Machine learning-based noise reduction for quantifying current fluctuations

Steenbergen, Jasper

Machine learning-based noise reduction for quantifying current fluctuations

THESIS

submitted in partial fulfillment of the
requirements for the degree of

BACHELOR OF SCIENCE
in
PHYSICS

Author :	J.C. Steenbergen
Student ID :	2352753
Supervisor :	PhD student W.O. Tromp
Second corrector :	Prof.dr. S.F. Portegies Zwart

Leiden, The Netherlands, July 23, 2022

Machine learning-based noise reduction for quantifying current fluctuations

J.C. Steenbergen

Huygens-Kamerlingh Onnes Laboratory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

July 23, 2022

Abstract

In this research, a machine learning approach to noise characterisation is presented. Specifically, the possibility of using a denoising autoencoder to quantify shot noise in scanning tunnelling microscopy measurements is explored. First it is shown that a neural network can be used to denoise an artificial dataset of time traces with Gaussian noise added to it. In a later stage, a signal generator in combination with a resonator circuit is used to measure noisy time traces in the MHz spectral regime. In this setting, the neural network is tested on its noise sensitivity. From this it is calculated that this particular neural network does not have the required sensitivity to reproduce regular noise measurements in high frequency STM, falling one order of magnitude short. As a first trial, this opens the door to further investigations in using neural networks to quantify shot noise.

Contents

Version of July 23, 2022– Created July 23, 2022 - 13:25

Chapter 1

Introduction

Machine learning and, in particular, artificial neural networks have become more and more prominent within science. The algorithms used by such systems are able to create predictive models by learning from data. Doing so, they have become a new tool for scientists in a range of technical fields. A milestone in AI-based science was reached when DeepMind's AlphaFold announced it had achieved highly accurate protein structure predictions, making unprecedented progress on the famous protein-folding problem [11]. Within (particle) physics, ML-based approaches have a long and productive history in accelerator experiments, where they play an important role in event and particle identification and energy estimation. Within astronomy and cosmology, different challenges are now being tackled by AI systems such as photometric redshift estimation, image analysis and feature extraction [1]. In particular, ML techniques have proven useful in the problem of image reconstruction. In image reconstruction, the goal is to reconstruct an image from a incomplete or noisy image. The first results of using artificial neural networks in medical image reconstruction (e.g. MRI, CT-scans) have been shown in the 1990's. As data acquisition and computing power have increased, such techniques have become increasingly more successful [9]. Within astronomy, image reconstruction has always posed a challenge as (i) astronomical images are a convolution of the observed object and what is called a point-spread-function (PSF) and (ii) noise from photon sources and telescope sensors pollute the image [8]. Neural networks and as of more recent, convolutional neural networks are now being increasingly more used for this task. An example of astronomical image reconstruction with such a neural network for Hubble deep-field images is shown in figure 1.1 [10]. Motivated by these promising examples, the goal of this research is to create a ML-based approach

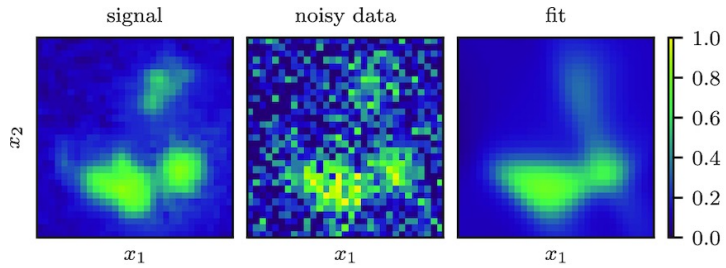


Figure 1.1: *Example of image reconstruction for modified images from the Hubble Space Telescope eXtreme Deep Field (Illingworth et al. 2013) using a neural network. The original signal (left) is turned into a noisy image (middle) by applying artificial, Gaussian noise. The neural network reconstructs the original signal from this noisy image (right). Adopted from [10].*

to noise characterisation in current measurements. We are interested in current fluctuations in scanning tunnelling microscopy (STM) measurements, as the noise in such measurements can reveal a lot about underlying physics. Specifically, shot noise is of key importance as it tells us something about the charge carriers and what processes are at play. We will see in section 2.2 how shot noise measurements allow us to identify Cooper pairs in superconducting junctions and how currently shot noise is extracted. In this research a neural network is trained to quantify noise in current time traces, using both generated data and real data from an elementary model of a high frequency (MHz) STM setup.
The outline will be as follows: we start with a theory chapter that includes a general outline of neural networks and a description of shot noise and its applications to quantum processes in scanning tunnelling junctions. In chapter 3 the neural network architecture that is used and the different learning processes involved are discussed in more detail. In addition, the experimental setup that is used to model a STM is described. In chapter 4 the neural networks performance on an artificial dataset and on circuit measurements is presented, including an analysis of the used resonator circuit. The following chapters will discuss the results in more detail and present its conclusions, also giving an outlook for future endeavours.

Chapter 2

Theory

We start with a description of neural networks: what they are and what they try to do and how they learn. A bit of mathematics is needed but we do not go into too great detail. After the treatment on neural network we turn our attention to the physics side of things. Shot noise will be discussed as well as other noise sources in STM setups. We will also give an example of how shot noise can reveal physical properties of the system and conclude with a brief description of how currently shot noise is extracted from spectral density measurements in a high frequency STM setup.

2.1 Neural networks: the basics

In general, neural networks are algorithms that, by learning from data, build one big function f_θ that predicts the output $\mathbf{y}$ from an input $\mathbf{x}$; the prediction being $f_\theta(\mathbf{x})$. Input and output are vectors, thus the natural language for neural networks is linear algebra. In general, $\mathbf{x} \subset \mathbb{R}^{d_0}$ and $\mathbf{y} \subset \mathbb{R}^{d}$ where d_0 and d are the input and output dimension respectively. We denote a dataset consisting of n input, output tuples $S = \{\mathbf{x_i}, \mathbf{y_i}\}_{i=1}^{n}$. The neural network takes the input vector $\mathbf{x}$ and manipulates it into an output prediction $f_\theta(\mathbf{x})$ by applying successive matrix multiplication. After a matrix multiplication, the resulting vector (which we will call a layer) feeds its components through a function σ (the activation function) after adding a bias vector $\mathbf{b}$. The output the first layer $l = 1$ is determined by matrix $\mathbf{W}^{[0]}$ and bias $\mathbf{b}^{[0]}$ (indexing the matrices and biases from 0 by convention [14]):

$$f_\theta^{[1]}(\mathbf{x}) = \sigma \circ (\mathbf{W}^{[0]}\mathbf{x} + \mathbf{b}^{[0]}) \tag{2.1}$$

Where $\circ$ indicates entry wise operation (i.e. every component is ran through the function σ). Instead of $\mathbf{x}$ we might as well have written $f_\theta^{[0]}(\mathbf{x})$ as this is just the input vector. We can now define f_θ for a neural network with L layers by recursion. The output at the first layer is a vector that is in turn the input to the next layer. In general, we can write for layer l:

$$f_\theta^{[l]}(\mathbf{x}) = \sigma \circ (\mathbf{W}^{[l-1]} f_\theta^{[l-1]}(\mathbf{x}) + \mathbf{b}^{[l-1]}) \qquad 1 \leq l \leq L-1 \tag{2.2}$$

Thus, for a neural network of layer size L: $f_\theta(\mathbf{x}) = f_\theta^{[L]}(\mathbf{x})$. Zooming in on a single node (a component of a layer) can be more illuminating to see how its value is determined (figure 2.3). Basically we take a weighted sum of the components of the previous layer:

$$f_{\theta,j}^{[l]}(\mathbf{x}) = \sigma \circ \Big(\sum_i^m w_i^{[l-1]} f_{\theta,i}^{[l-1]}(\mathbf{x}) + b^{[l-1]} \Big) \tag{2.3}$$

With $w_i^{[l-1]} = W_{ji}^{[l-1]}$, the i-th component of the j-th row of $\mathbf{W}^{[l-1]}$.

