UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Reversible Networks for Memory-efficient Image-to-Image Translation in 3D Medical Imaging

by

Tycho F.A. van der Ouderaa
10642757

January 18, 2019

36 EC
Mar 2018 -Jan 2019

Supervisor:
Prof. Dr. Bram van Ginneken
Daniel E. Worrall

Assessor:
Prof. Dr. Max Welling

University of Amsterdam
Diagnostic Image Analysis Group, Radboud University Medical Center

Acknowledgements

It has been a great pleasure to work on this master thesis. I would especially like to thank my two supervisors, Prof. Dr. Bram van Ginneken of the Diagnostic Image Analysis Group (DIAG) at the Radboud University Medical Center and Daniel E. Worrall of the University of Amsterdam for their great support, enthusiasm and guidance throughout the entire thesis project.

I would like to thank Prof. Dr. Bram van Ginneken for allowing me to do this project at the DIAG. His expertise bridges artificial intelligence and medical image analysis, which together with his pragmatic approach, was always very helpful. During meetings, he steered the project in the right direction, when needed.

I have also been blessed by having Daniel E. Worrall as superviser who encouraged me in an active way. He contributed with ideas and further insights, while consistently allowing the research to be my own work. Also, he challenged me to take and accept risks during the research which contributed to a better result.

Finally, I must express my gratitude to my parents for providing me with unfailing support and encouragement during my student time and my dear friends for making these years such a great and very enjoyable time. This accomplishment would not have been possible without them.

Tycho van der Ouderaa

Abstract

The Pix2pix [24] and CycleGAN [52] losses have vastly improved the qualitative and quantitative visual quality of results in image-to-image translation tasks. We extend this framework by exploring approximately invertible architectures which are well suited to these losses. These architectures are approximately invertible by design and thus partially satisfy cycle-consistency before training even begins. Furthermore, since invertible architectures have constant memory complexity in depth, these models can be built arbitrarily deep. We are able to demonstrate superior quantitative output on the Cityscapes and Maps datasets.

Additionally, we show that the model allows us to perform several memory-intensive medical imaging tasks, including a super-resolution problem on 3D MRI brain volumes. We also demonstrate that our model can perform a 3D domain-adaptation and 3D super-resolution task on chest CT volumes. By doing this, we provide a proof-of-principle for using reversible networks to create a model capable of pre-processing 3D CT scans to high resolution with a standardized appearance.

Contents

1 Introduction

Computer vision was once considered to span a great many disparate problems, such as superresolution [15], colorization [10], denoising and inpainting [49], or style transfer [17]. Some of these challenges border on computer graphics (*e.g.*style transfer), while others are more closely related to numerical problems in the sciences (*e.g.*superresolution of medical images [44]). With the new advances of modern machine learning, many of these tasks have been unified under the term of *image-to-image translation* [24].

Mathematically, given two image domains X and Y, the task is to find or learn a mapping $F : X \to Y$, based on either paired examples $\{(x_i, y_i)\}$ or unpaired examples $\{x_i\} \cup \{y_j\}$. Let's take the example of image superresolution. Here X may represent the space of low-resolution images, and Y would represent the corresponding space of high-resolution images. We might equivalently seek to learn a mapping $G : Y \to X$. To learn both F and G it would seem sufficient to use the standard supervised learning techniques on offer, using convolutional neural networks (CNNs) for F and G. For this, we require paired training data and a loss function ℓ to measure performance. In the absence of paired training data, we can instead exploit the reciprocal relationship between F and G. Note how we expect the compositions $G \circ F \simeq \mathrm{Id}$ and $F \circ G \simeq \mathrm{Id}$, where Id is the identity. This property is known as *cycle-consistency* [52]. The unpaired training objective is then to minimize $\ell(G \circ F(x), x)$ or $\ell(F \circ G(y), y)$ with respect to F and G, across the whole training set. Notice how in both of these expressions, we never require explicit pairs (x_i, y_i). Naturally, in superresolution exact equality to the identity is impossible, because the upsampling task F is one-to-many, and the downsampling task G is many-to-one.

The problem with the cycle-consistency technique is that while we can insert whatever F and whatever G we deem appropriate into the model, we avoid making use of the fact that F and G are approximate inverses of one another. In this work, we consider constructing F and G as approximate inverses, *by design*. This is not a replacement to cycle-consistency, but an adjunct to it. A key benefit of this is that we need not have a separate $X \to Y$ and $Y \to X$ mapping, but just a single $X \to Y$ model, which we can run in reverse to approximate $Y \to X$. Furthermore, note by explicitly weight-tying the $X \to Y$ and $Y \to X$ models, we can see that training in the $X \to Y$ direction will also train the reverse $Y \to X$ direction, which does not necessarily occur with separate models. Lastly, there is also a computational benefit that invertible networks are very memory-efficient [18]; intermediate activations do not need to be stored to perform backpropagation. As a result, invertible networks can be built arbitrarily deep, while using a fixed memory-budget—this is relevant because recent work has suggested a trend of wider and deeper networs performing better in image generation tasks [6]. Furthermore, this enables dense pixel-wise translation models to be shifted to memory-intensive arenas, such as 3D (see Section 6.2 for our experiements on dense MRI superresolution).

Our results indicate that by using invertible networks as the central workhorse in a paired or unpaired image-to-image translation model such as Pix2pix [24] or CycleGAN [52], we can not only reduce memory overhead, but also increase fidelity of the output. We demonstrate this on the Cityscapes and Maps datasets in 2D and on a diffusion tensor image MRI dataset for the 3D scenario (see Section 6).

Additionally, we evaluate applicability of our method by addressing two tasks involving CT imaging of the chest. Recent scanners allow for high-resolution scans with isotropic voxel sizes around or below 0.5 mm^3 and require the reconstruction of image matrices of 1024 by 1024, contrary to the traditionally used matrices of 512 by 512 pixels. In these high-resolution scans, fine parenchymal details such as small airway walls, vasculature and lesion texture, are better visible. At the same time, the vast majority of scans available today for training networks consist of 512 matrices and often thicker slices, around 2mm. In addition, a wide variety of CT reconstruction kernels are used in practice, from more noisy high frequency kernels to smoother (soft kernels), and both traditional filtered backprojection as well as iterative reconstruction methods, varying between scanner vendors, are used. To produce robust and accurate image processing results, it is therefore desirable to pre-process chest CT scans to a standardized high resolution, and to remove the structural differences resulting from the use of different reconstruction algorithms. In this work, we address both the tasks of super-resolution and domain adaptation, to remove kernel differences. We aim to provide a proof-of-principle that we can use reversible networks to create a model capable of pre-processing CT scans to high resolution with a standardized appearance.

1.1 Thesis Outline

This thesis is structured into the following parts.

In Chapter 2, **Related Literature**, we recap the basics behind Generative Adversarial Networks (GANs), cycle-consistency, and re-versible/invertible networks.

In Chapter 3, **Method**, we introduce our model, called RevGAN, as a general image-to-image translation method using reversible neural networks for paired and unpaired data. We explain the reasoning behind the model and provide the loss functions.

In Chapter 4, **Implementation**, we give a more detailed description of the model required for the implementation and reproduction of the experiments. This includes network architectures, optimization methods and hyper-parameters used in this study.

In Chapter 5, **Datasets**, we introduce the 2D and 3D datasets used in the experimental section of this study.

In Chapter 6, **Results**, we present and analyze the outcomes of the experiments accompanied with analysis.

In Chapter 7, **Limitations and Discussion**, we describe limitations of this study and discuss our findings.

In Chapter 8, **Negative Results**, we state the negative results ranging from novel ideas that did not have a desired outcome to engineering attempts that did not improve performance.

In Chapter 9, **Conclusion**, we summarize our findings.

2 Related Literature

In this section, we recap the basics behind generative adversarial networks, image-to-image translation, cycle-consistency and invertible/reversible neural networks.

2.1 Generative Adversarial Networks

Generative adversarial networks (GANs) [19] enjoy huge success in tasks such as image generation [6], image interpolation [26], and image re-editing [39]. They consist of two components, a generator $F : Z \to Y$ mapping random noise $z \in Z$ to images $y \in Y$ and a discriminator $D : Y \to [0, 1]$ mapping images $y \in Y$ to probabilities. Given a set of training images $\{y_1, y_2, ...\}$, the generator produces 'fake' images $y_* = F(z), z \sim p(z)$, where $p(z)$ is a simple distribution such as a standard Gaussian, and the discriminator tries to predict the probability that the image was from the true image distribution. For training, an *adversarial loss* L_{GAN} is defined:

$$L_{\text{GAN}}(F, D) = \mathbb{E}_y \log D(y) + \mathbb{E}_z \log(1 - D(F(z))) \tag{1}$$

This loss is trained using a minimax regime where intuitively we encourage the generator to fool the discriminator, while also training the discriminator to guess whether the generator created an image or not. Mathematically this game [19] is

$$F^* = \arg \min_F \max_D L_{\text{GAN}}(F, D). \tag{2}$$

At test time, the discriminator is discarded and the trained generator is used to hallucinate fake images from the same distribution [2] as the training set. The generator can be conditioned on an input image as well. This setup is called a *conditional GAN* [35].

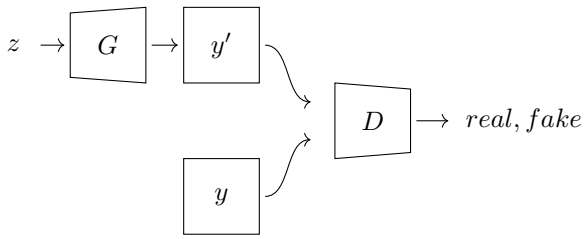


Figure 1: Illustration of typical Generative Adversarial Network: Generator G tries to generates an image y' from a random vector z that will be classified as real. Discriminator tries to classify a real sample y as real and a generated sample y' as fake.

2.2 Image-to-image Translation

In a standard (paired) image-to-image translation problem [24], we seek to learn the mapping $F : X \to Y$, where X and Y are corresponding spaces of images. It is natural to model F with a convolutional neural network (CNN). To train this CNN we minimize a loss function

$$L(F) = \frac{1}{n} \sum_{i=1}^{n} \ell(F(x_i), y_i) \tag{3}$$

where ℓ is a loss function defined in the pixel-space between the prediction $F(x_i)$ and the target y_i. Traditional image-to-image translation tasks relying on pixel-level loss functions are hampered by the fact that these losses do not typically account for inter-pixel correlations [51], for instance, L_1-losses treat each pixel as independent.

2.2.1 Pix2pix

GANs do not apply the loss per-pixel and can, therefore, account for inter-pixel correlational structures. GANs can be co-opted for image-to-image translation by adding the adversarial loss on top of a standard pixel-level L_1 loss function. This was first performed in the Pix2pix model [24], which is for paired image-to-image translation problems. Pix2pix replaces F with a conditional generator $F : X \times Z \to Y$, where Z is the domain of the random noise. The random noise vector enforces the output of the network to be stochastic and thereby allows the network to learn the distribution of output images. Unfortunately, the generator networks often learns to ignore the random noise and as a result does not learn a stochastic output. Although substituting the random noise vector by dropout in multiple layers may partially solve this issue, it is not a perfect solution and therefore this problem remains an important open research topic. In practice, as well as in this study, we usually ignore the additional noise input [52].

