
# ON DEEP LEARNING WITH NORMALIZING FLOWS FOR THE SUPER RESOLUTION OF OCEANIC CURRENT DATA

by

EVGENIOS TSIGKANOS

MASTER OF SCIENCE

A THESIS SUBMITTED TO
THE CYPRUS INSTITUTE TOWARDS THE
FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE

NICOSIA, SEPTEMBER 2021

## VALIDATION PAGE

**Master of Science Candidate:** Evgenios Tsigkanos
**Title of the thesis:** On Deep Learning with Normalizing Flows for the Super Resolution of Oceanic Current data

*This thesis was submitted towards the fulfilment of the requirements for the degree of Master of Science at the Graduate School of The Cyprus Institute and was approved on May 23, 2022 by the members of the Academic Committee of The Cyprus Institute.*

**Examination Committee:**

---

Asst. Prof. Michalis Nicolaou (Chair of the Committee)

---

Asst. Prof. Theodoros Christoudias (Member)

---

Dr Charalambos Chrysostomou (Member)

## DECLARATION

This thesis was submitted towards the fulfilment of the requirements for the award of a Master of Science Degree from The Cyprus Institute. It is the product of my own original work, unless otherwise mentioned through references, notes or other statements.

Evgenios Tsigkanos

# Abstract

High resolution representations of oceanic current data are paramount in applications such as ship routing and sea rescue operations. Techniques that reduce the computational complexity as well as the need for huge datasets in weather simulation are required. Super Resolution (SR) is such a technique, often employing statistical or learning methods; with recent developments adopting deep learning. With SR through deep learning, two main objectives are achieved; a lower complexity model than traditional weather simulations demand, as well as a reduction on the amount of required observations due to possible extrapolation. In this work, we create and train a model inspired by SRFlow – originally built for super resolution on human faces – targeting SR of oceanic current data. Subsequently, we assess its performance in this particular and challenging setting. In our evaluation we assess the effect of flows added to convolutions as well as different scaling factors on the resulting super resolution image, using application appropriate image degradation metrics. Finally, we compare the effectiveness of different architectures using MSE, PSNR, SSIM and LPIPS.

# Acknowledgements

I would like to thank Asst. Prof. M. Nicolaou for his guidance in this project, Andrea Littardi for his help and the Cyprus Institute HPCF for providing access to the computational resources.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Super resolution describes a family of methods with the goal of upscaling images or video. It essentially entails the increase of resolution using a variety of techniques by extracting information from several low resolution (LR) images in order to create/approximate a high resolution image (HR). The resulting super resolution (SR) image contains a higher pixel density and thus more information about a scene or subject. High resolution is sought after in many tasks such as in computer vision applications, because it often leads to better performance. Other applications that benefit greatly from high resolution images but are not always able to get them, include weather simulations, medical imaging, satellite imaging and surveillance.

As mentioned, in many fields high resolution images are not always obtainable or convenient to gather. This might be due to latency or bandwidth limitations, storage need, inherent sensor constrains or optics technology. Super resolution tries to overcome these constraints through image processing algorithms. The benefits offered include the reduction on the amount of required observations (and utilization of existing low resolution data) as well as the development of lower complexity models. Additionally, an after effect is that usually these techniques are less expensive to develop and maintain.

It is however important to note that Super Resolution is inherently an ill-posed problem. This essentially means that there exists more than one solution to a given problem - a low resolution image has more than one high resolution representation. This is what gives rise to the use of Normalizing Flows in this context. Bijective techniques are paramount when dealing with multiple solutions. Current SR methods such as GANs, while effective, only provide a single solution.

In this chapter we set forth to the reader the concepts delved into in this thesis. We briefly review the state-of-the-art methods that are used today in order to produce SR results and the applications where it is needed. Finally, we outline the methodology and contributions that are followed in the rest on the text.

## 1.1 Different Approaches

In order to achieve the objective of super resolution over the years, many techniques have been developed – ranging from bicubic and bilinear interpolation to Generative Adversarial Networks (GANs) and Normalizing Flows (NFs). In the following, we succinctly present these methods.

### 1.1.1 No training methods

In mathematics, bicubic interpolation is an extension of cubic interpolation (not to be confused with cubic spline interpolation) for interpolating data points on a two-dimensional regular grid as shown in [1]. The interpolated surface is smoother than corresponding surfaces obtained by bilinear interpolation or nearest-neighbor interpolation. Bicubic interpolation can be accomplished using either Lagrange polynomials, cubic splines, or cubic convolution algorithm.

In image processing as explained in [1], bicubic interpolation is often chosen over bilinear or nearest-neighbor interpolation in image resampling, when speed is not an issue. In contrast to bilinear interpolation, which only takes 4 pixels ($2\times2$) into account, bicubic interpolation considers 16 pixels ($4\times4$). Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts.

### 1.1.2 Feedforward Neural Networks & Deep Learning

Most modern deep learning models are based on artificial neural networks, specifically convolutional neural networks (CNN)s, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines.

In deep learning, each layer consists of a transformation of the input data towards a more abstract and composite representation. This transformation in the end has the goal of approximating the ground truth in for whichever input the model has gotten. In an image recognition application for faces, the raw input may be a 3-dimentional matrix of pixels (for RGB); the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own. This does not completely eliminate the need for user-tuning; for example, varying numbers of layers,layer sizes or hyperparameters can provide different degrees of abstraction and results.

The word "deep" in "deep learning" refers to the number of layers that data pass through in order to get transformed. More precisely [2] explains, deep learning systems have a substantial credit assignment path (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feedforward neural network, the depth of the CAPs is that of the network and is the number

of hidden layers plus one (as the output layer is also parameterized). For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited. No universally agreed-upon threshold of depth divides shallow learning from deep learning, but most researchers agree that deep learning involves CAP depth higher than 2. CAP of depth 2 has been shown to be a universal approximator in the sense that it can emulate any function. Beyond that, more layers do not add to the function approximator ability of the network. Deep models (CAP $>$ 2) are able to extract better features than shallow models and hence, extra layers help in learning the features effectively.

Deep learning architectures can be constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance. For supervised learning tasks [3], deep learning methods eliminate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures that remove redundancy in representation. Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabeled data are more abundant than the labeled data. Examples of deep structures that can be trained in an unsupervised manner are neural history compressors and deep belief networks.

### 1.1.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), a type of deep neural network, used for unsupervised learning, are among the most popular topics within Deep Learning. In unsupervised learning the network learns to discover patterns and unseen information in the data without explicit labels nor outside input. In recent years, there has been an exponential increase in research featuring GANs, with applications most notably in computer vision, pose transfer, style transfer, super resolution, artifact removal and image translation.

GANs typically consist of two opposing networks. The first network is the Generator $G(x)$, which as the name implies generates data similar to those in the training set, and the Discriminator $D(x)$ which tries to identify whether the data is real or not. The two opposing networks essentially play a game, with the Generator trying to fool and get past the Discriminator with data that look like the real thing (the training set) but are not. The input in GANs can vary from images, video, or audio.

The way the Generator creates similar input is by adding noise to the training set, usually by sampling a uniform or normal distribution. The Discriminator outputs the probability of the input being from the training set or not, with 1 being real and 0 being fake.

The mathematical basis of the GAN is the minmax game, described by the following function:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z))], \qquad (1.1)$$

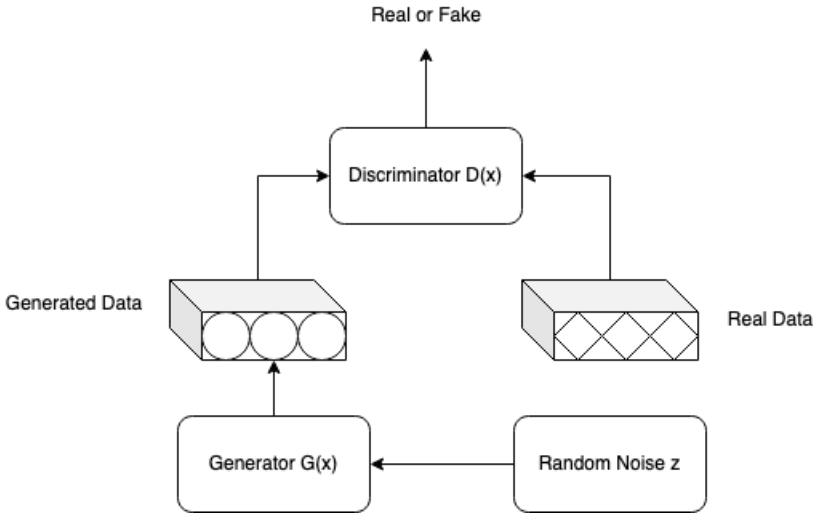



Figure 1.1: General architecture of GANs.

where the Generator tries to minimize the expected value:

$$\frac{1}{m}\sum_{1}^{m}\log(1 - D(G(z)), \tag{1.2}$$

and the Discriminator tries to maximize the expected value:

$$\frac{1}{m}\sum_{1}^{m}[\log D(x) + \log(1 - D(G(z))], \tag{1.3}$$

and output the probability of the given sample being from the training data.

In essence, G simply seeks to minimize the quantity $\log(1 - D(G(z))$ with high values of $D(G(z))$ such that D perceives that $G(z)$ belongs to the training data. The optimal objective of the network is that D outputs the probability $P(x) = 0.5$, which means that the output from G is indistinguible from the ground truth.

As for applications, GANs are perhaps best known for their contributions to image synthesis. StyleGAN [4], a model Nvidia developed, has generated high-resolution head shots of fictional people by learning attributes like facial pose, freckles, and hair. A newly released version — StyleGAN 2 [5] — makes improvements with respect to both architecture and training methods, redefining the state of the art in terms of perceived quality. In June 2019, Microsoft researchers detailed ObjGAN [6], a novel GAN that could understand captions, sketch layouts, and refine the details based on the wording. The coauthors of a related study proposed a system — StoryGAN [7] — that synthesizes storyboards from paragraphs. Such models have made their way into production. Startup Vue.ai's GAN susses out clothing characteristics and learns to produce realistic poses, skin colors, and other features. From snapshots of apparel, it can generate model images in every size up to five times faster than a traditional photo shoot.

Elsewhere, GANs have been applied to the problems of super-resolution (image upsampling) and pose estimation (object transformation). Tang says one of his teams used GANs

**Algorithm 1:** Minibatch stochastic gradient descent training of Generative Adversarial Networks. The number of steps to apply to the Discriminator, k, is a hyperparameter.

1: **for** number of iterations **do**
2: **for** k steps **do**
3: sample minibatch of $m$ noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.
4: sample minibatch of $m$ noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from data generating distribution $p_{data}(x)$.
5: update the Discriminator by ascending its stochastic gradient descent:

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\big(x^{(i)}\big) + \log\big(1 - D\big(G\big(z^{(i)}\big)\big)\big) \right]$$

6: **end for**
7: sample minibatch of $m$ noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.
8: update the Generator by descending its stochastic gradient descent:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log\big(1 - D\big(G\big(z^{(i)}\big)\big)\big)$$

9: **end for**

to train a model to upscale $200 \times 200$ pixel satellite imagery to $1000 \times 1000$ pixels, and to produce images that appear as though they were captured from alternate angles. Other applications include music, video, art, medicine, etc.