The bias, $\mathbf{b}$ is a vector with all its components the same value b shifting the output of the activation function either in the positive or negative direction. In figure 2.1 a simple example of a neural network is shown. The activation function $\sigma(x)$ that characterises a layer can take many different forms, depending on the type of learning problem the neural network tries to solve. The type of problem that we are trying to solve here is image reconstruction, which is a type of regression and for this task ReLU activation functions in its hidden layers are most often used. Two other examples of activation functions are given in figure 2.2.

2.1.1 The learning process

For each layer of size m there is an associated $m \times n$ matrix that transforms the layer before of size n. The components of these matrices determine the connection strength between nodes. With the goal of predicting a desired output from a given input in mind, the task of a neural network is to find the optimal weights and biases transforming the input vector in the desired output. This process is called learning and forms the core of the machine learning. All the weights and of the matrices and biases associated with the layers combined parameterise the function f_θ:

$$\boldsymbol{\theta} = \{\mathbf{W}^{[0]}, \mathbf{W}^{[1]}, \cdots, \mathbf{W}^{[L-1]}, \mathbf{b}^{[0]}, \mathbf{b}^{[1]}, \cdots, \mathbf{b}^{[L-1]}\} \tag{2.4}$$

The learning is done by training the neural network on a training set. This generally is a large dataset with of inputs and desired outcomes. In the

Version of July 23, 2022– Created July 23, 2022 - 13:25

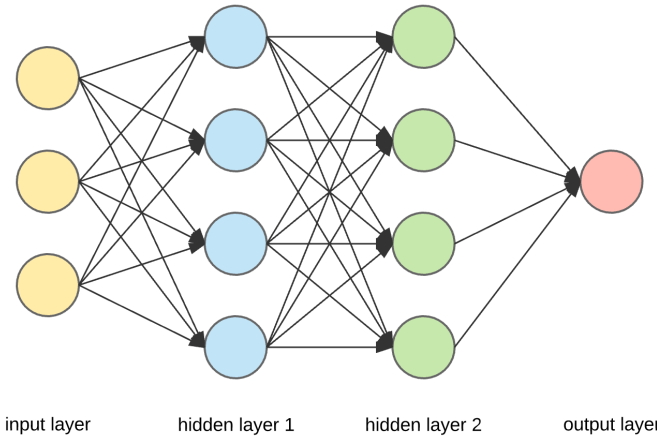


Figure 2.1: *A schematic overview of a simple neural network consisting of 2 hidden layers of size 4, going from an input vector of size 3 to a single output value. The arrows between each node show how the connections between the nodes. Credit: Arden Dertat [6].*

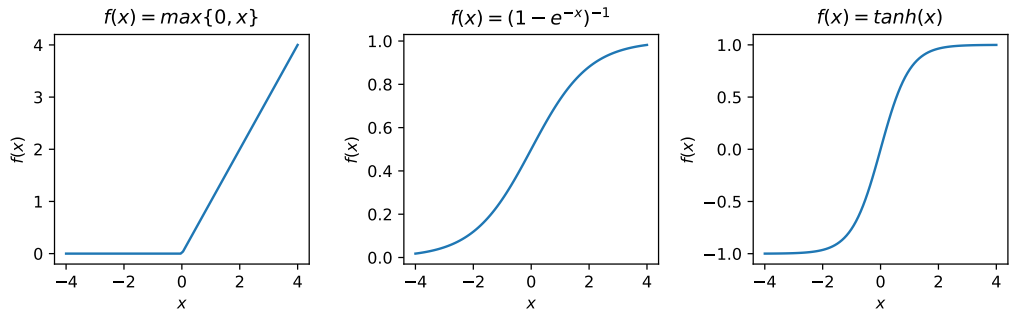


Figure 2.2: *Three examples of activation functions. From left to right: rectified linear unit (ReLU), sigmoid, hyperbolic tangent. Note the differences in range: ReLU has range* $[0, \infty)$*, the sigmoid* $(0, 1)$ *and* $\tanh x$ $(-1, 1)$*. As data is usually normalised between 0-1, the sigmoid activation function is often used in the output layer.*

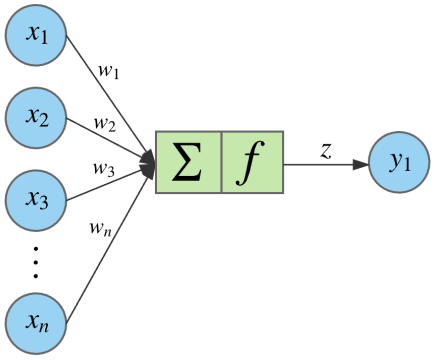


Figure 2.3: *A single node in a neural network: an input vector of length n is manipulated into a single output value y_1. Each node has its own set of weights and activation function f. The output y_1 in turn is itself a component of an input vector for another node. Credit: Arden Dertat [6].*

learning process, a batch (i.e. a subset) of inputs from the training set is run through the neural network, returning some outputs. The weights can have some chosen initial value or be randomised. For each batch of outputs the loss is calculated from a loss function that quantifies the difference between the models output f_θ and the desired output $\mathbf{y}$: $\ell(f_\theta(\mathbf{x}), \mathbf{y})$. Often in regression problems this is the root mean square error:

$$\ell(f_\theta(\mathbf{x}), \mathbf{y}) = \sqrt{\frac{\sum(f_\theta(\mathbf{x}) - \mathbf{y})^2}{n}} \tag{2.5}$$

and this is the loss function that is used here, but can take other forms depending on the use of the model. The learning task is to minimise this loss by changing the weights of the neural network, i.e. finding a local minimum in the loss function. Simplifying a rather complex process, for each training batch i the gradient of the loss function is calculated and subsequently the weights are changed by the negative of that gradient:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \nabla \ell(f_{\theta_i}(\mathbf{x}), \mathbf{y})) \tag{2.6}$$

This will move the weights in the most decreasing direction. After enough iterations, a minimum is achieved and this particular set of weights and biases makes up the predictive model.

2.1.2 Neural network architectures

In the description of neural networks above, we have considered general deep neural networks with arbitrary number of hidden layers and

layer size. In particular, we have only considered feed-forward neural networks: networks that transform an input consecutively into an output. However, a wide variety of arrangements are possible and signals can travel in both directions. For example, nodes can be arranged in a circle, i.e. all nodes are connected directly to each other. A specific type of neural network with a small hidden layer with symmetric surrounding layers is called an autoencoder [12] and will be discussed later. In addition to hidden layers captured by equation 2.3 there are recurrent, convolutional and memory layers, just to name a few. These layers all have their own use and can be put into different arrangements. In figure 2.4 an overview of the most important architectures are shown to illustrate the range of possibilities for neural networks.

2.2 Shot noise and its significance in STM measurements

In a scanning tunnelling junction, a tunnelling current is generated by electrons flowing between the sample and the tip as a result of a voltage difference applied to the sample (the bias voltage). As the current is made up of discrete, uncorrelated electrons, the number of electrons tunnelling within a time interval follows a Poissonian distribution. This gives rise to shot noise: a type of white noise (frequency independent) that is characteristic of discrete, random processes like this. For the tunnelling current I, the shot noise spectrum is given by:

$$S = 2q|I| \tag{2.7}$$

where q is the charge of the current carriers: in normal circumstances this is just the elementary electron charge e. However, in quantum materials electrons are not necessarily uncorrelated and in some circumstances we expect to see electrons tunnelling in pairs. This deviation from a purely Poissonian process is quantified by the normalised noise S_n, defined as:

$$S_n = \frac{S}{2e|I|} \begin{cases} > 1, & \text{bunching} \\ = 1, & \text{Poissonian} \\ < 1 & \text{anti-bunching} \end{cases} \tag{2.8}$$

where S is the measured noise. The normalised noise gives us information about the way in which electrons tunnel through the junction. Its value

