D:

Variational Methods

- Idea: A natural image u may have some properties a noisy image does not have. E.g. "locally constant".

Let J be a function which indicates the strength of the noise (data fidelity term).

Let g be a function which indicates how natural the image is (regularizer).

$$\Rightarrow \min_u J(u - \tilde{u}) + \lambda g(u), \qquad \lambda > 0$$

Variational Methods

- Problem: Find such functions J and g.

- Proposal given by Rudin, Osher and Fatemi:

$$J(u) = \frac{1}{2}\|u\|_2^2 = \frac{1}{2}\int_\Omega |u|^2\,dx \qquad\qquad g(u) = TV(u) = \int_\Omega \|\nabla u\|\,dx$$

$$\Rightarrow \min_u \frac{1}{2}\|\tilde{u} - u\|_2^2 + \lambda \int_\Omega \|\nabla u\|\,dx$$

Variational Methods

$\lambda = 0.06$

Neural Networks (NNs)

- Structure:

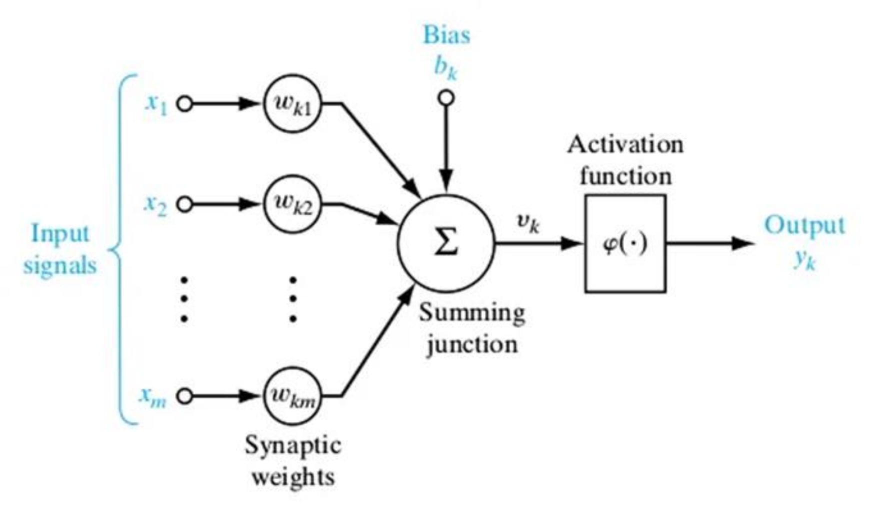


Figure: Structure of a neuron

  - Several neurons form a layer
  - Several layers form a neural network

- Problem description:
  - Given: samples $((x_i, y_i))_{i=1}^{N}$, loss L, network architecture
  - Task:

$$\sum_{i=1}^{N} L\big(NN_{(W,b)}(x_i), y_i\big) \rightarrow \min_{W,b} \;!$$

  - Idea: N very large → generalization

Training Algorithms

- Standard training algorithm: Stochastic Gradient Descent
  - Gradient Descent where the gradient is approximated using batches

- Problems of SGD:
  - Saddle points
  - Ill-conditioned problems

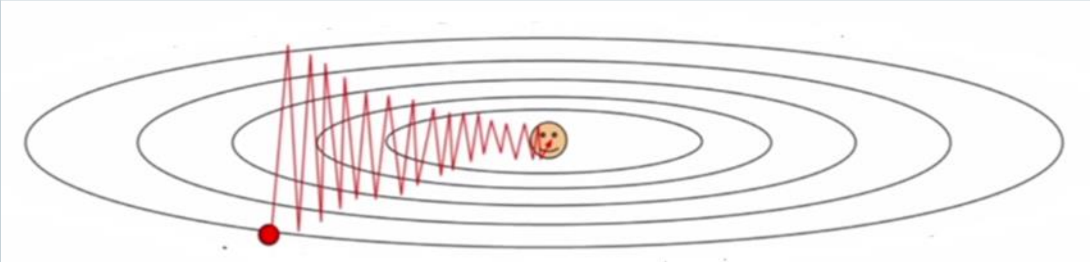


Figure: Performance of gradient descent on an ill-conditioned problem

- Improved algorithms, e.g.
  - SGD with momentum
  - ADAM algorithm

SGD with Momentum

- SGD with momentum:

$$v^{(k)} = \rho\, v^{(k-1)} - \alpha \nabla f\left(x^{(k)}\right)$$

$$x^{(k+1)} = x^{(k)} + v^{(k)}$$

with learning rate α, momentum ρ (standard choice: $\rho = 0.9$)

- Intuition: "$v \triangleq$ velocity"
  - In directions with partial derivatives of high absolute value but inconsistent sign oscillations are damped
  - In directions with partial derivatives of small absolute value but consistent sign "speed is gained"
  - May produce overshoot at minima but moves well past saddle points

Comparison between Training Algorithms

- Which training algorithm to pick?