The model combines a L_1-loss that enforces the model to map images to the paired translations in a supervised manner with an adversarial loss that enforces the model to adopt the style of the target domain. The loss is

$$F^* = \arg \min_F \max_D L_{\text{cGAN}}(F, D) + \lambda L_{L1}(F) \tag{4}$$

where

$$L_{L1}(F) = \mathbb{E}_{x,y} \|y - F(x)\|_1 \tag{5}$$

$$L_{\text{cGAN}}(F, D) = \mathbb{E}_x \left[\log D(x) + \log(1 - D(F(x))\right]. \tag{6}$$

and λ is a tuneable hyperparameter that determines the relative importance between both losses. It is typically set in the range $10 - 100$ [24].

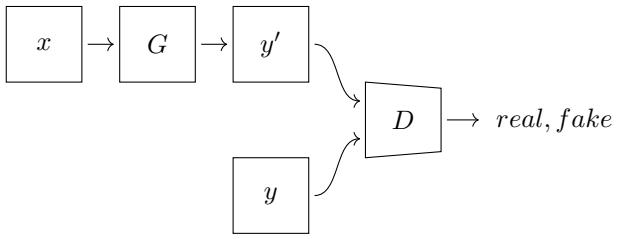


Figure 2: Illustration of a Pix2pix model: Generator G tries to generates an image y' from a random vector z and an image x that will be classified as real and close (L1-distance) to x. Discriminator tries to classify a real sample y as real and a generated sample y' as fake.

2.2.2 CycleGAN

The CycleGAN model was proposed as an alternative to Pix2pix for unpaired domains [52]. The mehtod has been applied within medical imaging [50], for instance to learn the mapping between CT and MRI volumes [48]. CycleGAN uses two generators F and G for the respective mappings between the two domains X and Y (so, $F : X \to Y$ and $G : Y \to X$), and two discriminators $D_X : X \to [0, 1]$ and $D_Y : Y \to [0, 1]$ trained to distinguish real and generated images in both domains. Since there are no image pairings between domains, we cannot invoke the Pix2pix loss and instead CycleGAN uses a separate *cycle-consistency loss* that penalizes the distances $L_{\text{cycle}}(G, F, x) = \|G \circ F(x) - x\|_1$ and $L_{\text{cycle}}(F, G, y) = \|F \circ G(y) - y\|_1$ across the training set. This encourages that the mappings F and G are loose inverses of one another. This allows the model to train on unpaired data. The total loss is

$$\begin{aligned} L_{\text{cycleGAN}} = & L_{\text{cGAN}}(F, D_Y) + L_{\text{cGAN}}(G, D_X) \\ & + \mathbb{E}_x L_{\text{cycle}}(G, F, x) + \mathbb{E}_y L_{\text{cycle}}(F, G, y). \end{aligned} \tag{7}$$

Given that F and G are loose inverses of one another, it seems wasteful to use separate models to model each. In this work, we model F and G as approximate inverses of one another. For this, we make use of the new area of invertible neural networks.

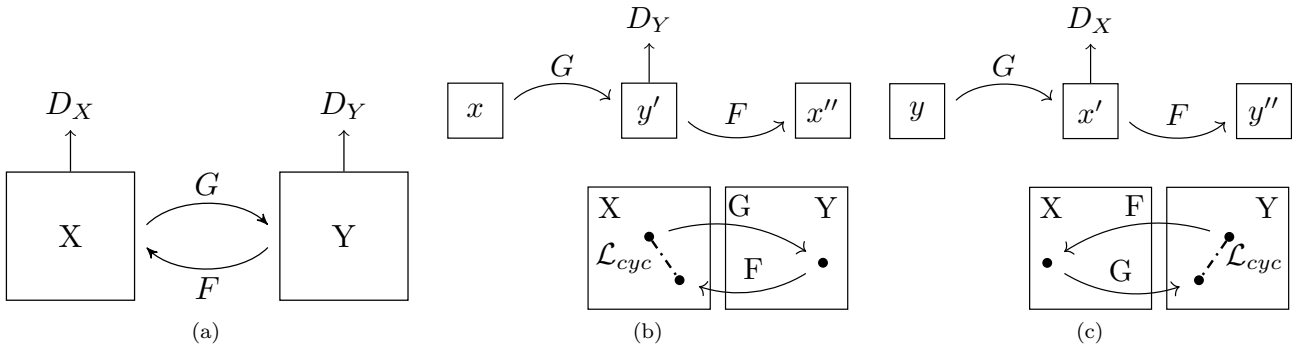


Figure 3: Illustrations of mappings, functions and domains in the CycleGAN model. (a) The mapping functions $G : X \to Y$ and $F : Y \to X$ and their corresponding discriminators D_X and D_Y. (b) Forward cycle-consistency loss: $x \to G(x) \to F(G(x)) = x'' \approx x$. (c) Backward cycle-consistency loss: $y \to F(y) \to G(F(y)) = y'' \approx y$

2.3 Reversible Neural Architectures

In recent years, several studies have proposed invertible neural networks (INNs) in the context of normalizing flow-based methods [40] [29]. It has been shown that INNs are capable of generating high quality images [28], perform image classification without information loss in the hidden layers [25] and analyzing inverse problems [1]. Most of the work on INNs, including this study, heavily relies upon the transformations introduced in NICE [13] later extended in RealNVP [14]. Although INNs share interesting properties they remain relatively unexplored.

Additive Coupling In our model, we obtain an invertible residual layer, as used in [18], using a technique called *additive coupling* [13]. As explained in [28], additive coupling is a special case of affine coupling introduced by [13]. The main idea is to split an input x (typically over the channel dimension) into (x_1, x_2) and then transform them using arbitrary complex functions $\texttt{NN}_1$ and $\texttt{NN}_2$ (such as a ReLU-MLPs) in the form (left):

$$\begin{aligned} y_1 &= x_1 + \texttt{NN}_1(x_2) & \qquad x_1 &= y_1 - \texttt{NN}_1(x_2) \\ y_2 &= x_2 + \texttt{NN}_2(y_1) & \qquad x_2 &= y_2 - \texttt{NN}_2(y_1). \end{aligned} \tag{8}$$

The inverse mappings can be seen on the right. Figure 4 shows a schematic of these equations.

Figure 4: Schematic of the forward pass (a) and inverse pass (b) of a residual block in a reversible residual layer. Note, functions $\texttt{NN}_1$ and $\texttt{NN}_2$ can be any volume-preserving transformation, such as a sequence of padded convolutions, normalization layers and non-linearities.

Memory efficiency Interestingly, invertible residual layers are very memory-efficient because intermediate activations do not have to be stored to perform backpropagation [18]. During the backward pass, input activations that are required for gradient calculations can be (re-)computed from the output activations because the inverse function is accessible. This results in a constant spatial complexity ($\mathcal{O}(1)$) in terms of layer depth (see Table 1). Effectively, we trade some additional computation time to obtain a constant spatial complexity ($\mathcal{O}(1)$) in terms of layer depth.

Technique	Spatial Complexity (Activations)	Computational Complexity
Naive	$\mathcal{O}(L)$	$\mathcal{O}(L)$
Checkpointing [32]	$\mathcal{O}(\sqrt{L})$	$\mathcal{O}(L)$
Recursive Checkpointing [9]	$\mathcal{O}(\log L)$	$\mathcal{O}(L \log L)$
Additive Coupling [18]	$\mathcal{O}(1)$	$\mathcal{O}(L)$

Table 1: Comparison of Spatial and Computational Complexity copied from [18]. L denotes number of residual layers. Notice how the spatial complexity of additive coupling is $\mathcal{O}(1)$ versus $\mathcal{O}(L)$ for a naive implementation.

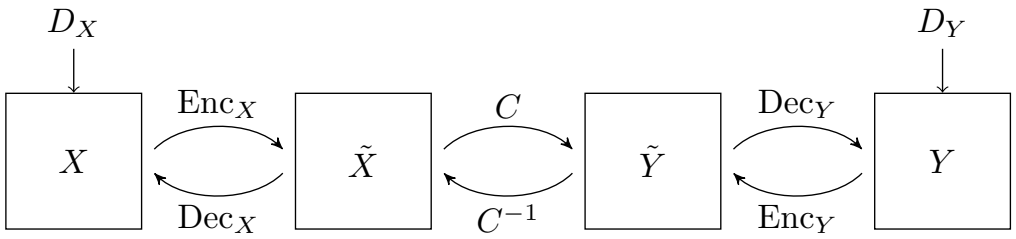


Figure 5: Schematic of our RevGAN model. Between the low-dimensional image spaces X and Y and their corresponding high-dimensional feature spaces $\tilde{X}$ and $\tilde{Y}$ we place non-invertible encoder and decoder networks $\text{Enc}_X, \text{Dec}_X, \text{Enc}_Y$ and Dec_Y. The feature spaces $\tilde{X}$ and $\tilde{Y}$ are of the same dimension, and between them we place an invertible core network C. We also attach to each image space, X and Y a domain-specific discriminator, which is used for training with the adversarial loss.

3 Method

Our goal is to create a memory-efficient image-to-image translation model, which is approximately invertible *by design*. Below we describe the basic outline of our approach of how to create an approximately-invertible model, which can be inserted into the existing Pix2pix and CycleGAN frameworks. We call our model *RevGAN*.

3.1 Lifting and Projection

In general, image-to-image translation tasks are not one-to-one. As such, a fully invertible treatment is undesirable, and sometimes in the case of dimensionality mismatches, impossible. Furthermore, it appears that the high-dimensional, overcomplete representations used by most modern networks lead to faster training [36] and better all-round performance [6]. We therefore split the forward $F : X \to Y$ and backward $G : Y \to X$ mappings into three components. With each domain, X and Y, we associate a high-dimensional feature space $\tilde{X}$ and $\tilde{Y}$, respectively. There are individual, non-invertible mappings between each image space and its corresponding high-dimensional feature-space; for example, for image space X we have $\text{Enc}_X : X \to \tilde{X}$ and $\text{Dec}_X : \tilde{X} \to X$. Enc_X *lifts* the image into a higher dimensionality space and Dec_X *projects* the image back down into the low-dimensional image space. We have used the terms *encode* and *decode* in place of 'lifting' and 'projection' to stay in line with the deep learning literature.

3.2 Invertible Core

Between the feature spaces, we then place an *invertible core* $C : \tilde{X} \to \tilde{Y}$, so the full mappings are

$$F = \text{Dec}_Y \circ C \circ \text{Enc}_X \tag{9}$$

$$G = \text{Dec}_X \circ C^{-1} \circ \text{Enc}_Y. \tag{10}$$

For the invertible cores we use invertible residual networks based on additive coupling as in [18]. The full mappings F and G will only truly be inverses if $\text{Enc}_X \circ \text{Dec}_X = \text{Id}$ and $\text{Enc}_Y \circ \text{Dec}_Y = \text{Id}$, which cannot be true, since the image spaces are lower dimensional than the feature spaces. Instead, these units are trained to be approximately invertible pairs via the end-to-end cycle-consistency loss. Since the encoder and decoder are not necessarily invertible they can consist of non-invertible operations, such as pooling and strided convolutions.