### 1.1.4 Normalizing Flows

A major goal of statistics and machine learning has been to model a probability distribution given samples drawn from that distribution [8]. This is an example of unsupervised learning and is sometimes called generative modelling. Its importance derives from the relative abundance of unlabelled data compared to labelled data. Applications include density estimation, outlier detection, prior construction, and dataset summarization.

Many methods for generative modeling have been proposed. Direct analytic approaches approximate observed data with a fixed family of distributions. Variational approaches and expectation maximization introduce latent variables to explain the observed data. They provide additional flexibility but can increase the complexity of learning and inference. Graphical models [9] explicitly model the conditional dependence between random variables. Recently, generative neural approaches have been proposed including Generative Adversarial Networks (GANs) [10] and variational autoencoders (VAEs) [11].

GANs and VAEs have demonstrated impressive performance results on challenging tasks

such as learning distributions of natural images. However [8] poses that, several issues limit their application in practice. Neither allows for exact evaluation of the probability density of new points. Furthermore, training can be challenging due to a variety of phenomena including mode collapse, posterior collapse, vanishing gradients and training instability [12] [13].

Normalizing Flows (NF) are a family of generative models with tractable distributions where both sampling and density evaluation can be efficient and exact. Applications include image generation [14] [15], noise modelling [16], video generation [17], audio generation [18] [19], graph generation [20], reinforcement learning [21] [22], and physics [23] Wong et al., 2020]. There are several survey papers for VAEs [24] and GANs [25] [26]. Furthermore, [8] affirms that Normalizing Flows were popularised by [27] in the context of variational inference and by [28] for density estimation. However, the framework was previously defined in [29] and [30], and explored for clustering and classification [31], and density estimation [Laurence et al., 2014 [32].

A Normalizing Flow is a transformation of a simple probability distribution (e.g., a standard normal) into a more complex distribution by a sequence of invertible and differentiable mappings. The density of a sample can be evaluated by transforming it back to the original simple distribution and then computing the product of i) the density of the inverse-transformed sample under this distribution and ii) the associated change in volume induced by the sequence of inverse transformations. The change in volume is the product of the absolute values of the determinants of the Jacobians for each transformation, as required by the change of variables formula. The result of this approach, as [8] expands, is a mechanism to construct new families of distributions by choosing an initial density and then chaining together some number of parameterized, invertible and differentiable transformations. The new density can be sampled from (by sampling from the initial density and applying the transformations) and the density at a sample (i.e., the likelihood) can be computed.

## 1.2 Applications

In our case, the improvement offered in oceanic current models aims to resolve two main problems. Firstly, an immediate benefit of deep learning methods in general over current statistical weather forecasting lies in the reduction of complexity. Deep learning methods, while requiring large amounts of data to train, have the advantage of greatly reduced time as well as computational complexity after the training phase. Secondly, specifically for super resolution in weather modeling, there exists the problem of missing data. While high resolution data exists for high traffic areas such as the Mediterranean sea or the Gulf of Mexico, the vast majority of areas do not have this advantage. This might be due to lack of instrumentation or less interest due to lower traffic. Nonetheless the lack of regional weather models for these areas remains a difficult problem.

In the field of satellite imaging it is often desired to have higher resolution images [33]. In order to execute this task SR plays a very significant role. Satellite image processing includes

image rectification, restorations, enhancement and also information extraction. All these areas require super resolution techniques. A super resolved image increases the number of pixels which enhances the display of the digital image, i.e., the visual interpretation increases. Moreover, this may aid in the removal of distortions, while geographical information can be further enhanced. In further processing, SR can also be combined in further classification of areas or geographical locations. It can also include learning-based techniques which are useful in land map constructions.

In the medical field, Super Resolution has its own particular role [33]. In CT scans, MRI and in other medical imaging techniques high contrast and image enhancement are needed, which can be fulfilled by SR methods. Most of the images in the medical field are of low resolutions, geometric deformations and with low contrast, i.e., X-ray has lower contrast, ultrasound having noisy images etc. Moreover as [33] explains, if more time is given for imaging to get more data, due to patient movement, blurring may also appear. So in order to tackle these issues, SR can be employed.

Super resolution is also playing an important role in microscopic image processing [34]. In this area recently much advancement has been done as per the literature to visualize the biological structures including cell and tissue. Super-resolution fluorescence microscopy is a significant field in microscopic imaging, where also the 2014 the Nobel Prize in Chemistry has been given in the area of Super-resolved fluorescence microscopy. A Flurescence microscope is one of the essential tools for examination of the pathways, biological molecules, living cells, tissues, and even whole subjects. It is more useful as compared to electron microscopy. Other techniques like MRI or OCT (optical coherence tomography) can give resolutions in 10s of centimeters and micrometers. However with super resolutions flurescence range can be further increased.

Nowadays, the prevalence of multimedia applications is increasing. Super resolution also involves the multimedia industry [33]. Time movies, animations, visual effects all require HD data. So SR can also be proved as the useful technique in video enhancements. Many methods used in multimedia based applications uses the SR method for the enhancement of images and videos. Cell phone based applications like image or videos also included SR based techniques to enhance their quality.

In the field of astronomical studies, Super Resolution is also involved as the significant technique [35]. High resolution astronomical images are always desirable for better computation. Many blurred and noisy images can be combined to get a better view. It has used SR for improvement of quality of astronomical images. Tightly grouped stars and far away objects can be visualized in an improved way. In this field, multiple unidentified objects may also be visualized.

Beyond the previous applications mentioned, SR is also seeing applications in areas like object detection, automotive industry, real time processing, scanning, surveillance, military and forensics. Furthermore, potential appears recently promising in the automotive industry, in classification and robotics where it is acting as a supportive technique. In forensic applications

SR-based methods are also being employed.

## 1.3 Methodology

The present thesis followed a systematic methodological procedure consisting of the following steps:

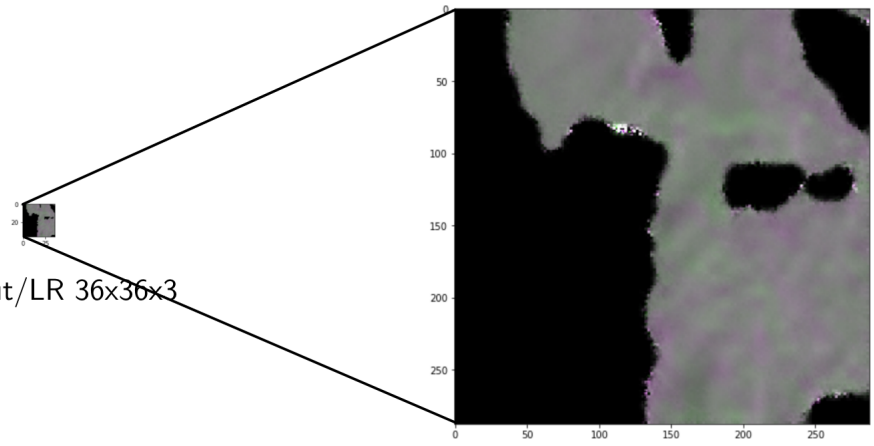
1. We identified the algorithms and models that are state-of-the-art and are used today for Super Resolution.
2. We chose the dataset and the model.
3. We created a preprocessing pipeline specific for the dataset.
4. We modified the architecture and the dataset in order to fit characteristics such as the difference in channels.
5. We created a testing pipeline specific for the dataset.
6. We produced and evaluated the results.
7. We distilled conclusions based on the aforementioned results.

The rest of this thesis is structured as follows. We present motivating examples and identify algorithms and models that are state-of-the-art and are used today for SR applications. In Chapter 2 we present related work that has been published about the subject at hand. Chapter 3 discusses the underlying theoretical model, Normalizing Flows, and in Chapter 4 we discuss the model that this thesis is based on, SRFlow. In Chapter 5 we analyze the data used and we present the findings of our experiments. Finally, Chapter 6 explores the results of the aforementioned experiments and concludes the thesis.

## 1.4 Contributions

Our key contribution lies in the proposal of a different method to achieve super resolution in regional weather models. To achieve this, the SRFlow architecture [36] is employed – an improvement over current super resolution methods that utilizes Normalizing Flows.

The SRFlow model was initially developed for other kinds of super resolution such as faces, trained and tested on CelebA [37]. We modify and adapt SRFlow in order to employ it for oceanic current data, and investigate the results. An example of the outcome of this thesis can be seen in Figure 1.2.




(a) Input/LR 36x36x3

(b) Output/SR 288x288x3

Figure 1.2: Realistic representation of the results of our contribution, with a scaling factor $r = 8$, on oceanic currents in the Mediterranean. The lower resolution image (left) is scaled using the model we create in order to produce the Super Resolution image on the right.

# Chapter 2

# Related Work

In this chapter we present and analyze the technical background in machine learning that the methods and techniques in this thesis use. We approach related work from three perspectives within the Deep Generative Models family; first, by considering Generative Adversarial Networks, then Normalizing Flows; finally, we review literature from the field of Autoencoders.

## 2.1 Deep Generative Models

Deep Generative Models are a relatively new form of models, born out of the combination of popular methods in machine learning; generative models and deep neural networks. The benefit of deep generative models over more traditional methods is that they minimize the number of parameters compared to the features of the input data. In essence these kind of models bring to light a concentrated abstraction of the data. Simply put, they gain the advantages of both deep neural networks as well as generative methods in the network parameters and latent encodings needed for generation. Well-known and state-of-the-art examples of deep generative models include GPT-2/GPT-3 [38] [39], BigGAN [40], VQ-VAE [41], Optimus [42] and jukebox [43].

### 2.1.1 Generative Adversarial Networks

In the following, we mention some of the landmark research in Generative Adversarial Networks. In [44] the authors define the architectural guidelines for GANs as well as feature visualization, and propose DCGAN, a latent space interpolation, using discriminator features to train classifiers. In [45], the authors explain how to stabilize training in GANs, and apply: feature matching, minibatch discrimination, historical averaging, one-sided label smoothing, and virtual batch normalization. In [46], the authors introduce Conditional GANs and image-to-image or text-to-image. In [47], multi-scale architecture is defined and methods to work aroung large resolution instability are proposed. In a follow-up work, [48], latent space training is established, along with Neural Style Transfer known as Adaptive Instance Normalization

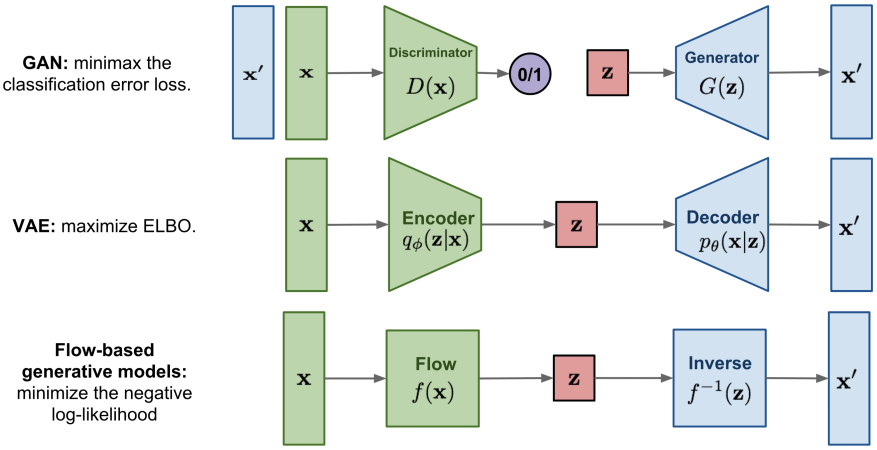



Figure 2.1: Flow-based Deep Generative models.