- Compare training losses for different algorithms
  - Network architecture: ResNet (see later)

- Standard SGD converges slowest
- ADAM algorithm works best!

  → Will be used in the rest of the results

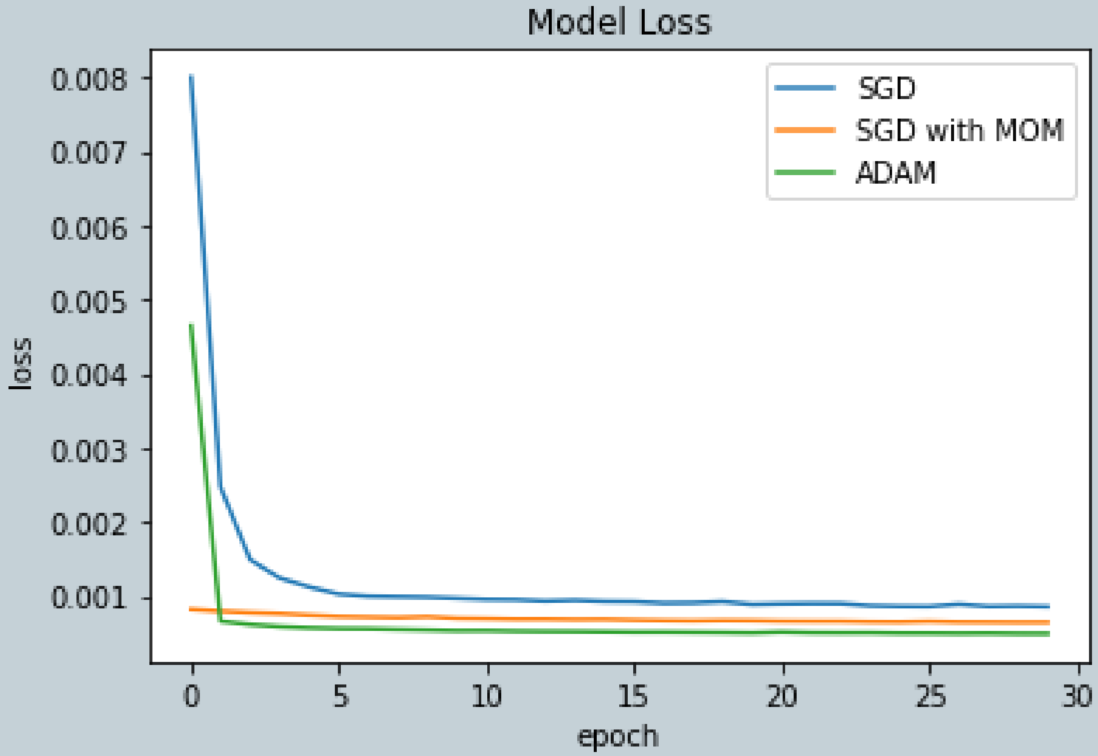


Figure: Example of the training error for different optimization algorithms

Convolutional Neural Networks (CNNs)

- Structure:
  - Input: d feature maps of size $L \times W$
  - Weights: k biases, k filters of size $(2\,l+1) \times (2\,l+1) \times d$
  - Output: k feature maps of size $L \times W$
  - Convolution operation:

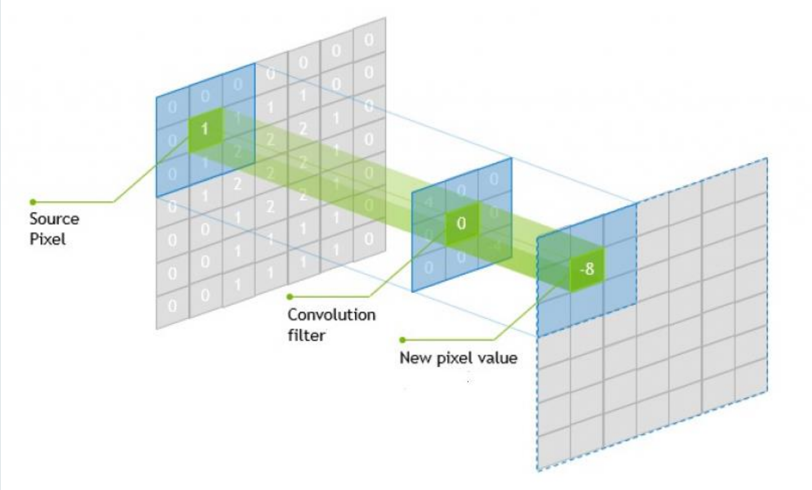


Figure: Convolution operation in the case $d = 1$

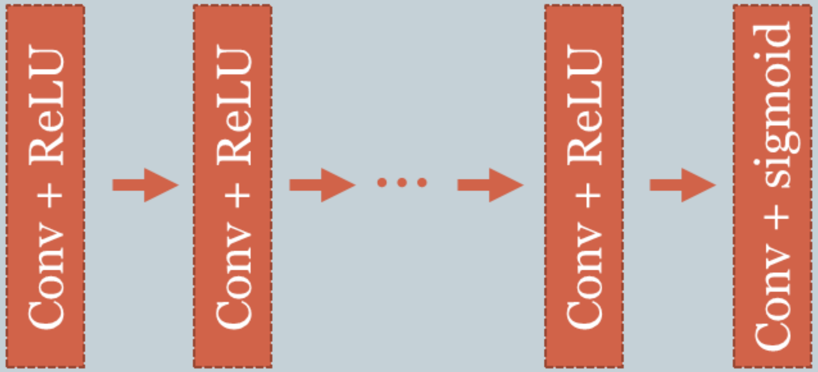
- Well-suited for image processing:
  - Take spatial structure into account
  - Use “translation invariance” of images

→ Integrate basic understanding of images into the network architecture

→ Reduce number of parameters significantly

Network Architectures: Convolutional

- Simple sequence of 17 convolution layers
- ReLU activation
- Dimensionality of data stays the same



Mean PSNR: 35.5

Mean MSE: 2.8×10^{-4}

Network Architectures: Convolutional

- "Degradation problem": At a certain depth, increasing the number of layers further leads to a decrease of the training accuracy.
- Counterintuitive
- Conjecture: Approximating the identity is hard!

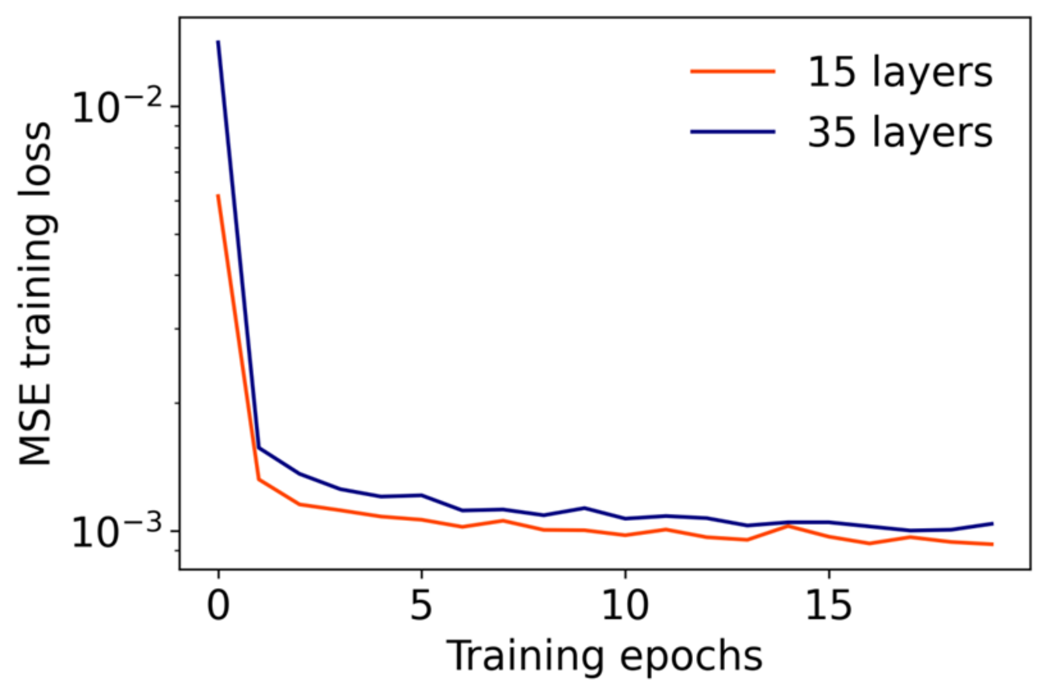
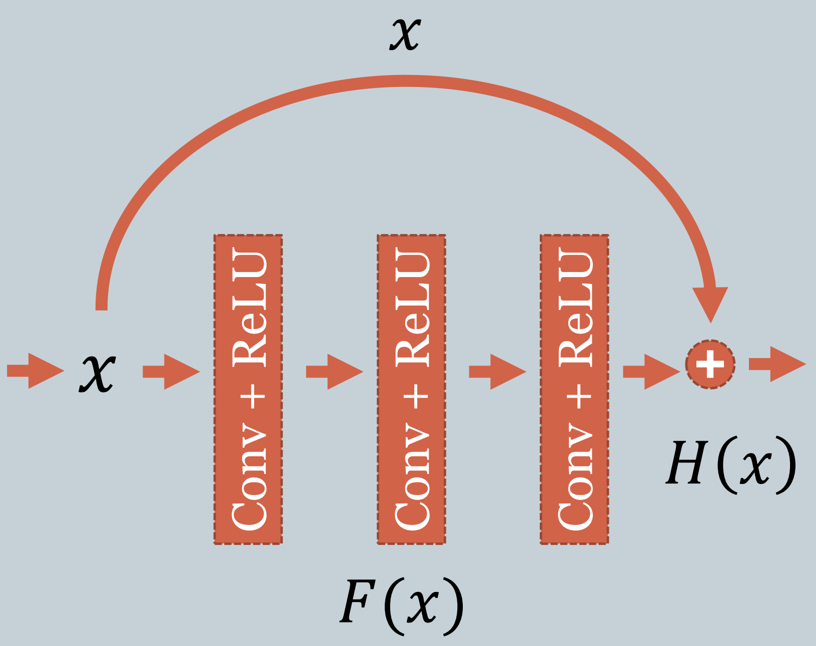