Because both the core C and its inverse C^{-1} are differentiable functions (with shared parameters), both functions can both occur in the forward-propagation pass and are trained simultaneously. Indeed, training C will also train C^{-1} and vice versa. The invertible core essentially weight-ties in the $X \to Y$ and $Y \to X$ directions.

Given that we use the cycle-consistency loss it may be asked, why do we go to the trouble of including an invertible network? The reason is two-fold: firstly, while image-to-image translation is not a bijective task, it is close to bijective. A lot of the visual information in an image x should reappear in its paired image y, and by symmetry a lot of the visual information in the image y should appear in x. It thus seems sensible that the networks F and G should be at least initialized, if not loosely coupled to be weak inverses of one another. If the constraint of bijection is too high, then the models can learn to diverge from bijection via the non-invertible encoders and decoders. Secondly, there is a potent argument for using memory efficient networks in these memory-expensive, dense, pixel-wise regression tasks. The use of two separate reversible networks is indeed a

possibility for both F and G. These would both have constant memory complexity in depth. Rather than having two networks, we can further reduce the memory budget by a rough factor of about two by exploiting the loose bijective property of the task, sharing the $X \to Y$ and $Y \to X$ models.

3.3 Paired and Unpaired Objective

In this section, we define the objective of the RevGAN model by specifying the different loss functions for paired and unpaired data.

Paired Loss We train our paired, reversible, image-to-image translation model, using the standard Pix2pix loss functions of Equation 4 from [24], applied in the $X \to Y$ and $Y \to X$ directions:

$$\begin{aligned} L_{\text{RevGANpaired}} = \lambda(L_{\text{L1}}(F) + L_{\text{L1}}(G)) \\ + L_{\text{cGAN}}(F, D_Y) + L_{\text{cGAN}}(G, D_X) \end{aligned} \tag{11}$$

where λ determines the relative importance between the L_1 loss and the conditional GAN loss L_{cGAN}. We also experimented with extra input noise for the conditional GAN, but found it not to help.

Unpaired Loss For unpaired RevGAN, we adapt the loss functions of the CycleGAN model [52], by replacing the L_1 loss with a cycle-consistency loss, so the total objective is:

$$\begin{aligned} L_{\text{RevGANunpaired}} = L_{\text{cGAN}}(F, D_Y) + L_{\text{cGAN}}(G, D_X) \\ + \mathbb{E}_x L_{\text{cycle}}(G, F, x) + \mathbb{E}_y L_{\text{cycle}}(F, G, y). \end{aligned} \tag{12}$$

where λ determines the relative importance between the cycle-consistency loss L_{cycle} and the conditional GAN loss L_{cGAN}.

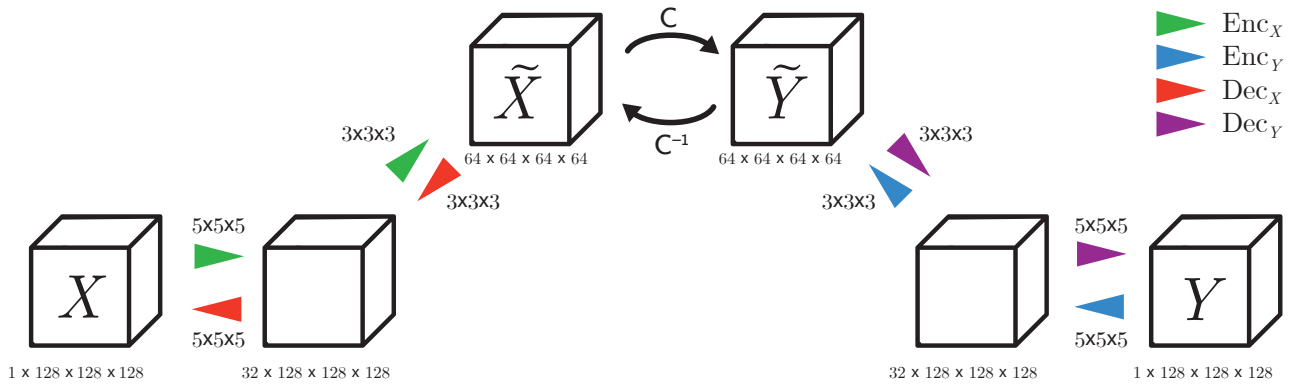


Figure 6: Illustration of the 3D ResNet architecture: Encoders Enc_X (green) and Enc_Y (blue) lift/encode from the image space features spaces $\tilde{X}$ and $\tilde{Y}$. Decoders Dec_X (red) and Dec_Y (purple) project/decode back to X and Y. The invertible mapping C transforms between $\tilde{X}$ and $\tilde{Y}$.

4 Implementation

We provide a Pytorch [38] implementation on Github. Our code extends the image-to-image translation framework from [52] with several reversible models in 2D and 3D. The reversible blocks are implemented using a modified version of MemCNN [46]. In this section we describe our RevGAN model in more detail and present specifics for both 2-dimensional and 3-dimensional architectures.

4.1 Generators

Reversible 2D ResNet For the 2-dimensional model, we tried to keep the architecture as close as possible to the ResNet used in the original CycleGAN paper [52]. The encoders $\text{Enc}_X, \text{Dec}_Y$ consist of a 7×7 convolutional layer that maps 3 input channels to 64 channels, followed by two 3×3 convolutional layers with stride 2 that spatially downsample ($/4$) the signal and increase the channel dimension ($\times 2$). For the reversible core C, we use R sequential reversible residual layers (with $R = 6$ for 128×128 *Cityscapes* data and $R = 9$ for 256×256 *Maps* data). We also refer to the amount of reversible residual layers in the core as the *depth* of the model. The decoders Dec_X, Dec_Y consist of two 3×3 fractionally-strided convolutional layers [1], followed by a 7×7 convolutional layer projecting the final features to 3 output channels.

We obtain our reversible residual layers using additive coupling from Equation 8, where both $\texttt{NN}_1$ and $\texttt{NN}_2$ are a 3×3 convolutional layer with reflection padding. Unlike the single convolutional layer in the residual blocks of the original CycleGAN ResNet [52], the reversible residual blocks contain two convolutional layers each halving the amount of channels.

Reversible 3D SRCNN Inspired by the SRCNN super-resolution model [15], we use a simple convolutional architecture to perform 3D super-resolution. We first apply a $3 \times 3 \times 3$ convolutional layer to increases the channel dimension to K channels, directly followed by an instance normalization layer [22] and a ReLU non-linearity. We then apply an arbitrary amount of 3D reversible blocks using additive coupling, where both $\texttt{NN}_1$ and $\texttt{NN}_2$ are a sequence of: a $3 \times 3 \times 3$ convolutional layer, an instance normalization layer, a ReLU non-linearity and another $3 \times 3 \times 3$ convolution. We use reflection padding of 1 to ensure that the input and output of $\texttt{NN}_1$ and $\texttt{NN}_2$ remain equally sized (and volume-preserving). Also, we initialize the weights of the last layer in the reversible block with zeros, so the entire block performs the identity mapping at the start of training. This trick has previously shown to be effective in the context of reversible networks [28]. Lastly, we perform a $1 \times 1 \times 1$ convolution to map the 32 channels back to the amount of channels of the input. A diagram of $\texttt{NN}_1$ and $\texttt{NN}_2$ used in the 3D reversible block is shown in Figure 7.

Reversible 3D ResNet For the 3-dimensional reversible ResNet model, the encoders Enc_X and Enc_Y are a sequence of: a $5 \times 5 \times 5$ convolutional layer that increases the channel dimension to 32, followed by an instance

[1] 'Fractionally-strided convolutional layers' or 'transposed convolutions' are sometimes referred to as 'deconvolutions' in literature. To avoid confusion, especially in the context of invertibility, we follow this [16] guide on convolutional arithmetic, and only refer to the term 'deconvolution' when we speak of the mathematical inverse of a convolution, which is different from the fractionally-strided convolution.

normalization layer [22] and a ReLU non-linearity, another $3 \times 3 \times 3$ convolution with stride 2, another instance normalization and a ReLU. The decoders Dec_X and Dec_Y consist of a transposed convolution with stride 2, an instance normalization layer, a ReLU nonlinearity and another $5 \times 5 \times 5$ convolutional layer followed by a Tanh.

We obtain our invertible core C by sequentially stacking reversible residual blocks using additive coupling (Equation 8), where both $\texttt{NN}_1$ and $\texttt{NN}_2$ are the sequence: a padded $3 \times 3 \times 3$ convolution, an instance normalization layer, a ReLU nonlinearity and a last convolution. The weights of this last convolution were initialized at zero, as also done in [28]. A diagram of $\texttt{NN}_1$ and $\texttt{NN}_2$ used in the 3D reversible block can be found in Figure 7.

An illustration of the whole 3D ResNet architecture is shown in Figure 6.

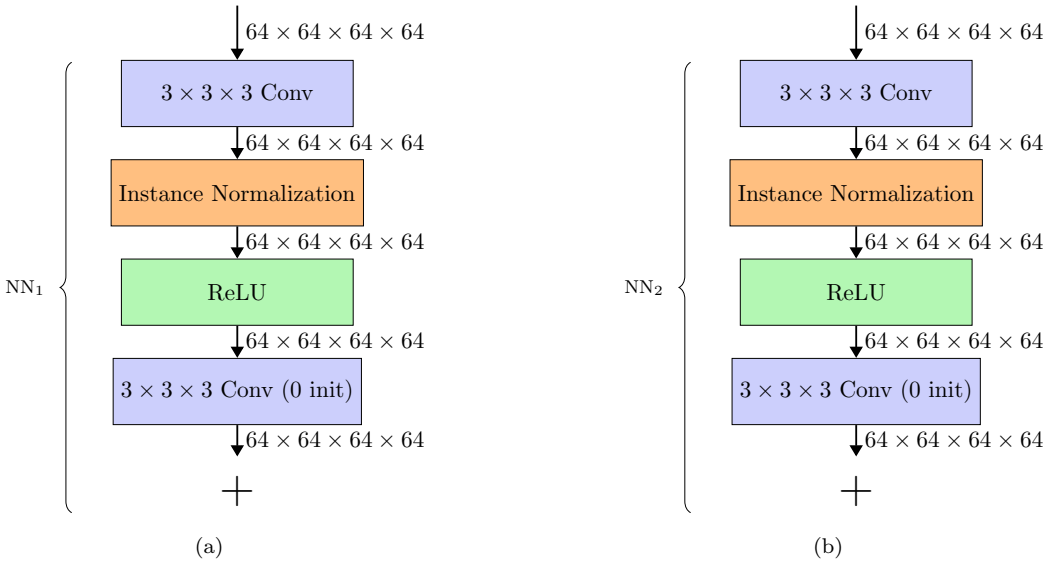


Figure 7: 3D Reversible Block

4.2 Discriminators

For fair comparison, we use the same PatchGAN as used in [24] and [52]. For the 3-dimensional version of our model we replace all two dimensional convolutions with three dimensional convolutions of equal radius. The architecture consists of four $4 \times 4(\times 4)$ convolutional layers with stride 2, each followed by a leaky ReLu non-linearity with a slope of 0.2. Every layer doubles the channel dimensions - while halving the spatial dimensions due to the used stride. The discriminator is referred to as PatchGAN, because its $70 \times 70(\times 70)$ receptive field does not cover the entire volume.

