Masters Thesis

Deep Learning based Super-Resolution for Sparse Photoacoustic Microscopy

Tara Vanhatalo

Under the supervision of Teemu Myllylä, Matteo Pedone and Arthur Leclaire

M2 Mathematical modelling for image and signal processing
Bordeaux University - University of Oulu

Acknowledgements

I would first and foremost like to thank my supervisors in Finland, Teemu Myllylä and Matteo Pedone. Thank you to Teemu for firstly the opportunity to carry out this Masters thesis at the University of Oulu and also for all the support, encouragement and understanding throughout my time in Finland. Thank you to Matteo for all the guidance and constructive criticisms. And a special thank you to Zuomin Zhao for all his help, patience and work with the photoacoustic device.

I would like to express my great appreciation to my supervisor in France Arthur Leclaire as well as Edoardo Provenzi for their help and support as well as to my other professors at Bordeaux University, Jean-François Aujol and Nicolas Papadikis, for making this second year of Masters as informative and enriching as it was. I am also grateful to Caroline Petit for her work and patience with administrative matters.

I also wish to acknowledge CSC – IT Center for Science, Finland, for computational resources.

Finally, I would also like to extend my thanks to my friends and family for always supporting me, believing in me and being there for me.

Abstract

Photoacoustic microscopy (PAM) is a rapidly emerging 3D imaging modality which has the beneficial ability to produce high-resolution images at depths beyond the 1mm penetration limit encountered with other similar imaging modalities, thus proving to be of valuable use in bio-medical research and clinical practices. However, image acquisition in PAM is slow and poses a problem for real-time imaging. Sparse scanning coupled with image reconstruction methods can be used to improve acquisition times. Recent work has largely focused on viewing the image reconstruction in this case as a matrix completion problem, that requires iterative algorithms to solve which can often be time-consuming. In this masters thesis, the approach proposed is to view the image reconstruction as a problem of single-image super-resolution (SISR) instead to be solved using deep learning. A generative adversarial network (GAN) is implemented, dubbed fast-SRGAN, which is a variation of the well-known SRGAN architecture optimised for improved speeds. To the best of our knowledge, this is the first time that the image reconstruction in sparse PAM has been considered as a SISR problem and the first use of deep learning in this context. This approach has yielded promising results that are both visually compelling and fast to obtain, in the aim of leading to the visualisation of dynamical changes in the scanned sample, something that is not yet visible in conventional PAM.

Contents

1. Introduction

Photoacoustic microscopy (PAM) is an imaging technique based on the photoacoustic effect: a pulsed laser beam illuminates the sample to be studied, subsequently heating it up and creating pressure waves which are then acquired by an ultrasound transducer thus leading to the construction of photoacoustic images with optical contrast and ultrasound resolution [1]. Unlike photoacoustic computed tomography (PACT), the data acquisition is direct and thus no image reconstruction is required in PAM in order to obtain the complete photoacoustic image. This process, if the photoacoustic scanning head remains immobile, captures a 1D depth image using the acoustic time-of-flight information. Two-dimensional raster scanning is then implemented in order to produce 3D volumetric images.

PAM bears numerous advantages over other optical microscopic imaging modalities such as confocal microscopy or optical coherence tomography (OCT). Firstly, it is a label-free technique [2], deeper imaging depths can be obtained with PAM (beyond the optical diffusion limit of 1mm) since ultrasonic scattering by tissue is over three orders of magnitude weaker than optical scattering [5]. Typical optical microscopic techniques rely on sharp optical focusing which becomes progressively more difficult the deeper we go into the tissue due to the scattering of photons in this media. Past the optical diffusion limit of 1mm under the skin, these photons have undergone so much scattering that their paths become randomised, thus preventing tight focusing. The weaker scattering in biological tissues of ultrasounds remedies this [1].

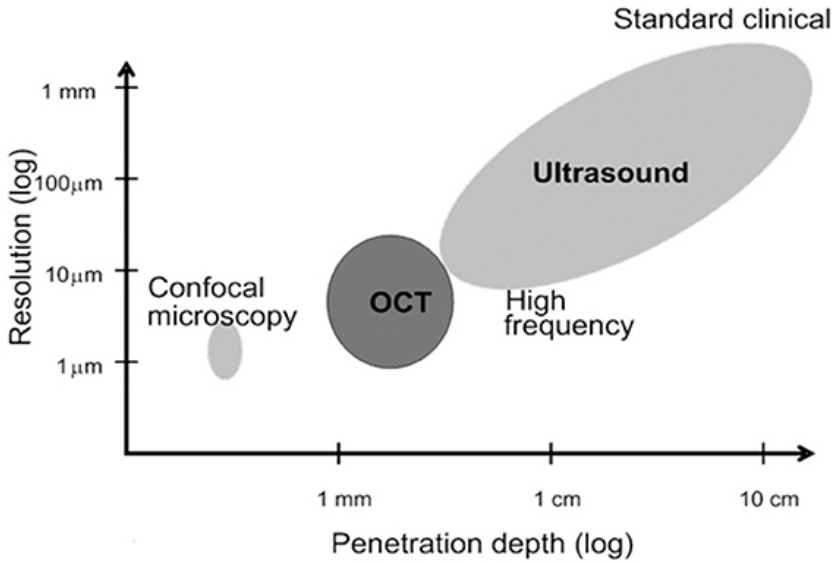


Figure 1.1: **Confocal vs. OCT vs. Ultrasound.** (Source: Karunamuni, Ganga & Gu, Shi & Ford, Matthew & Peterson, Lindsy & Ma, Pei & Wang, Yves & Rollins, Andrew & Jenkins, Michael & Watanabe, Michiko. (2014). Capturing structure and function in an embryonic heart with Biophotonic tools. Frontiers in physiology. 5. 351. 10.3389/fphys.2014.00351.)

Secondly, both structural and functional information is retained in PAM due to the intrinsic contrast provided by the photoacoustic effect [8]. And finally, there is no need for optical sectioning in order to obtain 3D images (due to the depth-resolved nature of the data acquired, 3D images are directly formed through raster scanning).

There are two main categories of PAM: optical-resolution PAM (OR-PAM), which uses a tighter optical focus than acoustic focus, and acoustic-resolution PAM (AR-PAM), which uses a tighter acoustic focus than optical one. OR-PAM has a maximum imaging depth limited to around 1mm [4], whereas AR-PAM's maximum imaging depth depends on a number of factors like pulse repetition rate [5], laser wavelength and ultrasound frequency, or the number of lasers used. Generally, the maximum imaging depth or AR-PAM is around a few millimetres [1].

PAM is a non-ionising *in vivo* imaging modality that is therefore safe for patients and applicable to a wide range of tasks such as tumour detection in breast tissue [8], oxy- and deoxy- haemoglobin monitoring, and the study the mechanism and functions of blood vessels, capillaries and tissue. At present, at the Biocenter in Oulu, the goal of the PAM device is to use it in studies of the blood brain barrier (BBB) [35] and the glymphatic system [36]. High-speed imaging in PAM is sought after to enable blood vessel dynamics monitoring and in order to avoid motion artefacts caused by the patient during scanning such as breathing or heart beat [1].

Unfortunately, the acquisition process in PAM is notoriously slow due to how the information is physically detected, therefore one of the main goals of image processing in PAM is to improve its acquisition speed while maintaining good spatial resolution and avoiding image artefacts. This will be the main focus of this Masters thesis. A way to speed up imaging would be to reduce the number of acquisition points and then implement a means of creating an adequate image from this sparse data, enabling for quick previews of the photoacoustic image. This acquisition process is dubbed sparse photoacoustic microscopy (sparse PAM).

The current techniques being studied to improve acquisition speed in PAM can be separated into two categories: modifying the physical scanning setup (e.g. by introducing a high-speed voice-coil stage [6]) or modifying the digital acquisition (e.g. acquiring sparser data as mentioned above [8]-[13]). In the current setup at the University of Oulu's Biocenter, voice-coil scanning has been implemented and thus this work will explore the digital and image processing techniques that aim to further improve acquisition speed.

Most attempts to improve the imaging speed of photoacoustic modalities rely on sparse scanning followed by iterative methods for image reconstruction, with the most of these methods aiming to solve a low-rank matrix completion problem (an approach which will be presented in Chapter 2). In this Masters thesis, the use of super-resolution as a means of reconstructing the photoacoustic image from sparsely acquired data is studied in an attempt to bypass the more time-consuming iterative approaches that are usually used and bring photoacoustic microscopy a step closer to imaging dynamical changes in the scanned sample.

The layout of this thesis will be as follows: firstly, the current image processing techniques used to speed up photoacoustic imaging will be detailed and compared, then the super-resolution approach will be discussed, followed by a presentation of the implementation including a description of the program used for the digital acquisition in the current PAM setup as well as its integration with the Python programming language for the image processing and the GPU resources provided, and finally the results of this approach will be presented and discussed as well as the optics for future work on this project.

Sample code and trained networks available at: https://github.com/tvanhatalo/PAM_SR.

2. Current Sparse PAM Techniques

In conventional photoacoustic microscopy, the measured data is acquired by point-to-point mechanical scanning of the sample in order to obtain high-resolution images (in our case, 2D raster scanning is implemented). This process tends to be time-consuming and thus poses a problem for real-time imaging. In order to find a way around this, alternate scanning schemes have been proposed.

In the OR-PAM setup developed at the Biocenter for high-speed imaging, two separate pulsed lasers (one at a wavelength of 532nm and another at 561nm) are used to heat up the sample in order to create pressure waves and the resulting photoacoustic data that is acquired in PAM. The two laser beams go through a series of optical elements (e.g. light filters and a fibre coupler) before going into a photoacoustic probe through a single mode fibre. The laser beams are redirected at a 90 degree angle with a prism before illuminating the sample. The photoacoustic signal is received by an ultrasound transducer and is then amplified by two amplifiers and before being acquired via a digital card in a personal computer. A FPGA is used to control the various elements of the setup and the data acquisition and visualisation is done in LabVIEW. The setup described here is illustrated in the figure below.

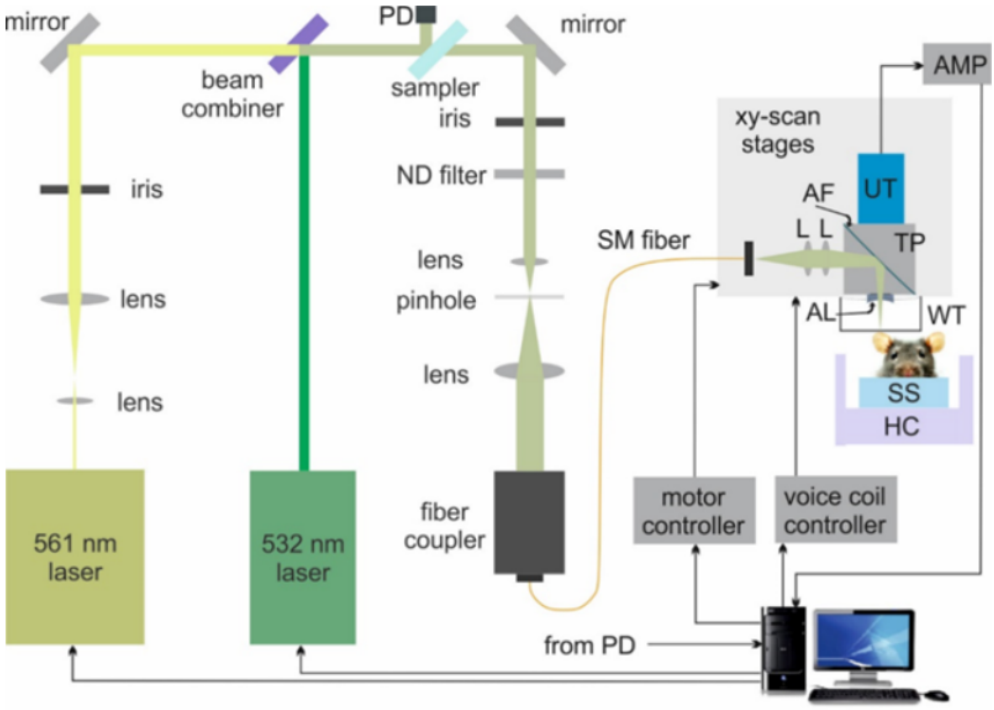


Figure 2.1: The OR-PAM setup developed at the Biocenter Oulu

As with for many other medical imaging modalities, the sparsity of PAM has been proven and explored with such techniques as compressive sensing, a technique widely used in photoacoustic computed tomography (PACT) although more rarely in PAM. When sparse sampling is utilised, it is then necessary to implement an image recovery algorithm in order to obtain the desired PAM image (i.e. an image as close as possible to what would be obtained in conventional PAM). Therefore, the two key points to be considered in the sparse PAM problem are the choice of sampling scheme and the appropriate image recovery algorithm [8].

2.1 The Sampling Scheme

In order to acquire sparse photoacoustic data, many researchers have proposed using a mask with a random uniform distribution for the sparse scanning due to the fact that this enables perfect reconstruction by low rank matrix completion [7],[8] which is the most common way of reconstructing the desired photoacoustic image.

When using a uniform sampling scheme, a mask K is created from a null matrix of same size as the PAM image that would of been acquired with conventional scanning (i.e. of same size as the desired output of the reconstruction), from which we pick a number of elements uniformly at random that we then set to a value of 1. So with this mask, if $K_{i,j} = 1$ then we acquire the information at position (i, j) of the field of view otherwise no information is acquired.

However, uniform sampling schemes can tend to ignore significant parts of the PAM image and recently a universal sampling scheme has been provided for matrix completion by leveraging expander graphs [14]. This has led to a number of researchers turning to non-uniform sampling schemes in order to preserve the important photoacoustic information during compressive sampling.

In [8], both a uniform sampling scheme and a non-uniform one are compared. In order to obtain the uniform sampling, a random uniform graph expander is used as the mask and for the non-uniform sampling scheme, a higher sampling rate is used in the region of interest (ROI) of the sample than in the background in order to retain more relevant information while minimising the sampling time. Using the non-uniform sampling scheme produced results closer to the desired PAM image than the results obtained with uniform scanning. Another non-uniform method [10] also focuses more on the ROI than the background by assigning weights to the acquired data.

2.2 Matrix Recovery

After acquiring the sparse data, the next task to be addressed is the reconstruction of the desired high-resolution PAM image. The most common way to do this is by using the low rank matrix completion (LRMC) model.

The compressive data acquired can be regarded approximately as a subset of the data measured using conventional PAM (i.e. the image obtained by implementing a full point-to-point scan) and therefore the reconstruction of the desired PAM image can be viewed as the following matrix recovery problem:

$$D = Kx$$

where x is the matrix with size $m \times n$ of data acquired during conventional PAM, K is the mask of random sampling ($K_{ij} = 1$ if we acquire data at point (i, j) and $K_{ij} = 0$ if we don't, $1 \leq i \leq m$ and $1 \leq j \leq n$), and D is the matrix of compressive data. In this system, x is the variable we wish to recover.

Recht et al. [7] proved that most matrices x of shape $m \times n$ with low rank can be recovered from D as the result of a minimization problem (i.e. minimizing the nuclear norm of the matrix x that fits the compressive data) if the entries of K are suitably random (e.g. i.i.d Gaussian). Therefore, if the distribution of

elements in the sampling mask used is suitably random and uniform, recovering the desired image x amounts to a simple minimisation problem.

Most of the sparse PAM techniques regard the PAM image recovery as a matrix completion problem, utilising random uniform scanning in order to acquire the sparse data. Although this sampling scheme tends to be the most common, as mentioned previously, there are ways proposed to improve sampling efficiency with non-uniform techniques, which still enable image reconstruction via the matrix completion problem.

Solving the low rank matrix completion problem amounts to solving a minimisation problem: that of minimising the nuclear norm under certain constraints. This can be done most simply and most efficiently using the singular value thresholding algorithm proposed by Cai et al. in [16].

An alternative way to solve the matrix completion problem is by using the GoDec algorithm instead of minimising the nuclear norm. This was proposed because usually in order to solve the minimisation problem, the Singular Value Decomposition (SVD) of x must be performed which is time-consuming.

However, when using the GoDec algorithm, the image could not be well recovered with just the low-rank constraint, especially for the edges. In addition, the PAM vascular image is not perfectly low-rank. In this case, the image recovered when using only a low-rank constraint will exhibit poor edge preservation and artefacts will appear due to lack of prior information [10]. To remedy this, Liu et al. in [11] propose combining the Total Variation norm as a sparse constraint with the nuclear norm (for the low-rank constraint) and solving the resulting problem using the Alternating Direction Method of Multipliers (ADMM) framework. In addition to remedying the edge preservation problem, this method could even prove to be more efficient than using GoDec.