Figure: Example of the training error of a simple convolutional network for 15 and 35 layers

Network Architecture: Residual Network

- Structure:



$$H(x) = F(x) + x$$

- Expectation for image denoising:

$$H(x) \approx x \Rightarrow F(x) \approx 0$$

→ Easier to learn:

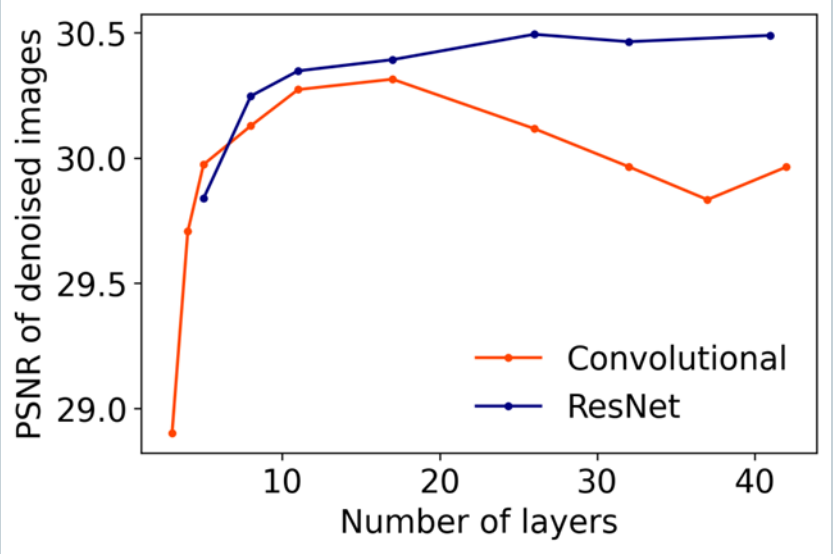
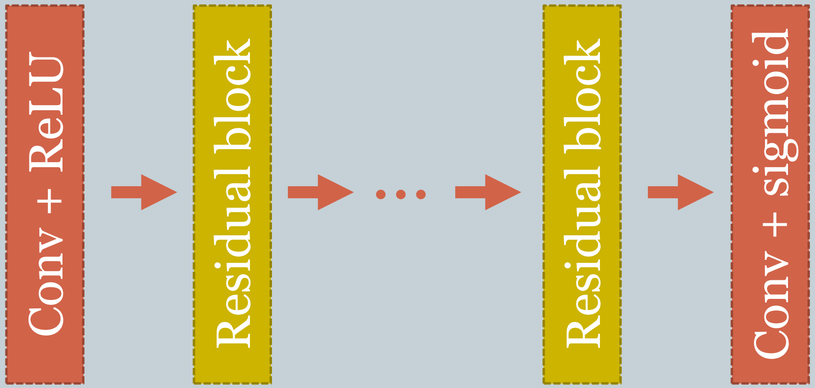