(AdaIN) [4]. In [40], a cornerstone paper, ImageNet generation is introduced along with self-Attention, spectral Normalization, cGAN with projection discriminators. In [49], the authors introduce image-to-image translation, PatchGAN, U-Net style generator architecture, ResNet-style skip connections in the generator and edge-maps to photo-realistic images

Specifically for super resolution, in [50], the authors present image-to-image translation, Cycle-Consistency loss formulation, GAN super-resolution and style transfer. Also in [51], SRGAN is presented, as another evolution of GANs with the goal of SR, along with [52].

### 2.1.2 Normalizing Flows

In the following, we mention some of the landmark research in Normalizing Flows. In [27], the authors describe approximations of distributions constructed through a normalizing flow, where they transform a simple initial density to a complex one. This is done through a series of invertible transformations, and consists of the basis of what we explore in this thesis. Following this, in [53], the authors extend the space of such models using real-valued non-volume preserving (real NVP) invertible and learnable transformations, and apply it to model natural images.

The authors in the survey article [8], give a complete review of the theoretical basis and use of Normalizing Flows for distribution learning. They review current state-of-the-art literature around the subject and explain its usage in a modern setting. Finally in the foundational paper of this thesis [36], the authors introduce SRFlow, a normalizing flow based super-resolution method capable of learning the conditional distribution of the output given the low-resolution input, while at the same time showing that it outperforms state-of-the-art GAN-based models.

### 2.1.3 Autoencoders

In the following, we mention some of the landmark research in Autoencoders. In [54], the authors provide a method of counteracting the blurriness that is usually present in gener-

ated images. They offer a principled method for jointly learning latent-variable models and corresponding inference models as done in VAEs, by adding a random variable that is a down-scaled version of the original image, while still use the log-likelihood function as the learning objective.

In [55] the authors propose a joint image denoising and super-resolution model via Variational AutoEncoder. This is implemented with a conditional variational autoencoder that encodes the reference for dense feature vector which then is transferred to the decoder for target image denoising and used as input to the discriminator acting as super-resolution sub-network.

In [56] the authods use an Autoencoder-inspired Convolutional Network in order to create SR images of MRIs. In [57], the authors design a data-driven model utilizing a coupled deep autoencoder (CDA) for single image Super Resolution. As cited, CDA simultaneously learns the intrinsic representations of LR and HR image patches and a big-data-driven function that precisely maps these LR representations to their corresponding HR representations, consisting of a great tool for single image SR.

# Chapter 3

# Normalizing Flows

In both statistics and machine learning, often the objective is to learn or accurately approximate an underlying distribution with as few data points as possible. Due to the nature of data availability, as in that labeled data are very labor intensive to generate or hard to find, frequently unsupervised techniques are employed. Such unsupervised techniques are often called generative modelling. Historically, these methods have been introduced with expectation maximization and variational approaches, and continued more recently with generative neural modelling that includes GANs and VAEs. These have been the state-of-the-art methods for tasks such as transferring style, generating data and generally learning the underlying distributions of images. However there exist drawbacks with these kind of methods. Specifically, some of the issues they suffer from are; vanishing gradients, training instability, mode collapse and inaccurate approximation of probability density on new data.

The newly proposed model is Normalizing Flow (NF). NF still is categorized as a generative technique, where the primary goal of is to sample the probability density and distribution accurately and efficiently. In this chapter we explore their mechanics. The fundamental principle of normalizing flow is the reconstruction of a complex probability distribution into a composite one. Essentially, with NF we can use a simple probability distribution as shown in Figure 3.1, and modify it sequentially in order to match the complex one. The key advantage that this process awards is firstly invertibility, and secondly the ability to differentiate.

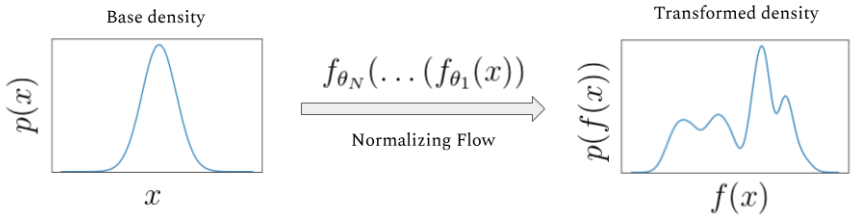



Figure 3.1: Density transformation example [8].

As an overview of the internal mechanics of NFs, we explore the probability density of the target. This is done by deconstructing it to basic components and calculating the density of

the inverse-transformed sample under this distribution and the associated change in volume induced by the sequence of inverse transformations. The change in volume is the product of the absolute values of the determinants of the Jacobians for each transformation, as required by the change of variables formula. The outcome of this process is the ability to generate similar distributions by re-using the aforementioned transformations and shifting the "initial condition". Then the new output can be re-evaluated with the same process in order to match the density of the input.

In the following, we recall foundational definitions from [8]. Specifically, we concretely specify normalizing flow concepts as used in the context of the present work. The interested reader is referred to [8] for verbose technical details.

## 3.1 Fundamentals

Following [8], let $Z \in \mathbb{R}^D$ be a random variable with a known and tractable probability density function $p_Z : \mathbb{R}^D \longrightarrow \mathbb{R}$. Let $g$ be an invertible function and $Y = g(Z)$. Then using the change of variables formula, one can compute the probability density function of the random variable $Y$:

$$p_Y(y) = p_Z(f(y))|\det Df(y)| = p_Z(f(y))|\det Dg(f(y))|^{-1} \tag{3.1}$$

where $f$ is the inverse of $g$, $Df(y) = \frac{\partial f}{\partial y}$ is the Jacobian of $f$ and $Dg(z) = \frac{\partial g}{\partial z}$ is the Jacobian of $g$. This new density function $p_Y(y)$ is called a pushforward of the density $p_Z$ by the function $g$ and denoted by $g_\star p_Z$.

In the context of generative models, the above function $g$ (a generator) "pushes forward" the base density $p_Z$ (sometimes referred to as the "noise") to a more complex density. This movement from base density to final complicated density is the generative direction. Note that to generate a data point $y$, one can sample $z$ from the base distribution, and then apply the generator: $y = g(z)$.

As per [8], the inverse function $f$ moves (or "flows") in the opposite, normalizing direction: from a complicated and irregular data distribution towards the simpler, more regular or "normal" form, of the base measure $p_Z$. This view is what gives rise to the name "normalizing flows" as $f$ is "normalizing" the data distribution. This term is doubly accurate if the base measure $p_Z$ is chosen as a Normal distribution as it often is in practice. Intuitively, if the transformation $g$ can be arbitrarily complex, one can generate any distribution $p_Y$ from any base distribution $p_Z$ under reasonable assumptions on the two distributions. This has been proven formally in [58] and [59].

Constructing arbitrarily complicated non-linear invertible functions (bijections) can be difficult. By the term Normalizing Flows people mean bijections which are convenient to compute, invert, and calculate the determinant of their Jacobian. One approach to this is to note that the composition of invertible functions is itself invertible and the determinant of its Jacobian

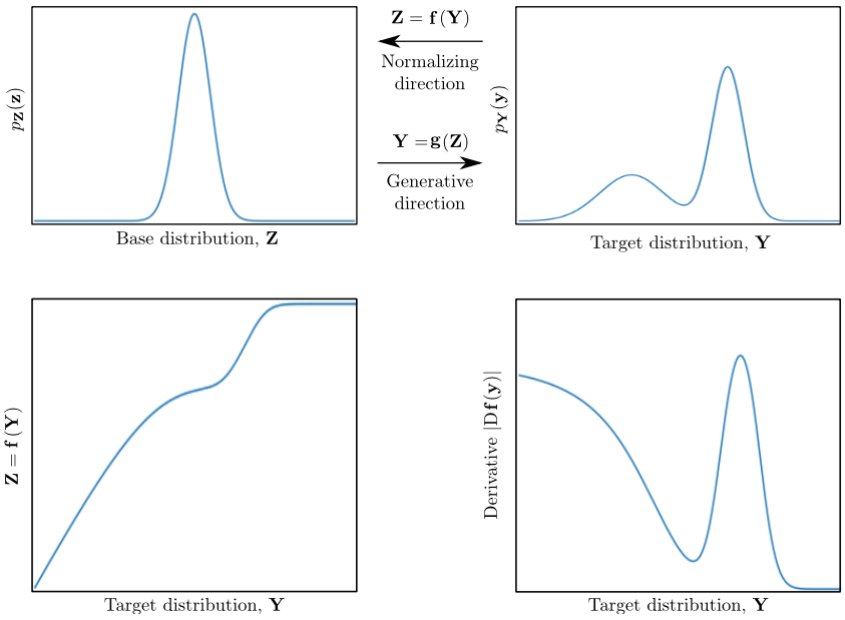



Figure 3.2: Change of variables example. Top-left: the density of the source $p_Z$. Top-right: the density function of the target distribution $p_Y(y)$. There exists a bijective function $g$, such that $p_Y = g * p_Z$, with inverse $f$. Bottom-left: the inverse function $f$. Bottom-right: the absolute Jacobian (derivative) of $f$ [8].

has a specific form.

In particular, let $g_1, g_2, ..., g_N$ be a set of $N$ bijective functions and define:

$$g = g_N \circ g_{N-1} \circ ... \circ g_1 \tag{3.2}$$

to be the composition of the functions. Then it can be shown that $g$ is also bijective, with inverse:

$$f = f_1 \circ ... \circ f_{N-1} \circ f_N \tag{3.3}$$

and the determinant of the Jacobian is:

$$\det Df(y) = \prod_{i=1}^{N} \det Df_i(x_i) \tag{3.4}$$

where $Df_i(y) = \frac{\partial f_i}{\partial x}$ is the Jacobian of $f$. We denote the value of the $i-th$ intermediate flow as:

$$x_i = g_i \circ ... \circ g_1(z) = f_{i+1} \circ ... \circ f_N(y) \tag{3.5}$$

and so $x_N = y$. Thus, a set of nonlinear bijective functions can be composed to construct successively more complicated functions, as shown in [8].

## 3.2 Density estimation and sampling

The natural and most obvious use of normalizing flows is to perform density estimation. For simplicity assume that only a single flow, $g$, is used and it is parameterized by the vector $\theta$. Further, assume that the base measure, $p_Z$ is given and is parameterized by the vector $\phi$. Given a set of data observed from some complicated distribution, $D = \{y^{(i)}\}_{i=1}^{M}$, we can then perform likelihood-based estimation of the parameters $\theta = (\theta, \phi)$. The data likelihood in this case simply becomes

$$\log p(\mathcal{D}|\Theta) = \sum_{i=1}^{M} \log p_Y(y^{(i)}|\Theta = \sum_{i=1}^{M} \log p_Z(f(y^{(i)}|\theta)|\phi) + \log \left| \det Df(y^{(i)}|\theta) \right| \tag{3.6}$$

where the first term is the log likelihood of the sample under the base measure and the second term, sometimes called the log-determinant or volume correction, accounts for the change of volume induced by the transformation of the normalizing flows. During training, the parameters of the flow $(\theta)$ and of the base distribution $(\phi)$ are adjusted to maximize the log-likelihood.