The ADMM framework has also been used in [9] to solve the matrix completion problem. It's used to solve an alternative non-convex minimisation problem obtained from the Lagrangian function corresponding to the aforementioned nuclear norm minimisation problem.

The following table summarises the methods mentioned:

Method	**Singular value thresholding algorithm**	**GoDec algorithm**	**TV & Nuclear norm + ADMM framework**	**ADMM for alternative minimisation problem**
Description	Used to solve the nuclear norm minimisation problem corresponding to the LRMC problem.	Used to solve the LRMC problem.	Used to solve the nuclear norm minimisation problem with Total Variation norm. The resulting problem is solved with ADMM.	Used to solve an alternative minimisation problem obtained from the Lagrangian function.
Advantages	Simple and efficient.	SVD of *x* not necessary.	- Better edge preservation. - Could prove to be more efficient than GoDec.	
Disadvantages	Necessitates the SVD of *x* which is time-consuming.	Poor edge preservation.		

3. Super-Resolution Using Deep Learning

3.1 Super-Resolution

The sparse sampling performed in PAM yields a lower resolution photoacoustic image that can be seen as the counterpart of the high resolution image that would typically be obtained through full resolution scanning of the sample. The aim of the reconstruction is to obtain the high-resolution from the single low-resolution input. Therefore this reconstruction is formulated as a single-image super-resolution (SISR) problem to be solved using deep learning.

SISR is a challenging problem as it is ill-posed because it is under-determined. Three types of SISR methods exist: interpolation-based methods (fast and simple but yielding low-quality results that over-smooth the details of the image), reconstruction-based methods in which the number of possible solutions to the problem is reduced with the introduction of some prior knowledge (generates the high-frequency details lacking in interpolation methods but are time-consuming and performs badly for large upsampling factors) and learning-based methods (computationally fast and good performance) [24]. It is this last category of methods that will be used in this work.

In learning-based methods, a large database of high-resolution images and their low-resolution counterparts (generally a downsampled version of the HR image with some blurring and noise added) is used. The network learns the statistical relationships between the two images and uses this knowledge to better predict high-resolution images from low-resolution ones.

In recent years, the success of using deep learning based methods for various data-driven tasks (particularly the use of CNNs) has been well documented and is continuously evolving, due in no small part to the increased capabilities and performance of modern-day computers. This is indeed the case also for image applications. In these image applications, the most widely used class of deep neural networks is the convolutional neural network (CNN), used to extract different feature maps of a given input image. The use of convolutional layers in the networks designed for SISR [17], enable the modeling of complex mappings between the low-resolution images and high-resolution ones while generating the high-frequency details lacking in interpolation-based methods and remedying the need for prior knowledge such as in the reconstruction-based methods.

In this work, two CNNs are utilised within a generative adversarial network (GAN) in order to improve accuracy. A GAN is a framework in which a generator network produces a prediction of the high-resolution input from the low-resolution one and a discriminator network aims to distinguish between this prediction and the high-resolution ground-truth. The network is trained so that the generator tries to trick the discriminator, thus pushing it to improve accuracy. This framework described above for SISR is the well-known SRGAN network [18] which, in addition to the GAN framework for improved accuracy, also uses residual blocks for improved performance and a perceptual loss for better perceptual quality of the output. In this thesis, a variation of this network for even better speeds is used wherein the residual blocks of SRGAN are replaced with bottleneck residual blocks that decrease the number of computations necessary in a single pass through the network. This modified architecture is called Fast-SRGAN [19].

3.2 SRGAN and Fast-SRGAN

3.2.1 SRGAN

SRGAN is a generative adversarial network (i.e. an architecture comprised of both a generator network and a discriminator network) in which the generator produces super-resolved images from low-resolution ones and the discriminator then aims to distinguish these super-resolved images from the high-resolution ground-truths in order to improve the accuracy of the network.

The input of the generator network goes through an initial stage comprising of a convolutional layer (with kernel size = 3; number of filters = 64; stride = 1) and a layer that applies the parametric rectified linear unit (PReLU) activation function before going through various residual blocks each with the following construction:

- convolutional layer (kernel = 9; number of filters = 64; stride = 1)
- Batch Normalisation
- Activation layer using PReLU function
- convolutional layer (kernel = 3; number of filters = 64; stride = 1)
- Batch Normalisation
- Element-wise Sum

The batch normalisation layers help to accelerate training times and decrease error by normalising the inputs and the element-wise sum is used to combine the residual blocks for the skip connections. The use of residual blocks with skip connections is to avoid accuracy saturation as very deep neural networks tend to decrease in accuracy as the number of layers increases. They do this by learning the residual of a layer as opposed to its true output:

$$R(x) = H(x) - x \tag{3.1}$$

where x is the input of a layer, $H(x)$ the true output of said layer and $R(x)$ is the residual [29].

After these residual blocks, there is a final layer bearing a similar construction to that of each residual block before the upsampling blocks that give the input the same dimensions as the desired high-resolution output. These upsampling blocks consist of a convolutional layer, a Pixel Shuffler layer which upsamples the image by a factor of 2 and a final PReLU activation layer.

The discriminator network is comprised of various blocks each bearing the same construction which is the following:

- Convolutional Layer
- Batch Normalisation
- Activation Layer using the Leaky ReLU activation function

After these blocks, two fully connected layers (each with their own activation layer following) are used for the classification of either high-resolution or super-resolution images.

The architecture of said model is the following:

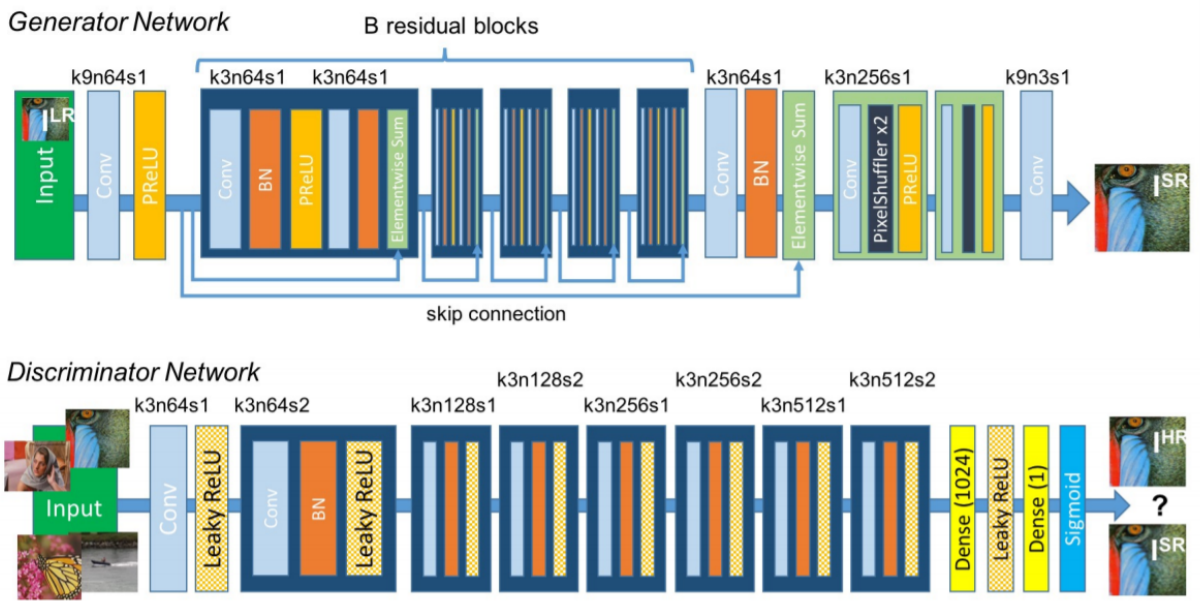


Figure 3.1: **The SRGAN architecture** (Source: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. [18] used with permission from the authors)

Coupled with this GAN architecture is the introduction of a perceptual loss that aims to increase the generated images' likeness to the high-resolution ground-truth ones. Most commonly used losses in deep-learning use a pixelwise loss such as the mean squared error (MSE). These kinds of losses are simple to implement but yield poor perceptual quality due to a lack of high-frequency content. SRGAN introduces a perceptual loss that combines a content loss (calculated from the features extracted using a separate neural network trained for classification, the VGG19 network) to take into account the high-frequency content and an adversarial loss which encourages the GAN to favour natural images by trying to trick the discriminator network.

Here are some of the results presented in the SRGAN paper:

Figure 3.2: **Results of SRGAN (upscaling factor of 4x):** Bicubic interpolation *(left)*, SRGAN *(middle)* and the original high-resoution image *(right)*. (Source: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. [18] used with permission from the authors)

The results obtained from SRGAN have a very high resemblance to the natural high-resolution images and it is the most latest step in the evolutionary chain of deep learning based super-resolution methods, therefore this was the architecture chosen for the super-resolution of the photoacoustic images.

3.2.2 Fast-SRGAN

Although SRGAN yields the some of the best results seen in super-resolution in recent years, our goal here is to make the image acquisition in PAM as fast as possible in order to enable real-time imaging and allow for this modality to be used in clinical contexts as well as in research. Therefore, a modified version of SRGAN, dubbed Fast-SRGAN [19], was chosen in order to reach a compromise between image quality and speed.

Fast-SRGAN is an implementation of SRGAN in which, instead of the standard residual blocks of SRGAN, bottle neck residual blocks that utilise depthwise separable convolutions are used. This is done in order to minimise the number of mathematical computations necessary and subsequently decrease computation time. This idea was inspired by the *Towards Real-Time image enhancement GANs* paper [20], which is itself drawing inspiration from Google's MobileNet [21], an architecture that aims to have the highest possible accuracy while keeping the number of computations as low as possible in order to be utilised on mobile devices.

The number of mathematical computations necessary for these inverted residual blocks is reduced compared to the usual residual blocks thanks to the depthwise separable convolutions. The way this works is the following: In a normal convolution for these kinds of image applications, we take a three-channel (i.e. RGB) image and apply a certain number of filters (for example 64). Without padding, this operation gives an output of 64 feature maps of smaller height and width than the input (e.g. a 32x32x3 input image using a kernel of size 5x5 with a stride of 1 would give and output of shape 28x28x64). In depthwise separable convolutions, we separate the operation into two: a depthwise convolution (we perform the convolution on the input without changing the depth e.g. with 3 kernels of shape 5x5x1 giving an output of shape 28x28x3) and a pointwise convolution (using 64 kernels of shape 1x1x3 that go through every point of the input, this process gives an output of shape 28x28x64).

In order to illustrate how this reduces the number of computations, intuition can be drawn from spatial separable convolutions with the following example:

$$\begin{bmatrix} 4 & 8 & 12 \\ 5 & 10 & 15 \\ 6 & 12 & 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

A convolution performed with the kernel on the left involves nine multiplications whereas two convolutions performed with the vectors on the right, the first convolution with the column vector and the second with the row vector, yields the same result with a total of six multiplications (three multiplications per convolution). Depthwise separable convolutions reduce the number of necessary mathematical computations in a similar way. Whereas spatial separation isn't possible for all convolutional kernels, depthwise separation is and improved speeds are achieved thanks to these separations.

3.2.3 Loss functions

One of the factors that makes SRGAN produce results so close to the ground-truth images is its perceptual loss, inspired by [26], which is comprised of a content loss, a discriminator loss and an adversarial loss. The Fast-SRGAN network chosen for the super-resolution in PAM also uses this perceptual loss.

Typically in training deep neural networks for image applications, pixel-wise losses such as the mean squared error (MSE) are used in order to check the effectiveness of the network with respect to some ground-truth. In single-image super-resolution, the output of the network is the super-resolved image and the ground truth would be the high-resolution image we introduce the network to and we would use the MSE, for example, to assess how close the super-resolved image was to the high-resolution one. The goal then for the network is to learn how to minimise this loss in order to achieve results as close as possible to the ground-truths.

The problem with the pixel-wise losses such as MSE is that, as they measure pixel-to-pixel, they often neglect to account for such things as the global structure of the image or its finer details. In SRGAN, these pixel-wise losses are replaced with the perceptual loss mentioned previously. This perceptual loss is the sum of three other losses: a content loss, an adversarial loss and a discriminator loss.

The content loss is what is used to remedy the the pixel-to-pixel comparison usually used. It does this by utilising a separate, pre-trained network, the well-known VGG network. The network implemented here uses the VGG19 network (19 for 19 layers as VGG16 has 16 and VGG22 has 22). VGG19 is a very deep convolutional neural network originally designed for image recognition [25]. In the context of SRGAN, it is used to extract the features of both the super-resolved output and the high-resolution ground truth and then returns the MSE of these feature maps, thus retaining important structural information that would otherwise be ignored when computing the regular MSE on these two images. The VGG19 network used for this is directly imported, pre-trained from Keras so no extra training was required for the computation of this loss.

In SRGAN (and subsequently also Fast-SRGAN), accuracy is improved through a minimax problem between the generator and the discriminator. The discriminator network D is trained to maximise the probability of assigning the correct label to both the high-resolution training examples and the samples produced by the generator network G and simultaneously the generator's goal is to minimise the error between generated samples and the ground-truths with cross-entropy. This minimax problem is the following [34]:

$$\min_{G} \max_{D} \mathbb{E}\big[\log D(y)\big] + \mathbb{E}\big[\log\big(1 - D(G(x))\big)\big] \tag{3.2}$$

where x is the low resolution input of the network, y is it's high-resolution counterpart. This minimax problem gives us the following optimistion problem for discriminator loss:

$$\max \frac{1}{N}\sum_{i=1}^{N}\Big(\log D(y^{(i)}) + \log\big(1 - D(G(x^{(i)}))\big)\Big) \tag{3.3}$$

And the following problem for the generator loss:

$$\min \frac{1}{N}\sum_{i=1}^{N}\log\Big(1 - D\big(G\big(x^{(i)}\big)\big)\Big) \tag{3.4}$$

where N is the number of samples in the training batch.

A problem that arises when training a GAN is that in the early stages of training, the discriminator can distinguish the fake super-resolved images created by the generator from the training samples with high confidence and thus causing $\log\big(1 - D\big(G(x)\big)\big)$ to saturate. To remedy this, $-\log\big[D\big(G(x)\big)\big]$ is minimised instead

[34] to obtain the following adversarial loss described in [18] equal to $10^{-3}\ell_{\text{adversarial}}$ where

$$\ell_{\text{adversarial}} = \frac{1}{N}\sum_{i=1}^{N} -\log D\big(G(x)\big) \tag{3.5}$$

The overall perceptual loss for Fast-SRGAN is the sum of the aforementioned content and adversarial losses and the MSE between the high-resolution ground-truth and the super-resolved image produced by the generator:

$$\ell_{\text{perceptual}} = \ell_{\text{content}} + \ell_{\text{MSE}} + 10^{-3}\ell_{\text{adversarial}} \tag{3.6}$$

The optimisation problem used to compute the discriminator loss involves a gradient ascent so it is more commonly implemented as a traditional binary classification problem which is the case in Fast-SRGAN.

3.3 Modifications for the PAM Context

Only very minor modifications were made to the Fast-SRGAN source code, if any at all. The main modification was made to upsampling layer in this network. In the original code, the upsampling is applied to both the height and the width of the low-resolution image. In our application, the speed and resolution in the y-axis is fixed and thus sparse sampling is not performed in this direction. As a result, the low-resolution obtained with this sparse acquisition is one-dimensional and we only need to super-resolve in this dimension. Fast-SRGAN, in its training phase, takes as input the high-resolution training dataset and creates the low-resolution counterparts in the dataloader.py module. This was modified to create a low-resolution dataset in which the images were only downsampled along the x-axis.

This one-dimensional super-resolution has the advantageous effects of both retaining more information than the original 2D Fast-SRGAN, thus producing more accurate results, and of reducing some of computations performed in the network and thus speeding up the process.

The source code was also slightly modified in order to be integrated into the LabVIEW software which will be presented in detail (along with its Python integration) in the following chapter. The super-resolution network used was trained to perform on three-channel RGB images whereas the data presented in the LabVIEW front panel is a one-channel intensity graph of the maximum amplitude projection (MAP) of the 3D photoacoustic information along the depth axis. Therefore, in order to be able to use the network on this one-channel intensity graph, it was copied to each channel of a RGB input image (this is the most simple and commonly used way of using networks trained for RGB images on grayscale ones). Once the network has been applied to this artificial three-channel image, only one channel of the super-resolved output is then passed back into the LabVIEW code and presented to the end-user on the front panel as the new MAP in the intensity graph.

4. The Data Acquisition Software

4.1 A presentation of the LabVIEW programming language

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a development environment for a graphical programming language and software for engineering applications that require communication with hardware for a range tasks such as testing, measurements and controls [37]. Typical LabVIEW code, called a VI, comprises two parts: the front panel which is in essence the graphical interface that the end-user interacts with and the block diagram which is the graphical source code. The dataflow programming paradigm is used in LabVIEW so each part of the source code is executed when data becomes available to it. This means that, in a simple block diagram with no loops or sequences, execution of the various parts of the code happens in parallel as a standard and the code must be modified if sequential execution is required.