Figure: Comparison of the performance of CNN and ResNet with increasing number of layers

Network Architecture: Residual Network

- Activation function: still ReLU
- 5 blocks of 3 convolutions each
- One convolution in front and at the end



Mean PSNR: 35.8

Mean MSE: 2.6×10^{-4}

Results: Noise Levels

Gaussian Noise

- ResNet was trained on datasets with different noise levels

Denoising performance still good even for large noise levels!

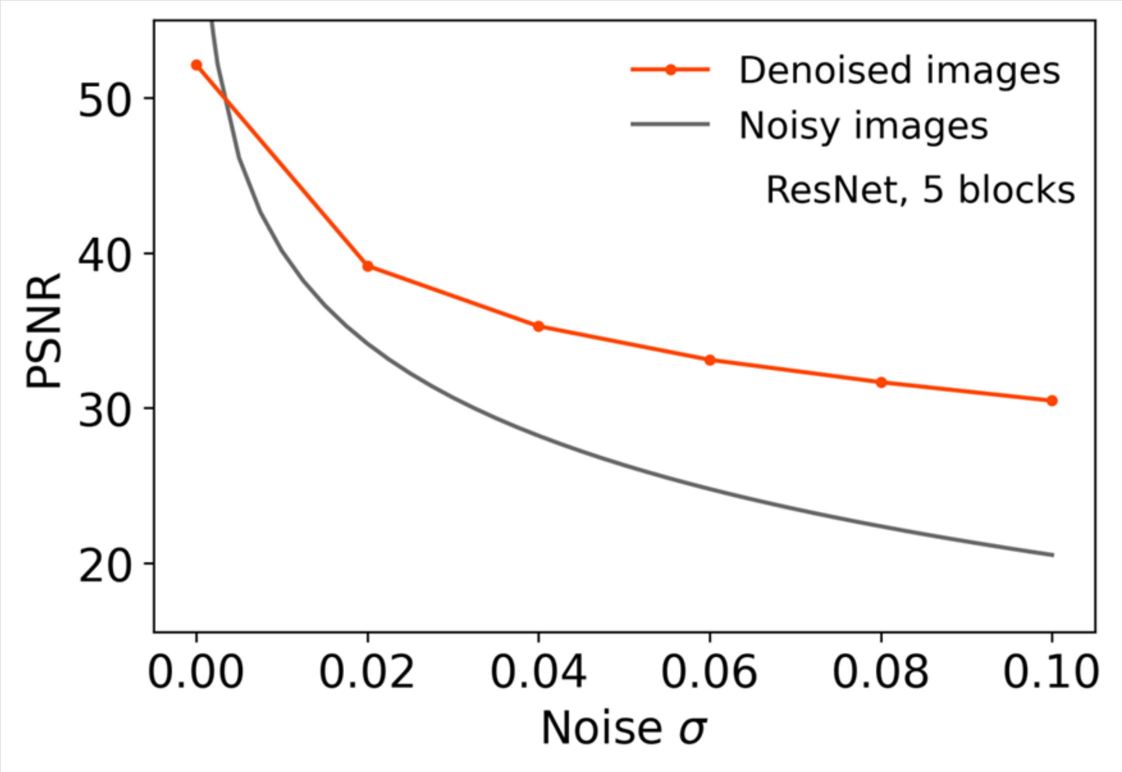
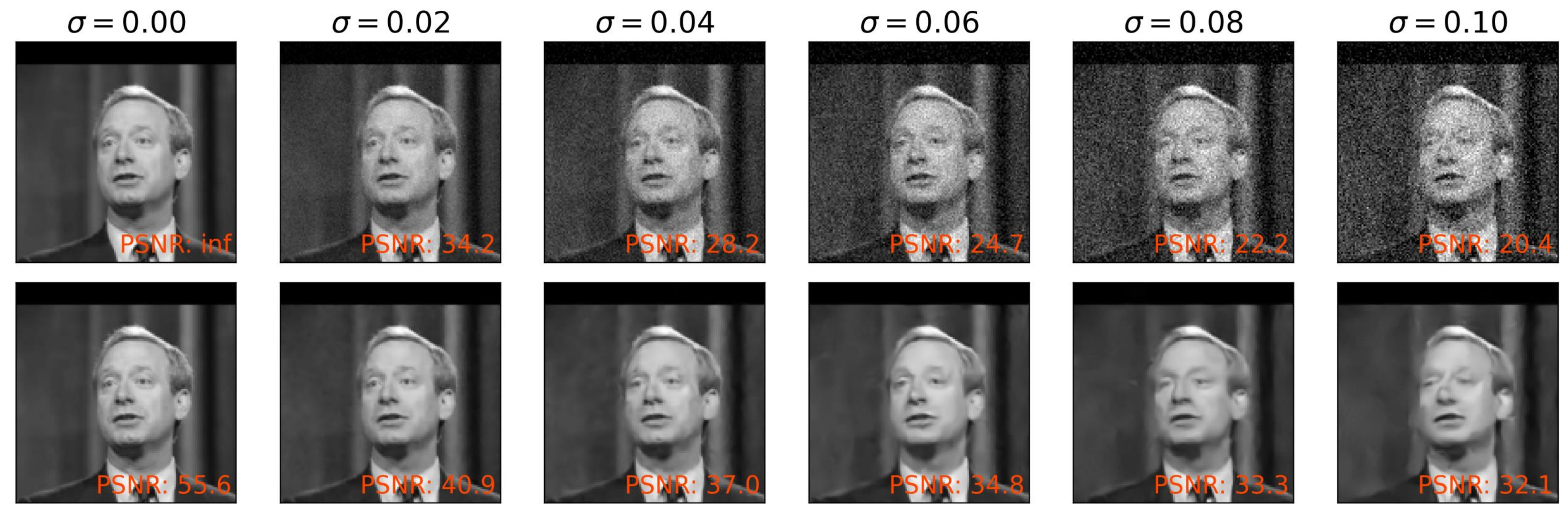