4.3 Optimization

The initial model parameters were sampled from a $\mathcal{N}(\mu = 0, \sigma = 0.02)$ Gaussian distribution. We used the Adam [27] optimizer with a 0.0002 learning rate (and $\beta_1 = 0.5, \beta_2 = 0.999$). All models were trained with batch size 1 for 200 epochs, where we linearly decay the learning rate to zero over the last 100 epochs. For λ we used a factor of 100 on the paired models and 10 on the unpaired models.

5 Datasets

We run tests on two 2D datasets and three 3D dataset. All five datasets have paired X and Y domain images, and we can thus extract quantitative evaluations of image fidelity.

5.1 Cityscapes

The Cityscapes dataset [12] is comprised of urban street scenes with high quality pixel-level annotations. For comparison purposes, we used the same 2975 image pairs as used in [52] for training and the validation set for testing. All images were downsampled to 128×128.

For evaluation, we adopt commonly used semantic segmentation metrics: *per-pixel accuracy*, *per-class accuracy* and *class intersection-over-union*. The outputs of photo→label mappings can directly be evaluated. For the reverse mapping, label→photo, we use the *FCN-Score* [52], by first passing our generated images through a FCN-8s semantic segmentation model [31] separately trained on the same segmentation task. We then measure the quality of the obtained segmentation masks using the same classification metrics. The intuition behind this (pseudo-)metric is that the segmentation model should perform well if images generated by the image-to-image translation model are of high quality.

5.2 Maps

The Maps dataset contains 1096 training images and an equally sized test set carefully scraped from Google Maps in and around New York City by [24]. The images in this dataset are downsampled to 256×256.

We evaluate the outputs with several commonly used metrics for image-quality: *mean absolute error* (MAE), *peak signal-to-noise ratio* (PSNR) and the *structural similarity index* (SSIM).

5.3 HCP Brains

The Human Connectome Project dataset consists of 15 $128 \times 128 \times 128$ brain volumes, of which 7 volumes are used for training. The value of each voxel is a $6D$ vector representing the 6 free components of a 3×3 symmetric diffusion tensor (used to measure water diffusivity in the brain). The brains are separated into high and low resolution versions. The low resolution images were upsampled using $2\times$ nearest neighbour so the input and output equal in size. This is a good task to trial on, since superresolution in 3D is a memory intensive task. For training, we split the brain volumes into patches of size $24 \times 24 \times 24$ omitting patches with less than 1% brain matter, resulting in an average of 112 extracted patches per volume.

We evaluate on full brain volumes with the *root mean squared error* (RMSE) between voxels containing brain matter in the ground-truth and the up-sampled volumes. We also calculate the error on the interior of the brain, defined by all voxels that are surrounded with a $5 \times 5 \times 5$ cube within the full brain mask, to stay in line with prior literature [44] [5].

5.4 1024-CT

The *1024-CT* dataset contains high-resolution CT imaging and artificially downsampled versions of the same scans. We aim to increase the resolution of these downsampled scans in all three dimensions, but focus on improving the resolution along the z-direction. Resolution is often lower in the z-direction than within the axial plane, especially with data from older scanner models without advanced techniques such as dose modulation and iterative reconstruction. In general, increasing resolution in CT is often infeasible due to the risk associated with high-dose CT radiation [47]. Therefore, we spatially downsample the CT volumes most along the z-direction and evaluate the scans in the coronal plane to clearly see the improvement in resolution.

The *1024-CT* dataset contains 18 train volumes and 5 test volumes each consisting of 671 ($\pm$49) slices of size 1024×1024 obtained with a high-end Canon CT scanner (Aquilion ONE). The values lie in the range [-1, 1], uniformly redistributed from [-1150, 350] Hounsfield Units (HU). The source domain was generated by aggressively down-sampling 4 times in the z-dimension and 2 times in the x and y dimensions (corresponding to a typically used resolution of 512 by 512).. We trained and evaluated on $128 \times 128 \times 128$ chunks. For evaluation, the chunks corresponds to the context around a $64 \times 64 \times 64$ moving target window used to construct full CT

volumes. To measure image quality, we compared the full volumes with our ground-truths using mean absolute error (MAE) and peak signal-to-noise ratio (PSNR).

5.5 NLST

The *NLST* dataset contains 17 CT scans for training and 3 CT scans for testing from The National Lung Screening Trial (NLST) [45]. Our goal is to learn the mapping between scans created with different configurations, that can be used as a pre-processing step to standardize CT scan appearance. All scans were obtained by a Siemens scanner and both a smooth (B30f) and a sharper (B50f) reconstruction kernel were available. Each scan contains an average of 169 ($\pm$14.7) axial 512×512 slices. All values are were normalized in a range of [-1, 1], uniformly redistributed from [-900, 400] HU. We trained and evaluated on $128 \times 128 \times 128$ cubes. For evaluation, we reconstructed by passing equally sized cubes around a $64 \times 64 \times 64$ moving target window and visualize slices from this fully constructed volume in the coronal plane.

6 Results

In this section, we evaluate the performance of our method on a range of image-to-image translation tasks. First, we compare the performance of the paired and unpaired 2-dimensional RevGAN model, both qualitatively and quantitatively, against the Pix2pix and CycleGAN baselines on the *Maps* and *Cityscapes* dataset. We repeat the experiment to test the performance of our 3D RevGAN model at different network depths using the *HCP Brains* dataset. Then we evaluate our model on a 3D CT super-resolution task and a 3D domain-adaptation task. Lastly, we study the scalability of our method in terms of memory-efficiency and model depth.

6.1 Comparison study

In this section, we compare our paired and unpaired RevGAN model with existing Pix2pix [24] and CycleGAN [52] baselines. For fair comparison, we evaluate on the *Maps* and *Cityscapes* datasets using the same architectures and hyper-parameters as in the original papers.

6.1.1 Qualitative Results

For the qualitative results, we picked the first images from the test sets that were also used in the original Pix2pix [24] and CycleGAN [52] papers to avoid 'cherry-picking' bias. The images are generated by models with equal parameter counts, indicated with a '†' symbol in the quantitative results of the next section (Table 3, Table 2). The qualitative results of the *Maps* dataset can be found in Figure 3. The results on the *Cityscapes* dataset are shown in Figure 2.

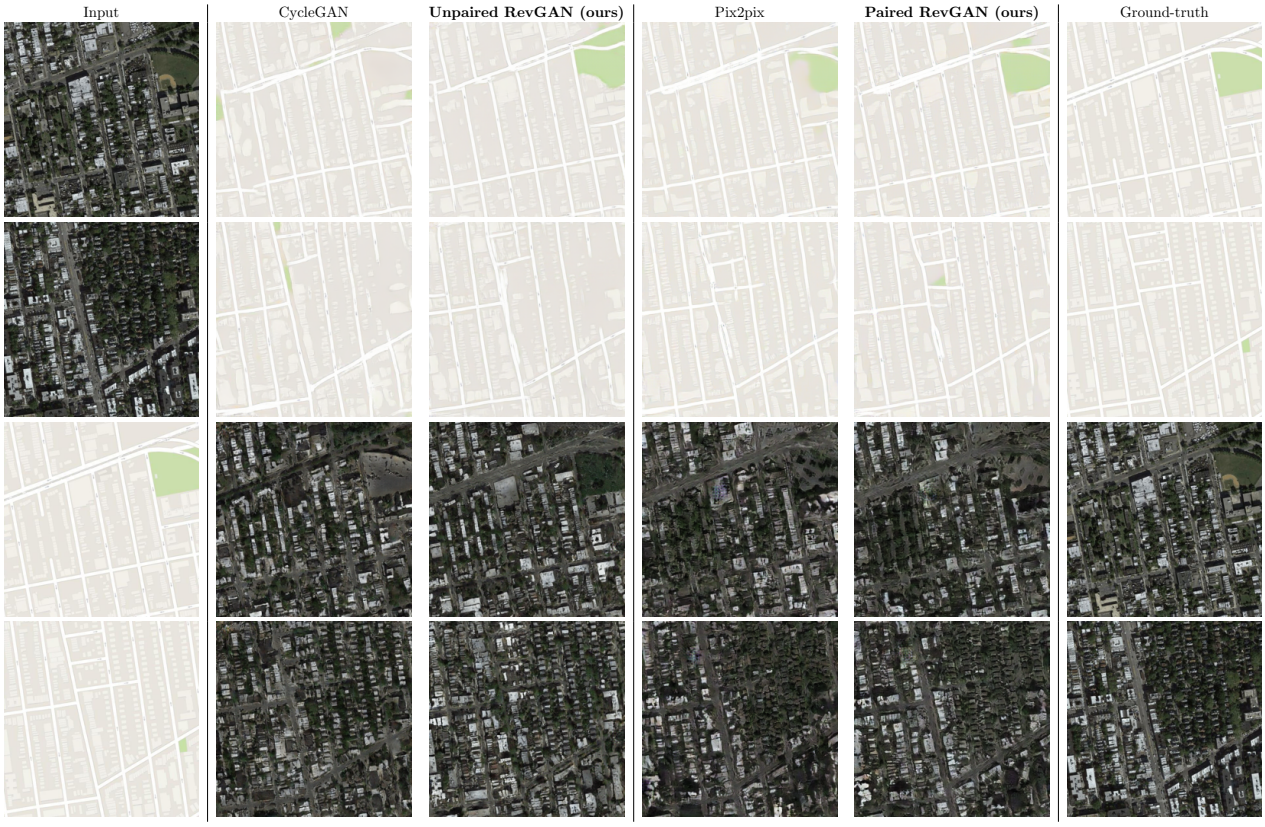


Figure 8: Test set image mappings on the *Maps* dataset. We see from this panel of images that there is no obvious degradation in the quality of the translated images between the baselines (Pix2pix and CycleGAN) and the reversible variants.