Data is propagated through the block diagram using wires and operations are performed using terminals, which are blocks assigned a certain function e.g. in the example pictured below, the "Multiply" terminal takes as input integer values from two wires and computes their product. The output of this terminal can then be propagated further through the code.

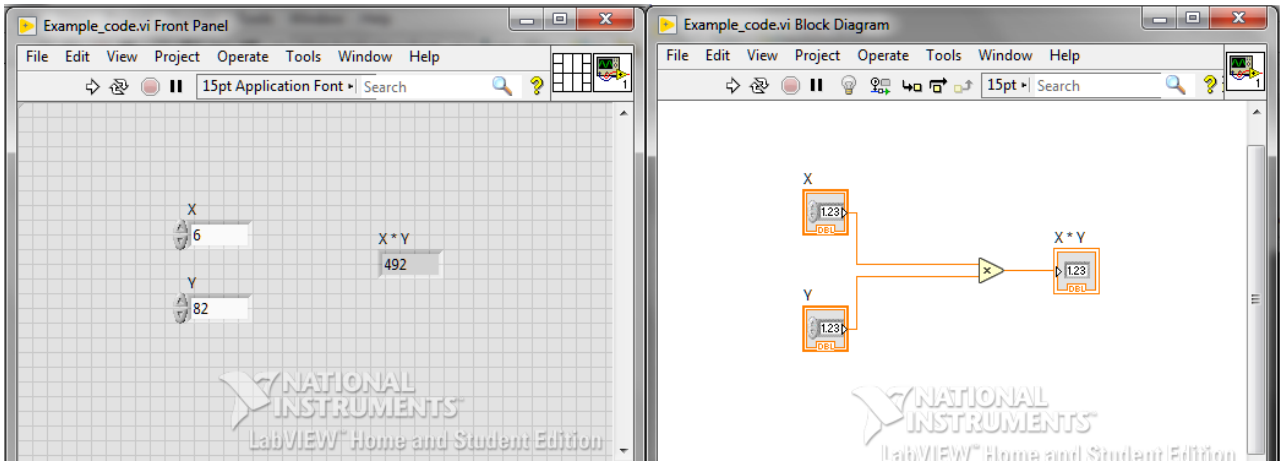


Figure 4.1: An example of LabVIEW code with its Front Panel *(left)* and its Block Diagram *(right)*.

There are two types of front panel elements in LabVIEW, controls which take data inputs from the user and indicators which indicate to the user information from the block diagram terminals. In the example pictured above, the "X" and "Y" boxes are controls and the "X*Y" box is an indicator.

4.2 The Graphical User Interface

In the original LabVIEW code, there were two main processes. The first prepares for and implements the acquisition and also saves the measurement data if the user chooses to do so before acquisition into the appropriate file format (the TDMS file format being the recommended one). The second communicates with the Field Programmable Gate Array (FPGA) which is the hardware custom programmed for the signal generation. The changes implemented do not affect these original processes, only modify the structure of the program to enable the front end of the program to be clearer for the user and add some additional functionalities to improve ease of use.

Regarding the structure of the program, a simple "if" loop was put in place to remove ambiguity about which button to press in order to launch acquisition. Indeed, if the "Run Acquisition" button is pressed, the acquisition code is launched as originally the acquisition was launched as soon as the code started running. Now, once the LabVIEW code starts running, the program is in an idle state until the user chooses to commence acquisition.

A tab system was put in place to separate the program's configuration from the actual acquisition phase. All the parameters necessary for the signal generation are set in the configuration tab and functionalities that enable the user to save and load their own personal configuration as well as reset to the default configuration were added. The two buttons, one that initialises the sine wave generation and the other that begins the signal generation, are from the original code and need to be pressed before acquisition in order to activate the sine wave generation from the FPGA target. This sine wave generation is for the voice coil scanning along the x-axis of the PAM setup and thus the number of sine waves generated equals the number of y-axis scan lines.

As for the acquisition tab, a number of functionalities have also been added. Previously, in order to save the 3D photoacoustic data, the user had to choose to save this before running the acquisition and although this is still the case, there is now an option to save the maximum amplitude projection (MAP) photoacoustic image that appears on screen after acquisition as a snapshot. The button that enables this, when pressed, will open the file explorer so that the user can choose the name and location of this image. Options to zoom in or out and drag the view have also been added for the MAP image.

The PAM setup allows for acquisition with either one or two lasers of wavelengths 532nm and 561nm. The 532nm laser is activated directly by the LabVIEW code, whereas the other necessitates activation via a terminal emulator, the one used here is Termite. Previously, the user had to open Termite before acquisition and enter the commands to activate this laser before running the LabVIEW code (and subsequently the second 561nm laser). Now there is a switch which allows the user to choose to use either one or two lasers. When the switch is set for acquisition with both the 532nm laser and the 561nm laser, the light turns green to indicate this state and when the user runs the acquisition, Termite immediately opens (pausing acquisition with a message displayed to the user) and the user can then enter the necessary command to quickly activate the second laser before clicking "continue" in the message box which will relaunch the acquisition. This effect was achieved by implementing a flat sequence in the LabVIEW block diagram.

Below are the two tabs present in the current PAM user interface.

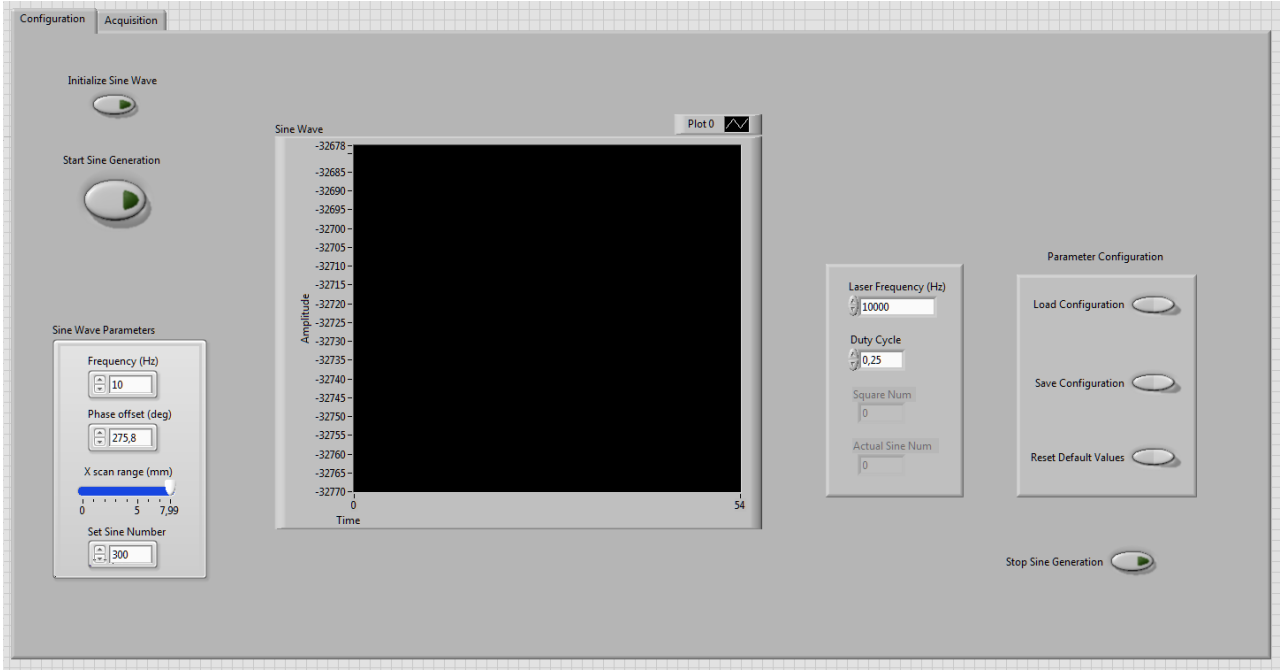


Figure 4.2: The Configuration tab

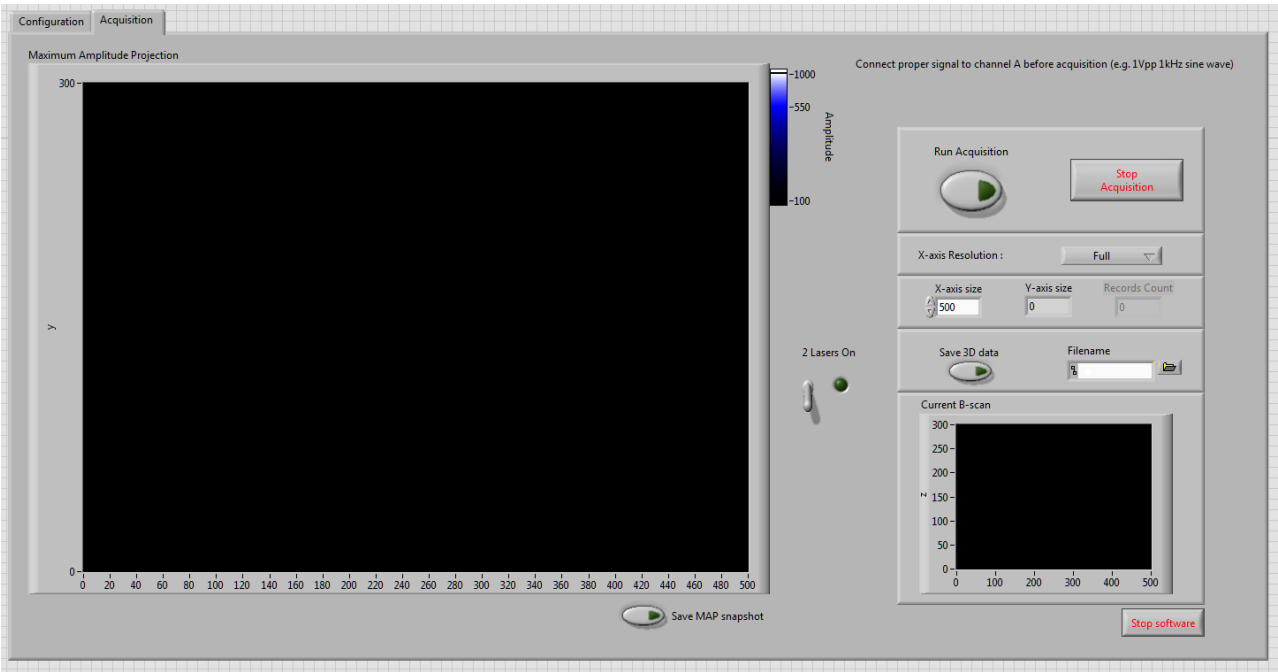


Figure 4.3: The Acquisition tab

4.3 Integrating LabVIEW and Python

In order to use the Fast-SRGAN [19] Python code in the PAM acquisition process, it must be integrated into LabVIEW. Communication between LabVIEW and Python code is mostly simply done using the Python Node in LabVIEW.

Integrating these two languages is not in itself a complicated task, however certain factors had to be taken into account for everything to work correctly. First of all, in order for LabVIEW and Python to work together, they must both have the same bitness. Moreover, the Tensorflow module requires the Python version to be 64-bit therefore the LabVIEW installation must also be 64-bit. The Python node for LabVIEW is only available in LabVIEW 2018 or newer and is only compatible with native Python versions 2.7 or 3.6 (any Python version installed via Anancoda for example won't work with LabVIEW [22]). LabVIEW also requires a specific set of installation options for native Python which are detailed in [23]. The following instructions must be followed to ensure compatibility with LabVIEW when installing Python:

1. Download Python 3.6 only from Python.org (any other python installation won't work e.g. via pip or conda)
2. Make sure it's 64-bit (also make sure that the LabVIEW version used has same bitness) otherwise it won't work with Tensorflow
3. Download for example version 3.6.8 and make sure to pick option: "Windows x86-64 executable installer"
4. Open Python setup and click box for option "Add Python 3.6 to PATH" then choose "Customize installation"
5. Choose "Optional Features" as desired, there are no specific requirements here
6. In "Advanced Options", click box for "Install for all users" and modify the installation location to: "C:/Python3.6" ← this exactly otherwise LabVIEW can't find the python version properly (e.g. make sure it's "Python3.6" in path and not "Python36")
7. Install and after installation run test code "test_install.py" to verify correct installation

LabVIEW requires a specific license to work on any Linux OS therefore the LabVIEW and Python integration necessitates using Windows. The integration detailed here was tested on both Windows 7 and Windows 10. The following Python packages must also be installed in order to run Fast-SRGAN:

- Tensorflow version 2.0.1
- Keras version 2.3.1
- Open-cv2 (version 4.2.0 used here)
- Numpy (version 1.18.3 used here)

Once this installation is completed and all other compatibility requirements are met, Fast-SRGAN's 'infer.py' code (which is used to predict the super-resolution images from the low-resolution ones once the network has been trained) was slightly modified for LabVIEW purposes.

The LabVIEW vi written for this task first opens a Python session before using the 'Build Path' function that takes as input the 'infer.py' Python file. This 'infer.py' file as well as the necessary argument parameters (the ones usually written in the command line under the Linux OS) for prediction are inputs to the Python Node. The super-resolution prediction outputs can then be shown on screen using the usual LabVIEW terminals for such a task.

5. Computing Clusters and GPU Resources

In order to train Fast-SRGAN, a need for GPU resources was acutely present. Indeed, in order to obtain the results presented in Chapter 6, the network had to be trained for 10,000 epochs with a batch size of 14 whereas when training on the CPU, training for 100 epochs lasted five days and five hours and any batch size over 8 (for the x4 upsampling network) resulted in out of memory (OOM) problems. Therefore a smaller batch size would be required at least for the x2 network as the images that are used in this case are larger and occupy more memory space and longer training times would also result from the greater image sizes.

The GPU resources used were provided by CSC Finland's Puhti supercomputer and all training of the final networks was performed on Puhti's GPU partition.

5.1 Computing clusters and parallel computing

Puhti is a supercomputer made up of inter-connected computing clusters or nodes for computationally expensive tasks. Generally, the architecture of such a computing cluster is the following: one or more "head-nodes", i.e. login nodes which are accessed with the Secure Shell (SSH) protocol from a personal computer, and a number of computing nodes that are reserved for particular tasks like for more or less computationally expensive tasks, tasks that require a certain number of cores to be performed or tasks that necessitate manipulation of larger amounts of data [30].

Storage on computing clusters can also be broken up into smaller parts. Although, this partition varies from system to system, some examples are the following: Home directory storage (only for small amounts of storage), project storage (storage attached to specific projects with specific storage volume and file count allotment), scratch spaces (the bulk of the user's storage in these volatile spaces and they are frequently cleaned e.g. on the CSC computers they are cleared after 90 days).

All storage needed for Fast-SRGAN was done in the scratch space allocated for this project except when it came to manipulating the training dataset. For lots of small files to be read, the fast local drive present in every GPU node needs to be used instead of the scratch storage [31]. Therefore the compressed DIV2K dataset was extracted to this local storage at the start of the batch job for the training task.

5.2 Slurm and batch jobs

The queue system for jobs on the CSC computing clusters is operated by the open source cluster management and job scheduling system Slurm [32]. Slurm allocates resources and provides a framework for all stages of running jobs using said resources. It consists of a daemon (a daemon is a computer process that runs when certain conditions are met instead of being invoked by a user [33]) running on each node and a central daemon. The Slurm daemons manage: nodes, partitions, jobs, or allocations of resources assigned to a user for a specific amount of time, and job steps (sets of tasks within a job).

The queues managed by Slurm are set up for different tasks and configurations (e.g. for 1 GPU per node) and it is up to the user to establish what set up is necessary for their task. Jobs are submitted via submission scripts. A scheduler sends jobs to compute nodes by finding the optimal schedule for computation (maximise the use of the computer while maintaining equitable access).

There are two ways of operating a computing cluster: Shells (command line interface e.g Bash) or Scripts (script.sh file that are written with UNIX commands). To navigate the various nodes and partitions and to perform simple tasks like importing files and modifying them the Shells are used but for the main jobs themselves scripts must be written and submitted to the queue.

In the submission script, various elements of information on the job must be indicated such as the time needed to perform all job steps, the partition desired, the number of CPUs per task and the memory required. In the submission scripts for Fast-SRGAN, three job steps were performed: load the required module for Tensorflow on Puhti, extract the dataset to the appropriate local scratch space on the GPU partition and then run the Python code for training.

Training for 10,000 epochs for each upscaling factor was done with a learning rate equal to 1e-4 and a batch size of 14 and training lasted approximately 48 hours (or a little over) for each upscaling factor (factors x2, x4 and x8).