As [8] notes, evaluating the likelihood of a distribution modelled by a normalizing flow requires computing $f$ (i.e., the normalizing direction), as well as its log determinant. The efficiency of these operations is particularly important during training where the likelihood is repeatedly computed. However, sampling from the distribution defined by the normalizing flow requires evaluating the inverse g (i.e., the generative direction). As a concequence, sampling performance is determined by the cost of the generative direction. Even though a flow must be theoretically invertible, computation of the inverse may be difficult in practice; hence, for density estimation it is common to model a flow in the normalizing direction (i.e., $f$). Finally, it is notable that while maximum likelihood estimation is often effective (and statistically efficient under certain conditions) other forms of estimation can and have been used with normalizing flows. In particular, adversarial losses can be used with normalizing flow models e.g., in Flow-GAN [60].

# Chapter 4

# Methodology

In this chapter we present and analyze the need as well as the architectural details of the model this thesis is based, SRFlow. We approach this by exploring three fundamental perspectives; first, by considering why such a model is needed and what the specific uses are, then presenting the key components/layers that make up the network and finally we review the data that it is trained on. We then analyze the structure and origins of the data used in the experiments with the model while specifying the preprocessing pipeline. We present a short description about the metrics that we evaluate the model on, as well as the specifications upon which the model ran. Finally we discuss any problems that were faced during this process and how they were handled.

## 4.1 Super Resolution

In most electronic imaging applications as [61] details, images with high resolution (HR) are desired and often required. HR means that pixel density within an image is high, and therefore an HR image can offer more details that may be critical in various applications. For example, HR medical images are very helpful for a doctor to make a correct diagnosis. It may be easier to distinguish an object from similar ones using HR satellite images, and the performance of pattern recognition in computer vision can be improved if an HR image is provided. The approach that uses signal processing techniques to obtain an HR image (or sequence) from observed multiple low-resolution (LR) images has been one of the most active research areas, and it is called super resolution (SR), (or HR) image reconstruction or simply resolution enhancement.

Super-resolution (SR) essentially refers to the task of restoring high resolution images from one or more low-resolution observations of the same scene. For example, when considering satellite data, there is often a trade-off to be made against higher resolution. Frequently data volume, obstruction, flyover time or bandwidth are limiting factors over the capture of the most information a sensor is capable of. However super resolution is inherently an ill-posed problem. Because of its nature, that is filling in missing data to create a more complete image,

there exists a unique solution whose value changes only slightly if initial conditions change slightly. This creates a challenge that current methods have not overcome.

SRFlow [36] proposes a flow-based method for super-resolution. Contrary to conventional methods, this approach learns the distribution of photo-realistic SR images given the input LR image. This explicitly accounts for the ill-posed nature of the SR problem and allows for the generation of diverse SR samples. Moreover, this enables techniques developement for image manipulation, exploiting the strong image posterior learned by SRFlow. In comprehensive experiments, this approach achieves improved results compared to state-of-the-art GAN-based approaches.

## 4.2 Architecture

The design of flow-layers $f_\theta^n$ proposed by the authors in [36], requires care in order to ensure a well-conditioned inverse and a tractable Jacobian determinant. It starts from the unconditional Glow architecture [62], which is itself based on the RealNVP [53]. The flow layers employed in these architectures can be made conditional in a straight-forward manner.

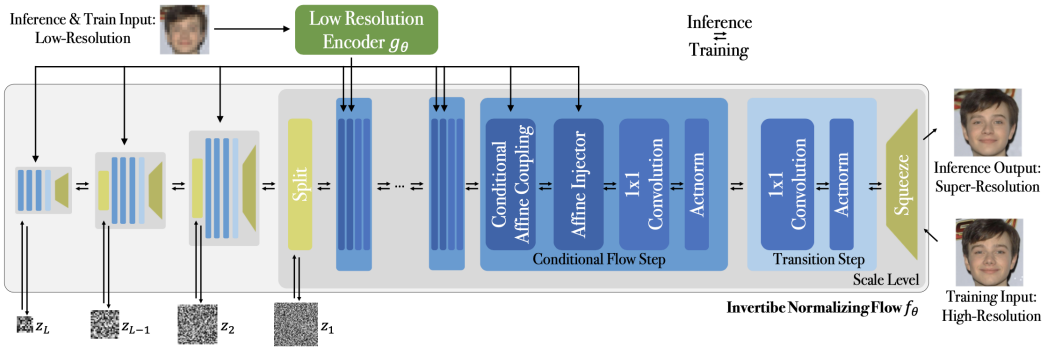



Figure 4.1: SRFlow's conditional normalizing flow architecture [36]

**Conditional Affine Coupling.** The affine coupling layer provides a simple and powerful strategy for constructing flow-layers that are easily invertible. It is trivially extended to the conditional setting as follows:

$$\begin{aligned} h_A^{n+1} &= h_A^n \\ h_B^{n+1} &= \exp(f_{\theta,s}^n(h_A^n; u)) \cdot h_B^n + f_{\theta,b}^n(h_A^n; u) \end{aligned} \tag{4.1}$$

Where, $h^n = (h_A^n, h_B^n)$ is a partition of the activation map in the channel dimension. Moreover, $u$ is the conditioning variable, set to the encoded LR image $u = g_\theta(x)$. Note that $f_{\theta,s}^n$ and $f_{\theta,b}^n$ represent *arbitrary* neural networks that generate the scaling and bias of $h_B^n$. The Jacobian of (4.1) is triangular, enabling the efficient computation of its log-determinant as

$$\sum_{ijk} f_{\theta,s}^n(h_A^n; u)_{ijk} \tag{4.2}$$

**Invertible $1 \times 1$ Convolution**. General convolutional layers are often intractable to invert or evaluate the determinant of. However, [21] demonstrated that a $1 \times 1$ convolution $h_{ij}^{n+1} = Wh_{ij}$ can be efficiently integrated since it acts on each spatial coordinate $(i, j)$ independently, which leads to a block-diagonal structure. Non-factorized formulation is used.

**Actnorm**. This provides a channel-wise normalization through a learned scaling and bias. This layer is kept in its standard un-conditional form.

**Squeeze**. It is important to process the activations at different scales in order to capture correlations and structures over larger distances. The squeeze layer provides an invertible means to halving the resolution of the activation map $h^n$ by reshaping each spatial $2 \times 2$ neighborhood into the channel dimension.

**Affine Injector**. To achieve more direct information transfer from the low-resolution image encoding $u = g_\theta(x)$ to the flow branch, the affine injector layer is introduced. In contrast to the conditional affine coupling layer, the affine injector layer directly affects all channels and spatial locations in the activation map $h^n$. This is achieved by predicting an element-wise scaling and bias using only the conditional encoding $u$:

$$h^{n+1} = \exp(f_{\theta,s}^n(u)) \cdot h^n + f_{\theta,b}^n(u) \tag{4.3}$$

Here, $f_{\theta,s}$ and $f_{\theta,b}$ can be any network. The inverse of (4.3) is trivially obtained as:

$$h^n = \exp(-f_{\theta,s}^n(u)) \cdot (h^{n+1} - f_{\theta,b}^n(u)) \tag{4.4}$$

and the log-determinant is given by:

$$\sum_{ijk} f_{\theta,s}^n(u)_{ijk} \tag{4.5}$$

Here, the sum ranges over all spatial $(i, j)$ and channel indices $k$.

**Loss**. The model [36] utilizes the Negative Log Likelihood (NLL). This essentially mean that the optimization happens by minimizing the NLL for each training sample $(x, y)$.

$$\mathcal{L}(\theta; x, y) = \log p_{y|x}(y|x, \theta) = -\log p_z(f_\theta(y; x)) - \log \left| \det \frac{\partial f_\theta}{\partial y}(y; x) \right| \tag{4.6}$$

## 4.3 Datasets

### 4.3.1 Datasets used in SRFlow

SRFlow [36] is primarily trained in faces. Specifically the primary dataset that the authors use is Celeb2A, a 200k RGB HR image dataset, containing 10k identities in a variety of poses and backgrounds.


Figure 4.2: Sample images from Celeb2A [37].

SRFlow is also trained with the DIV2K dataset, a 1k RGB HR image collection with very different scenes. DIV2K comes with corresponding LR images, obtained with a variety of methods to simulate realistic degradation for 2,3 and 4 scaling factors.

### 4.3.2 Datasets employed

As is illustrated in the concurrent work of [63] the dataset employed in this thesis consists of regional models of oceanic currents of the Mediterranean Sea Physics Analysis and Forecast 1 (referred to as MEDSEA) provided by the Copernicus Marine Environment Monitoring Service (CMEMS) as well as the Navy Coastal Ocean Model (NCOM) 2 for the Gulf of Mexico and Caribbean Sea (referred to as AMSEA), provided by the Fleet Numerical Meteorology and Oceanography Center (FNMOC). Details such as resolution and geographical longitude/latitude coordinates for the area used are illustrated in Table 5.1.

For the MEDSEA and AMSEA datasets, we used consecutive data ranging from October 2019 to February 2021. As described in the concurrent work of [63] we thinned the data such that we have data at fixed 6 hours intervals. This was done in order to have the same temporal sample rate in both datasets as well as to remove consecutive images with too much

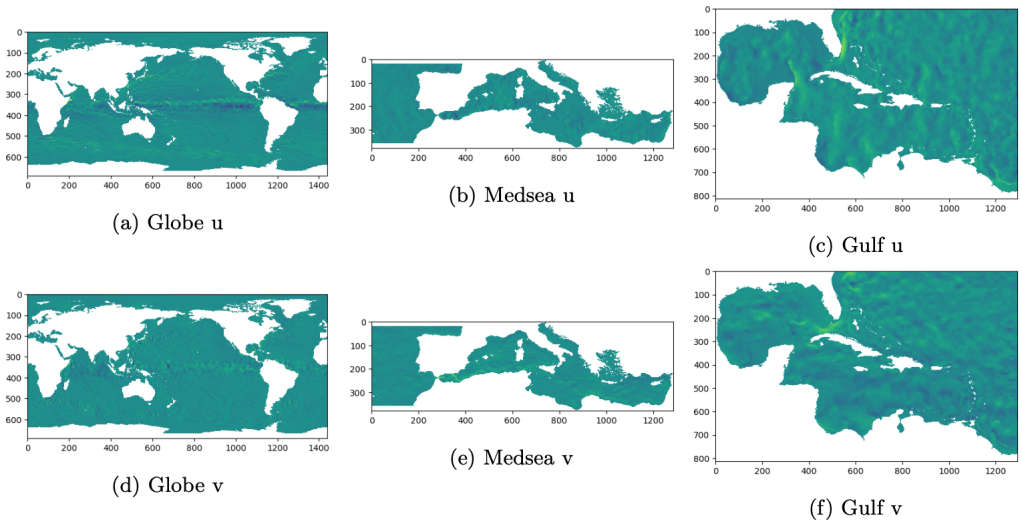
| Data | Area | Resolution | Lat range | Lon Range | Latitude (km) |
|---|---|---|---|---|---|
| Medsea | Mediterranean | 380x1287 | 30/46 | -17/36 | 4.62 |
| Amseas | Gulf of Mexico | 814x1294 | 5/32 | -98/-55 | 3.7 |

Table 4.1: Dataset description as stated in [63]

correlation.