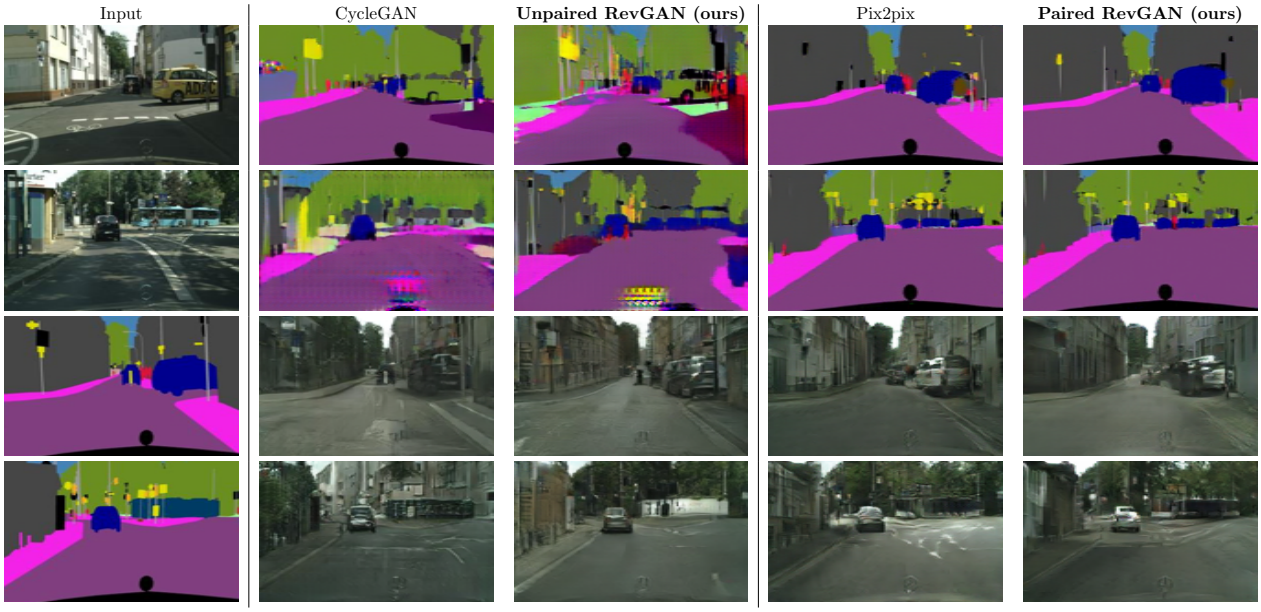


Figure 9: Test set image mappings on the *Cityscapes* dataset for the CycleGAN and Pix2pix models, compared to our reversible variants. Top: The photo→label mapping. Bottom: The label→photo mapping. Notice how in the greatest improvement is between the CycleGAN and our unpaired RevGAN variant; whereas, both the Pix2pix and paired RevGAN models are of comparative visual fidelity. More results can be found in the supplementary material.

All models are able to produce images of similar or better visual quality. The greatest improvement can be seen in the unpaired model (compare CycleGAN with Unpaired RevGAN). Both paired tasks are visually more appealing than the unpaired tasks, which make intuitive sense, since paired image-to-image translation is an easier task to solve than the unpaired version. We therefore conclude that the RevGAN model does not seem to under-perform our non-reversible baselines in terms of observable visual quality. A more extensive collection of model outputs can be found in Appendix A.

6.1.2 Quantitative Results

We provide quantitative evaluations of the performance of our RevGAN model on the *Maps* and *Cityscapes* datasets. To ensure fairness, the baselines use the code implementations from the original papers. For our model, we provide two versions, a low parameter count version and a parameter matched version.

Cityscapes In Table 2 the performance of our RevGAN model on the *Cityscapes* dataset is shown. The photo→label mapping is given by segmentation scores in the center columns and the performance on the label→photo is given by the *FCN-Scores* in the righthand columns.

Model	Width	Params	photo→label			label→photo		
			Per-pixel acc.	Per-class acc.	Class IOU	Per-pixel acc.	Per-class acc.	Class IOU
CycleGAN (baseline)†	32	3.9 M	0.60	**0.27**	**0.19**	0.42	0.15	0.10
Unpaired RevGAN	32	1.3 M	0.52	0.21	0.14	0.36	0.14	0.09
Unpaired RevGAN†	56	3.9 M	**0.66**	0.25	0.18	**0.65**	**0.24**	**0.17**
Pix2pix (baseline)†	32	3.9 M	**0.82**	0.43	0.32	**0.61**	**0.22**	**0.16**
Paired RevGAN	32	1.3 M	0.81	0.41	0.31	0.57	0.20	0.15
Paired RevGAN†	56	3.9 M	**0.82**	**0.44**	**0.33**	0.60	0.21	**0.16**

Table 2: Center Classification scores on *Cityscapes* photo→label. Right FCN-scores on *Cityscapes* label→photo. Top Unpaired models. Bottom Paired models. Bold numbers indicate where the best model in that section. Notice that in the sections where the baseline beats our model, the differences in values are only very small. † Parameter matched architectures

In Table 2, we see that on the low parameter and parameter matched RevGAN models outperform the CycleGAN baselines on the per-pixel accuracy. This matches our qualitative observations from the previous section. For

per-class and class IOU, we also beat the baseline on label→photo, and from similar or marginally worse on the photo→label task.

On the paired tasks we see that the results are more mixed and we perform roughly similar to the Pix2pix baseline, again matching our qualitative observations. We presume that the paired task is already fairly easy and thus the baseline performance is saturated. Thus introducing our model will do nothing to improve the visual quality of outputs. On the other hand, the unpaired task is harder and so the provision of by-design, approximately-inverse photo→label and label→photo generators improves visual quality. On the paired task, the main benefit comes in the form of the memory complexity (see Section 6.5.1), but on the unpaired task the RevGAN maintains low memory complexity, while generally improving numerical performance.

Maps Results on the *Maps* dataset are shown in Table 3, which indicate that the RevGAN model performs similarly and sometimes better compared to the baselines. Again, similarly to the *Cityscapes* experiment, we see that the biggest improvements are found on the unpaired tasks; whereas, the paired tasks demonstrate comparable performance.

Model	Width	Params	maps→satellite			satellite→maps		
			MAE	PSNR	SSIM	MAE	PSNR	SSIM
CycleGAN †	32	5.7 M	139.85 ± 15.52	**14.62** ± 1.16	0.31 ± 0.05	138.86 ± 20.57	**26.25** ± 3.64	0.81 ± 0.06
Unpaired RevGAN	32	1.7 M	**133.57** ± 18.09	14.59 ± 0.96	0.31 ± 0.05	142.56 ± 18.94	26.23 ± 3.89	0.81 ± 0.06
Unpaired RevGAN †	58	5.6 M	134.63 ± 14.25	14.54 ± 1.09	0.30 ± 0.06	148.98 ± 16.83	25.47 ± 4.27	0.80 ± 0.08
Unpaired RevGAN	64	6.8 M	135.48 ± 19.19	14.55 ± 1.24	0.26 ± 0.04	**133.12** ± 17.18	23.66 ± 2.80	0.67 ± 0.10
Pix2pix †	32	5.7 M	139.63 ± 13.14	14.78 ± 1.08	0.30 ± 0.05	129.16 ± 16.11	27.11 ± 3.11	0.82 ± 0.04
Paired RevGAN	32	1.7 M	**139.23** ± 12.76	14.73 ± 1.07	0.30 ± 0.05	129.80 ± 15.54	26.84 ± 3.35	0.81 ± 0.05
Paired RevGAN †	58	5.6 M	140.74 ± 12.45	**14.91** ± 1.13	0.31 ± 0.05	**128.55** ± 12.71	27.27 ± 3.12	0.82 ± 0.05
Paired RevGAN	64	6.8 M	140.59 ± 13.64	14.85 ± 1.20	0.31 ± 0.06	133.09 ± 12.09	**27.37** ± 3.06	0.82 ± 0.04

Table 3: Image quality on *Maps* dataset. Notice how in most of the experiments that the RevGAN performs better than the baseline. † Parameter matched architectures

6.2 3D HCP Brain Volumes

To evaluate our 3-dimensional RevGAN model we compare the model at different depths against the *HCP Brains* dataset of [44]. We compare the model with a simple SRCNN model [15] (see Section 4.1). In Figure 10, we show the outputs of both a paired and unpaired RevGAN model and with and without reversible layers. For each upsampling, we provide a visualization of the error. Higher quality results are obtained using models with additional reversible residual layers. Of course, it is not unusual that deeper models result in higher quality predictions. Increasing the model size, however, is often unfeasible due to memory constraints. Fitting the activations in GPU memory can be particularly difficult when dealing with large 3D volumes. This study suggests that we can train deeper neural image-to-image translation models by adding reversible residual layers to existing architectures, without requiring more memory to store model activations.

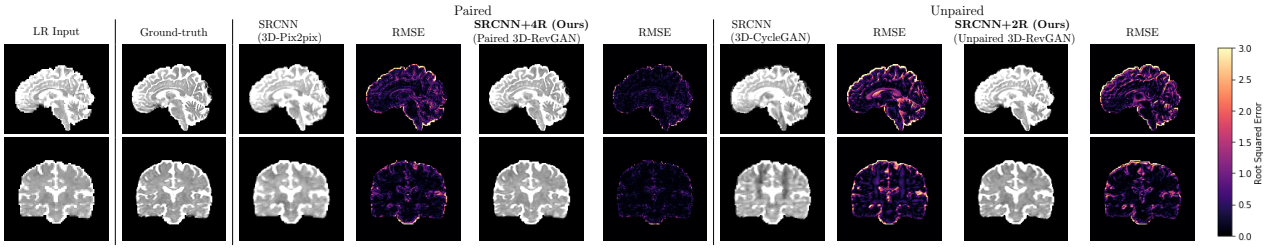


Figure 10: Visualization of mean diffusivity maps on an sagittal slices (top) and axial slices (bottom) of the first brain in the *HCP Brain* test set. From left to right: low-resolution input, high-resolution ground-truth, paired model without reversible layers (SRCNN-3D-Pix2pix), paired model with reversible layers (Paired 3D-RevGAN), unpaired model without reversible layers (SRCNN-3D-CycleGAN) and an unpaired model with reversible layers (Unpaired 3D-RevGAN).

The same conclusions can be drawn upon inspection of the quantitative results, shown in Table 4. As we can see, adding reversible residual layers in the core improves the overall performance, while not increasing the amount of memory required to store model activations. This effect, however, seems to be limited to a few (≈ 2 or 4) reversible residual layers, in the unpaired case, after which the performance slowly seems to degrade again. We do hypothesize that deeper reversible architectures may be more beneficial when training more complex functions or for a longer time period on a larger dataset.

Model	RMSE (Interior)	RMSE (Total)
Paired w/o L_{GAN} (3D-SRCNN)	7.03 ± 0.31	12.41 ± 0.57
Paired+2R w/o L_{GAN}	7.02 ± 0.32	12.41 ± 0.57
Paired+4R w/o L_{GAN}	**6.68** ± 0.30	**11.85** ± 0.56
Paired+8R w/o L_{GAN}	18.43 ± 1.03	21.40 ± 0.98
Paired (3D-Pix2pix)	11.94 ± 0.65	20.73 ± 1.05
Paired+2R	9.61 ± 0.40	17.36 ± 0.76
Paired+4R	8.43 ± 0.37	14.81 ± 0.61
Paired+8R	**7.82** ± 0.35	**13.76** ± 0.60
Unpaired (3D-CycleGAN)	17.23 ± 0.73	26.94 ± 1.20
Unpaired+2R	**11.05** ± 0.51	**17.76** ± 1.38
Unpaired+4R	18.98 ± 1.22	28.06 ± 1.44
Unpaired+8R	18.96 ± 0.85	27.94 ± 1.09

Table 4: Mean and standard deviation of RMSE scores measured on the 8 brains in the *HCP Brains* test set. Notice how in each experiment that the shallowest model is the not the highest performing. We are able to improve performance, by using deeper models at the same level of memory complexity as shallow models.