Figure: Performance of ResNet for different noise levels

Results: Noise Levels

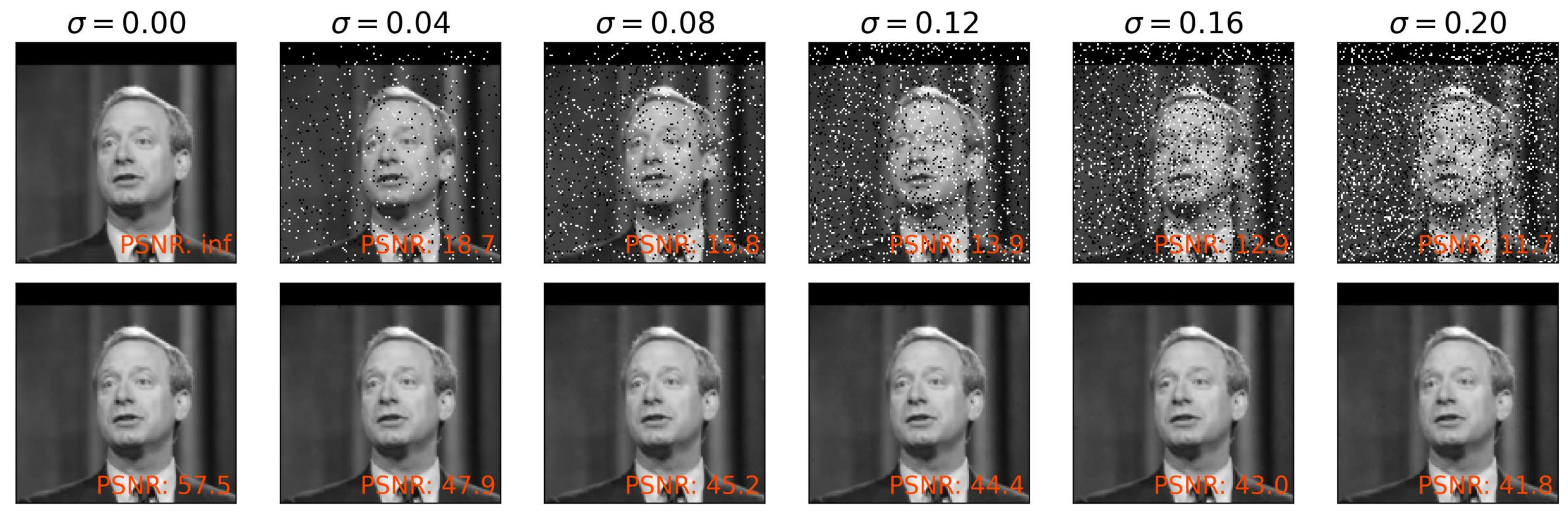
Gaussian Noise



For large noise levels, the output images lose detail and appear smoothed

Results: Noise Levels

Salt-and-Pepper Noise

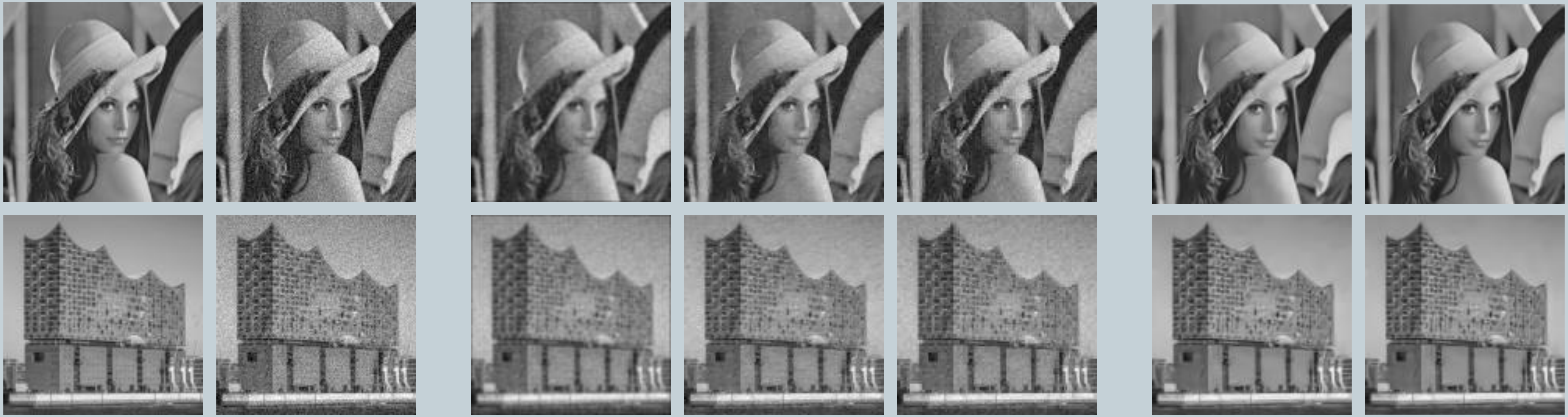


With salt-and-pepper noise, the output is a lot clearer!

Neural Networks vs. Classical Methods

Classical Methods | Neural Networks

Original | Noisy | Linear Filter | ROF | Wavelets | Conv. | ResNet



PSNR: | 27.9 / 27.9 | 29.0 / 27.7 | 31.0 / 29.7 | 30.0 / 29.2 | 34.2 / 31.8 | 34.6 / 32.0

Neural Networks widely outperform the classical methods!

Summary

- Best choice of training algorithm: ADAM

- Good choice of architecture: Residual Network
  - Solves degradation problem
  - Can be used to train deeper networks

- Neural Networks are well-suited to image denoising
  - Same architectures can be used for different noise types
  - Give much better results than classical methods

Who did what?

Adelowo:	Implementation of the networks (Autoencoder & ResNet) Comparison between training algorithms
Anton:	Theory of classical denoising Implementation of classical denoising (Linear filtering, ROF method, Wavelet method, Perona-Malik diffusion)
Gesine:	Theory of ResNets Theory of Batch Normalization Theory of training algorithms
Laurids:	Implementation of the networks (Simple Conv. & ResNet) Comparison between noise levels & network depths Project Manager