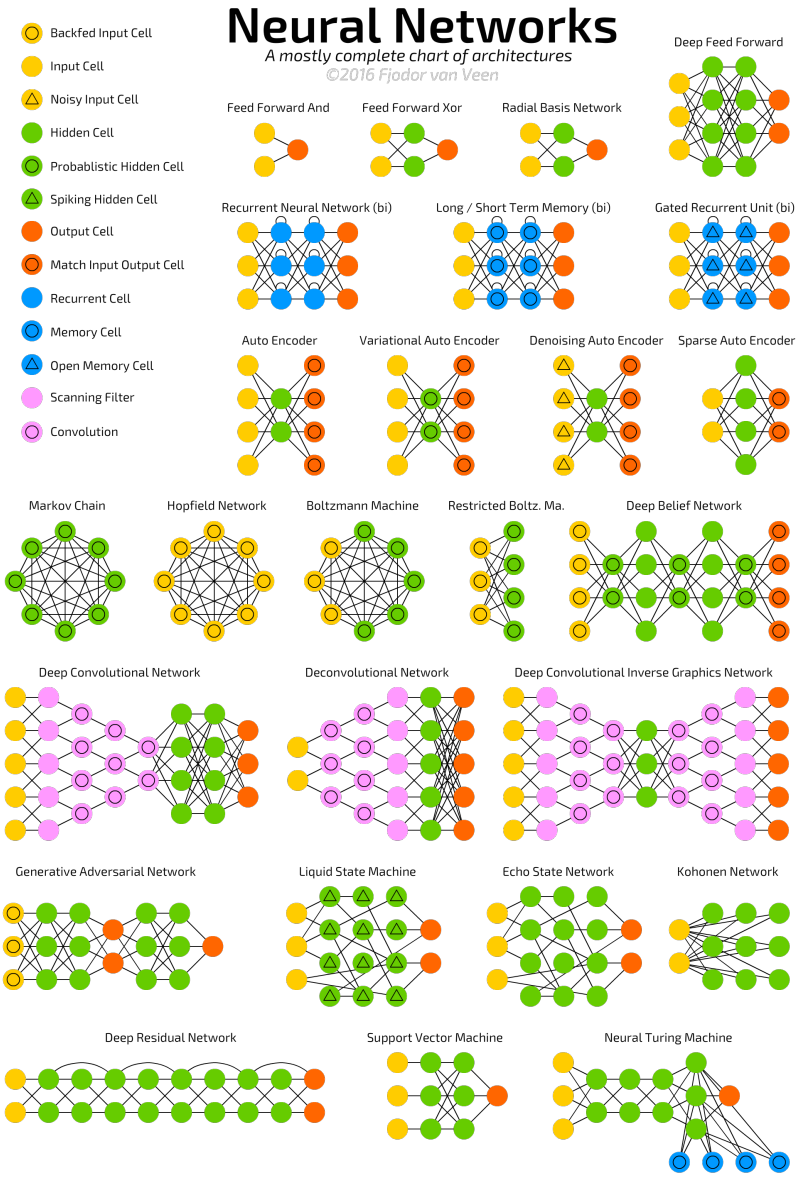


Figure 2.4: *An overview of the many different neural network architectures [13]. In this research we only work with a very simple neural network architecture: the denoising autoencoder (third row). There is a wide variety in neural network architectures: different types of layers, cells and arrangements. A complete description of each of the presented architectures can be found in the accompanying paper by Stefan Leijnen and Fjodor van Veen [12]*

tells us if electrons are tunnelling randomly, in groups (bunching) or the reverse (anti-bunching).

2.2.1 Cooper-pairs in Josephson junctions and shot noise doubling

Without going in too much detail on superconducting scanning tunnelling microscopy (STM), an important application of noise measurements in this field is described briefly here. In a standard STM system, a current flow between tip and sample is induced by a bias voltage on the tip. This current is present due to the quantum mechanical effect of tunnelling. In superconducting STM, both tip and sample are made of superconducting materials. At temperatures below a certain critical temperature T_c, a superconductor is described as a condensate of electron pairs called Cooper-pairs [5]. Cooper pairs are curious particles in their own right: the electrons that form a pair carry opposite spin and opposite momentum. Because of the opposite spin of the constituent electrons, Cooper pairs have integer spin and thus are Bosons. This means that they can condensate into the ground state: this is what happens in the superconducting regime. The state of a superconductor can be described by a wavefunction:

$$\Psi = \sqrt{n}e^{i\phi} \tag{2.9}$$

where n is density of Cooper pairs and ϕ is the phase of the condensate. The combination of superconducting tip and sample creates a so called Josephson junction: two superconductors separated by an insulator. The Josephson effect dictates that the tunnelling (super)current is given by:

$$I_S = I_C \sin \Delta\phi \tag{2.10}$$

where I_C is a maximum supercurrent and $\Delta\phi$ is the phase difference between the two superconductors. The charge carriers of this supercurrent are the Cooper-pairs described above and thus carry charge $2e$. Referring to equation 2.7 this means that the shot noise is doubled. This is an example to illustrate how complex physical phenomena can be captured by noise measurements: the main motivation for this research into the application of neural networks to noise characterisation.

2.2.2 Other noise sources in STM

Shot noise being the noise source of main interest, there are other noise sources to keep in mind. In the lower frequency range, up to $\sim$ 1 kHz,

flickering and mechanical noise are dominant, both inversely proportional to frequency. This flicker 1/f noise is present in all active and most passive electric components [4]. Furthermore, there is thermal noise present across the whole spectrum (thus also a source of white noise). For example, the noise spectrum for the current across a resistance R is given by:

$$S_I(f) = \frac{4k_BT}{R} \tag{2.11}$$

where k_B is Boltzmann's constant and T the temperature. The thermal noise can be distinguished from shot noise as its independent of current. That means; if we measure the noise at zero current we find the thermal noise and increasing the current from there we measure shot noise. Also, note from equation 2.11 that thermal noise $\propto T$ and can be minimised by decreasing T. This is done in ultra low temperature cryostats where temperatures $\sim$ 4K are met. From these considerations, it is easy to see why high frequency measurements can be beneficial: the 1/f and mechanical noise contributions are increasingly less significant to the thermal and shot noise.

2.2.3 High frequency noise scanning tunnelling microscopy (NSTM)

We now describe briefly the process of noise scanning tunnelling microscopy, as it performed in a high frequency STM setup from the Milan Allan research group at LION [2]. A bias voltage over the STM junction generates a tunnelling current. This current is then manipulated in three steps. First, the current signal is separated into high and low frequency components. The low frequency component is used for feedback: keeping the tip and sample at fixed distance. The high frequency part then goes through a resonator tank circuit, where the current is converted into a voltage at a resonance frequency of around 3 MHz. This voltage signal, peaked at this resonance frequency, is converted to a current and amplified by a high electron mobility transistor (HEMT) and amplified further by a 40dB amplifier. This signal is used for power spectral density measurements, from which shot noise can be extracted. A detailed calculation [5], specific to the resonator circuit used in this particular setup shows that the current noise is given by:

$$S_I(I,\omega) = 2q^*I \coth\left(\frac{q^*V}{2k_BT}\right) + \frac{4k_BT}{\left|Z_{res}(\omega)\right|} + S_{amp} \tag{2.12}$$

Version of July 23, 2022– Created July 23, 2022 - 13:25

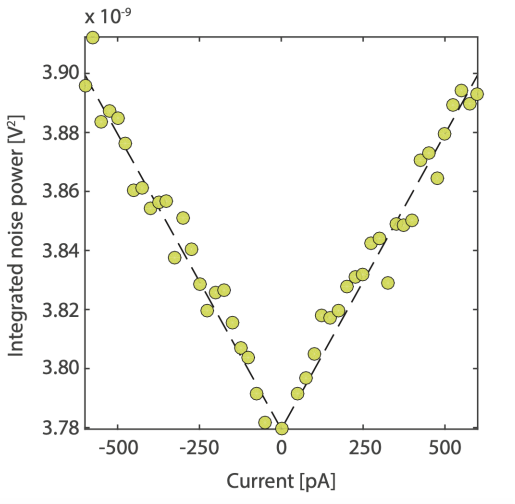


Figure 2.5: *Noise measurements taken at a random location in an Au sample using the setup described above [2]. The dashed line indicates purely Poissonian shot noise. The vertical axis measures the integrated noise power as function of current.*

where q^* is the effective charge carrier, V the bias voltage applied to the tip, Z_{res} the resonator impedance and S_{amp} the amplifier noise. The particular form of equation 2.12 is not of concern here, but note that it consists of three components: a current-dependent, shot noise part (compare with equation 2.7), a thermal part (compare with equation 2.11) and the amplifier noise. The shot noise vanishes at zero current, so subtracting $S_I(I = 0)$ then gives us the shot noise part only. Again, we can extract the carrier charge q^* from the shot noise. In figure 2.5 an example of shot noise measurements using the setup as described above on a Au sample.