As is standard, the dataset is divided into a training set, a validation set and a test set. We allocate 66% of the original dataset to the training set. The remaining third we further split in 30% validation and 70% test sets. The division of the dataset is performed in a rolling manner such that out of a three week period, the first two weeks are used for training, then 2 days are used for validation, and the last 5 days for testing. In this way, even though there will still be temporal dependencies between points in the different data sets, the inter-dependency between datasets are greatly reduced. Specifically, for the MEDSEA this yields 1262 separate time instances for the training set, 430 for the test set, and 164 for the validation set. Likewise, for the AMSEA this yields 1069 time instances, 380 for the test set and 136 for the validation set.

In this thesis, the aforementioned data points used as input (LR images) are generated using average pooling. As proposed in the concurrent work of [63], implicit in our choice lies the assumption of the average being the value represented in global models. A critical aspect of this process is the lack of information in locations corresponding to land. When averaging over an area containing land locations, we can either choose to ignore the entire area, labelling it as land, with the result of a significant loss of information, and land being over-represented in the LR input, or, conversely, to ignore the land values, choosing the average of the available values instead. We opted for the latter, this resulting in land disappearing from LR representations, in particular for large SR factors. This of course also leads to NaN values that we describe the method for handling in section 4.5.1.



(a) Globe u (b) Medsea u (c) Gulf u

(d) Globe v (e) Medsea v (f) Gulf v

Figure 4.3: Whole dataset representation [63].

## 4.4 Benchmarks

A variety of metrics are used in order to measure the error, similarity and noise of the produced SR. Below they are defined.

**MSE.** Comparing restoration or resolution results requires a measure of image quality. One commonly used measure is Mean-Squared Error [64]. The mean-squared error (MSE) between two images $g(x, y)$ and $\hat{g}(x, y)$ is:

$$e_{\text{MSE}} = \frac{1}{MN} \sum_{n=1}^{M} \sum_{m=1}^{N} \left( \hat{g}(n, m) - g(n, m) \right)^2 \tag{4.7}$$

where M and N are the number of rows and columns in the input images. The closer $e_{\text{MSE}}$ is to 0, the better the result.

**PSNR.** Another common comparison metric of image quality, derived from MSE, is the Peak Signal-to-Noise Ratio (PSNR) [64]. The Peak Signal-to-Noise Ratio between two images is:

$$e_{\text{PSNR}} = 10 \log_{10} \frac{R^2}{e_{\text{MSE}}} \tag{4.8}$$

where $R$ is the maximum fluctuation in the input image data type. For example, if the input image has a double-precision floating-point data type, then R is 1. If it has an 8-bit unsigned integer data type, R is 255, etc. PSNR is measured in decibels (dB). The higher the PSNR, the better the quality of the reconstructed image.

**SSIM.** Structural Similarity Index (SSIM) is used as a metric to measure the similarity between two given images. As expained in [65], unlike PSNR, SSIM is based on visible structures in the image. SSIM works by extracting 3 key features from an image, Luminance, Contrast and Structure. We explore how these key features are calculated and combined to create a final deterministic score.

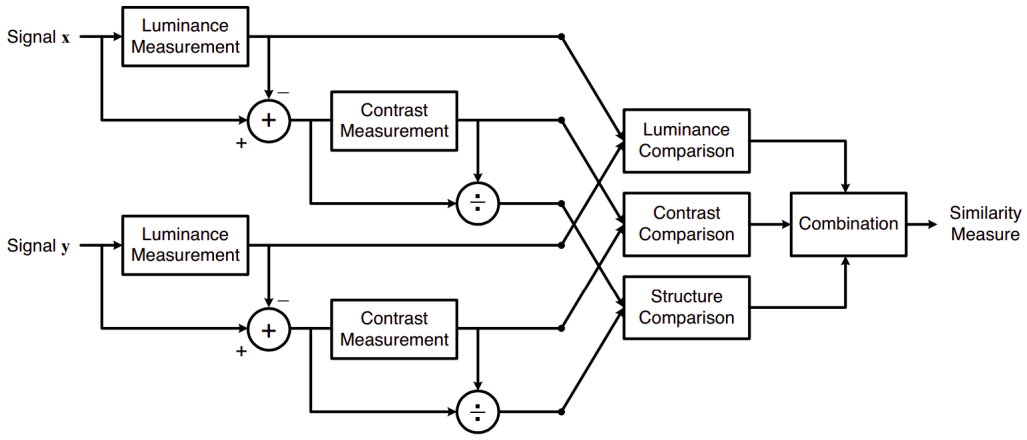



Figure 4.4: How the key features used by SSIM (Luminance, Contrast and Structure) are used. Signal X and Signal Y refer to the Reference and Sample Images [65].

Luminance is defined as such:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.9}$$

where $x_i$ is the i-th pixel value of image x with size N. Contrast is defined as such:

$$\sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2} \tag{4.10}$$

and structure is defined as such:

$$\frac{(x - \mu_x)}{\sigma_x} \tag{4.11}$$

Based on those definitions the comparison functions are:

Luminance comparison function:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.12}$$

where $\mu_x, \mu_y$ the luminance of images $x, y$ respectively and $C_1$ a stability constant defined by:

$$C_1 = (K_1 L)^2 \tag{4.13}$$

where L is the dynamic range of the pixel values and K a constant. Contrast comparison function:

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.14}$$

with $C_2$ defined similarly to $C_1$. Structure comparison function:

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{4.15}$$

and combining all the above [65] defines SSIM as:

$$\text{SSIM}(x, y) = \big[l(x, y)\big]^{\alpha} \cdot \big[c(x, y)\big]^{\beta} \cdot \big[s(x, y)\big]^{\gamma} \tag{4.16}$$

with $\alpha, \beta, \gamma$ being positive, non-zero constants denoting the importance of each component.

The resulting SSIM value is between $[-1, 1]$. A value of $1$ indicates high similarity while a value of $-1$ indicates the 2 given images are very different. Often these values are normalized to be in $[0, 1]$ range, where the low/high ends of the spectrum mean the same as before.

**LPIPS.** Learned Perceptual Image Patch Similarity (LPIPS) [66] (version='0.1'), is a recently proposed human perception metric to oppose PSNR and SSIM. It serves as a good quantitative evaluator, relating well with human perception of image quality, in contrast to

other metrics. It works by using deep features of visual networks (i.e. ImageNet-trained VGG features) as a training loss. Higher means further/more different. Lower means more similar.

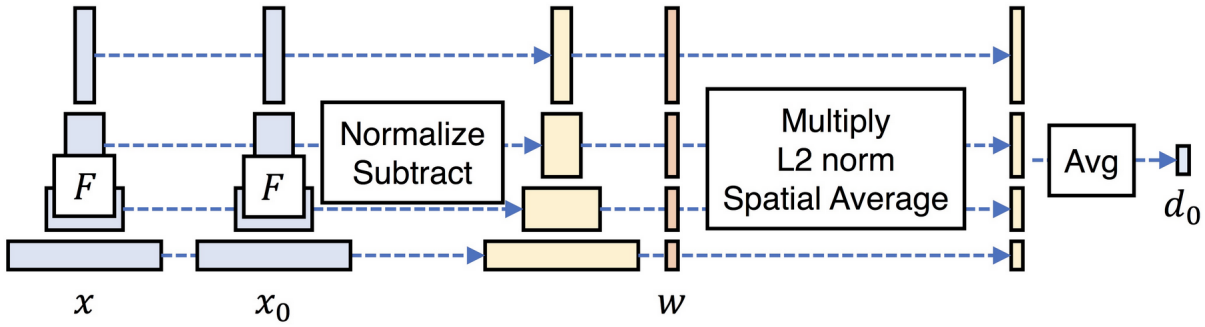



Figure 4.5: LPIPS architecture [66].

## 4.5 Modifications

### 4.5.1 Preprocessing

A new preprocessing code had to be created in order to accomodate the different dataset. Because of the totally different preprocessing pipeline of SRFlow, a new approach had to be taken. Instead of creating single files of LR and HR images, we created a pickle file for each train input, train target, validation input, validation target, test input and test target, according to the split we defined in Section 4.3.2. This was done of course after creating the LR images, after the normalization, randomization, NaN handling, noise addition, channel manipulation and scaling. Then the task would be handed over to a loading function in the training code that created the actual dataset to be used in memory by pytorch's dataloader that also seperated the batches and reshuffled the data. Below some of the issues as well as modifications to the model are presented.

**Exploding gradients.** An persistent issue faced and dealt with was exploding gradients. The solution to this had multiple parts. Firstly NaNs (representing land in the dataset) were dealt with, by replacing them with Gaussian noise. Secondly, with normalization to $[0, 1]$ we avoided outliers with very large or very small values. Thirdly, we lowered the learning rate to $1e-5$. Finally we added Gaussian noise to the target images (scaled and adjusted). Gradient clipping was also implemented in order to deal with exploding gradients, a solution also recommended by SRFlow's author, as this was a persistent issue even in the original model.

**Channel mismatch.** Because the data that we seek to train are represented with 2 channels (the uo and vo vectors respectively) and SRFlow's architecture is built to accept 3 channel data (mainly RGB) modifications had to be made to our dataset in order to fit the architecture. In the end 2 versions of the network were built; the first that accepts 3-channel

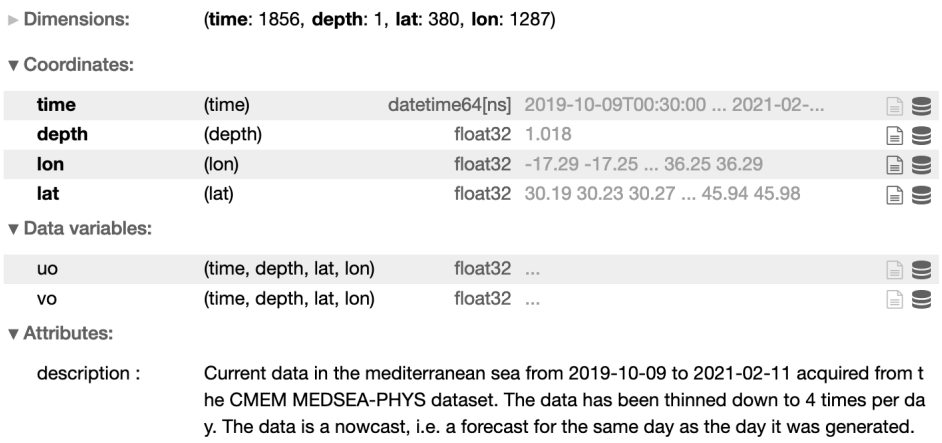



Figure 4.6: Data from oceanic currents of the Mediterranean Sea Physics Analysis and Forecast 1 description in xarray.

data and the second that accepts 2-channel data. Different methods such as appending a 3rd channel with singular values (zeros/ones) that were attempted did not yield good results, due to the non-convergence of singular value decomposition (SVD) internal method in the calculation of loss. A straightforward solution that did work was to append a copy of one original channel as the 3rd channel.