6. Results

The Fast-SRGAN architecture has shown relatively good results and the use of these computationally optimised residual blocks has resulted in both faster training and faster passes through the trained network: On the DIV2K validation dataset of low-resolution images (108 images of shape 510x339 mainly), the total time taken to produce the super-resolved counterparts, with an upscaling factor of x4, was around 342 seconds, i.e. roughly 3 seconds per image.

The bicubic interpolation was obtained using the resize function from Python's Open-cv2 library on the same downsampled image as the one used for Fast-SRGAN.

Visually, the Fast-SRGAN result is far less pixelated than the result obtained via bicubic interpolation. To further assess the effectiveness of this network, two metrics were computed the structural similarity index and the peak-signal-to-noise-ratio with respect to the high-resolution ground truth.

Metric	Fast-SRGAN	Bicubic Interpolation
SSIM	0.44908652	0.5815471
PSNR	13.500970087990769	14.926127836336818

The bicubic interpolation outperformed Fast-SRGAN for both the PSNR and the SSIM with regards to the ground truth. However, these metrics (particularly the PSNR) are not accurate measurements of the perceptual quality of an image. This has previously been illustrated in [18] when SRResNet outperformed SRGAN for both of these metrics when clearly the perceptual quality of the SRGAN results was higher. Indeed, this was backed up by a higher mean opinion score (MOS) for all results obtained with SRGAN than those obtained with SRResNet.

A comparison of the results obtained with both Fast-SRGAN and bicubic interpolation with respect to the high-resolution ground-truth is visible hereafter:

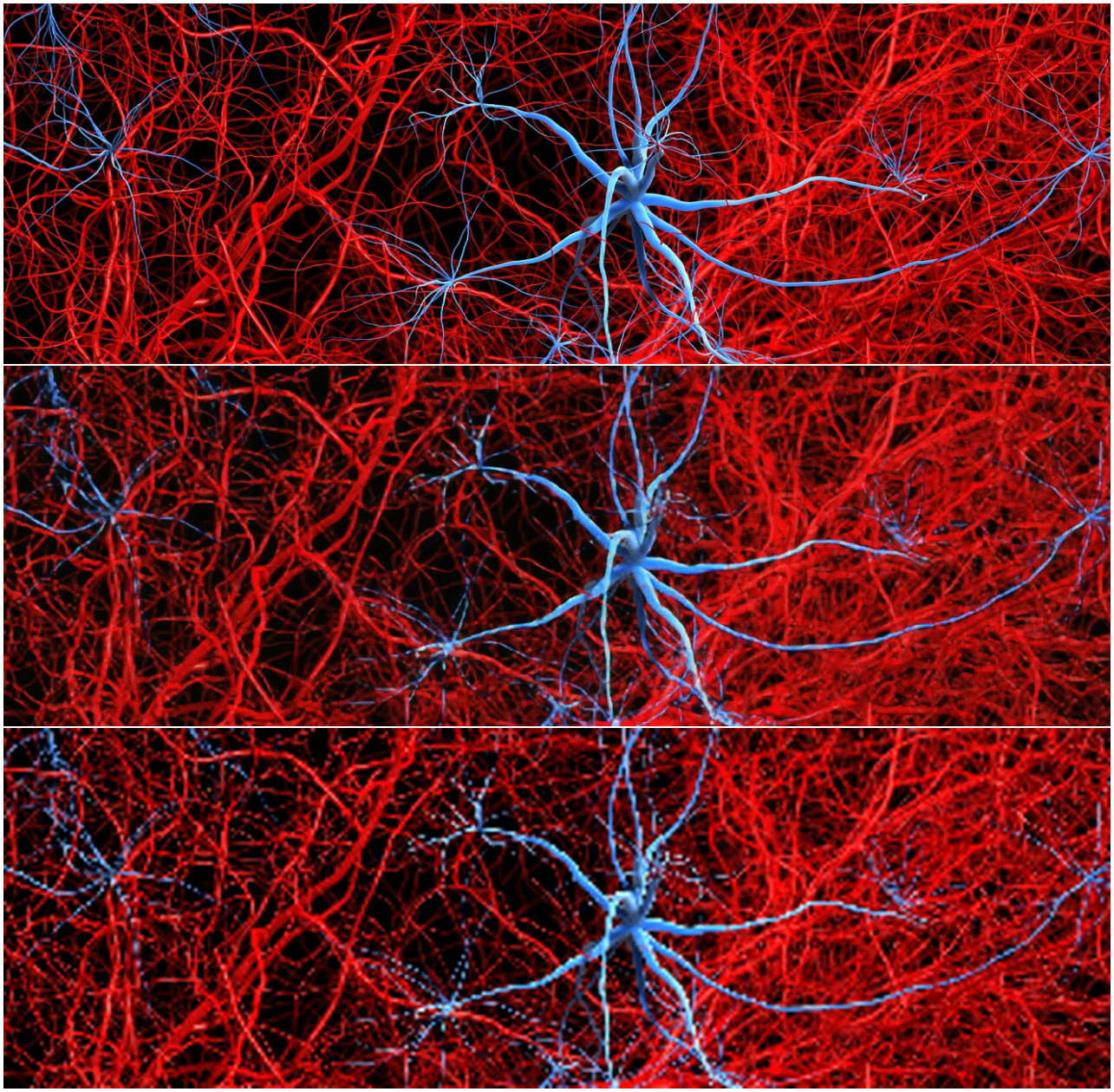


Figure 6.1: **2D Comparison with Upsampling Factor 4:** High-Resolution *(top)* 2D Super-Resolution *(middle)* & Bicubic Interpolation *(bottom)*.

As mentioned in Section 3.3, the one-dimensional super-resolution has the advantageous effect of reducing some of computations performed in the network and thus speeding up the process. As a result, the time taken for an image to make a single pass through the network was reduced: when testing the newly trained network for one-dimensional super-resolution on the DIV2K validation dataset, it took around 3min26sec to super-resolve 100 images with an upscaling factor of 4x along the x-axis (the low-resolution images had a size of around 510x339) i.e. approximately 2 seconds per image compared to 3 seconds for the two-dimensional super-resolution.

These processing times (3 seconds for 2D processing and 2 seconds for 1D processing) were obtained when computing on the CPU. With a CUDA-enabled GPU, these processing times would be significantly reduced. For example, the following results, presented in [19], were obtained using an Nvidia GTX1080 GPU:

Input Image Size	Output Size	Time (s)
128x128	512x512	0.019
256x256	1024x1024	0.034
384x384	1536x1536	0.068

The 1D super-resolution produced results that were perceptually closer to the high-resolution ground truth image, which was to be expected as more information was retained along the y-axis.

The same metrics as in the previous method (SSIM and PSNR) were computed again for this 1D super-resolution, in the same manner as previously:

Metrics	Fast-SRGAN	Bicubic Interpolation
SSIM	0.64759177	0.7348551
PSNR	17.0141782201672	16.942355295152922

Comparison of the 1D and 2D methods:

Metrics	1D Fast-SRGAN	2D Fast-SRGAN
SSIM	0.64759177	0.44908652
PSNR	17.0141782201672	13.500970087990769

Although bicubic interpolation still outperformed the 1D Fast-SRGAN for the SSIM with regards to the ground-truth image, 1D Fast-SRGAN outperformed it for the PSNR and it also outperformed 2D Fast-SRGAN for both these metrics. This improvement in performance was to be expected due again to the retention of more of the original information of the high-resolution ground-truth in the one-dimensional method.

The following is the comparison of both the 1D and 2D fast-SRGAN results on the example PAM-like image shown above.

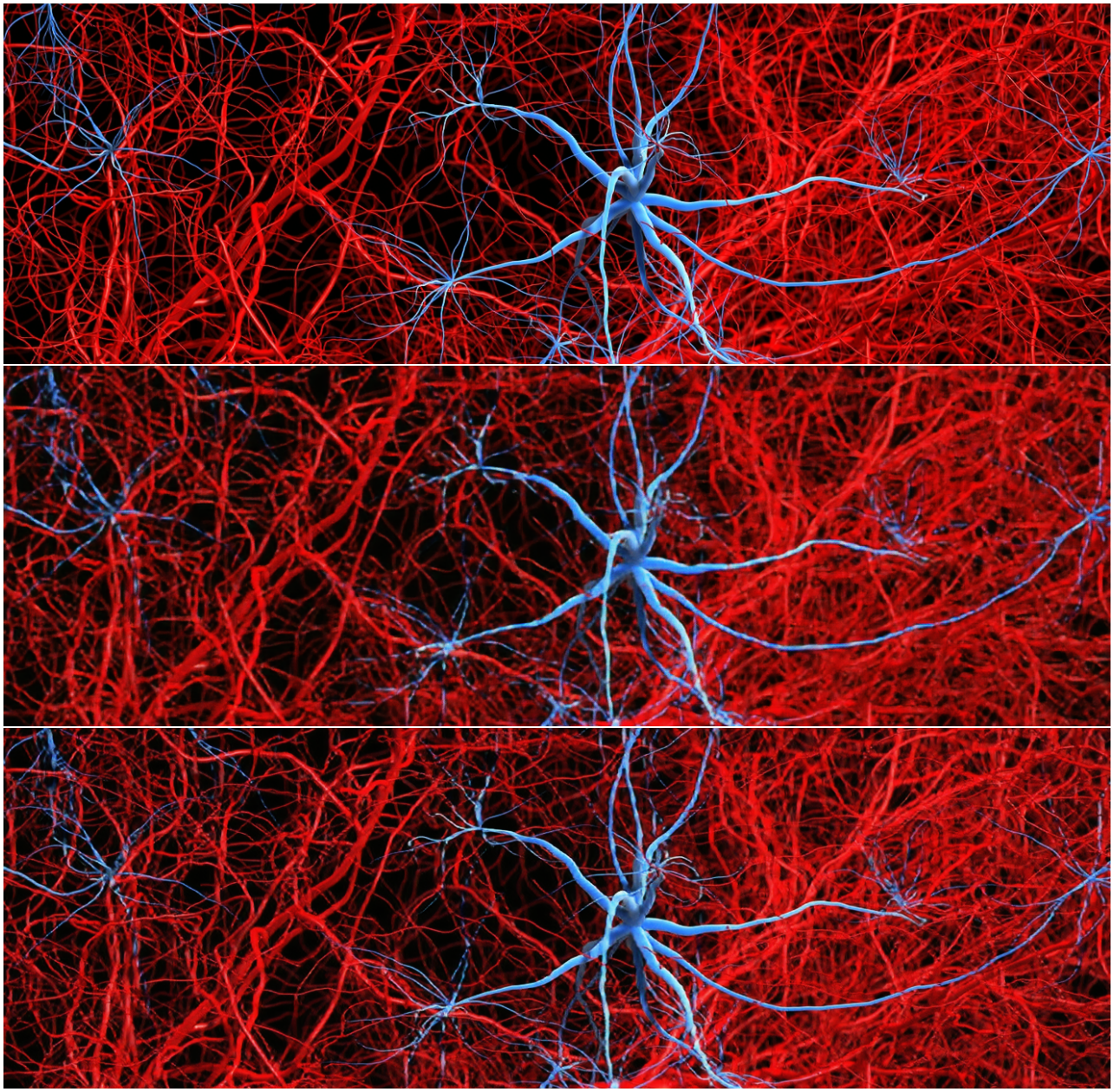


Figure 6.2: **2D and 1D Comparison with Upsampling Factor 4:** High-Resolution *(top)* 2D Super-Resolution *(middle)* & 1D Super-Resolution *(bottom)*.

All of the following results were obtained using the one-dimensional Fast-SRGAN.

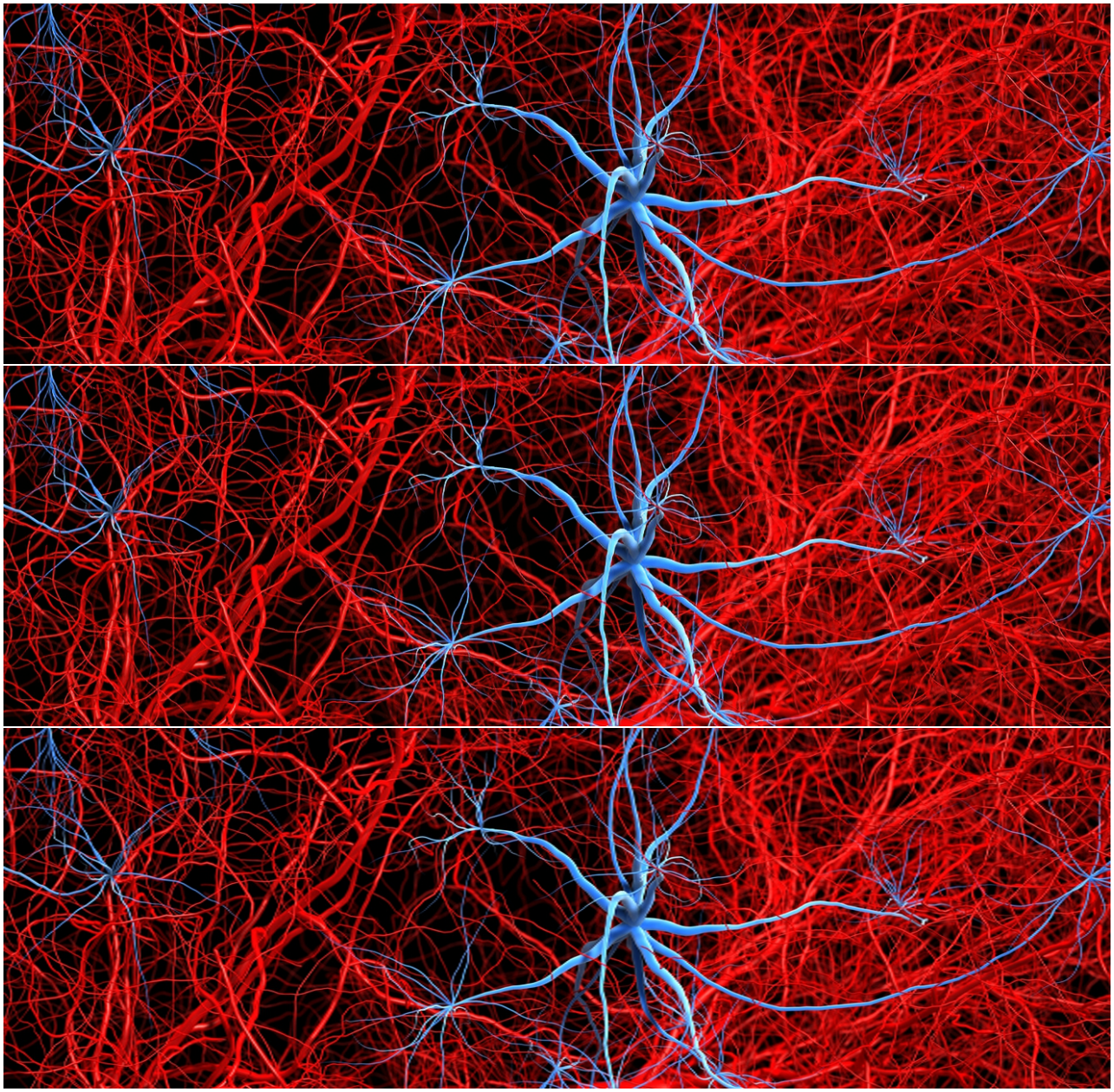


Figure 6.3: **Upsampling Factor 2:** High-Resolution *(top)*, Super-Resolution *(middle)* & Bicubic Interpolation *(bottom)*.

The differences between bicubic interpolation and the result obtained via super-resolution for this scale factor are only really perceivable when enlarging these images.