Chapter 3

Methods

3.1 Neural network architecture, learning processes and training data

3.1.1 The denoising autoencoder

The type of neural network that is used in this research is an autoencoder. This neural network has a specific layer structure: it consists of an encoder, that transforms the input through a number of hidden layers into a substantially smaller sized layer called the code, and a decoder that transforms the code into an output of the same size as the input. The code is also referred to as the latent space. A simple visual representation of an autoencoder is given in figure 3.1. Autoencoders are often used for denoising images. The autoencoder essentially learns to compress the image into a more elementary representation (the code or latent space) and by doing so it can take a image with noise, transform it to a reduced representation and decode that into an image with reduced noise. This process is shown schematically for 2D images in figure 3.2. Using the output of the denoising autoencoder, we can calculate the noise present in a signal: we can calculate the signal that is filtered out (input - output) and determine its standard deviation. This gives us the neural networks estimation of the noise level in the original signal.

3.1.2 Supervised learning

Neural networks can be trained using training data that has a clear desired output for each input vector. This type of training is called supervised

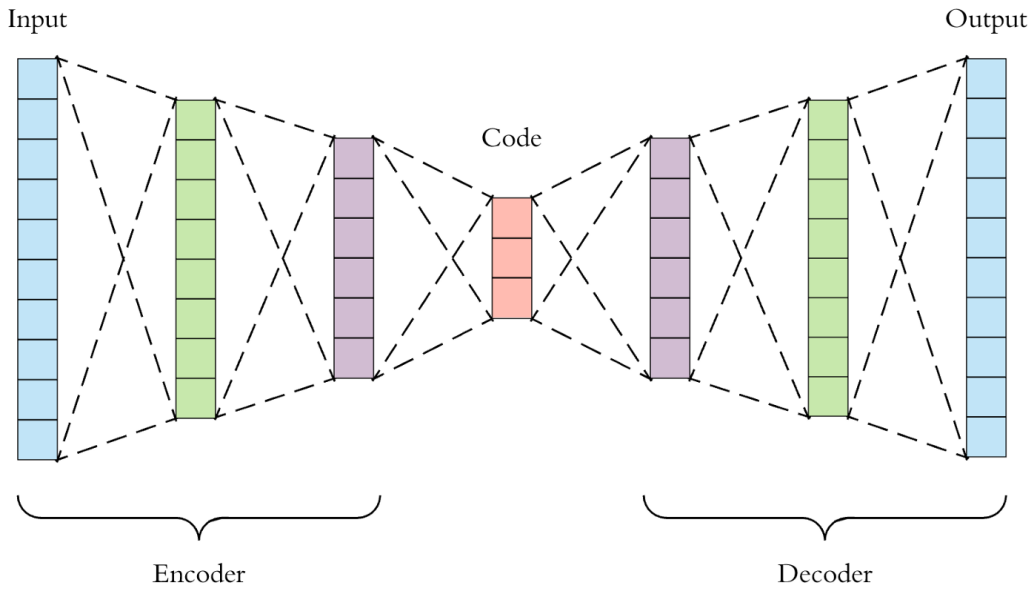


Figure 3.1: *An example of a basic autoencoder structure. Here, both encoder and decoder consist of two hidden layers, transforming the input vector to a smaller sized code layer that is in turn decoded into an output of the same size as the input. In general, there can be any number of hidden layers and layer size may vary. Credit: Arden Dertat [7].*

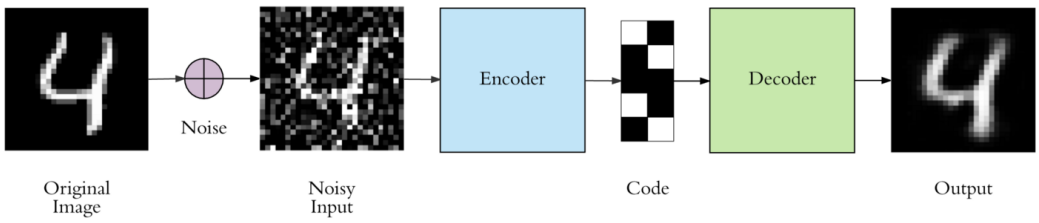


Figure 3.2: *The denoising power of an autoencoder. A noisy image is created by adding noise to a clean original of a handwritten number. The autoencoder encodes the image to a smaller representation and from that the decoder extracts a less noisy image. Credit: Arden Dertat [7].*

learning, where the loss function quantifies the difference between the two (see section 2.1.1). Within the context of noise suppression, this desired output is referred to as a ground truth: the input without noise. Evidently, such data is not accessible for experimental data. In order to create and test a supervised form of noise suppression, we therefore generate a dataset with artificial noise. This will be our starting point from which to train a denoising autoencoder neural network.

3.1.3 Self-supervised learning

The final goal is to learn the autoencoder to suppress noise using only noisy data. Learning without a certain desired output is called self-supervised or blind learning. The technique used here is taken from the general framework proposed by Joshua Batson and Loic Royer 2019 [3]. The idea is to manipulate the original dataset in a specific way, thereby creating a new input while the original data can be used as "ground truth" from which to calculate the loss function. The assumption being that the noise is uncorrelated, the procedure is as follows: at a regular interval input vector components are assigned the mean value of its neighbours (see figure 3.3). The original (unchanged) vector now is the desired output. In the learning process, the model's outputs are only evaluated at those components that have been changed. This way, we can achieve self-supervised learning and filter out the uncorrelated noise: exactly what we want in our scanning tunnelling junction's measurements.

3.1.4 Generating data, splitting the data and cross-validation

As discussed above, we need to generate some artificial data in order to create a set of inputs with a noiseless output. The type of data generated is intended to mimic that of a signal time trace and can either be a constant signal, a little skewed or very slightly sinusoidal. To these functions a Gaussian noise pattern is added with a characteristic amplitude which we will call the training noise level, which will play an important role later. As is standard procedure, the data is normalised between 0-1. From the dataset with such time traces we split the data in three parts: a training set, a validation set and test set. This is also standard practice in machine learning: it is insightful to see how the network performs on unseen data whilst it is learning (validation) and of course when testing a model, the test data should not contain training data. To give a numeric description of the type of data: the input dimension is set to 1000, of which there are

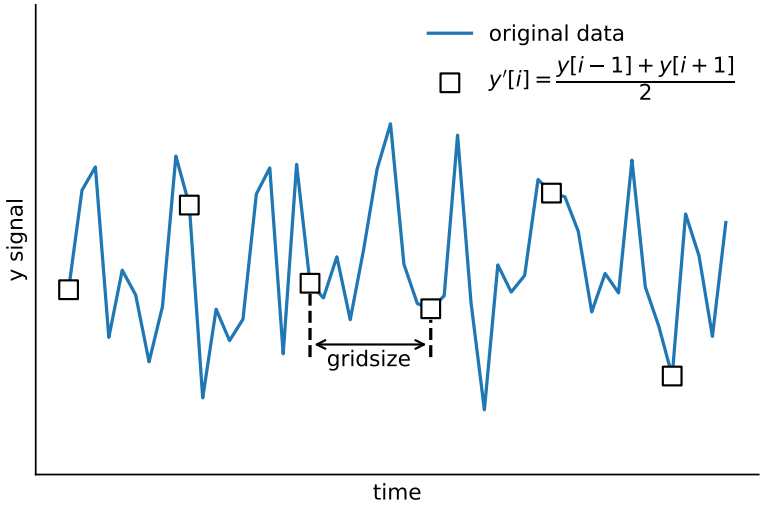


Figure 3.3: *Illustration of the process behind the self-supervised learning used here. A new dataset is generated from the original by assigning the mean value of neighbouring datapoints for a set of points in the data spaced evenly (spacing = gridsize). When training, the loss function is only evaluated at the square datapoints as indicated in this plot.*

Version of July 23, 2022– Created July 23, 2022 - 13:25

10 000 training examples from that a subset of 15% is used for validation. The test set contains 1000 examples of the same input dimension. Both the training and test set examples are evenly split in straight lines, skewed and curved time traces. The noise is added with `np.random.normal`.

