dortmund university

Zero-Shot Denoising of Distributed Acoustic Sensing Data using Deep Priors

Jannik Mänzer

Bachelor of Science

March 2025

Supervisors:
Prof. Dr. Stefan Harmeling
Sebastian Konietzny

Artificial Intelligence (VIII)
Department of Computer Science
TU Dortmund University

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Contents

Chapter 1

Introduction

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Chapter 2

Background

This chapter provides the necessary background for understanding unsupervised denoising. We begin by defining the general denoising problem and discussing its inherent challenges. We then present key deep learning concepts and techniques that are used in the context of denoising. Finally, we introduce Digital Acoustic Sensing (DAS) as a real-world application and highlight the unique difficulties it presents.

2.1 Denoising

In general, denoising refers to the process of recovering a clean signal from a noisy observation. Formally, it can be described by the inverse problem

$$y = x + n \tag{2.1}$$

where y is the noisy observation, x is the underlying clean signal and n represents some form of noise, for example Gaussian noise $n \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. Denoising is an inherently ill-posed problem, as the noise and, in many cases, the noise distribution are unknown. Therefore, additional assumptions about the solution are essential for solving the denoising problem. This process is known as regularization and typically involves imposing certain constraints on the solution space to favor more natural solutions [7]. The choice of regularizer depends on the specific problem setting and the type of data involved.

2.2 Deep Learning

Deep learning is a subfield of machine learning that utilizes deep neural networks to learn complex patterns from data. Over the past decade, deep learning has established itself as the state-of-the-art approach for a wide range of problems across various different fields.

2.2.1 Deep Neural Networks

In its most basic form, a (fully-connected) neural network consists of neurons organized in layers, where each neuron applies a linear transformation followed by a non-linear activation function. The output of a single neuron is given by

$$y = \varphi(\mathbf{w}^T\mathbf{x} + b) \tag{2.2}$$

for an input $\mathbf{x}$, a weight vector $\mathbf{w}$, a bias value b and an activation function φ. The activation function is needed in order to avoid the network collapsing into a single linear transformation. The outputs of each layer are then passed as inputs to the next layer. Therefore, a neural network can be described as a function $f_\theta : \mathcal{X} \to \mathcal{Y}$ parameterized by θ, where θ represents the weights and biases across all layers [2].
In order to optimize these parameters, a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is defined, which measures the difference between the predicted output and the target value. Now, the gradient of the loss function with respect to the parameters, $\nabla_\theta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta}$, represents the direction of steepest ascent. Therefore, by moving the parameters in the opposite direction of the gradient, the loss function can be minimized. Typically, the gradient is not calculated for a single data point or for the whole dataset, but instead for a small subset of the dataset, which is why this approach is referred to as (mini-batch) gradient descent. Backpropagation [5] is used to efficiently compute the gradient by making use of the chain rule, enabling fast optimization.
While traditionally neural networks only consisted of a few layers and required hand-crafted features to work effectively, advances in computing power allow modern architectures to automate feature extraction by using additional layers, hence the term *deep* neural network.

2.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) [4] are a specific type of neural network that learns features using kernels. Prior to the rise of deep learning, such kernels were designed manually for various computer vision tasks, for example the Sobel kernel [6] used for edge detection. In CNNs, these kernels are automatically learned from data. In contrast to fully-connected layers, the output of a convolutional layer is obtained by convolution with one or multiple kernels, replacing the matrix multiplication. For a kernel $\mathbf{K} \in \mathbb{R}^{n \times m}$, the convolution is defined as

$$(\mathbf{X} * \mathbf{K})_{i,j} = \sum_n \sum_m X_{i+n,j+m} \cdot K_{n,m}. \tag{2.3}$$

The output of the convolution is then passed through a non-linear activation function, just like in fully-connected layers. CNNs provide two main advantages: First, since the weights are shared, convolutional layers drastically reduce the number of parameters compared to

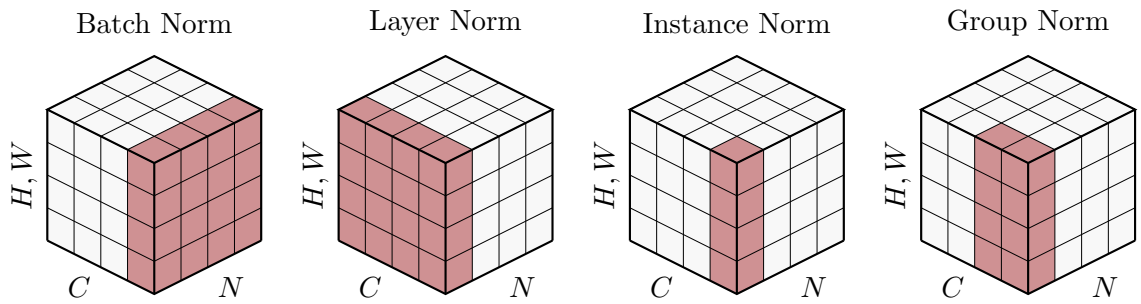

Figure 2.1: Different normalization techniques. N is the batch dimension, C is the channel dimension and H and W are the spatial dimensions of a 4D tensor. The input is normalized across the dimensions highlighted in red. Figure adapted from [9].

fully-connected layers. Second, convolutions are translationally equivariant, meaning that local patterns in the input can be recognized regardless of their position, which makes CNNs very suitable for image data. [2]

2.2.3 Normalization

During the training process, the inputs of each layer change with each iteration as the parameters are optimized. That slows down the training because now each layer has to adapt to the new distribution of its inputs. This process is often referred to as internal covariate shift. To counteract this issue, Ioffe et al. proposed Batch Normalization (BN) [3]. The idea behind BN is to normalize the inputs across the whole mini-batch and their spatial dimensions. The normalized input for a channel i is given by

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mu^{(i)}}{\sigma^{(i)}}, \tag{2.4}$$

where $\mu^{(i)}$ and $\sigma^{(i)}$ are the per-channel mean and standard deviation of the mini-batch, respectively. In order to allow the model learn the identity if that is the optimal transformation, two additional learnable parameters, γ and β, are introduced. The output of the BN layer is then defined as

$$y^{(i)} = \gamma^{(i)}\hat{x}^{(i)} + \beta^{(i)}. \tag{2.5}$$

While BN is widely used, there exist various similar normalization techniques [1, 8, 9] mainly differing in the dimensions across which they are applied. A selection of them is visualized in Figure 2.1.

2.2.4 Attention Mechanisms

2.3 Digital Acoustic Sensing

Chapter 3

Related Work

3.1 Supervised Approaches

3.2 Unsupervised Approaches

3.2.1 Noise2Noise

3.2.2 Noise2Void

Chapter 4

Methods

Chapter 5

Experimental Setup

Chapter 6

Results

Chapter 7

Discussion

Chapter 8

Conclusion

Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML]. URL: https://arxiv.org/abs/1607.06450.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[3] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[4] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[5] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (Oct. 1986), pp. 533–. URL: http://dx.doi.org/10.1038/323533a0.

[6] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford AI Lab (SAIL)* (1968).

[7] "TODO". In: ().

[8] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: *CoRR* abs/1607.08022 (2016). arXiv: 1607.08022. URL: http://arxiv.org/abs/1607.08022.

[9] Yuxin Wu and Kaiming He. "Group Normalization". In: *CoRR* abs/1803.08494 (2018). arXiv: 1803.08494. URL: http://arxiv.org/abs/1803.08494.

Appendix A

Appendix