6.3 Chest CT Super-resolution

In this section, we evaluate our RevGAN model both qualitatively and quantitatively on the *1024-CT* dataset. We compare model outputs with high-resolution ground-truth volumes in the test set.

Qualitative Results In Figure 11, we present a comparison between trilinear upsampling, a paired RevGAN and an unpaired RevGAN. We present both a full coronal slice as well as a patched slices originating from 3D volumes. The outputs are compared with the high-resolution ground-truth.

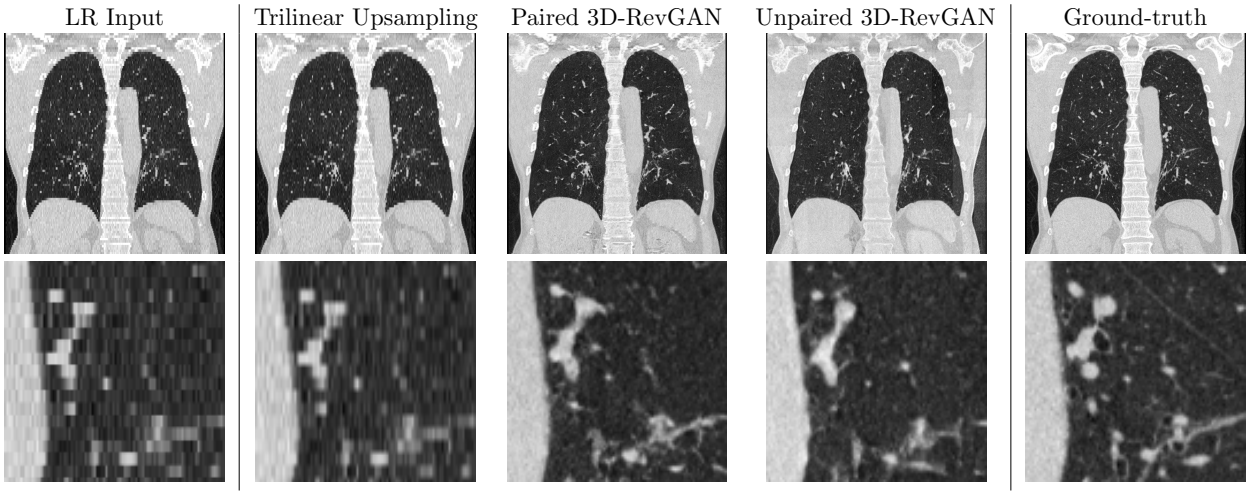


Figure 11: Visualization of a coronal slice (top) a patch from a coronal slice slices (bottom) of the first volume in the *1024-CT* test set. From left to right: low-resolution input, trilinear upsampling, paired 3D-RevGAN, unpaired 3D-RevGAN and high-resolution ground-truth.

We can see that the model achieves visual compelling results, with fine detail structures. Upon closer inspection of the more detailed structures in the CT volumes, we find that the model is able to correctly fill-in parenchymal details such as small airway walls, vasculature and tissue texture. More slices of mapped volumes from the test set, including some in the axial and lateral plane [TYC]: add axial+lateral in appendix, can be found in Appendix A.

Quantitative Results To quantitatively evaluate the model performance on the super-resolution task, we compare the model outputs with ground-truth volumes using *mean absolute error* (MAE), *peak signal-to-noise ratio* (PSNR) and *structural similarity index* (SSIM). Results of this comparison can be found in Table 5.

Model	MAE	PSNR	SSIM
Unpaired	0.24 ± 0.007	15.43 ± 0.39	0.44 ± 0.008
Paired	0.18 ± 0.001	15.89 ± 0.16	0.46 ± 0.008
Paired+2R	**0.15** ± 0.000	**18.19** ± 0.08	**0.48** ± 0.008
Paired+4R	0.15 ± 0.000	18.09 ± 0.08	0.47 ± 0.007

Table 5: Mean and standard deviation of MAE, PSNR and SSIM scores measured over the CT volumes in the *1024-CT* test set. Notice how in each experiment that the shallowest model is the not the highest performing. We are able to improve performance, by using deeper models at the same level of memory complexity as shallow models.

We found that the model performance increases after increasing the network depth by adding two reversible blocks. However, increasing adding even more reversible blocks does not seem to further increase the performance. Note that there even seems to be a slight decrease in performance when adding 4 reversible blocks. Overall, our results indicate that we can increase performance of super-resolution model by adding reversible blocks while keeping constant memory requirements for activation storage.

6.4 Chest CT Domain-adaptation

In this section, we provide the qualitative and quantitative results of our domain-adaptation experiments on the *NLST* dataset. The objective is to learn the mappings between scans with a smooth (B30f) and a sharper (B50f) reconstruction kernel.

Qualitative Analysis We visualize full and patched coronal slices of model outputs and compare them with ground-truths. The mapping from smooth to sharp reconstruction kernels, B50f→B30f, can be found in Figure 12. As we can see, the model is able to succesfully mimic the more grainy structure of scans created with the sharper reconstruction kernel. The results of reverse mapping B50f→B30f, from sharp to smooth reconstruction kernels, are shown in Figure 13.

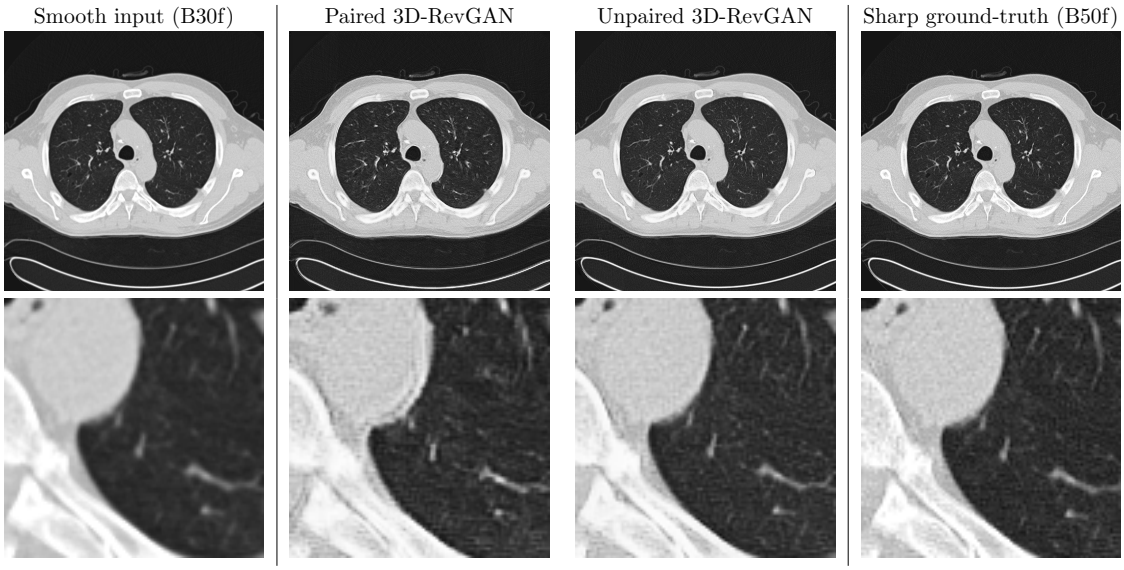


Figure 12: Smooth to sharp. Visualization of a coronal slice (top) a patch from a coronal slice slices (bottom) of the first volume in the *1024-CT* test set. From left to right: low-resolution input, paired 3D-RevGAN, unpaired 3D-RevGAN and high-resolution ground-truth.

Our results indicate that both the paired and unpaired RevGAN model can succesfully map smooth to sharp reconstruction kernels, B50f→B30f, and its reverse mapping, from sharp to smooth B50f→B30f reconstruction kernels. Not surprisingly, the model outputs of the paired model seems to be slightly better. Also, we found that the paired model converged quicker and showed more stable loss curves during training.

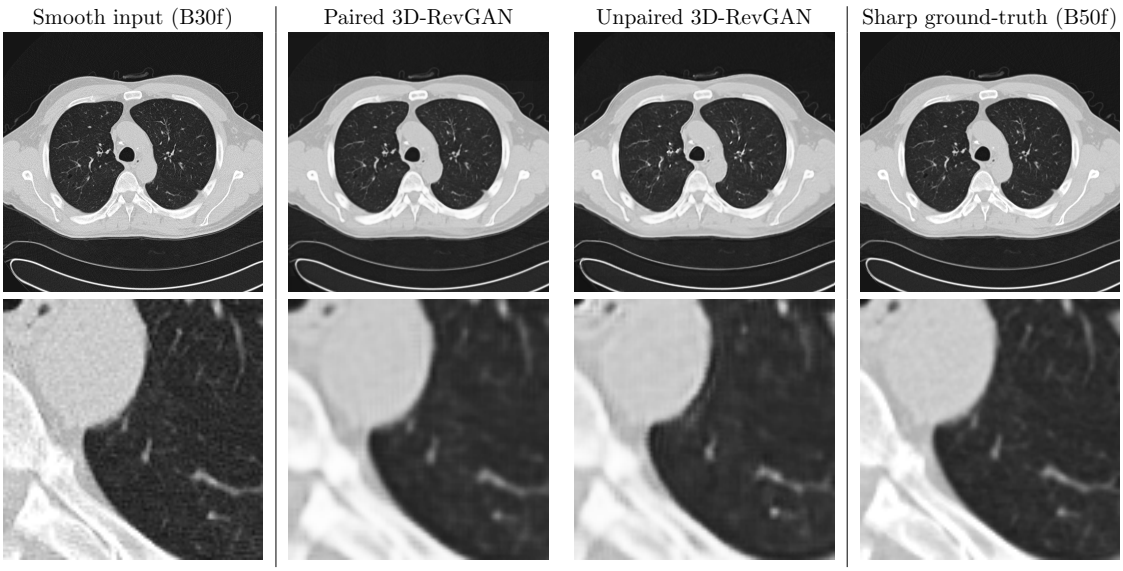


Figure 13: Sharp to smooth. Visualization of a coronal slice (top) a patch from a coronal slice slices (bottom) of the first volume in the *1024-CT* test set. From left to right: low-resolution input, paired 3D-RevGAN, unpaired 3D-RevGAN and high-resolution ground-truth.

The mapping from the sharper to the smoother kernel does not appear to be much more complicated than a simple blurring operation. Therefore, it can be expected that the mapping could also be learned by fitting a simple blurring function, such as a gaussian blur with radius as free parameter, on the training data. We showed, however, that our model is able to learn the mapping without making any assumptions based on prior-knowledge of the function that is being learned.

Quantitative Analysis To quantitatively assess the quality of the results, we compare the model outputs with ground-truth volumes using *mean absolute error* (MAE), *peak signal-to-noise ratio* (PSNR) and the *structural similarity index* (SSIM). We perform this comparison for the mappings from B30f (smooth) to B50f (sharp) kernels and for the reverse mapping from B50f (sharp) to B30f (smooth) kernels. The results of this comparison are shown in Table 6.