3.2 Modelling a high frequency STM setup

In order to benchmark how well the denoising neural network performs on real experimental data, it is tested using an elementary model of a typical high frequency STM setup. The fluctuations in the tunnelling current are modelled by a signal generator outputting white noise. To move into the MHz regime, this signal is passed through a resonator circuit with a characteristic resonance frequency $f_{res} \sim 1$ MHz. Using a lock-in amplifier the filtered signal is moved to DC. Voltage/current time traces are then fed through the neural network as input in order to characterise the noise in these time traces $I(t)$. The resonator circuit used here is a parallel LCR circuit (see fig. 3.4). Its impedance is peaked around a resonance frequency $f_{res} = (2\pi\sqrt{LC})^{-1}$. The current response over R is given by:

$$I_R(\omega)/I(\omega) = H_{res}(\omega) = \frac{i\omega L}{R - \omega^2 LCR + i\omega L} \tag{3.1}$$

We also need to take into account the internal low-pass filter used by the lock-in amplifier. After the signal is moved from ω_{res} to DC, the lock-in amplifier applies a low pas filter in order to only measure signals that originate from frequencies close to ω_{res}. This low-pass filter takes the shape of a n-th order coupled RC-circuit with -3dB frequency $\omega_{RC} = (RC)^{-1}$:

$$H_{lock-in}(\omega, n) = \left[1 + \left(\frac{\omega}{\omega_{RC}}\right)^2\right]^{-n/2} \tag{3.2}$$

In our measurements we have used a low pass filter in the lock-in of order 8 at a -3dB frequency of 99.51 Hz. Combining 3.1 (shifted so that resonance is at DC) and 3.2 gives the total transfer function $H_{tot}(\omega) = H_{res}(\omega + \omega_{res})H_{lock-in}(\omega)$. As the input noise is white noise, the input noise spectrum $S(\omega) = S$ and relates to the total noise via an integral over the absolute square of the transfer function of the circuit:

$$\sigma^2 = S \int_0^{\Delta\omega} \left|H_{tot}(\omega)\right|^2 d\omega \tag{3.3}$$

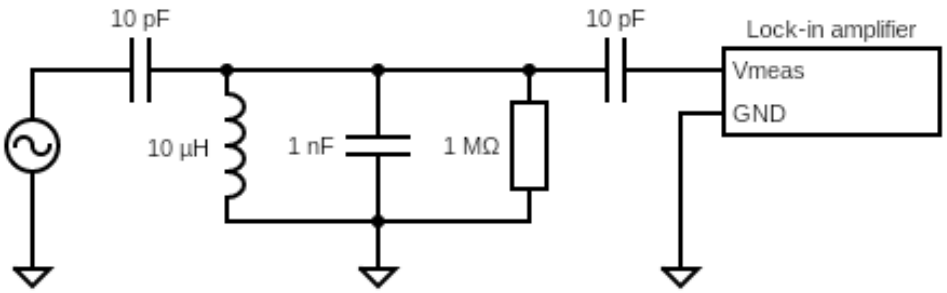


Figure 3.4: *The resonance circuit used to measure in the MHz range. The source of the signal comes from a signal generator and goes through a parallel arrangement of L, C and R components that are "isolated" using two 10 pF capacitors.*

where $\Delta\omega$ is the measurement bandwidth of in radians ($\Delta\omega = 2\pi\Delta f$). The neural network estimates σ^2 from current time traces in A^2 and equation 3.3 then gives us the noise spectrum level in A^2/Hz. In practice however, we skip these steps and put the whole transfer function in a factor that translates the noise amplitude from the signal generator into a standard deviation in the measured signal on the lock-in. This is done by measuring the standard deviation at different amplitudes and applying a linear fit. Also, we do not make current time trace measurements but measure voltages. After demodulating with the lock-in amplifier we measure the real part of the demodulated signal.

Chapter 4

Results

4.1 Hyperparameter search

A neural networks structural layout is determined by the number of layers and the sizes of those layers; its performance is determined further by the choice of activation functions, loss function, number of learning epochs, training data and some more sophisticated parameters that go into the learning process. Those include: learning optimiser (which has learning rate as its own parameter), batch size, weight initialisation and for the self-supervised learning method that is used here: the gridsize (the spacing between the modified components in the process of creating a separate dataset). It would be computationally expensive to go over all the possible combinations of parameters. Instead, we start with a set of parameters that performs reasonably well and tweak a number of parameters individually from there. The starting point for our neural network structure and parameters is taken from an autoencoder used for denoising 2D images. The results are given in table 4.1 and a plot of a training learning curve is given in figure 4.1a.

4.2 Performance on artificial dataset

Both neural networks (supervised and self-supervised) are learn from a training set with a constant input noise level of 0.3. We then measure the performance a denoising neural network in two ways. The first is the mean squared error value of the network's output on the test set and we vary the standard deviation of the noise. The second measures how accurately the network predicts the noise in the input time traces. The standard

input dimension	1000
no. of hidden layers	4
size hidden layer 1	128
size hidden layer 2 (code size)	32
size hidden layer 3	128
size hidden layer 4 (output)	1000
total no. of parameters	265480
activation function hidden layers 1-3	ReLU (rectified linear unit)
activation function output layer	Sigmoid
optimizer	Adam
learning rate	0.00005
loss function	mean squared error
no. of epochs	25
batch size	32
gridsize (self-supervised learning)	5

Table 4.1: *An overview of all the parameters choices for the denoising neural network.*

deviation in the filtered out signal is calculated, again varying the input noise. A perfect denoiser would have a one-to-one correspondence which is indicated by a $y = x$ curve in the plots. Of interest is the point where the neural network intersects the y-axis. As the amplitude of the fluctuations becomes very small, the neural network at some point is not able to recognise the noise properly; the value at this intersection will give us the minimum noise level the neural network can predict. The results of neural networks trained by both supervision and self-supervision are presented in figure 4.1. In the appendix, a plot of the different noisy input traces and the neural networks output is given as well as a learning curve.