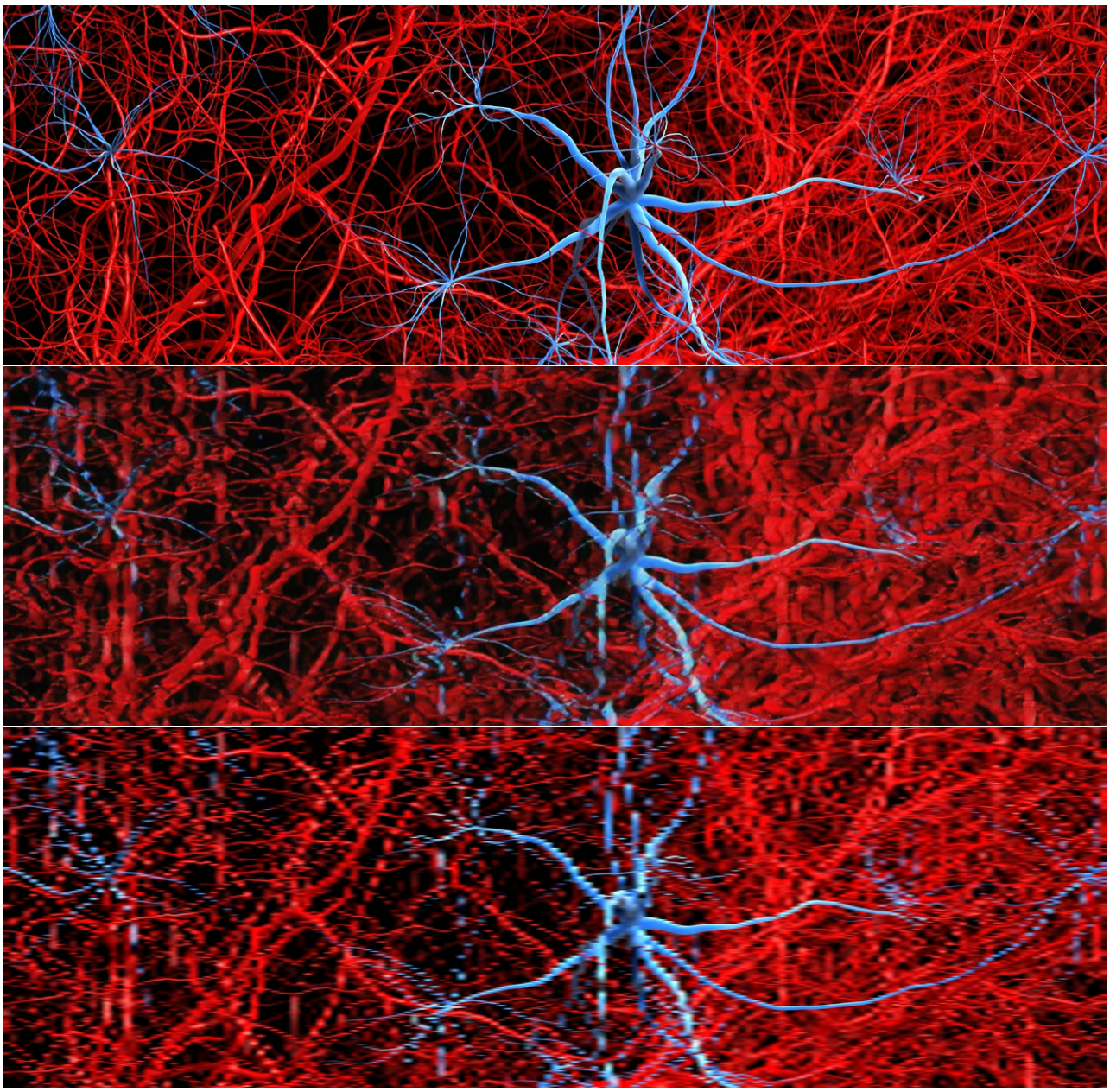
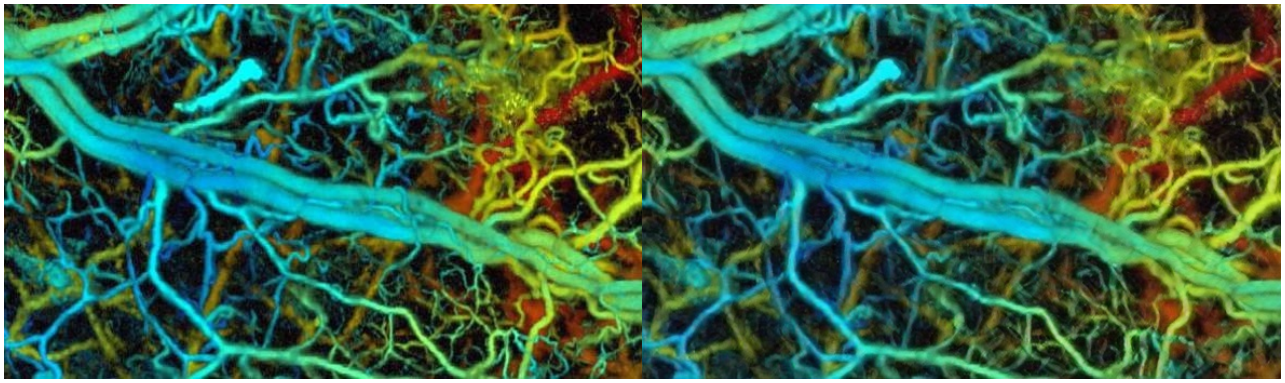


Figure 6.4: **Upsampling Factor 8:** High-Resolution *(top)*, Super-Resolution *(middle)* & Bicubic Interpolation *(bottom)*.

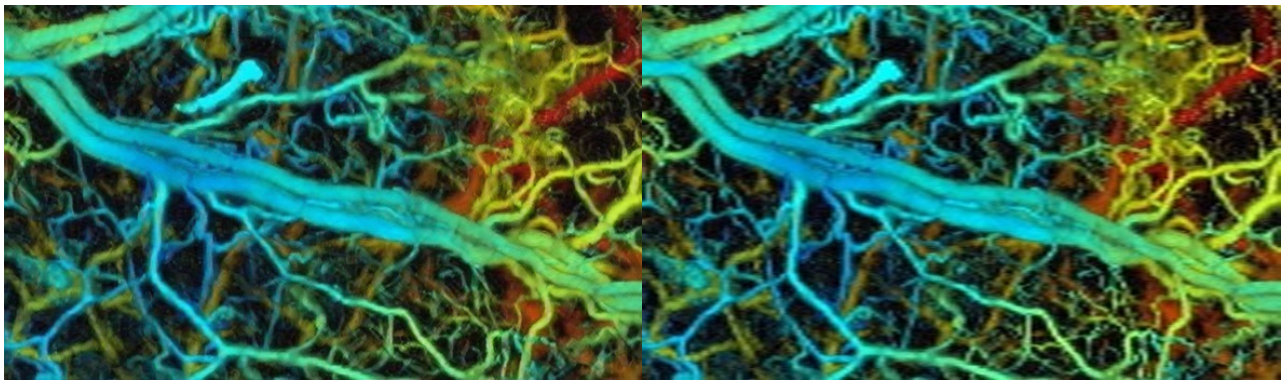
6.0.1 Upsampling factor x4

Additional results obtained on different PAM like images of blood vessels when super-resolving by a scale factor (SF) of 4x.



(a) High-Resolution (b) Super-Resolved

Figure 6.5: High-Resolution v Super-Resolution (SF = 4).

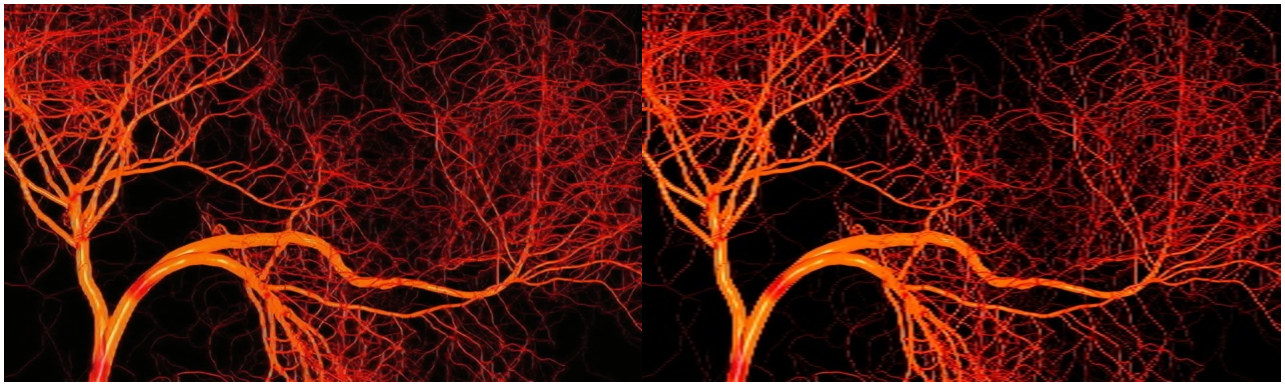


(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.6: Super-Resolution v Bicubic Interpolation (SF = 4).

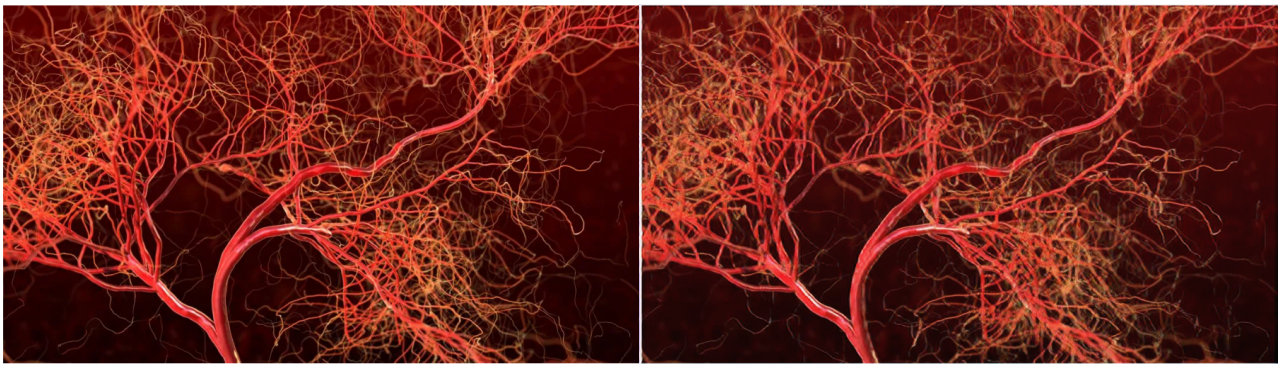
(a) High-Resolution (b) Super-Resolved

Figure 6.7: High-Resolution v Super-Resolution (SF = 4).



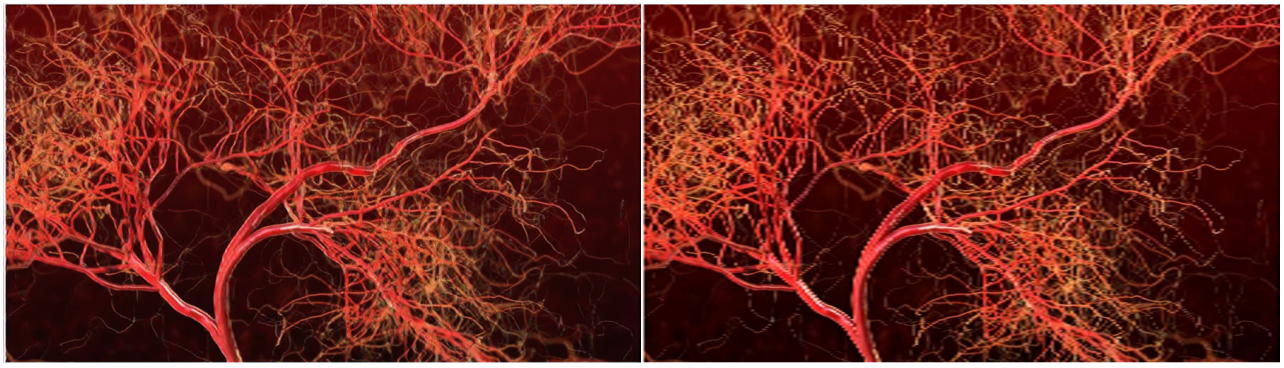
(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.8: Super-Resolution v Bicubic Interpolation (SF = 4).



(a) High-Resolution (b) Super-Resolved

Figure 6.9: High-Resolution v Super-Resolution (SF = 4).

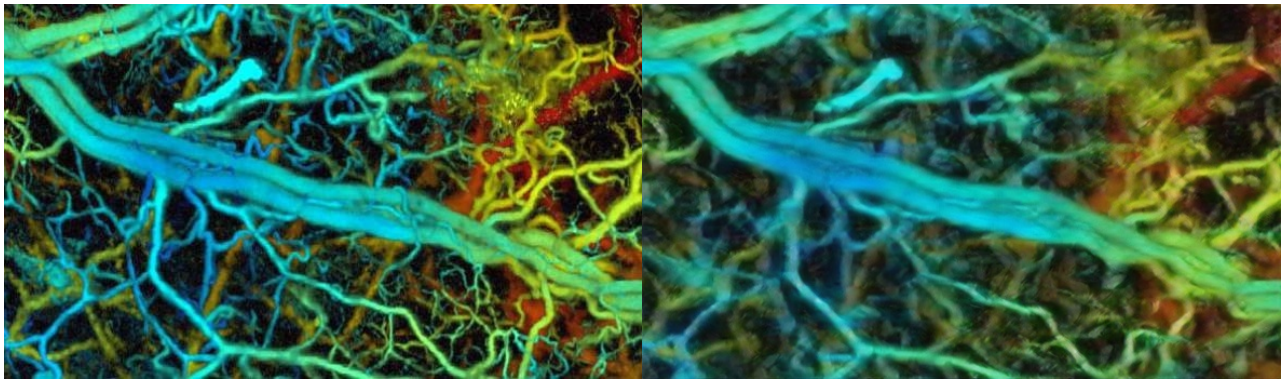


(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.10: Super-Resolution v Bicubic Interpolation (SF = 4).

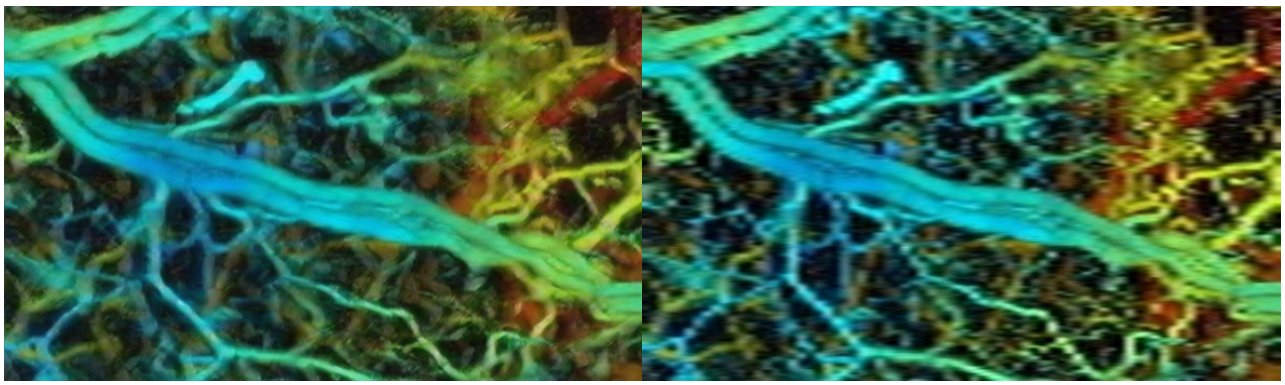
6.0.2 Upsampling factor x8

Additional results obtained on different PAM like images of blood vessels when super-resolving by 8x.



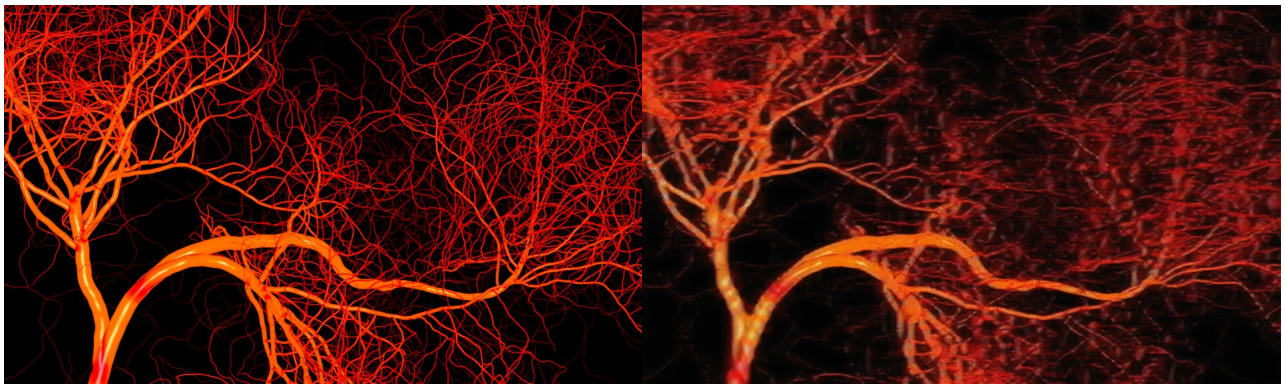
(a) High-Resolution (b) Super-Resolved

Figure 6.11: High-Resolution v Super-Resolved (SF = 8).