Model	Width	Params	B50f→B30f			B30f→B50f		
			MAE	PSNR	SSIM	MAE	PSNR	SSIM
Paired RevGAN	0	0.27 M	0.14	18.29	0.33	0.09	20.60	0.62
Paired RevGAN†	2	0.35 M	**0.10**	**23.24**	**0.46**	0.09	**26.88**	**0.77**

Table 6: Center Image-quality scores on *NLST* B50f→B30f (smooth to sharp). Right Image-quality scores on *NLST* B30f→B50f (sharp to smooth). Top Unpaired models. Bottom Paired models. Bold numbers indicate where the best model in that section. Notice that in the sections where the baseline beats our model, the differences in values are only very small. † Parameter matched architectures used in qualitative evaluation.

As we can see, increasing the model depth by adding reversible blocks increases the performance of our model, without increasing the memory cost to store activations. Also, our quantitative results confirm that the paired model performs better than the unpaired model.

6.5 Introspection

In this section, we provide an analysis of the RevGAN model memory-usage in practice and compare it with the CycleGAN baseline. Additionally, we compare how the memory-efficiency of our model relates to performance by varying the model depth.

6.5.1 Memory usage

To evaluate the memory-efficiency of our RevGAN model, we measure the GPU memory consumption for increasingly deeper models. We perform the same experiment on a CycleGAN model for comparison. The widths of both models were kept fixed at such a value that the model parameters are approximately equal (both ∼3.9 M) at depth 6.

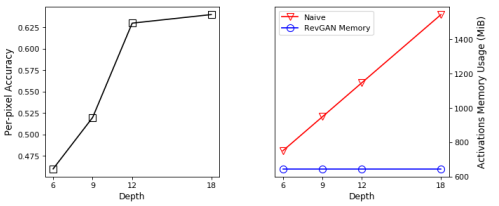
As can be seen from Table 7, the total memory usage increases for deeper networks in both models. In contrast to CycleGAN, however, the memory cost to store activations stays constant on the RevGAN model. A 6 layer CycleGAN model has the same total memory footprint of an unpaired RevGAN with 18-30 layers. Note that for convolutional layers the memory cost of storing the model is fixed given the network architecture, while the memory usage cost to store activations also depends on the size of the data. Therefore, reducing the memory cost of the activations becomes particularly important when training models on larger data sizes (e.g. higher image resolutions or increased batch sizes).

Depth	CycleGAN		Unpaired RevGAN	
	Model	Activations	Model	Activations
6	434.3	+ 752.0	374.4	+ **646.1**
9	482.3	+ 949.0	385.4	+ **646.1**
12	530.3	+ 1148.1	398.5	+ **646.1**
18	626.3	+ 1543.9	423.4	+ **646.1**
30	818.7	+ 2335.8	626.3	+ **646.1**

Table 7: Memory usage on GPU measured in MiB on a single Nvidia Tesla K40m GPU on the *Maps* dataset (lower is better). Both the CycleGAN and unpaired RevGAN have a similar number of parameters.

6.5.2 Scalability

Reversible architectures can be trained arbitarily deep without increasing the memory cost needed to store activations. We evaluate the performance of larger RevGAN models on the *Cityscapes* dataset.



(a) Per-pixel accuracy against depth.

(b) Memory usage against depth.

Figure 14: Comparison of per-pixel accuracy for a width 64 RevGAN evaluated after 75 epochs and memory usage on *Cityscapes* dataset.

As shown in Figure 14, with successive increases in depth, the performance of the RevGAN model increases on the *Cityscapes* task. This effect seems to hold up until a certain depth ($\sim 12 - 18$) after which we find a slight decrease in performance again. We presume this decrease in performance is due to the longer training times of deeper models, which we have not be able to train to full convergence due to time-budgeting issues. Keep in mind that we tried to keep our network architectures and training parameters as close as possible to networks used in the original Pix2pix and CycleGAN models. Other research suggests that training models with much deeper reversible architectures can be very effective [7]. We leave the exploration of alternative reversible architectures to future work.

7 Limitations and Discussion

7.1 Notes on memory-efficiency

Our results indicate that we can train image-to-image translation models with close to constant memory requirements in depth (see Table 7). This enables us to scale up to very deep architectures. Our ablation studies also show that increasing depth can lead to higher quantitative results in terms of various semantic segmentation metrics. This ability to scale up, however, trades memory for time, and so there is a trade-off to be considered in practical situations where we may be concerned about how long to spend in the development phase of such models. This is evident in our ablation study in Figure 14, where we were not able to wait until full convergence of the deepest models. We have also demonstrated empirically that given a constrained budget of trainable parameters, we are able to achieve improved performance on the Cityscapes and Maps datasets, especially for of an unpaired training regime.

7.2 Notes on the quantitative evaluation

For the quantitative evaluation, we compare the output of our models with available paired ground-truth images in the test sets. Unfortunately, such analysis is not possible on datasets in which no paired samples exist. The used pixel-based metrics *mean absolute error* (MAE), *mean squared error* (MSE) and *peak signal-to-noise ratio* (PSNR), are easy to interpret, but treat all pixels independently. The measure is sensitive to slight alignment differences that often occur in CT imaging and does not take structural information into account [23]. The *structural similarity index measure* (SSIM) is a slightly more sophisticated measure, but is also far from a perfect measure from image quality by human observers [21]. Nevertheless, the metrics are sufficient enough to be used for model comparison, which is the main purpose of the quantitative analysis.

Unlike likelihood-based generative models, such as variational auto-encoders [30], GANs do not have an explicit objective function. As a result, evaluation of the model using classic measures, such as log-likelihood, is not desired. Alternative evaluation metrics, such as Inception Score [41] and Fréchet Inception Distance [20] were suggested and have shown to correlate well with the human judgement of visual quality. Yet, both of these measures rely on a separate network pretrained on a collection of natural images, which differs greatly from the datasets used in this study. Therefore, we chose not to include the metric in our quantitative analysis. Properly evaluating the performance of GANs remains an open research question [3].

7.3 Notes on the adversarial loss

We performed the experiments on paired models with and without the adversarial loss L_{GAN} . We found that models without such loss generally perform better in terms of pixel-distance (MAE, MSE, PSNR, etc.), but that models with an adversarial loss typically obtain higher quality results upon visual inspection. A possible explanation of this phenomenon could be that models that solely minimize a pixel-wise distance, such as L_1 or L_2, tend to 'average out' or blur the aleatoric uncertainty (natural diversity) that exists in the data, in order to obtain a low average loss. An adversarial loss enforces the model to output an image that could have been sampled from this uncertain distribution (thereby introduce realistic looking noise), often resulting in less blurry and visually more compelling renderings, but with a potentially higher pixel-wise error.

Another issue with our setup is that two discriminators are required during training time (one of each domain). These are not used at test time, and can thus be considered as superfluous networks, requiring a lot of extra memory. That said, this is a general problem with CycleGAN and Pix2pix models in general.

7.4 The use of GANs in medical imaging

Although the CycleGAN model has proven capable of generating high-quality imaging, the model outputs were not always flawless. It is known that deep models that incorporate a GAN loss, such as CycleGAN, can lead to mis-diagnosis of medical conditions. Most dramatically, it has been shown [11] that biased data can result in tumors that are being added and removed from MRI images after the use of unpaired image translation. Consequently, the image output of a GAN should not directly be used for interpretation by doctors or automated software without taking steps to prevent such biases from wrongly altering data.

8 Negative Results

In this section we describe our negative results. In particular, we mention several attempts to replace the cycle-consistency loss all together using invertible neural networks. It should be stated that, for most ideas, we can only share an intuitive explanation of the cause of failure, instead of a thorough analysis. Nevertheless, the author of this manuscript believes that sharing these ideas can be beneficial to future research. Additionally, we provide a list of engineering tricks that did not significantly improve performance and were therefore also dropped during the research.

8.1 Invertibility as alternative to cycle-consistency loss

Due to the nature of the problem, our network is not fully invertible. As a result, we still need to use the cycle-consistency loss, which requires two forward propagation passes and two backward passes through the model. In future work we plan to explore techniques to get rid of the cycle-consistency loss.

Fully-invertible mapping The cycle-consistency loss enforces the forward and backward mapping to be close to the identity, in other words $F \circ G \approx \text{Id}$ and $G \circ F \approx \text{Id}$. Now imagine that we replace the generators F and G by a single function $C : X \to Y$, which inverse is responsible for the backward mapping $C^{-1} : Y \to X$. The cyclic mapping equals the identity by definition ($C^{-1} \circ C = \text{Id}$), and the cycle-consistency loss is thereby constraint to be zero.

Constructing a fully invertible network (or flow) can be technically challenging. We attempted to take an existing architecture, the 2D ResNet (Section 4.1), and replace each non-invertible transformation one-by-one with an invertible alternative. It has been shown by [25] that transposed convolutions used for downsampling and the fractionally-strided convolutional layers used for upsampling, which are typically not invertible, with sub-pixel re-arragements first introduced by [42] in a way that roughly preserves spatial ordering. Unfortunately, we did not find such suitable replacement for the first and last convolutional layer. Although the spatial-dimensions of the activations in remain fixed in these layers, they are not volume-preserving since they heavily increase or decrease the amount of channels. We found that training a network without them resulted in poor performance, which can be interpreted as additional evidence that lifting and projection of data to higher-dimensional domains is crucial for efficient learning (as discussed in Section 3.1).

Alternatively, we tried to use the multi-scale Glow architecture by [28] to act as an encoder and applying the inverse of the same (invertible) architecture as decoder to create an invertible auto-encoder. We trained the model with the loss functions in Equation 4 and Equation 7, but found that the performance was poor. Also, we found the Glow model difficult to train, due to the large amount of parameters (over 30 times more parameters than the CycleGAN ResNet).

We also found that we could replace the invertible core with a continuous residual layer using neural ordinary differential equations from [8].

An fully invertible model does not allow any information to be added or lost in the intermediate activations. Consequently, inputs propagated through an invertible network must always be fully recoverable from the output. Image-to-image translation mappings are typically not one-to-one and neural networks are known to perform well in higher-dimensional spaces. Taken together, we do not believe that full invertibility on its own is a desired property in typical image-to-image translation models.

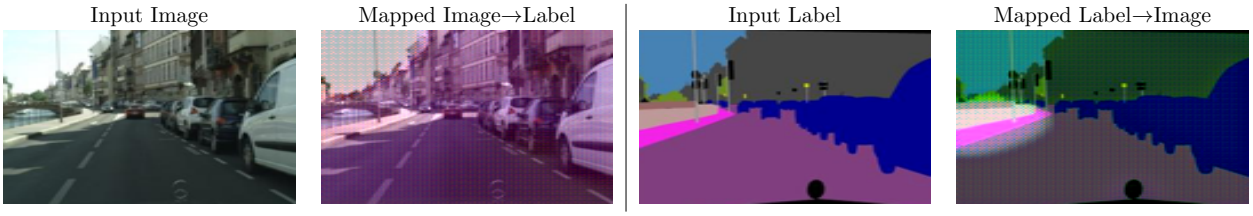


Figure 15: Train set image mappings on the *Cityscapes* dataset for fully-invertible model. LEFT: The photo→label mapping. RIGHT: The label→photo mapping. Notice how outputs stay really close to the inputs besides from some colour changes..