4.3 Benchmark on a STM model

4.3.1 Circuit analysis

First we start by analysing the resonance circuit. Using the sweep function (amplitude = 1V) on the lock-in amplifier, the circuits transfer function (amplitude, phase) is determined (figure 4.2b). A fit to both curves yields the characteristic values for the resonator (figure 4.2c). The resonance bandwidth Δf is the full-width-half-maximum (FWHM) of the peak. Using the lock-in amplifier, a white noise signal from the genera-

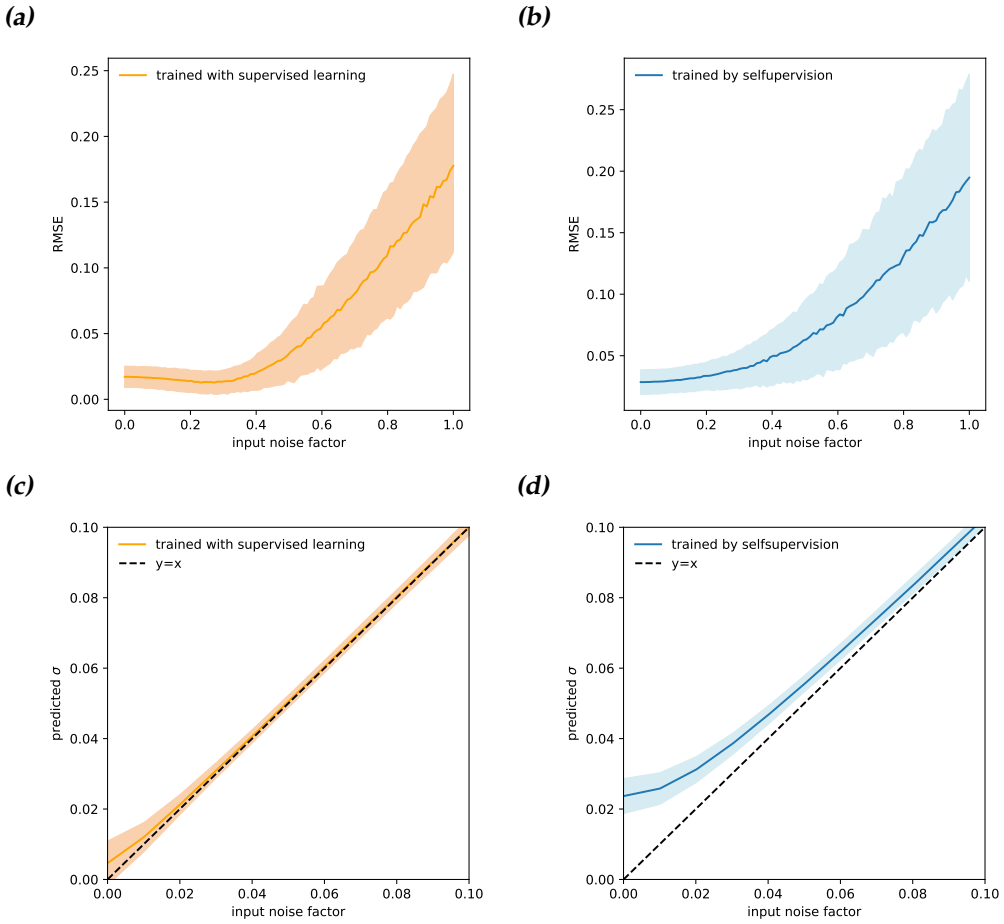


Figure 4.1: *Neural network performance when trained (supervised and self-supervised) and tested on artificial data.* ***a-b.*** *Varying over the input noise factor on test data (corresponding to the input σ) the neural networks (trained by both supervision and self-supervision) root mean square error (RMS) is plotted, at a training input noise factor of 0.3. A bias towards this noise amplitude is slightly visible by a minimum at input noise factor $\sim$ 0.3 in* ***a****.* ***c-d.*** *Noise characterisation by the neural network. The noise estimation of the neural network σ_{diff} plotted against the input noise factor. The intersection of the curves indicates the smallest noise amplitude the neural network is able to detect.*

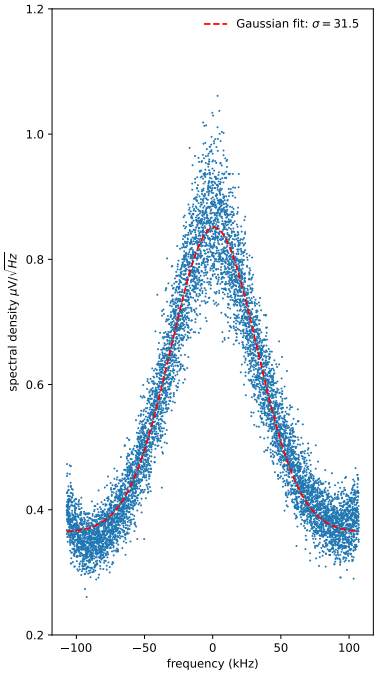
tor (amplitude = 1V, bandwidth = 10MHz) is measured through the circuit and its power spectral density around resonance frequency is plotted in (4.2a).

4.3.2 Testing the neural network on physical data

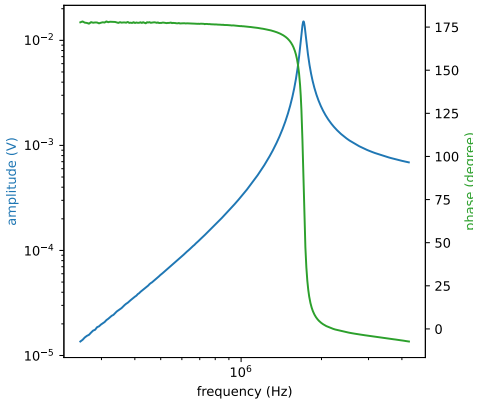
We now train our neural network on physical data; circuit time traces with varying amplitude. Here the method is slightly different than before: we can only use the self-supervised learning technique and in this case we train and test the neural network on the whole dataset, thus already containing different noise levels while training. The results are presented in figure 4.3.

Version of July 23, 2022– Created July 23, 2022 - 13:25

(a)



(b)



(c)

curve fit	amplitude	phase
f_{res} (MHz)	1.714	1.712
Δf (kHz)	73.8	69.5
Q factor	23.2	24.6
fit error	1.3%	0.16%

Figure 4.2: *Analysis of the circuit used to model a high frequency STM setup.* ***a.*** *Power spectral density measured around resonance frequency, dashed line shows a Gaussian fit to the data. The Gaussian fit yields $\sigma = 31.5$ kHz, corresponding to a FWHM bandwith of $\Delta f = 73.6$ kHz.* ***b.*** *Frequency response: amplitude and phase showing clear resonance curves (amplitude peaked around resonance frequency, phase shift of -180 degrees at resonance) used to determine the resonators resonance frequency f_{res}, its bandwidth and Q factor.* ***c.*** *Curve fit results. The fit to the phase frequency response curve yields the best fit, with fit error almost 10 times lower than the amplitude fit.*

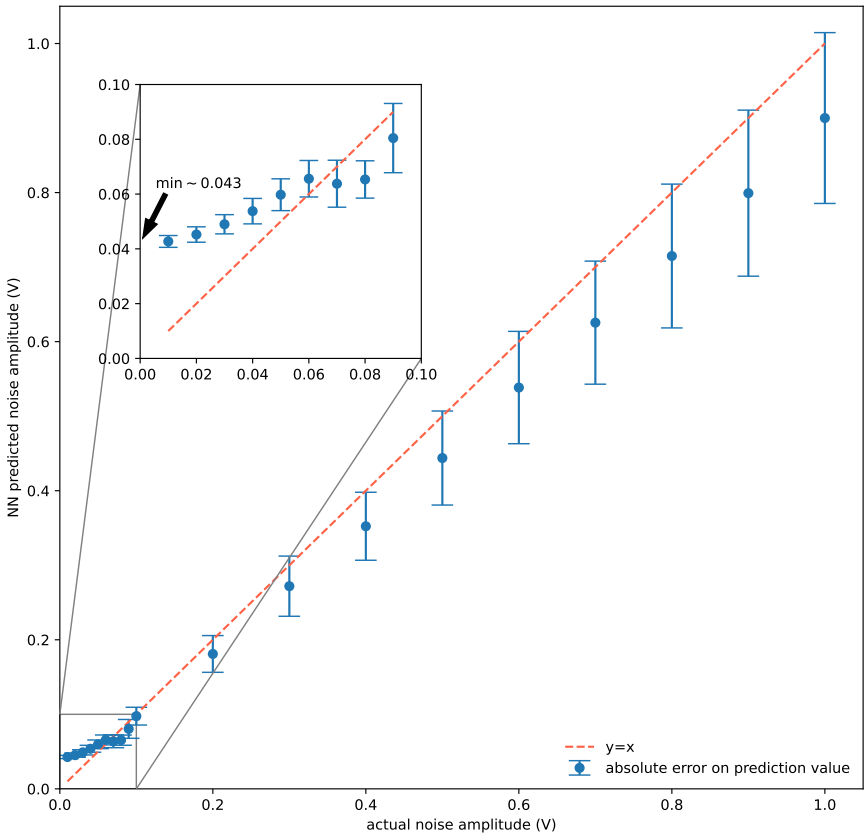


Figure 4.3: *Neural network (trained by selfsupervision) predicting the noise amplitude of the signal generator's input. The input signal goes through the resonator circuit and is measured and demodulated by the lock-in. We see a clear linear response up to a point ∼ 0.06 V, where the datapoints begin to branch off. Zooming in on the datapoints at a smaller input noise range (0-0.1 V), we see an (extrapolated) intersection with the y-axis of 0.043 V. This value sets the minimal noise level the neural network is able to detect.*

Chapter 5

Discussion

5.1 The neural network and its simulation on artificial data

We start our discussion with a few remarks about the neural network itself and the way it is trained for its performance measurements. This is followed by a discussion of the results of the neural networks performance on the artificial dataset.