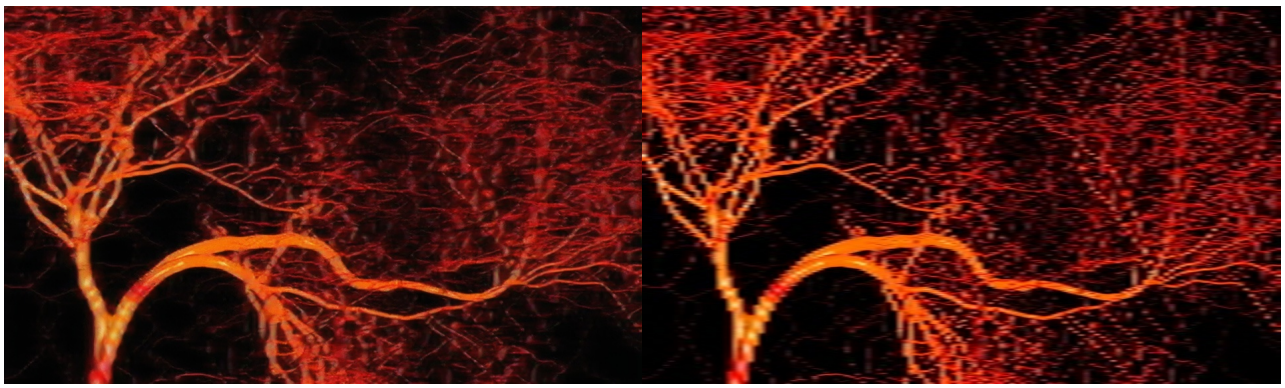
(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.12: Super-Resolution v Bicubic Interpolation (SF = 8).



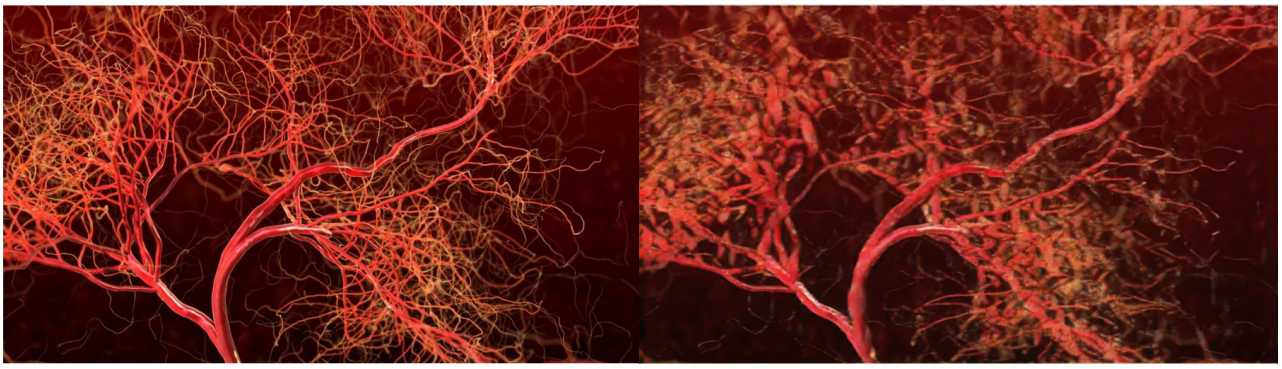
(a) High-Resolution (b) Super-Resolved

Figure 6.13: High-Resolution v Super-Resolution (SF = 8).



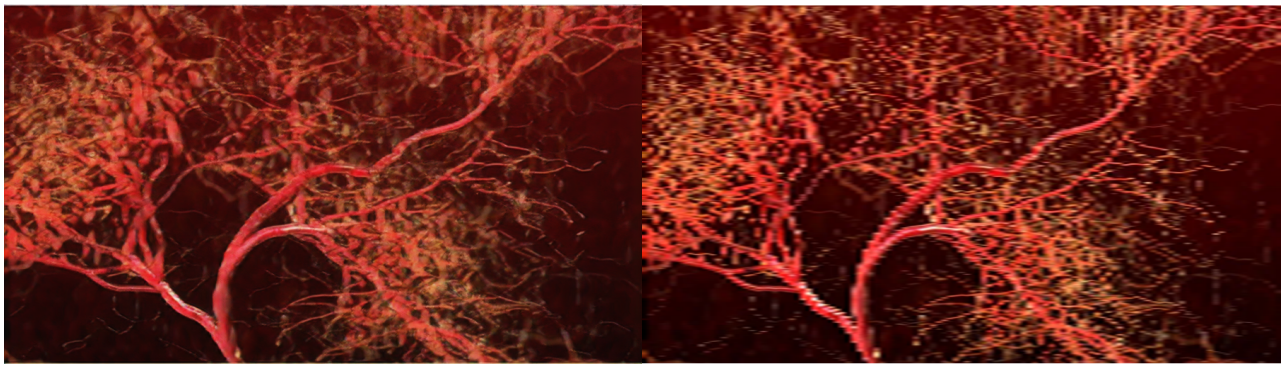
(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.14: Super-Resolution v Bicubic Interpolation (SF = 8).



(a) High-Resolution (b) Super-Resolved

Figure 6.15: High-Resolution v Super-Resolution (SF = 8).

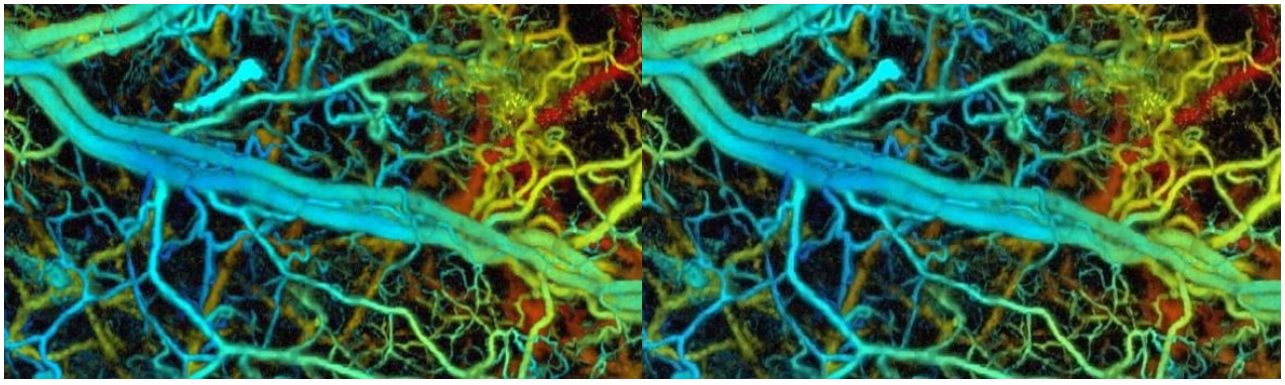


(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.16: Super-Resolution v Bicubic Interpolation (SF = 8).

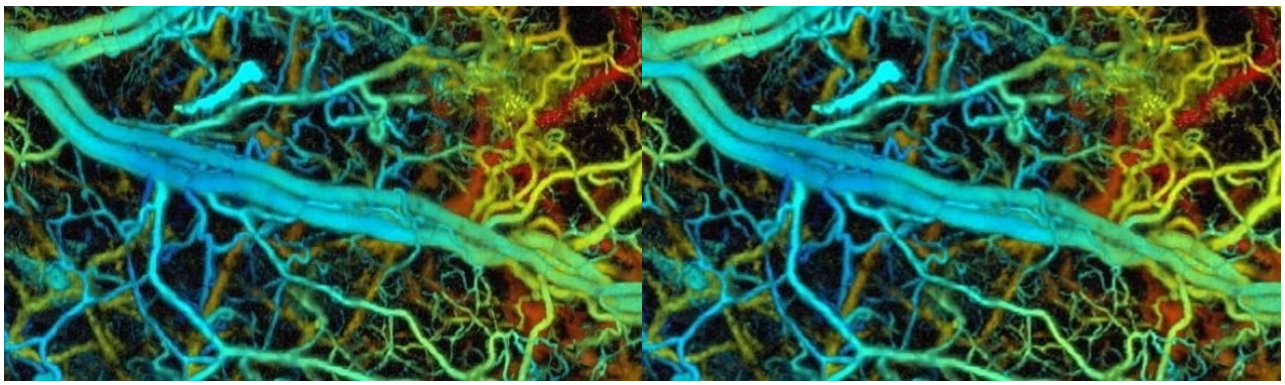
6.0.3 Upsampling factor x2

Additional results obtained on different PAM like images of blood vessels when super-resolving by 2x.



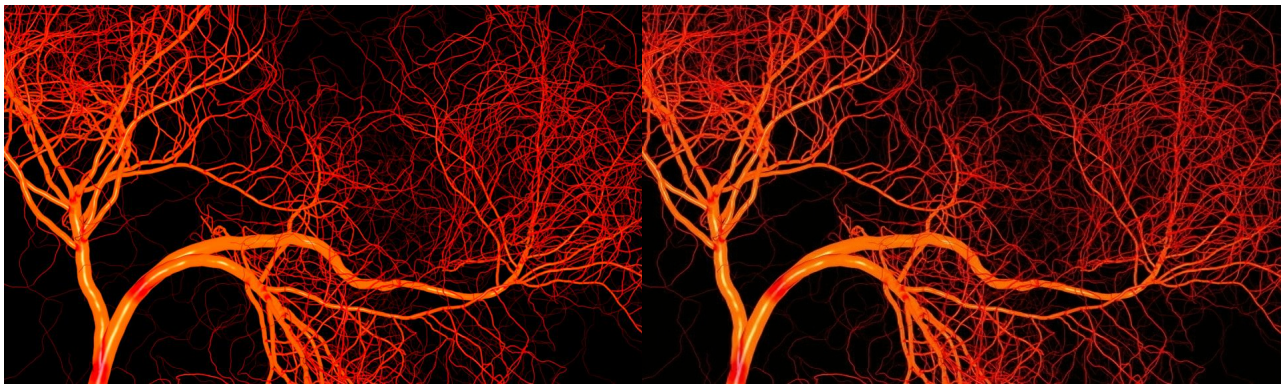
(a) High-Resolution (b) Super-Resolved

Figure 6.17: High-Resolution v Super-Resolved (SF = 2).



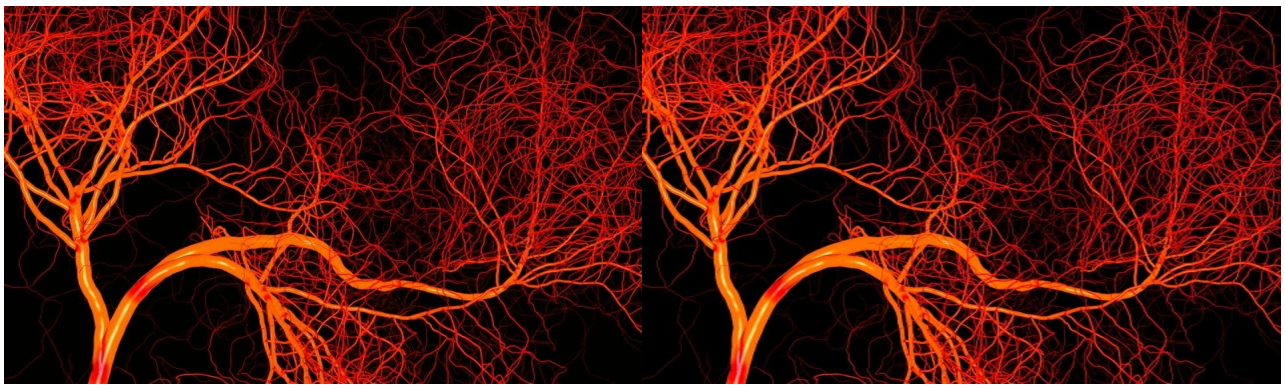
(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.18: Super-Resolution v Bicubic Interpolation (SF = 2).



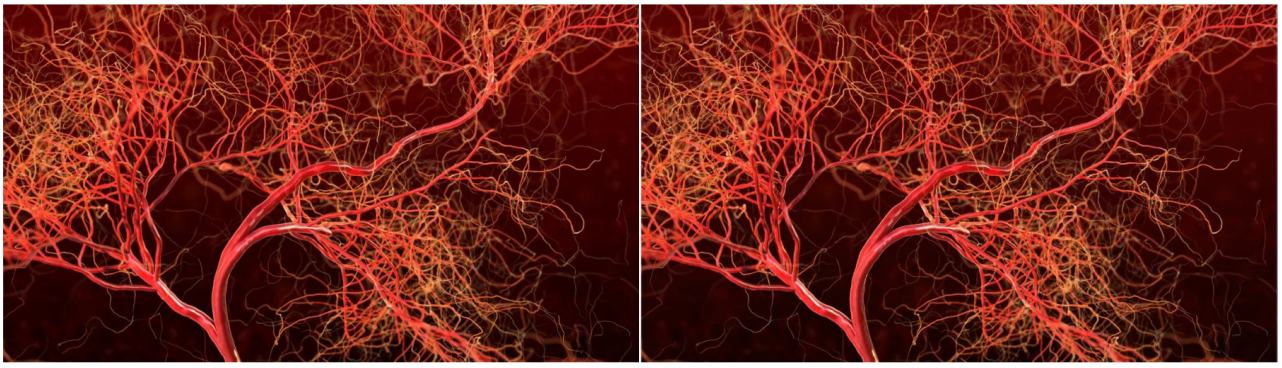
(a) High-Resolution (b) Super-Resolved

Figure 6.19: High-Resolution v Super-Resolution (SF = 2).



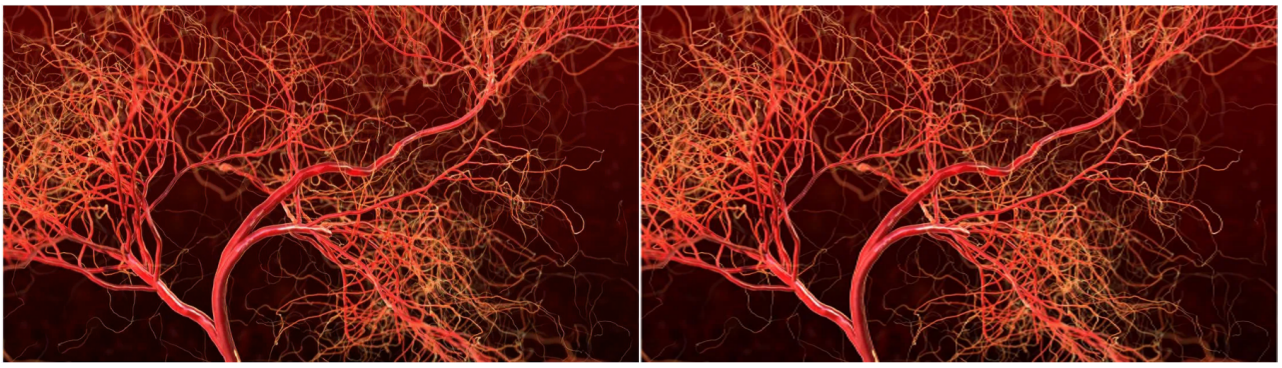
(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.20: Super-Resolution v Bicubic Interpolation (SF = 2).



(a) High-Resolution (b) Super-Resolved

Figure 6.21: High-Resolution v Super-Resolution (SF = 2).



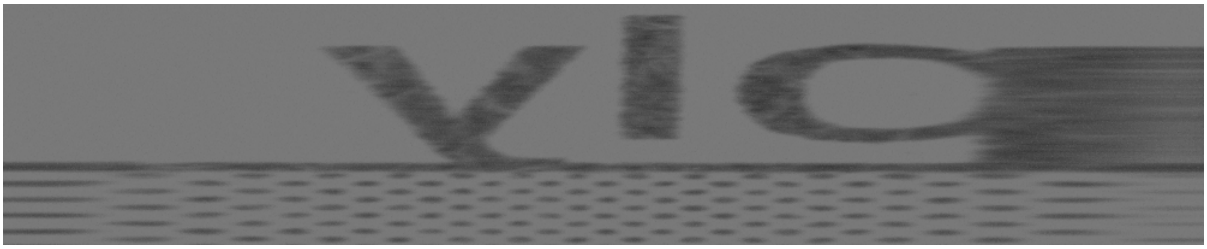
(a) Super-Resolved (b) Bicubic Interpolation

Figure 6.22: Super-Resolution v Bicubic Interpolation (SF = 2).

6.0.4 Experimental Results

In order to acquire sparser data, the scanning frequency of the voice-coil in the PAM setup has to be increased. The same region of an ink-printed image phantom was scanned using the photoacoustic microscope at a frequency of 5Hz and then again at a frequency of 10Hz, which corresponds to half of the full resolution, and finally 20Hz, which corresponds to a quarter of the full resolution. The phantom used was the following:

The results obtained with each frequency was then super-resolved with the corresponding scale factor. The following are the results obtained before super-resolving:



(a) Full-Resolution

(b) Half-Resolution

(c) Quarter-Resolution

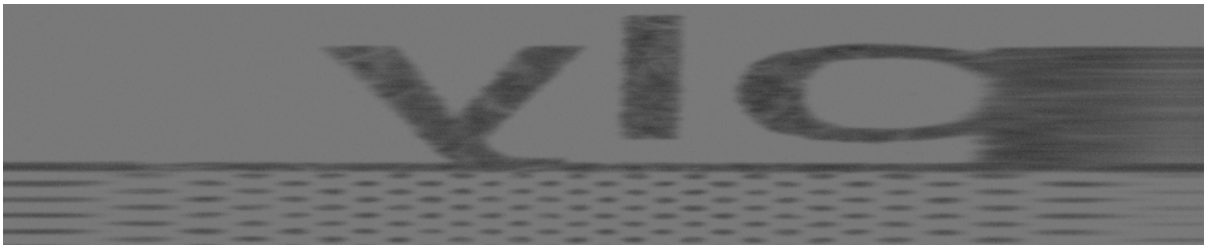
Figure 6.23: Comparison of full-resolution *(top)*, half-resolution *(bottom left)* & quarter-resolution *(bottom right)* (45% of original size).