Literature Sources

- K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang. *Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising*. In: IEEE Transactions on Image Processing, vol. 26, no. 7, pp. 3142-3155, July 2017, doi: 10.1109/TIP.2017.2662206.
- K. He, X. Zhang, S. Ren and J. Sun. *Deep Residual Learning for Image Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
- M. Lotz. *Mathematics of Machine Learning*. Lecture notes, Chapters 23 – 26, March 2020, URL: `http://homepages.warwick.ac.uk/staff/Martin.Lotz/files/learning/lectnotes-all.pdf`.
- C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Chapters 3.5 and 8.1 – 8.2, 2018
- F. Li, J. Johnson and S. Yeoung. *Convolutional Neural Networks for Visual Recognition*. Lecture Notes, Lectures 5, 7, 2018, URL: `http://cs231n.stanford.edu/slides/2018/`
- J. Duchi and Y. Singer. *Proximal and first-order methods for convex optimization*. 2013, URL: `https://ppasupat.github.io/a9online/uploads/proximal_notes.pdf`
- G. Hinton. *Neural networks for machine learning*. Video lecture, lectures 6.3 and 6.5, URL: `https://www.youtube.com/playlist?list=PLLssT5z_DsK_gyrQ_biidwvPYCRNGI3iv`
- G. Peyré. *Advanced Signal, Image and Surface Processing*. January 2010, Ceremade, Université Paris-Dauphine.
- K. Bredies and D. Lorenz. *Mathematische Bildverarbeitung*. 1. Auflage, 2011, Vieweg+Teubner Verlag, Wiesbaden.