5.1.1 Remarks regarding the neural network parameters and architecture

The neural networks parameter choices as listed in table 4.1 are somewhat arbitrary. For the most part, very generic choices of activation functions, optimizer and weight initializer are made with a few tweaks. For example, the learning rate turned out to be an influential factor: we started from a more generic rate of 0.001 and increasing it made the neural networks prediction less sensitive (i.e. the intersection with the y-axis in figures 4.1c-d) and also have increased spread. Lowering it substantially to a value of 0.00005 seemed a good value. This example is illustrative of how most parameters are chosen: starting from a standard value, experimenting a bit while keeping other parameters the same. Of course, this is not the most thorough way to go about selecting parameters. It is, however, more practical and less time consuming than iterating over possible combinations. That said, it might be effective to do a full hyperparameter grid-search.
The neural network structure can also be more fine-tuned. The number of hidden layers and layer size is kept quite low, for two reasons: i) it reduces

training times to < 5 min and ii) to prevent overfitting to the training data (this happens when the number of nodes is on the order of the total number of training datapoints). Also, experimenting a bit with an extra hidden layer did not seem to increase performance. However, there are many possibilities for the neural network architecture (such as an arrangement of more, but smaller hidden layers) that can still be tested. While the neural network used in this research is build only from input, output and hidden layers, there are also different types of layers (as pointed out in section 2.1.2), that can be experimented with.

5.1.2 Remark about the method of measuring neural network performance

The reason to create the plots given in figure 4.1 is that we want to test the neural network performance on different input noise levels. However, there is one subtlety at play: the neural network is trained on a training dataset with a specific input noise level (referred to as training noise level). This creates a bias towards noise levels close to this value, as can be seen in the slight minimum in figure 4.1a around the training noise level of 0.3. It is interesting to see that training on one noise level generalises quite well when testing on other noise levels (σ-predictions being accurate on higher and lower noise levels). Again, this is also time effective as we are only training the network once per learning method. That said, it might be interesting to see how these plots would change if the network is trained on each input noise level separately.

5.1.3 Performance results on artificial dataset

As might be expected, the neural network trained with supervised learning outperforms its self-supervised counterpart in both tests. It has a consistently lower RMSE across all input noise levels, and its standard deviation predictions follow the input noise factor more closely and branches of at a lower input noise. The intersection with the y-axis in figures 4.1(c-d) is lower for the supervised neural network, giving a lower errorbar on σ-predictions as well as being the more sensitive denoiser. That said, the neural network trained with self-supervision still performs quite well, when you keep in mind that it has never seen any ground truths in the training process and denoises input signals solely by its masking technique as described in section 3.1.3.
Two things to point out about error margins (indicated by the shaded re-

gions in figure 4.1): i) the error margins on the RMSE plots (figure 4.1a-b) grow linearly with the input noise for input noise levels $>$ training level and remain constant for lower levels. This indicates that the relative error is constant for levels $>$ training level and starts to increase for lower levels. ii) In the σ-prediction plots (figure 4.1c-d) we see that the (absolute) error margins increase for small input noise factors close to the branching-off point.

5.2 The neural network in combination with physical system

We now turn to our discussion of the physical application of our neural network. First we discuss the quality of the used resonator circuit and how well it matches that of a high frequency STM setup, followed by a discussion of the neural network performance on circuit data, concluded by a discussion on the possibility of using this neural network for detecting shot noise.

5.2.1 The STM-model circuit

The circuit used in this research (figure 3.4) is at best a toy-model of the experimental setup with which high frequency STM measurements are made as described in section 2.2.3. The quality factor of the circuit used here of ~ 24 is far below the Q = 600 of the mentioned setup [2]. There are a number of reasons for this:

1. the circuit design of course is different alltogether
2. the circuit is not shielded from external electric fields interfering with the circuits electronics
3. the circuit is not at low temperature, which means the thermal noise is two orders of magnitude higher
4. the circuits components are soldered onto each other via their connecting wires, not onto a printed circuit board. These wires also have self inductance that interfere with the circuit.

That being said, the goal of this research is not to design a high frequency amplifier system for STM measurements, but to test a denoising neural networks performance on time traces somewhat resembling those from

a high frequency STM setup. For those purposes, the resonator circuits qualtity factor of 24 is high enough to measure time traces in a spectral region around the resonance frequency of 1.71 MHz.

5.2.2 Performance neural network on circuit data

The neural network performs in quite a similar manner to physical time traces as it does on the artificial dataset. Its noise amplitude-predictions scale linearly with the actual noise amplitude as they should, as expected branching off at a certain point when its denoising sensitivity limit is reached (figure 4.3). One thing to point out is that the predictions at levels above the branching-off point are consistently a bit below the actual value, which indicates that there is some systemic error in the neural networks predictions. Another unexpected result is that the absolute errors on the predictions grow linearly; the relative error more or less constant. We do not see a clear increase in relative errors with smaller noise amplitudes, something that we did see in the simulation.
The minimum error on predictions, as indicated by the arrow in figure 4.3 serves as a detection limit: lower input noise amplitudes are not detectable. Given that the maximum input noise amplitude is 1V, we get a minimum relative noise sensitivity of 4%. This a bit higher than the result on the artificial dataset, where this level was just above 0.02 for the self-supervised neural network. This might be explained by the fact that in that case there are more generated training samples for the neural network to learn on.
We now give a back-of-the-envolope calculation to see if this neural network can be used to detect shot noise in the setup described in section 2.2.3. For this purpose, we take a closer look at the shot noise measurement presented in figure 2.5. We see a zero-current measurement with an integrated noise power of $3.78 \times 10^{-}9\ \mathrm{V}^2$. We know our neural network has a relative sensitivity of 4%. This means that in this context, the neural network can distinguish integrated noise power values differing by $0.04 \times 3.78 \times 10^{-9}\ \mathrm{V}^2 = 0.15 \times 10^{-9}\ \mathrm{V}^2$. Looking at figure 2.5, we see that this level of sensitivity is not quite enough to reproduce the same plot. The level of sensitivity falls short to ~ 1 order of magnitude. One might still be able to detect shot noise when measuring at bigger intervals of current, say steps of 500 pA, but that might not be an option.

Version of July 23, 2022– Created July 23, 2022 - 13:25

Chapter 6

Conclusions and outlook

From the results we conclude the following points:

1. Proof of concept: neural networks, specifically denoising autoencoders are capable of characterising noise in noisy time-traces. The σ-predictions of the neural network scale with the noise present in the signal.

2. In the simulation, we have seen that supervised and self-supervised neural networks perform in a similar manner. The supervised performs better, measured both in RMSE and σ-prediction sensitivity.

3. Time traces from a physical system can be used to train a neural network by self-supervision. The results are very similar to simulated data; given that there is less training data, the slightly lesser performance is to be expected.