The following parameters were used in order to obtain experimental results from the PAM device. The scanning range of the y-axis was 3.2mm approximately with 200 data points acquired. The x-axial scanning range for images of all resolutions (i.e. full, half and quarter) is $\sim$4mm. The full resolution image was obtained with a scanning frequency of 5Hz for the voice-coil, the half resolution with a frequency of 10Hz and quarter resolution with a frequency of 20Hz.

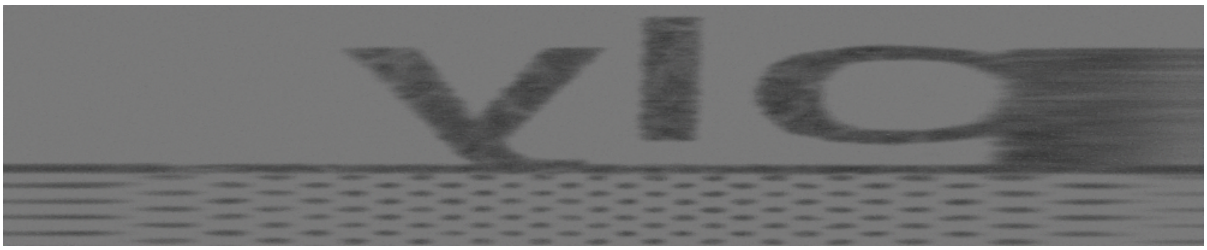
The following table summarises the resolution with their corresponding frequencies and scanning times:

Resolution	Frequency	Scanning Speed	Number of data points
Full	5 Hz	0.32 mm/s	1000
1/2	10 Hz	0.16mm/s	500
1/4	20 Hz	0.08mm/s	250

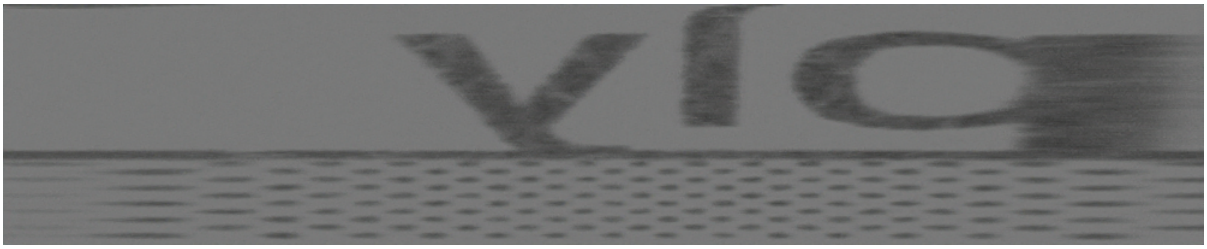
The following are the super-resolved results compared to the full-resolution image:



(a) Full-Resolution



(b) Half-Resolution super-resolved (SF = 2)



(c) Quarter-Resolution super-resolved (SF = 4)

Figure 6.24: Comparison of full-resolution *(top)* and the super-resolved half-resolution *(middle)* & quarter-resolution *(bottom)* results (45% of original size).

The theoretical scanning time is obtained using the following formula:

$$(\text{scan time}) = (\text{sine number}) \times \frac{1}{(\text{scanning frequency})} \tag{6.1}$$

where the sine number corresponds to the number of acquisition points along the y-axis. So for example, with a sine number of 20 and a scanning frequency of 5Hz, the scanning time is $\frac{200}{5} = 40$s.

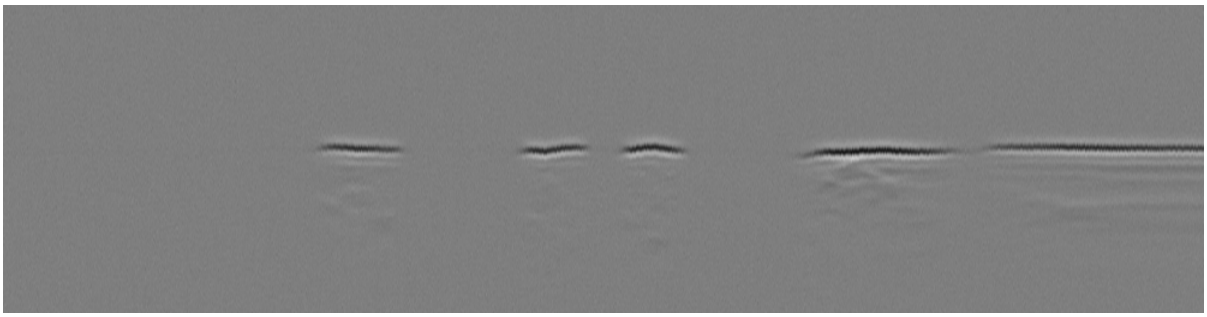
The following is a comparison of the acquisition times achieved with the different scanning resolutions followed by super-resolving (by a factor of 2 for half resolution and a factor of 4 for quarter resolution):

	Full Resolution	**1/2 Resolution**	**1/4 Resolution**
Scan Time	40s	20s	10s
Super-Resolution Time	0s	1.1s	0.7s
Total Time	40s	21.1s	10.7s

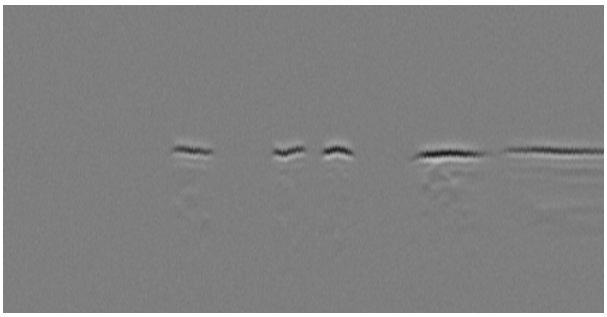
The use of sparse acquisition in PAM followed by image reconstruction via deep-learning based super-resolution provides significantly faster overall acquisition times than full photoacoustic scanning when visualising the maximum amplitude projection (MAP) and this is the case when computing on the CPU so utilising GPU acceleration will only serve to improve these speeds. Moreover, sparser acquisition not only leads to faster overall times but the time needed for the image reconstruction using the super-resolution also decreases with the resolution of the image.

However, the use of this super-resolution technique to recover the full 3D data (i.e. by super-resolving each two-dimensional image contained in the full data) is still not possible as super-resolving is still too slow. Indeed, the full 3D data contains 200 B-scans amounting in a super-resolution time of 3min 42s for half resolution scanning and 2min 20sec for quarter resolution, times significantly greater than the 40s of full scanning.

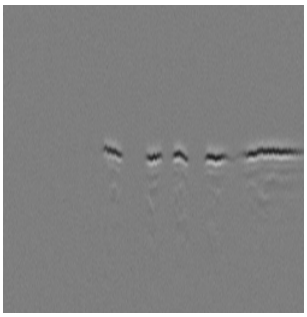
To illustrate the approach mentioned above, super-resolution of a single B-scan (the 36th B-scan out of 200) is presented. Firstly, the original B-scans for each scanning frequency:



(a) B-scan at 5Hz



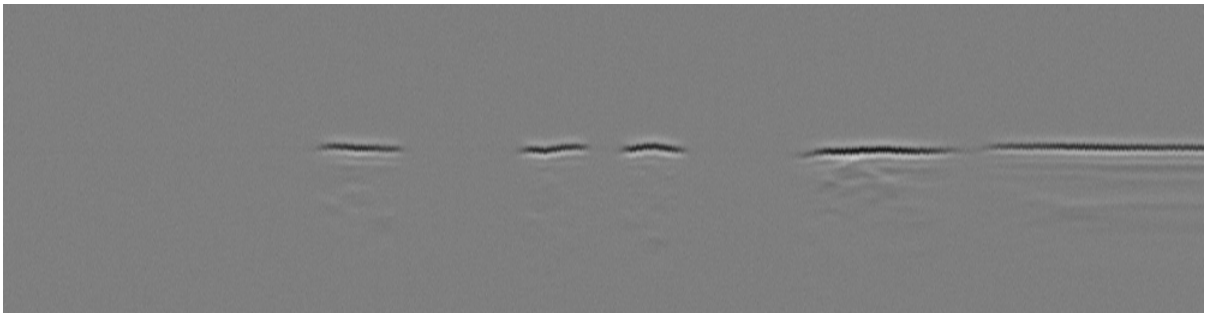
(b) B-scan at 10Hz



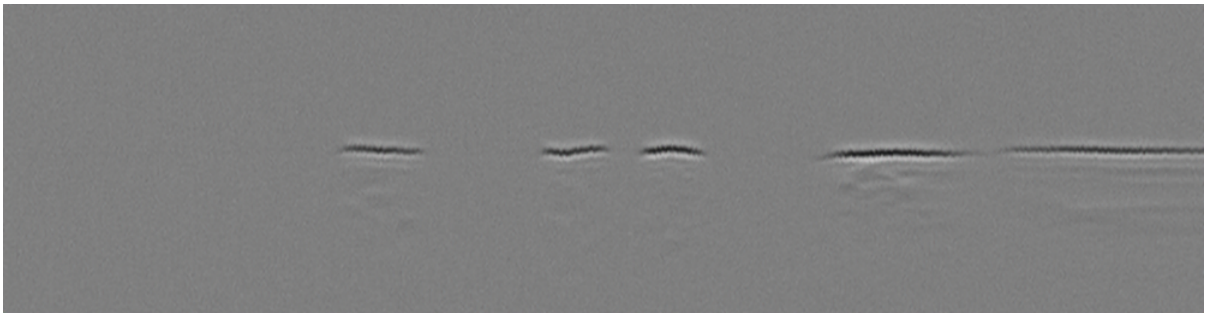
(c) B-scan at 20Hz

Figure 6.25: Comparison of full-resolution *(top)*, half-resolution *(bottom left)* & quarter-resolution *(bottom right)* (45% of original size).

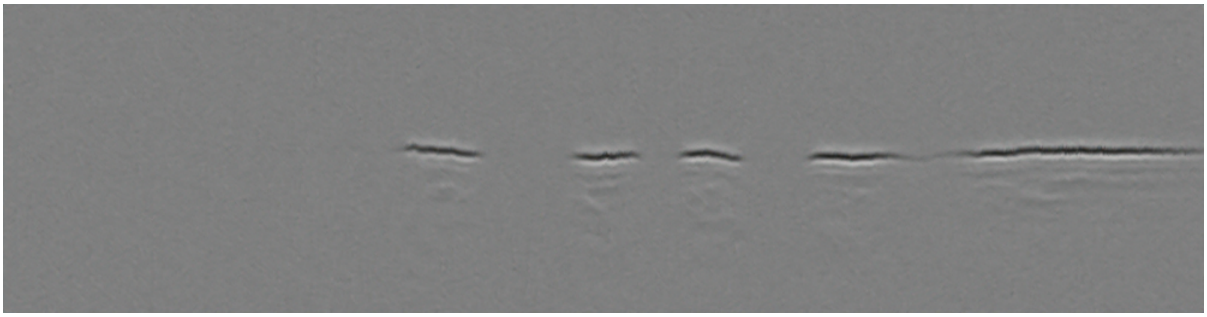
After applying super-resolution:



(a) Full Resolution



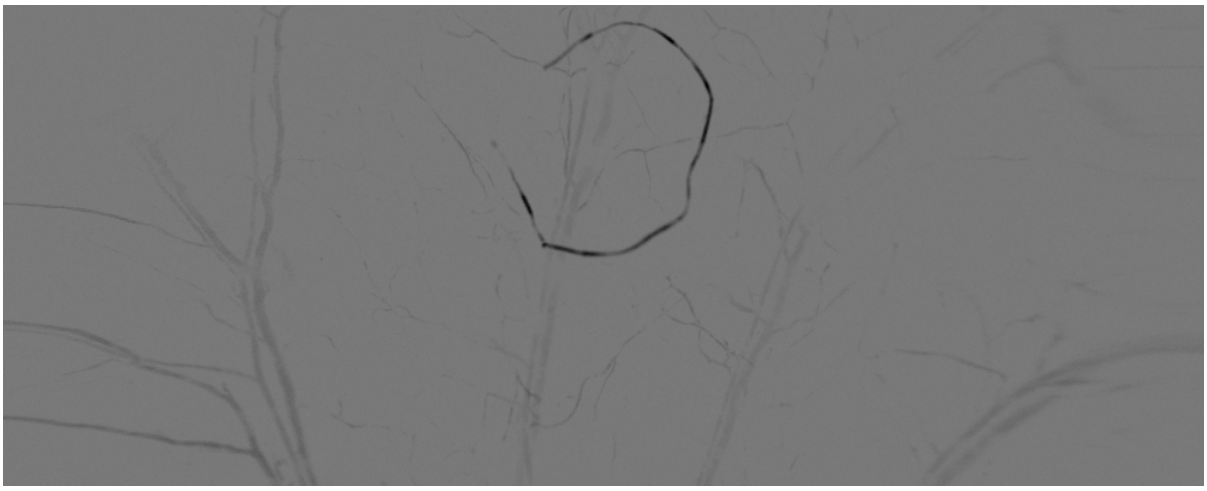
(b) Half Resolution Super-Resolved (SF = 2)



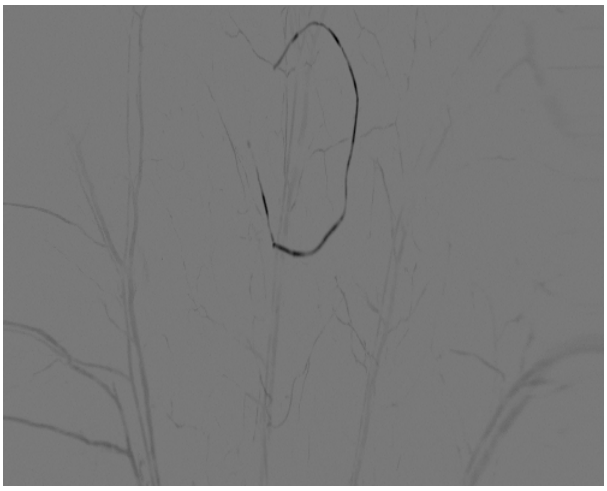
(c) Quarter Resolution Super-Resolved (SF = 4)

Figure 6.26: Comparison of full-resolution *(top)* and the super-resolved half-resolution *(middle)* & quarter-resolution *(bottom)* results (45% of original size).