Due to modifications in SRFlows architecture, we also gained the ability to fit the original 2 channel data in the model, without the need for a 3rd channel. Still the results of the 3 channel method are presented, as they serve as a benchmark, are interesting to study due to the additional channel's effect. Additionaly the 3 channel results are easier to present as they resemble RGB channels and thus can be plotted and the effects seen. The 2 channel data have to be plotted 1 channel at a time,. because no suitable color standard exists that constists of 2 values.

**NaNs.** Another issue that had to be dealt with was the presence of NaNs in the dataset at coordinates that represent land. One choice contemplated as a solution was the application of an exemption mask in the calculation of loss/gradients such that these parts of the image would be excluded from calculation. It is important to note that if these values were left unchanged, the model would after some iterations face an exploding gradient problem that led to floating point errors (overflows). The method that was finally implemented was replacing the NaN values with small Gaussian noise. This solution was also suggested by the authors of SRFlow as an improvement over the original model published as a response to an issue. This holds, as the added noise, scaled and shifted to the data range, helps in the generalization of the model. The noise added wa standard normal $\epsilon \sim \mathcal{N}(0, 1)$, multiplied with $1e-5$ and the values outside of the range $[0, 1]$ cut off.

**Normalization.** Normalization was also applied as part of the preprocessing pipeline, scaling the dataset to the [0,1] range. This was motivated due to the presence of the RRDB modules, as CNNs work best with normalized data in that range [67]. This is done because

in a CNN setting, the ranges of the distributions of feature values are likely be different depending on the feature, and thus the learning rate multiplication would cause corrections in each dimension that would proportionally differ.

**Memory.** Memory issues also had to be dealt with due to the size of each data point as well as limitations in hardware. Preprocessing each batch, saving it and then handing it to the dataloader was contemplated but that solution proved slow and created unnecessarily large batch files. Instead, the whole input (LR) and target (HR) was encoded into pickle files for each train, validation and testing section, and subsequently fed into the dataloader to break into batches. Batch size needed also to be tuned as not to cause memory issues.

### 4.5.2 Architecture

Architecturally, changes also had to be made in order to fit the new data in the new network. For example a major change would be the size of the input HR image. This effected the size of filters as well as layers, so it had to be matched to the new input size. In order to fit 2-channel data in the model, the CNN filter size had to be changed such that the input and the output had the desirable channel size in the latent space.

### 4.5.3 Testing

Furthermore, new testing code had to be created in order to accomodate the different dataset. New measuring functions had to be implemented, due to the difference in dimensionality, as well as shape conversions that made the images to work on the metrics stated in Section 5.2. Also new plotting functions had to be implemented, due to for example the 2 channel data that did not fit any regular RGB (3 channel) approaches.

## 4.6 Training Details

The network is trained using the negative log-likelihood loss [Eq. 4.6]. Patches of 288x288 of the original images are used, scaled down according to the scaling factor $r = 4, 8, ....$ These patches generate training image pairs $(x, y)$ of size 8 batches of LR-HR images.

For the optimization step, Adam [68] is used with a learning rate of $5 \cdot 10^{-5}$, which is reduced to half an the 50%, 75%, 90% and 95% of the total training iterations.

The training is split into two parts, the warm-up and RRDB as well as full flow training. The goal of the warm-up phase is to increase efficiency by pre-training the LR encoder. Here the network makes use of the standard feed-forward SR architecture based on Risidual-in-Risidual Dense Blocks (RRDB) [69]. As SRFlow [36] mentions, the LR image that first gets encoded by a shared deep CNN $g_\theta$, which extracts a rich representation suitable for conditioning in all flow-layers, as detailed in [Section 4.2]. The warm-up phase is trained for 100k iterations,

using the $L_1$ loss:

$$L_1 = \sum_{i=1}^{N} |y - \hat{y}| \tag{4.17}$$

Subsequently, the full architecture is trained for another 100k iterations using the NLL loss [Eq. 4.6].

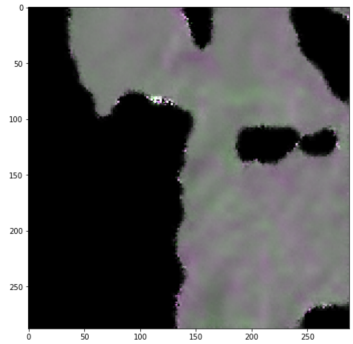
The training take place in the CyI's Cyclone cluster on a NVIDIA Corporation GV100GL [Tesla V100 SXM2 32GB] GPU. The training time for the full training period is 2 days, for 100k iterations for warm-up and RRDB training and another 100k iterations for the flow full network training. The training/validation/test split is 60% for training and the rest 40% is split 30% for validation and 70% for testing. The validation is applied every 500 iterations.

# Chapter 5

# Experimental Results

In this chapter we present the experiment results. We put forward the outcomes of model experimentation with different architectures and evaluation based on the benchmarks defined in . Additionally, the respective plots and tables are presented, showcasing visible pay-offs of the SR undertaking.

(a) Input/LR 36x36x3



(b) Output/SR 288x288x3

Figure 5.1: Realistic representation of size difference with scaling factor $r = 8$.

## 5.1 Results

Table 5.1 records the results of the models trained on MEDSEA for 200k iterations (100k warm-up [RRDB] and 100k full network) as well as the comparison between scaling factors. As is expected as the scaling factor increases, we expect to see a degradation in the results. It is also important to note for these experiments, regarding the datasets MEDSEA and AMSEA, that the former was primarily used for training and the latter for testing. This was done to maintain the region integrity and because MEDSEA contains more data points. Training on a combination dataset is explored in as future work.

From the results shown in Table 5.2, we can see that the addition of RRDB in the network does improve the results. This is done as mentioned in Section 4.6 in order to capture the

| Model info | MSE | PSNR | SSIM | LPIPS |
|---|---|---|---|---|
| 4x; 72x72x3 | 0.0040 | 24.4295 | 0.9193 | 0.1410 |
| 8x; 36x36x3 | 0.0067 | 22.2228 | 0.8832 | 0.2101 |

Table 5.1: Results of training on MEDSEA with different scaling factors.

| Model type | MSE | PSNR | SSIM | LPIPS |
|---|---|---|---|---|
| 4x; 72x72x3 to 288x288x3 | | | | |
| Flow | 0.0042 | 24.4038 | 0.9243 | 0.1282 |
| Flow & RRDB | 0.0040 | 24.4295 | 0.9193 | 0.1410 |
| 8x; 36x36x3 to 288x288x3 | | | | |
| Flow | 0.0073 | 21.8305 | 0.8846 | 0.1896 |
| Flow & RRDB | 0.0067 | 22.2228 | 0.8832 | 0.2101 |

Table 5.2: Results of training on MEDSEA with different scaling factors and architectures.

LR latent space encoding. As we can see in Table 5.2, in 4x 3-channel scaling, this offers a 4.76% increase in MSE, a 0.1% increase in PSNR, a 0.54% in SSIM and a 9.98% decrease in LPIPS. In 8x 3-channel scaling the effect in MSE is 8.21% decrease, 1.79% increase in PSNR, 0.15% decrease in SSIM and 10.81% increase in LPIPS.

In Figure 5.1 and Figure 5.2, we can see that visually the results are very significant. This is especially evident in Figure 5.6 where the size difference of the input LR image and the resulting SR output can be seen. These visual results of course are validated with the application appropriate image degradation metrics shown in Table 5.1 and Table 5.2. The black background that can be seen in the Figures of this chapter, represents the land.

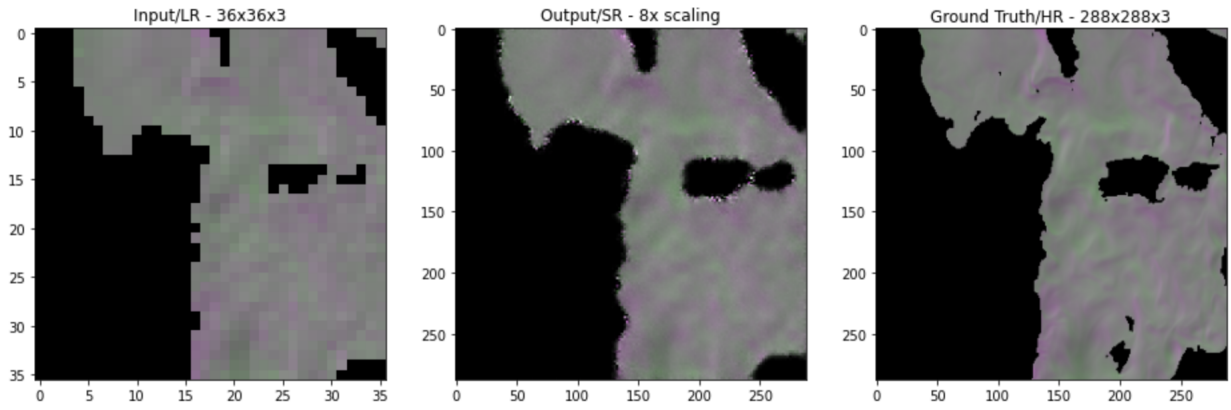



Figure 5.2: Results of training with 200k iterations (100k warm-up, 100k full) on 8x SR on MEDSEA.

In Figure 5.4 the resulting SR can be see for different temperatures (heat parameter). The temperature $\tau$ denotes the variance of the desired latent distribution that the model samples

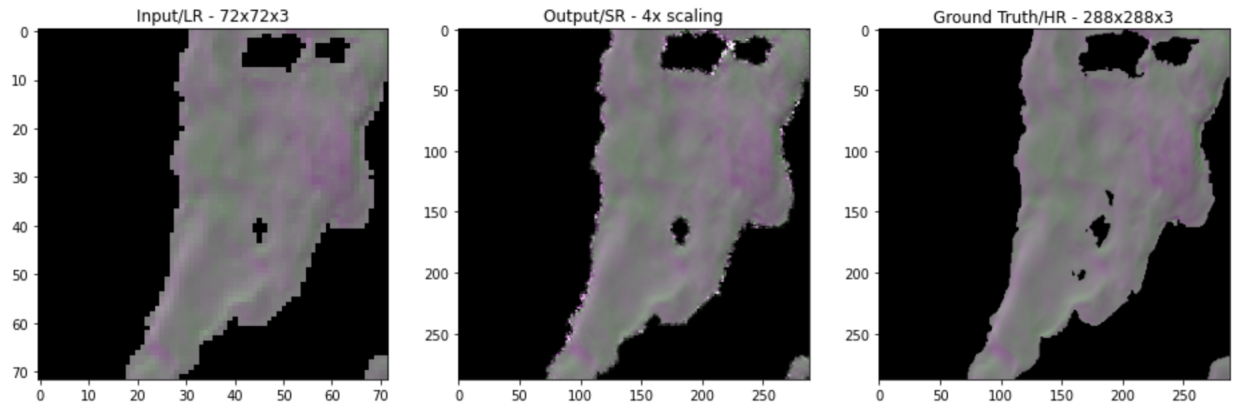



Figure 5.3: Results of training with 200k total iterations (100k warm-up, 100k full) on 4x SR on MEDSEA.

from, in order to create the SR result. As explained additionally in SRFlow, temperature controls the variance of the Gaussian latent variable used when sampling SR images as

$$y = f_\theta^{-1}(z; x), \quad z \sim \mathcal{N}(0, \tau) \tag{5.1}$$

A slightly reduced temperature $\tau < 1$, increases image quality. When further decreasing the temperature to $\tau = 0$, the sampling process becomes deterministic. A temperature $\tau = 0$ generates predictions with high fidelity, in terms of PSNR and SSIM. However, the results are blurry, as seen in Figure 5.4, explaining the poor perceptual quality (LPIPS) for this setting. In this example, we can see that the region representing Italy appears to have a lot more defined edges in the higher heat. Increasing the temperature leads to a drastic improvements in perceptual quality in terms of LPIPS distance. This is also clearly seen in the visual results in Figure 5.4 and Figure 5.5. Nonetheless higher accuracy, which is more useful in applications, rather than visually pleasing results sometimes comes at a cost. In the $\tau = 0.9$ plot in Figure 5.4 we can see some white values in the edge of land, in the south-eastern part of Sicily. This is caused by NaN values, when the model samples the latent space in order to capture the SR, with greater variance. According to SRFlow's authors this may be corrected by increasing the number of iterations.