4. The neural network is not yet capable to reproduce regular shot noise measurements. Its sensitivity falls short about 1 order of magnitude.

We point out the following challenges/ideas for further endeavours in this field:

1. The main challenge is to modify the neural networks architecture (amount of layers, types of layers and layer sizes) and parameters in order to improve its sensitivity. For parameters, a full grid-search for hyperparameters might be useful.

2. For simulations, different ways of training the network might be interesting to look at: how will the performance change when the neural network is trained at different noise levels individually?

Chapter 7

Appendix

We show two more figures: the learning curve of a supervised neural network and the neural networks output to illustrate how the traces look like before and after denoising.

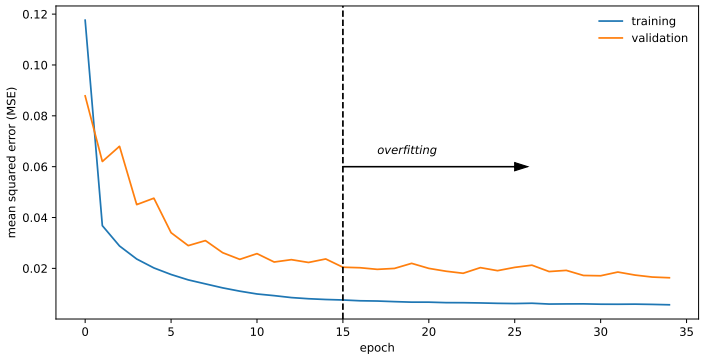


Figure 7.1: *Learning curve of supervised training of the neural network. After epoch $\sim$ 15, we see validation loss not decreasing anymore. Training further would overfit the network to the training data.*

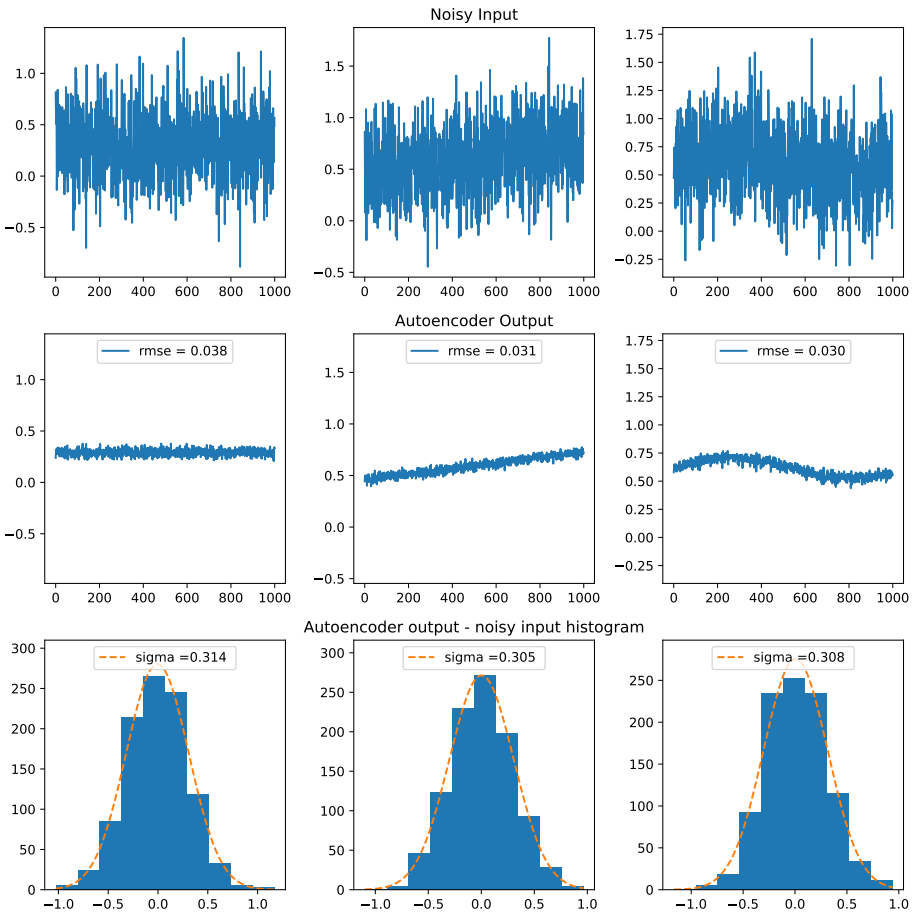


Figure 7.2: *Here show a plot containing generated noisy time-traces (of three types), the neural networks denoised output and a histogram showing the distribution of the filtered out signal with a Gaussian fit to them. The autoencoder gets as input noisy (generated) time traces at noise level amplitude = 0.3. The neural network here is trained by self-supervision.*

Bibliography

[1] Argonne National Laboratory: Linda Colin et al. *Report on the Department of Energy (DOE) Town Halls on Artificial Intelligence (AI) for Science*. Feb. 2019. URL: https://cs.lbl.gov/assets/AI-Report/AI-for-Science-Report.pdf.

[2] Koen Bastiaans. *Probing quantum materials with novel scanning tunneling microscopy techniques*. Dec. 2019. URL: http://hdl.handle.net/1887/81815.

[3] Joshua Batson and Loic Royer. *Noise2Self: Blind Denoising by Self-Supervision*. 2019. DOI: 10.48550/ARXIV.1901.11365. URL: https://arxiv.org/abs/1901.11365.

[4] Bruce Carter. "Chapter 12 - Op Amp Noise Theory and Applications". In: *Op Amps for Everyone (Third Edition)*. Ed. by Ron Mancini and Bruce Carter. Third Edition. Boston: Newnes, 2009, pp. 163–188. ISBN: 978-1-85617-505-0. DOI: https://doi.org/10.1016/B978-1-85617-505-0.00012-0. URL: https://www.sciencedirect.com/science/article/pii/B9781856175050000120.

[5] Damianos Chatzopoulos. *Josephson and noise scanning tunneling microscopy on conventional, unconventional and disordered superconductors*. Nov. 2021. URL: https://hdl.handle.net/1887/3243474.

[6] Arden Dertat. *Applied Deep Learning - Part 1: Artificial Neural Networks*. Aug. 2017. URL: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6.

[7] Arden Dertat. *Applied Deep Learning - Part 3: Autoencoders*. Oct. 2017. URL: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

[8] Remi Flamary. "Astronomical image reconstruction with convolutional neural networks". In: Aug. 2017, pp. 2468–2472. DOI: 10.23919/EUSIPCO.2017.8081654.

[9] Kerstin Hammernik and Florian Knoll. "Chapter 2 - Machine learning for image reconstruction". In: *Handbook of Medical Image Computing and Computer Assisted Intervention*. Ed. by S. Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger. The Elsevier and MICCAI Society Book Series. Academic Press, 2020, pp. 25–64. ISBN: 978-0-12-816176-0. DOI: https://doi.org/10.1016/B978-0-12-816176-0.00007-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128161760000077.

[10] Edward Higson et al. "Bayesian sparse reconstruction: a brute-force approach to astronomical imaging and machine learning". In: *Monthly Notices of the Royal Astronomical Society* 483.4 (Dec. 2018), pp. 4828–4846. ISSN: 0035-8711. DOI: 10.1093/mnras/sty3307. eprint: https://academic.oup.com/mnras/article-pdf/483/4/4828/27441042/sty3307.pdf. URL: https://doi.org/10.1093/mnras/sty3307.

[11] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (Aug. 2021), pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. URL: https://doi.org/10.1038/s41586-021-03819-2.

[12] Stefan Leijnen and Fjodor Veen. "The Neural Network Zoo". In: *Proceedings* 47 (May 2020), p. 9. DOI: 10.3390/proceedings47010009.

[13] Fjodor van Veen. *The Neural Network Zoo*. Apr. 2019. URL: https://www.asimovinstitute.org/neural-network-zoo/.

[14] Zhi- Qin John Xu. *Suggested Notation for Machine Learning*. 2020. URL: http://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/mlmath/mlmath.pdf.