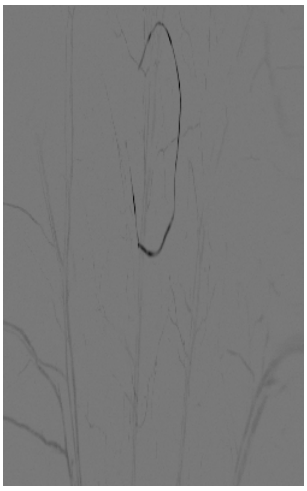
And finally, to illustrate this method on *ex vivo* blood vessel imaging, a full resolution scan of a mouse ear was made at a frequency of 5Hz before being artificially downsampled (by factors 2 and 4 along the x-axis) to simulate scanning at frequencies of 10Hz and 20Hz:



(a) 5Hz



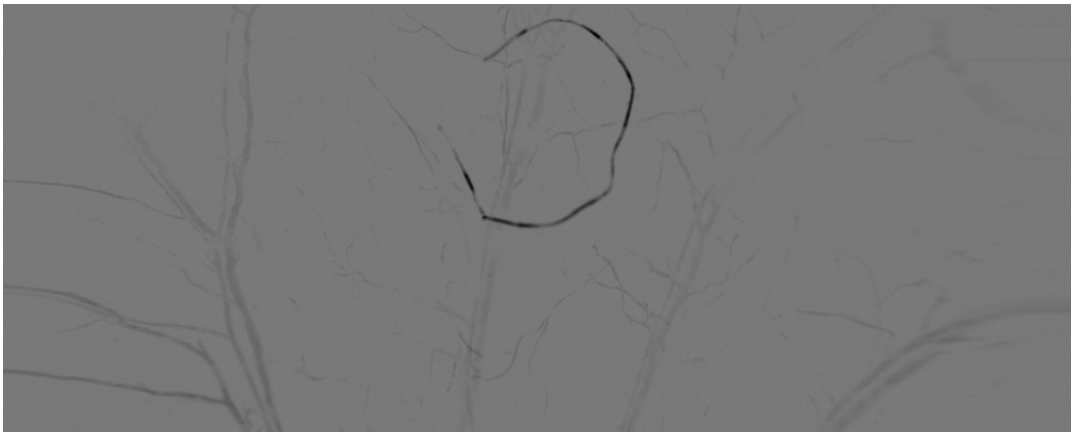
(b) Artificial 10Hz



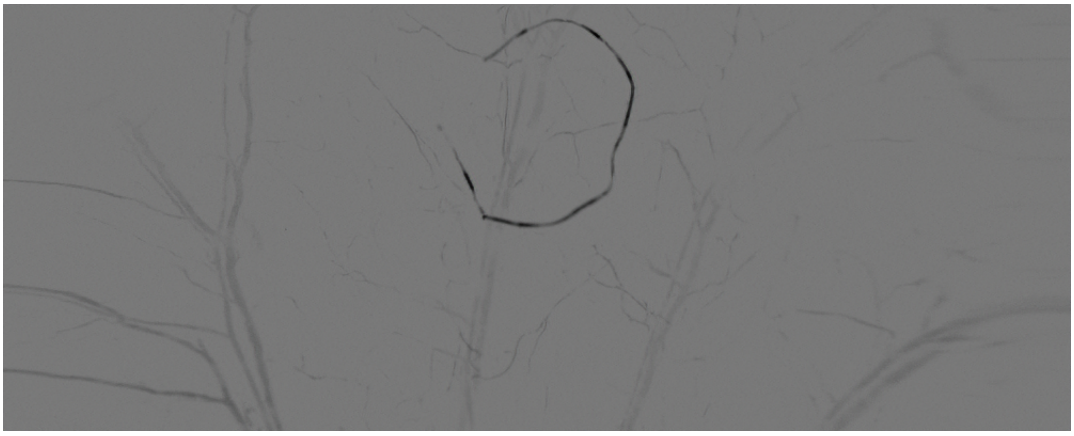
(c) Artificial 20Hz

Figure 6.27: Comparison of full-resolution *(top)*, half-resolution *(bottom left)* & quarter-resolution *(bottom right)* both accomplished via downsampling (45% of original size).

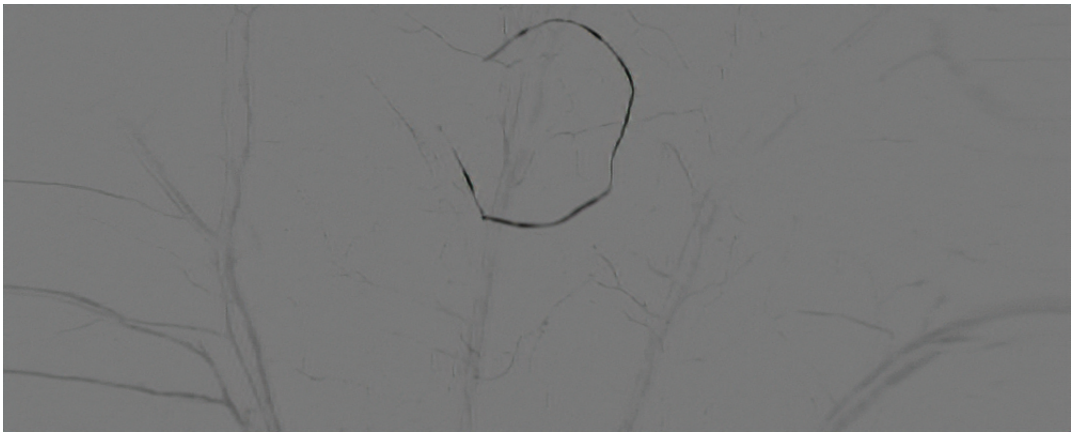
Super-resolution on these artificial sparse PAM images yielded the following results:



(a) Full Resolution



(b) Artificial 10Hz Super-Resolved (SF = 2)



(c) Artificial 20Hz Super-Resolved (SF = 4)

Figure 6.28: Comparison of full-resolution *(top)* and the super-resolved half-resolution *(middle)* & quarter-resolution *(bottom)* results (40% of original size).

7. Discussion and Future Work

In this work, the maximum amplitude projection (MAP) of the PAM data is what is reconstructed from sparse data using super-resolution, not the full 3D data. The MAP, being a 1-channel image, needs to be adapted in order to use the pre-trained network to super-resolve it. This is done by triplicating the 1-channel intensity image from the LabVIEW code in order to be used in the fast-SRGAN network originally trained for 3-channel RGB images. It has been shown in certain medical imaging applications [38], that retraining an RGB network to work on grayscale images before transfer learning leads to improvements in inference speed and accuracy in the case of image classification. Therefore retraining Fast-SRGAN on grayscale images could be an interesting next step towards an increase in performance (the VGG19 network would also have to be retrained for 1-channel images in this case).

In order to apply this method to the full 3D photoacoustic data acquired during sparse scanning of the sample, the network would have to work on images with a depth of 256 (for the number of depth acquisition points along the z direction of scanning) as opposed to 3 (for the RGB colour information). Doing this will undoubtedly be slower than full scanning unless GPU-accelerated deep-learning can be utilised. Retraining for use directly on 256-channel images doesn't seem like the most realistic approach as a dataset of 256-channel images would have to be created (which could be done by stacking the RGB channels of the natural images of the dataset until reaching 256 channels) but mainly, if this method is used, the VGG network would also have to be retrained for 256-channel images.

Alternatively, the original network trained on and for RGB images can be used on each 2D image corresponding to each acquisition point along the y-axis, i.e. on each individual B-scan. This, however, would be very slow as it amounts to super-resolving for example 200 images in order to obtain the full reconstructed 3D data if we have 200 acquisition points along this axis and even using GPU-acceleration, it seems likely this method would be slower than directly acquiring the full data during the scanning as the times obtained for this method when computing on the CPU were much greater than those obtained when fully scanning the sample.

One limitation of this method to improve speed in PAM is that increased sparsity and speed of the scanning is not possible over a scanning frequency of 20Hz as the setup is not yet stable enough to endure such high frequencies.

A next step could be to setup the photoacoustic device to work directly on a computer with a CUDA-enabled GPU for faster passes through the trained network thanks to GPU-acceleration.

One final limitation concerns the evaluation of the accuracy of the super-resolution approach. As mentioned in Chapter 6, both SRGAN and Fast-SRGAN were outperformed by SRResNet and simple bicubic interpolation respectively when comparing metrics such as PSNR and SSIM when clearly both super-resolution GANs produce better perceptual results. A metric that better evaluates the perceptual quality of images other than MOS could prove to be very beneficial as the MOS relies on the ratings given by a subjective quality test by individuals.

8. Conclusion

This Master's thesis aimed to explore the possibility of using deep learning based super-resolution as a means of reconstructing photoacoustic data after sparse acquisition in the hopes of improving imaging times to enable monitoring of blood flow dynamics in the scanned sample.

The choice of the neural network used to solve the super-resolution problem provides a good compromise between speed and accuracy as this network, Fast-SRGAN, is based on the state-of-the-art in super-resolution, SRGAN, optimised for higher speeds by reducing the number of operations needed for a forward pass through the network. The time benchmarks of this network when utilising GPU-acceleration are very impressive and when applied to the task of sparse photoacoustic microscopy, the results are obtained faster than full scanning, even without GPU-acceleration. Thus, this method shows promising results for increasing speed of the photoacoustic acquisition process.

The results produced with Fast-SRGAN adapted to the sparse-PAM context are also of good quality (at least when super-resolving by factors 2 and 4) as they are very close to the ground-truth images of the test dataset of natural images and the experimental results obtained from the photoacoustic device are also promising. Indeed, the results obtained when scanning half the number of points along the x-axis followed by super-resolving with an upscaling factor of 2 as well as those obtained when scanning a quarter of the points along the x-axis and then super-resolving with a factor of 4 were both perceptually very similar to the photoacoustic image obtained with full-resolution scanning.

Thus, the super-resolution approach to sparse-PAM image reconstruction appears to be a viable way of improving imaging times and consequently enabling the visualisation of blood flow dynamics in the brain, which would not currently be possible with full photoacoustic scanning of the sample due to the long acquisition times.

Based on these results, future studies in the field of super-resolution applied to sparse-PAM image reconstruction should aim to further improve speeds thanks to GPU-acceleration and greater computing power or modifications to the Fast-SRGAN architecture itself (e.g. optimising the network for one-channel images) and the possibility of applying this method to the full three-dimensional photoacoustic data should be explored in more detail once faster super-resolution times can be achieved via the use of one of these two methods.

References

[1] Yao, J., & Wang, L. V. (2013). Photoacoustic Microscopy. *Laser & photonics reviews*, 7(5), 758-778. https://doi.org/10.1002/lpor.201200060

[2] Karthikeyan, P., Moradi, S., Ferdinando, H., Zhao, Z., & Myllylä, T., (2020) Optics Based Label-Free Techniques and Applications in Brain Monitoring. *Applied Sciences* 10(6), 2196 https://www.mdpi.com/2076-3417/10/6/2196/htm

[3] Liu, Y., Nie, L., & Chen, X. (2016). Photoacoustic Molecular Imaging: From Multiscale Biomedical Applications Towards Early-Stage Theranostics. *Trends in biotechnology*, 34(5), 420–433. https://doi.org/10.1016/j.tibtech.2016.02.001

[4] Hu, S., Maslov, K., & Wang, L. V. (2011) Second-generation optical-resolution photoacoustic microscopy with improved sensitivity and speed. *Optics Letters*, 36(7), 1134-1136. https://doi.org/10.1364/OL.36.001134

[5] Wang, L. V., & Hu, S. (2012). Photoacoustic tomography: in vivo imaging from organelles to organs. *Science (New York, N.Y.)*, 335(6075), 1458–1462. https://doi.org/10.1126/science.1216210

[6] Wang, L., Maslov, K., Yao, J., Rao, B., & Wang, L.V. (2011). Fast voice-coil scanning optical-resolution photoacoustic microscopy. *Optics Letters*, 36(2), 139-141. https://doi.org/10.1364/OL.36.000139

[7] Recht, B., Fazel, M., & Parrilo, P. A. (2010). Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52(3), 471-501. https://people.eecs.berkeley.edu/ brecht/papers/07.rfp.lowrank.pdf

[8] Liu, T., Sun, M., Meng, J., Wu, Z., Shen, Yi., & Feng, N. (2016). Compressive Sampling Photoacoustic Microscope System based on Low Rank Matrix Completion. *Biomedical Signal Processing and Control*, 26, 58-63. https://dx.doi.org/10.1016/j.bspc.2015.12.008

[9] Fu, Y., Feng, N., Shi, Y., Liu, T, & Sun, M. (2018). Sparse Photoacoustic Microscopy Reconstruction Based on Matrix Nuclear Norm Minimization. *Machine Learning and Intelligent Communications MILICOM 2017 Part I*, LNICST 226, 49-56. https://doi.org/10.1007/978-3-319-73564-1_6

[10] Liu, T., & Zhao, Y. (2018). Non-uniform Sampling Scheme Based on Low-Rank Matrix Approximation for Sparse Photoacoustic Microscope System. *Communications, Signal Processing and Systems*, Lecture Notes in Electrical Engineering 516, 1364-1375. https://doi.org/10.1007/978-981-13-6504-1_162

[11] Liu, T., Sun, M., Liu, Y., Hu, D., Ma, Y., Ma, L., & Feng, N. (2019). ADMM based low-rank and sparse matrix recovery method for sparse photoacoustic microscopy. *Biomedical Signal Processing and Control*, 52, 14-22. https://doi.org/10.1016/j.bspc.2019.03.007

[12] Liu, T., Sun, M., Feng, N., Wang, M., Chen, D., & Shen, Y. (2016). Sparse photoacoustic microscopy based on low-rank matrix approximation. *Chinese Optics Letters*, 14(9).
https://doi.org/10.3788/COL201614.091701

[13] Wang, M., Liu, X., Wang, Q., Sun, M., Zhao, R., & Wu, Z. (2017). Weighted Singular Value Thresholding for Sparse Photoacoustic Microscopy. *IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Turin*
https://doi.org/10.1109/I2MTC.2017.7969718

[14] Bhojanapalli, S., & Jain, P. (2014). Universal Matrix Completion. *arXiv e-prints.*
https://arxiv.org/pdf/1402.2324.pdf

[15] Zhou, T., & Tao, D. (2011). GoDec: Randomized Low-rank & Sparse Matrix Decomposition in Noisy Case. *Proceedings of the 28th International Conference on Machine Learning.*
http://www.icml-2011.org/papers/41_icmlpaper.pdf

[16] Cai, J-F., Candes, E.J., & Shen, Z. (2008). A Singular Value Thresholding Algorithm for Matrix Completion. *arXiv e-prints.*
https://arxiv.org/pdf/0810.3286.pdf

[17] Dong, C., Loy, C. C., He, K., & Tang, X. (2015). Image Super-Resolution Using Deep Convolution Networks. *arXiv e-prints.*
https://arxiv.org/pdf/1501.00092.pdf

[18] Ledig, C., Theis, L., Huszár, F., Caballaro, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv e-prints.*
https://arxiv.org/pdf/1609.04802.pdf

[19] https://github.com/HasnainRaz/Fast-SRGAN

[20] Galteri, L., Seidenari, L., Bertini, M., & Bimbo, A. (2019). Towards Real-Time Image Enhancement GANs. *Computer Analysis of Images and Patterns, 18th International Conference, Proceedings, Part I,* 183-195
http://www.micc.unifi.it/seidenari/wp-content/papercite-data/pdf/caip_2019.pdf

[21] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv e-prints.*
https://arxiv.org/pdf/1704.04861.pdf

[22] https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000015C6tSAE&l=fi-FI

[23] Link to download pdf of instructions: https://forums.ni.com/ni/attachments/ni/170/1141240/

[24] Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.-H., & Liao, Q. (2019). Deep Learning for Single Image Super-Resolution: A Brief Review. *arXiv e-prints.*
https://arxiv.org/pdf/1808.03344.pdf

[25] Simonyan, K., & Andrew Zisserman, A., (2015) Very Deep Convolutional Networks For Large Scale Image Recognition. *arXiv e-prints.*
https://arxiv.org/pdf/1409.1556.pdf

[26] Johnson, J., Alahi, A., & Fei-Fei, L., (2016) Perceptual Losses for Real-Time Style Transferand Super-Resolution. *arXiv e-prints.*
https://arxiv.org/pdf/1603.08155.pdf

[27] Yang, J., Wright, J., Huang, T., & Ma, Y., (2010) Image Super-Resolution Via Sparse Representation. *IEEE Transactions on Image Processing,* vol. 19, no. 11, pp. 2861-2873 https://doi.org/10.1109/TIP.2010.2050625

[28] Zaeemzadeh, A., Rahnavard, N., & Shah, M., (2020) Norm-Preservation: Why Residual Networks Can Become Extremely Deep? *arXiv e-prints* https://arxiv.org/pdf/1805.07477.pdf

[29] He, K., Zhang, X., Ren, S. & Sun, J., (2015) Deep Residual Learning for Image Recognition. *arXiv e-prints* https://arxiv.org/pdf/1512.03385.pdf

[30] https://researchcomputing.princeton.edu/faq/what-is-a-cluster

[31] https://docs.csc.fi/support/faq/

[32] https://slurm.schedmd.com/overview.html

[33] http://catb.org/ esr/jargon/html/D/daemon.html

[34] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y., (2014) Generative Adversarial Nets. *arXiv e-prints* https://arxiv.org/pdf/1406.2661.pdf

[35] Myllylä, T. S., Kaakinen, M., Vihriälä, E., Jukkola, J., Ferdinando, H., Korhonen, V., & Eklund, L., (2020) Functional NIRS study of blood brain barrier disruption when induced by focused ultrasound and intra-arterial mannitol infusion. *In Tissue Optics and Photonics* Vol. 11363, p.113630T. International Society for Optics and Photonics.

[36] Myllylä, T., Harju, M., Korhonen, V., Bykov, A., Kiviniemi, V., & Megkinski, I., (2018) Assessment of the dynamics of human glymphatic system by near - infrared spectroscopy. *Journal of biophotonics,* 11(8), e201700123.

[37] https://www.ni.com/getting-started/labview-basics/

[38] Xie, Y., & Richmond, D., (2018) Pre-training on Grayscale ImageNet Improves Medical Image Classification. *ECCV 2018 Workshop.* https://openaccess.thecvf.com/content_ECCVW_2018/papers/11134/