As can be seen from Figure 5.2, as expected, the results are much better with 4x scaling. This is due to the fact that as the required scaling increases, less and less information is present in the LR image that will help recreate the HR representation.

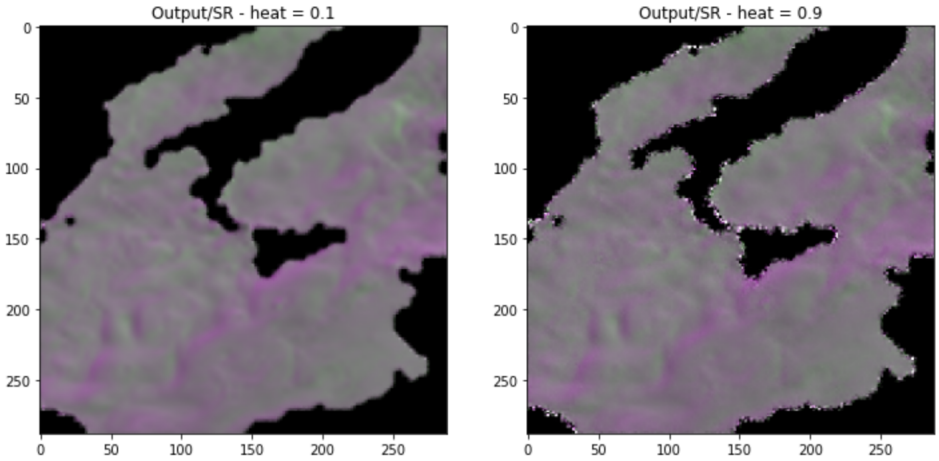



Figure 5.4: Heat (temperature $\tau$) effect comparison.

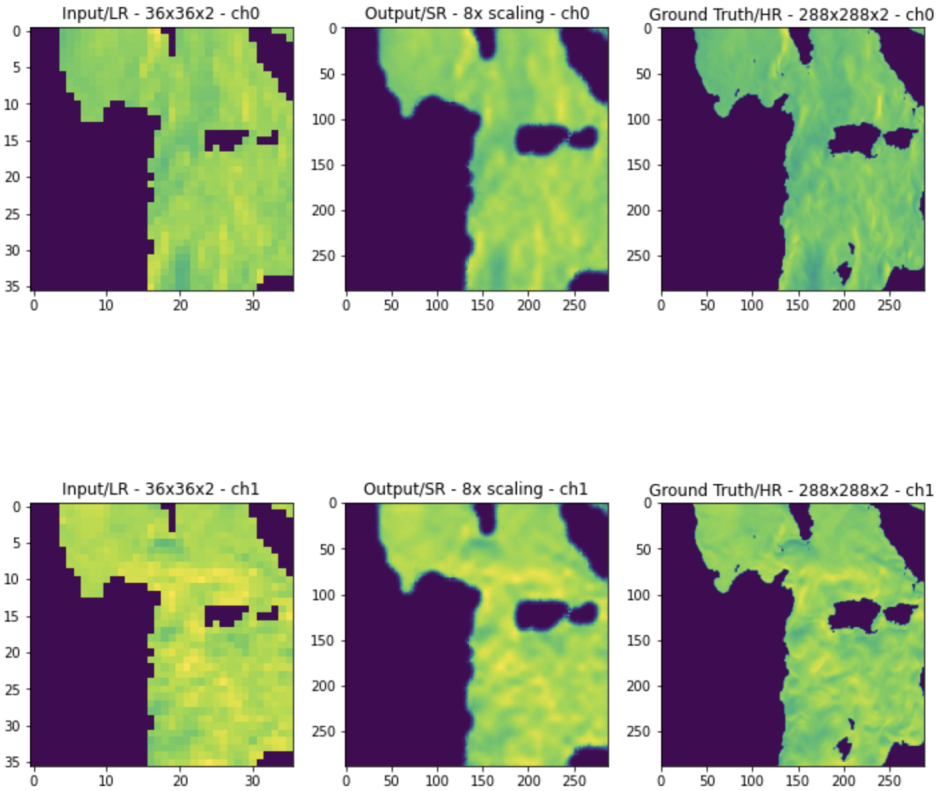



Figure 5.5: Results of training with 200k total iterations (100k warm-up, 100k full) on 8x SR on MEDSEA with 2 channel data.

From Figure 5.6 we can observe that we do not overfit. By observing the validation loss, in a cross-validation setting we can infer that due to the fact the *val_loss* keeps decreasing as *train_loss* is decreasing we have not reached the full potential of the network and we can still keep training without incurring the drawbacks of overfitting.

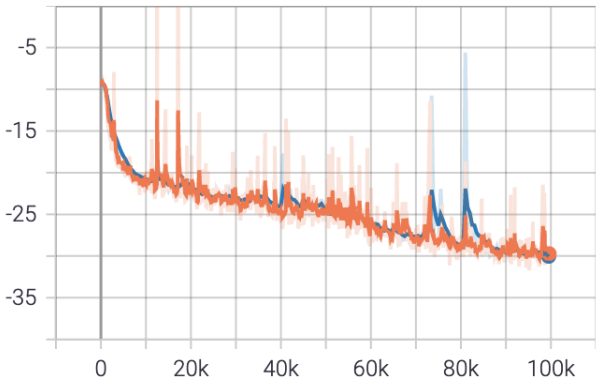



Figure 5.6: NLL of training (red line) as well as validation (blue line) for the 200k iteration run (100k warm-up, 100k full - only full visible) on 4x SR on MEDSEA with 3 channel data.

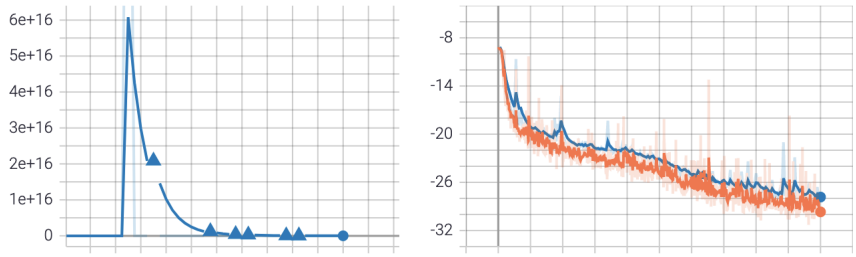



Figure 5.7: Plots of MSE (left) and NLL (right) (train and validation) with 200k total iterations (100k warm-up, 100k full) on 8x SR on MEDSEA with 3 channel data.

In Figure 5.5 we can see the results from the 2-channel architecture, with each triplet of plot representing an low resolution, super resolution and high resolution for channel 0 and channel 1.

From Figure 5.9, we can clearly see that the model generalizes. The network in trained on a completely different part of the world MEDSEA, with different oceanic current patterns, and yet the results when applying the pretrained model on AMSEA the SR work. This is significant because in DL in general generalization is a challenging goal to achieve.

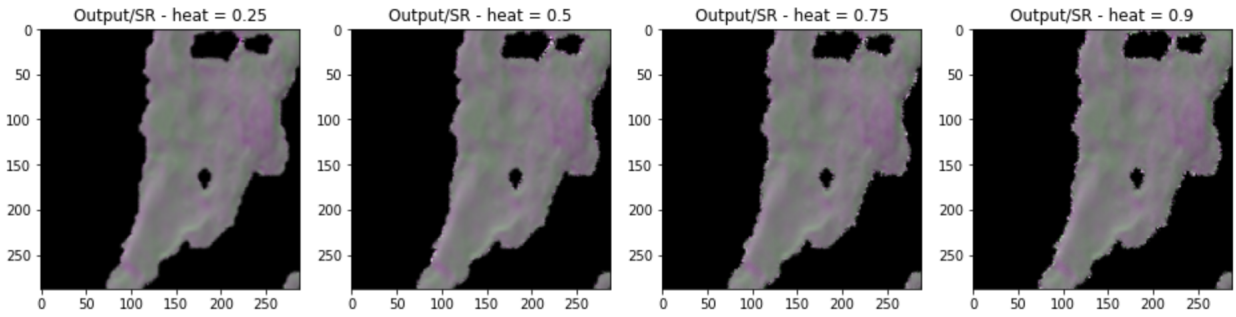



Figure 5.8: Results of training with 200k total iterations (100k warm-up, 100k full) on 4x SR on MEDSEA for different heat.

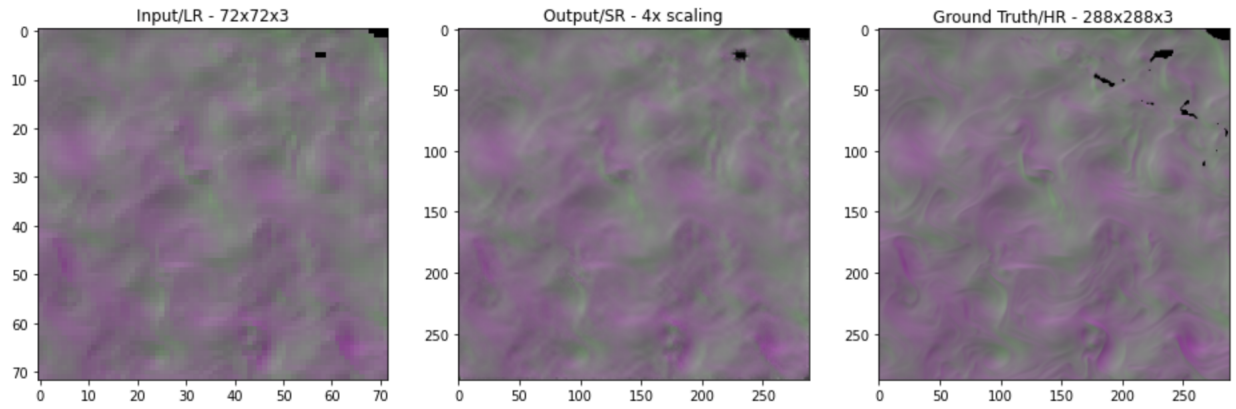



Figure 5.9: The results of a model trained on MEDSEA and tested on AMSEA for scaling factor $r = 4$.