All internet sources retrieved on 12th Jul 2020

Other Sources

- The Neural Networks were implemented using KERAS in Tensorflow:
  https://keras.io/
  https://www.tensorflow.org/

- As training data, we used the LFW dataset:
  G. Huang, M. Ramesh, T. Berg and E. Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. University of Massachusetts, Amherst, Technical Report 07-49, October, 2007. URL: http://vis-www.cs.umass.edu/lfw/

- Image sources for our test images:
  Lenna: https://en.wikipedia.org/wiki/File:Lenna_(test_image).png
  Elbphilharmonie: https://commons.wikimedia.org/wiki/File:Elbphilharmonie,_Hamburg.jpg

- Image sources for the presentation:
  Neural Network: https://www.opensourceforu.com/2017/03/neural-networks-in-detail/
  CNN: https://www.edge-ai-vision.com/2018/09/whats-the-difference-between-a-cnn-and-an-rnn/
  Training algorithms: from F. Li, J. Johnson and S. Yeoung. *Convolutional Neural Networks for Visual Recognition*. Lecture Notes, Lecture 7, 2018, URL: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture07.pdf

All internet sources retrieved on 12th Jul 2020

Appendix: ResNet Code

```
block_count = 5

x = Conv2D(32, 3, activation="relu", padding="same") (Input(input_shape))

for i in range(block_count):

    x_shortcut = x

    x = Conv2D(32, 3, activation="relu", padding="same") (x)
    x = Conv2D(32, 3, activation="relu", padding="same") (x)
    x = Conv2D(32, 3, activation="relu", padding="same") (x)

    x = Add() ([x, x_shortcut])

x = Conv2D(1, 3, activation="sigmoid", padding="same") (x)

model = keras.Model(inp, x)
```

- Implemented in KERAS
- 5 blocks with 3 convolutions each
- 32 filters per convolution

Appendix: Implementation of Variational Methods

The most straightforward method is gradient descent.
This gives the following sequence:

$$u^{k+1} = u^k - \tau\left(J'(u^k) + \lambda g'(u^k)\right)$$

We have $J'(\tilde{u} - u) = \tilde{u} - u$ and $g'(u) = -\mathrm{div}\left(\frac{\nabla u}{\|\nabla u\|}\right)$.

Problem: If $\nabla u = 0,$ then g' cannot be calculated.

→ Solve by setting for $\epsilon > 0$

$$g_\epsilon(u) := \int \sqrt{(\epsilon^2 + \|\nabla u\|^2)}\, dx \quad \Rightarrow \quad g'_\epsilon(u) = -\mathrm{div}\left(\frac{\nabla u}{\sqrt{\epsilon^2 + \|\nabla u\|^2}}\right)$$

Appendix: Adam Algorithm

algorithm:

$$v_i^{(0)} = 0,\ g_i^{(0)} = 0$$

$$\boxed{v_i^{(k)} = \rho\, v_i^{(k-1)} + (1-\rho)\, \partial_i f(x^{(k)})}$$

idea of momentum

$$\boxed{g_i^{(k)} = \rho_g\, g_i^{(k-1)} + (1-\rho_g)\, \left(\partial_i f(x^{(k)})\right)^2}$$

idea of pre-conditioning

$$x_i^{(k+1)} = x_i^{(k)} - \boxed{\left(\frac{\sqrt{1-\rho^k}}{1-\rho_g^k}\right)} \frac{\alpha}{\sqrt{g_i^{(k)}} + \varepsilon}\, v_i^{(k)}$$

correction factor

with standard choices $\rho = 0.999,\ \rho_g = 0.9$

Appendix: Varying Noise Levels

Gaussian Noise

- How do networks perform on images outside of the training data?

- Network trained on fixed noise level

- Evaluated on images with different noise levels

- For low noise levels: NN makes PSNR worse!