# Chapter 6

# Conclusions & Future Work

Within applications such as ship routing and sea rescue operations, high resolution representations of oceanic current data are highly important, and thus techniques that reduce the computational complexity as well as the need for huge datasets in weather simulation are required. Super Resolution (SR) is such a technique, often employing statistical or learning methods. In this thesis, we tackled SR through deep learning and normalizing flows, having two main objectives; a lower complexity model than traditional weather simulations demand, and a reduction on the amount of required observations due to possible extrapolation. We adapted SRFlow – originally built for super resolution for faces – such that oceanic current data can be fitted. Subsequently, we assessed its performance in this particular setting. Our evaluation assessed the effect of flows added to convolutions as well as the effect of the scaling factor on the resulting super resolution image, against typical image degradation metrics. From the results that were discussed we can clearly see that this method for super resolution is indeed promising.

Regarding future work, as was discussed in the previous chapter (5), in the experiments performed the model did not yet reach it's full potential. If we observe the validation loss curve Figures 5.5 and 5.7, it is evident that accuracy can be improved and less MSE and higher PSNR can be achieved. Another improvement that can be pursued is to reduce the ration of training/validation/testing split such that the training dataset is larger.

Another item to be considered in followup work is the dataset. Comparing the 160k-image-sized dataset that SRFlow is using to achieve desireable results with 2000 of our images is lacking. A solution to this could be investigating the results of the combination MEDSEA & AMSEA datasets. Also in order to improve on training time due to the size of the models at hand, an implementation of distributed training could also be envisioned.

# Bibliography


[1] R. Keys. "Cubic convolution interpolation for digital image processing". *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.6 (1981), 1153–1160.

[2] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". *Neural Networks* 61 (2015), 85–117.

[3] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. "Accurate Image Super-Resolution Using Very Deep Convolutional Networks". *CoRR* abs/1511.04587 (2015). arXiv: 1511.04587.

[4] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". *CoRR* abs/1812.04948 (2018). arXiv: 1812.04948.

[5] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". *CoRR* abs/1912.04958 (2019). arXiv: 1912.04958.

[6] Wenbo Li et al. "Object-driven Text-to-Image Synthesis via Adversarial Training". *CoRR* abs/1902.10740 (2019). arXiv: 1902.10740.

[7] Yitong Li et al. "StoryGAN: A Sequential Conditional GAN for Story Visualization". *CoRR* abs/1812.02784 (2018). arXiv: 1812.02784.

[8] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (2020), 1–1.

[9] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press, 2009. ISBN: 9780262013192.

[10] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[11] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[12] Samuel R. Bowman et al. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, 632–642.

[13] Tim Salimans et al. "Improved Techniques for Training GANs". *CoRR* abs/1606.03498 (2016). arXiv: 1606.03498.

[14] Jonathan Ho et al. *Cascaded Diffusion Models for High Fidelity Image Generation*. 2021.

[15] Diederik P. Kingma and Prafulla Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. arXiv: 1807.03039 [stat.ML].

[16] Abdelrahman Abdelhamed, Marcus Brubaker, and Michael Brown. "Noise Flow: Noise Modeling With Conditional Normalizing Flows". In: 2019, 3165–3173.

[17] Manoj Kumar et al. "VideoFlow: A Flow-Based Generative Model for Video". *CoRR* abs/1903.01434 (2019). arXiv: 1903.01434.

[18] Sungwon Kim et al. "FloWaveNet : A Generative Flow for Raw Audio". *CoRR* abs/1811.02155 (2018). arXiv: 1811.02155.

[19] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. "WaveGlow: A Flow-based Generative Network for Speech Synthesis". *CoRR* abs/1811.00002 (2018). arXiv: 1811.00002.

[20] Kaushalya Madhawa et al. *GraphNVP: An Invertible Flow Model for Generating Molecular Graphs*. 2019. arXiv: 1905.11600 [stat.ML].

[21] Bogdan Mazoure et al. *Deep Reinforcement and InfoMax Learning*. 2020.

[22] Patrick Ward, Ariella Smofsky, and Avishek Bose. *Improving Exploration in Soft-Actor-Critic with Normalizing Flows Policies*. 2019.

[23] Danilo Jimenez Rezende et al. *Normalizing Flows on Tori and Spheres*. 2020. arXiv: 2002.02428 [stat.ML].

[24] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". *Foundations and Trends® in Machine Learning* 12.4 (2019), 307–392.

[25] Antonia Creswell et al. "Generative Adversarial Networks: An Overview". *IEEE Signal Processing Magazine* 35.1 (2018), 53–65.

[26] Ruohan Wang et al. *MAGAN: Margin Adaptation for Generative Adversarial Networks*. 2017. arXiv: 1704.03817 [cs.LG].

[27] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML].

[28] Laurent Dinh, David Krueger, and Yoshua Bengio. *NICE: Non-linear Independent Components Estimation*. 2015. arXiv: 1410.8516 [cs.LG].

[29] Esteban G. Tabak and Eric Vanden-Eijnden. "Density estimation by dual ascent of the log-likelihood". English (US). *Communications in Mathematical Sciences* 8.1 (2010), 217–233.

[30] E. G. Tabak and Cristina V. Turner. "A Family of Nonparametric Density Estimation Algorithms". *Communications on Pure and Applied Mathematics* 66.2 (2013), 145–164. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.21423.

[31] J. Agnelli et al. "Clustering and Classification through Normalizing Flows in Feature Space". *Multiscale Modeling & Simulation* 8 (2010), 1784–1802.

[32] Oren Rippel and Ryan Prescott Adams. *High-Dimensional Probability Estimation with Deep Density Models*. 2013. arXiv: 1302.5125 [stat.ML].

[33] Amanjot Singh and J. S. Sidhu. "Super Resolution Applications in Modern Digital Image Processing". *International Journal of Computer Applications* 150 (2016), 6–8.

[34] Rongcheng Han et al. "Recent Advances in Super-Resolution Fluorescence Imaging and Its Applications in Biology". *Journal of Genetics and Genomics* 40 (2013).

[35] Zhan Li et al. "Super resolution for astronomical observations". *Astrophysics and Space Science* 363 (2018).

[36] Andreas Lugmayr et al. *SRFlow: Learning the Super-Resolution Space with Normalizing Flow*. 2020. arXiv: 2006.14200 [cs.CV].

[37] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

[38] Vanya Cohen and Aaron Gokaslan. "OpenGPT-2: Open Language Models and Implications of Generated Text". *XRDS* 27.1 (2020), 26–30.

[39] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[40] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2019. arXiv: 1809.11096 [cs.LG].

[41] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. *Neural Discrete Representation Learning*. 2018. arXiv: 1711.00937 [cs.LG].

[42] Chunyuan Li et al. *Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space*. 2020. arXiv: 2004.04092 [cs.CL].

[43] Prafulla Dhariwal et al. *Jukebox: A Generative Model for Music*. 2020. arXiv: 2005.00341 [eess.AS].

[44] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[45] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[46] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG].

[47] Tero Karras et al. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018. arXiv: 1710.10196 [cs.NE].

[48] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: 2019, 4396–4405.

[49] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[50] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2020. arXiv: 1703.10593 [cs.CV].

[51] Christian Ledig et al. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. 2017. arXiv: 1609.04802 [cs.CV].

[52] Ayan Kumar Bhunia et al. "Texture synthesis guided deep hashing for texture image retrieval". *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019).

[53] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density estimation using Real NVP*. 2017. arXiv: 1605.08803 [cs.LG].

[54] Ioannis Gatopoulos, Maarten Stol, and Jakub M. Tomczak. *Super-resolution Variational Auto-Encoders*. 2020. arXiv: 2006.05218 [cs.LG].

[55] Zhi-Song Liu et al. *Unsupervised Real Image Super-Resolution via Generative Variational AutoEncoder*. 2020. arXiv: 2004.12811 [cs.CV].

[56] Seonyeong Park et al. "Autoencoder-Inspired Convolutional Network-Based Super-Resolution Method in MRI". *IEEE Journal of Translational Engineering in Health and Medicine* PP (2021), 1–1.

[57] Zeng Kun et al. "Coupled Deep Autoencoder for Single Image Super-Resolution". *IEEE Transactions on Cybernetics* 47 (2015), 1–11.

[58] V I Bogachev, A V Kolesnikov, and K V Medvedev. "Triangular transformations of measures". *Sbornik: Mathematics* 196.3 (2005), 309–335.

[59] C. Villani. "Topics in Optimal Transportation Theory". 58 (2003).

[60] Aditya Grover, Manik Dhar, and Stefano Ermon. "Flow-GAN: Bridging implicit and prescribed learning in generative models" (2017).

[61] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. "Super-resolution image reconstruction: a technical overview". *IEEE Signal Processing Magazine* 20.3 (2003), 21–36.

[62] Jian Cong et al. *Glow-WaveGAN: Learning Speech Representations from GAN-based Variational Auto-Encoder For High Fidelity Flow-based Speech Synthesis*. 2021. arXiv: 2106.10831 [eess.AS].

[63] Waqas Qazi Andrea Littardi Anders Hildeman and Mihalis Nicolaou. "Methods for super-resolution of oceanic currents and their ability for spatial generalization". *Unpublished manuscript* (2021).

[64] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X.

[65] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". *IEEE Transactions on Image Processing* 13.4 (2004), 600–612.

[66] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *CVPR*. 2018.

[67] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[68] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[69] Xintao Wang et al. *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks*. 2018. arXiv: 1809.00219 [cs.CV].

# .1 List of Abbeviations

| | |
|---|---|
| **SR** | **S**uper **R**esolution |
| **HR** | **H**igh **R**esolution |
| **LR** | **L**ow **R**esolution |
| **CPU** | **C**entral **P**rocessing **U**nit |
| **NF** | **N**ormalizing **F**low |
| **CNN** | **C**onvolution **N**eural **N**etwork |
| **DNN** | **D**eep **N**eural **N**etwork |
| **ANN** | **A**rtificial **N**eural **N**etwork |
| **RNN** | **R**ecurrent **N**eural **N**etwork |
| **GAN** | **G**enerative **A**dversarial **N**etwork |
| **VAE** | **V**ariational **A**uto **E**encoder |
| **AE** | **A**uto **E**encoder |
| **DL** | **D**eep **L**earning |
| **CAP** | **C**redit **A**ssignment **P**ath |
| **CT** | **C**omputerized **T**omography |
| **MRI** | **M**agnetic **R**esonance **I**maging |
| **NaN** | **N**ot **A** **N**umber |
| **MSE** | **M**ean **S**quared **E**rror |
| **PSNR** | **P**eak **S**signal **N**oise **R**atio |
| **SSIM** | **S**tructural **S**imilarity **I**ndex |
| **LPIPS** | **L**earned **P**erceptual **I**mage **P**atch **S**imilarity |
| **RRDB** | **R**isidual in **R**isidual **D**ense **B**locks |
| **CyI** | **C**yprus **I**nstitute |
| **GPU** | **G**raphics **P**rocessing **U**nit |
| **RGB** | **R**ed **G**reen **B**lue |
| **SVD** | **S**ingular **V**alue **D**ecomposition |
| **LR** | **L**earning **R**ate |
| **CDA** | **C**oupled **D**eep **A**utoencoder |
| **MEDSEA** | **M**editerranean **S**ea |
| **AMSEA** | **A**merican **S**eas |
| **CMEMS** | **C**opernicus **M**arine **E**nvironment **M**onitoring **S**ervice |
| **NCOM** | **N**avy **C**oastal **O**cean **M**odel |
| **FNMOC** | **F**leet **N**umerical **M**eteorology and **O**ceanography **C**enter |