Local cycle-consistency The forward and backward cycle-consistency losses enforce the generators to be each others approximate left-inverses: $G \circ F \approx \mathrm{Id}$ and $F \circ G \approx \mathrm{Id}$. Let us write out the forward cyclic mapping $F \circ G = \mathrm{Dec}_X \circ C^{-1} \circ \mathrm{Enc}_Y \circ \mathrm{Dec}_Y \circ C \circ \mathrm{Enc}_X \approx \mathrm{Id}$. Trivially, we can derive that the forward cycle matches the identity ($F \circ G = \mathrm{Id}$), if the following components match the identity: $\mathrm{Dec}_X \circ \mathrm{Enc}_X = \mathrm{Id}$, $C^{-1} \circ C = \mathrm{Id}$ and $\mathrm{Enc}_Y \circ \mathrm{Dec}_Y = \mathrm{Id}$. Because $C^{-1} \circ C = \mathrm{Id}$ is true by definition, we aim to achieve cycle-consistency in the forward mapping by only approximating $\mathrm{Dec}_X \circ \mathrm{Enc}_X \approx \mathrm{Id}$ and $\mathrm{Enc}_Y \circ \mathrm{Dec}_Y \approx \mathrm{Id}$. We call this *local cycle-consistency*, because we aim to accomplish cycle-consistency by enforcing cycle-consistency only on a local subset (in this case the encoders and decoders). By symmetry, we can achieve cycle-consistency of the backward mapping $G \circ F = \mathrm{Dec}_Y \circ C \circ \mathrm{Enc}_X \circ \mathrm{Dec}_X \circ C^{-1} \circ \mathrm{Enc}_Y \approx \mathrm{Id}$, by enforcing the encoder and decoder to be locally cycle-consistent in the backward direction (i.e. $\mathrm{Dec}_Y \circ \mathrm{Enc}_Y \approx \mathrm{Id}$ and $\mathrm{Enc}_X \circ \mathrm{Dec}_X \approx \mathrm{Id}$).

Unfortunately, we were not able to successfully apply local cycle-consistency. The biggest obstacle seemed to be the fact that $\mathrm{Dec}_X \circ \mathrm{Enc}_X \approx \mathrm{Id}$ and $\mathrm{Enc}_Y \circ \mathrm{Enc}_Y \approx \mathrm{Id}$ require a minimization in feature space which we can not trivially combine with the differently scaled (L_1) losses in image space from Section 3.3. Although we did experiment with some weighted L_1 and L_2 in feature space, we were not able to successfully combine the loss with the other losses.

We also tried to only optimize $\mathrm{Dec}_X \circ \mathrm{Enc}_X \approx \mathrm{Id}$ and $\mathrm{Dec}_Y \circ \mathrm{Enc}_Y \approx \mathrm{Id}$, which is equivalent to a typical auto-encoders loss [4]: $L_{\text{auto}} = \mathbb{E}_x ||\mathrm{Dec}_X \circ \mathrm{Enc}_X(x) - x||_1 + \mathbb{E}_y ||\mathrm{Dec}_Y \circ \mathrm{Enc}_Y(y) - y||_1$. Unfortunately, we found that training a model with such loss performed significantly worse than a model trained with the full cycle-consistency loss.

Fully-invertible network with noisy channels As previously explained, modeling the mapping between X and Y as a fully invertible function is undesirable as image-to-image translation tasks are typically not one-to-one and because higher-dimensional, overcomplete representations are known to train more effectively [36] [6]. Alternatively, we could increase the dimensionality of the input domains X and Y (to lift) by concatenating them with noisy channels Z_X and Z_Y and then model the mapping between the new higher-dimensional domains with an invertible mapping $C : X \odot Z_X \rightarrow Y \odot Z_Y$. Optionally, we can still apply the paired and unpaired losses from Section 3.3 to the subset of the concateted channel that belongs to the original data. Intuitively, the noisy output channels could allow the model to remove (dump) information from the input that does not exist in the output. At the same time, the noisy input channel can act as a source from which additional information can be generated (as is typically done in GANs).

Unfortunately, we found that the additional noise hampered training significantly. In our experiments, we sampled noise from a uniform distribution which we then concatenated to original data in the channel direction. Although we did not obtain high-quality results, it would be interesting to further investigate this idea.

Convolutions as Pseudo-invertible Touplitz Matrices Lastly, we explored (psuedo-)invertible convolutions as a parameterized alternative to the sub-pixel rearrangement introduced by [42] used for invertible down-sampling in [25]. If we formulate the (strided-)convolutions as Toeplitz matrix–vector products [33], we can easily obtain the (pseudo-)inverse of the convolution operation by calculating inverse or pseudo-inverse (Moore-penrose inverse) of the Toeplitz matrix. We succesfully implemented a differentiable pseudo-inverse in PyTorch, but found that exact pseudo-invertibility is computationally too slow to run. Future work could investigate whether such transformations can be applied in a computationally efficient way. Moreover, such a transformation might be useful when designing more expressive architectures in flow-based density estimations, such as [28].

8.2 Negative engineering results

In this section we list some of engineering attempts that did not turn out to improve performance.

Affine coupling We tried to replace *additive coupling* with *affine coupling*, which has been applied succesfully in the context of reversible networks by [28]. In theory, affine coupling is more general and more expressive than additive coupling. We found, however, that affine coupling degraded performance and made training more unstable. Therefore, it would be interesting to see whether affine coupling outperforms additive coupling for other architectures or hyper-parameters.

Sub-pixel convolutions We tried to replace the down-sampling and up-sampling layers with sub-pixel convolutions [42] in our 2D and 3D models, which have also been applied succesfully in the context of invertible architectures [25], but found that it degraded performance. Sub-pixel convolutions were originally proposed to save memory in super-resolution problems by applying convolutions in lower-dimensional space rather than in the higher-dimensional target space. The RevGAN model, on the other hand, saves memory by not having to store the activations of the reversible layers.

Nearest-neighbours and Bilinear Upsampling We tried to replace the transposed convolutions used for up-sampling in our model with nearest-neighbour and bilinear upsampling to prevent checkerboard-like aftifacts as explained in [37], but found that it degraded performance. Furthermore, we observed that the checkerboard appeared in early training stages, but that they disappeared after a sufficient amount of training iterations.

Consensus Optimization We tried *Consensus Optimization* [34] to stabilize training by encouraging agreement between the two players in the mini-max GAN game. Consensus optimization boils down to regularization term over the second-order derivative over our gradients, which is a computationally intensive task. We stopped using it because it slowed down training too much.

Identity Loss An additional loss known as the *identity loss*, introduced by [43], encourages the generators to be near an identity mapping: $L_{\text{identity}}(G, F) = \mathbb{E}_x \, ||x - G(x)||_1 + ||y - F(y)||_1$. It is known to be helpful by guiding early training and encourage the mapping to preserve color composition between the input and output. We did not use it in the 2D experiments to stay in line with the Maps and Cityscapes hyper-parameters from [52] for comparison. We did not find the identity loss to improve performance early on in training and therefore stopped using it.

Neural Ordinary Differential Equations We found that the invertible core can be replaced with a continuous-depth residual networks introduced in [8] of which the forward and inverse pass are trained using an ordinary differential equation (ODE) solver. Due to time constraints, we were not able to evaluate the performance of this method. Some benefits of the method are constant $\mathcal{O}(1)$ memory cost as a function of depth (comparable with additive coupling in Section 2.3) and explicit control over the numerical error. In future work we plan to explore the use of neural ordinary (or even stochastic) differential equations in the context of image-to-image translation.

9 Conclusion

In this thesis we have proposed a new image-to-image translation model using reversible residual layers. The proposed model is approximately invertible *by design*, essentially weight-tying in the forward and backward direction, hence training from domain X to domain Y simulaneously trains the mapping from Y to X. We demonstrate equivalent or improved performance in terms of image quality, compared to similar non-reversible methods. We show that our model is more memory efficient, because activations of reversible residual layers do not have to be stored to perform backpropagation.

Additionally, we demonstrate that our model can also be applied on memory-intensive tasks in medical imaging. For instance, we perform a 3-dimensional super-resolution task on brain MRI volumes using a model at depths that would not have been possible without the memory savings of the reversible layers. Lastly, we provide a proof-of-principle for using a reversible neural network model capable of pre-processing 3D chest CT volumes to high resolution with a standardized appearance.

References

[1] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

[2] Sanjeev Arora and Yi Zhang. Do gans actually learn the distribution? an empirical study. *CoRR*, abs/1706.08224, 2017.

[3] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[5] Stefano B Blumberg, Ryutaro Tanno, Iasonas Kokkinos, and Daniel C Alexander. Deeper image quality transfer: Training low-memory neural networks for 3d images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 118–125. Springer, 2018.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[7] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. *arXiv preprint arXiv:1709.03698*, 2017.

[8] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

[9] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[10] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 415–423, 2015.

[11] Joseph Paul Cohen, Margaux Luck, and Sina Honari. How to cure cancer (in images) with unpaired image translation. *Medical Imaging with Deep Learning*, 2018.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[13] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[14] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[15] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[17] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2414–2423, 2016.

[18] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, pages 2214–2224, 2017.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[21] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *Pattern recognition (icpr), 2010 20th international conference on*, pages 2366–2369. IEEE, 2010.

[22] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1510–1519, 2017.

[23] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[24] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[25] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[26] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[29] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[32] James Martens and Ilya Sutskever. Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012.

[33] MA Matuson. Svd pseudo-inverse deconvolution of two-dimensional arrays. Technical report, Pennsylvania State University Park Applied Research Lab, 1985.

[34] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835, 2017.

[35] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[36] Peter Ochs, Tim Meinhardt, Laura Leal-Taixé, and Michael Möller. Lifting layers: Analysis and applications. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, pages 53–68, 2018.

[37] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS-W*, 2017.

[39] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible conditional gans for image editing. *CoRR*, abs/1611.06355, 2016.

[40] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[41] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[42] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[43] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[44] Ryutaro Tanno, Daniel E Worrall, Aurobrata Ghosh, Enrico Kaden, Stamatios N Sotiropoulos, Antonio Criminisi, and Daniel C Alexander. Bayesian image quality transfer with cnns: Exploring uncertainty in

dmri super-resolution. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 611–619. Springer, 2017.

[45] National Lung Screening Trial Research Team. The national lung screening trial: overview and study design. *Radiology*, 258(1):243–253, 2011.

[46] Sil C van de Leemput, Jonas Teuwen, and Rashindra Manniesing. Memcnn: a framework for developing memory efficient deep invertible networks. *International Conference on Learning Representations (ICLR) Workshop Track*, 2018.

[47] Zhimin Wang, Suicheng Gu, Joseph K Leader, Shinjini Kundu, John R Tedrow, Frank C Sciurba, David Gur, Jill M Siegfried, and Jiantao Pu. Optimal threshold in ct quantification of emphysema. *European radiology*, 23(4):975–984, 2013.

[48] Jelmer M Wolterink, Anna M Dinkla, Mark HF Savenije, Peter R Seevinck, Cornelis AT van den Berg, and Ivana Išgum. Deep mr to ct synthesis using unpaired data. In *International Workshop on Simulation and Synthesis in Medical Imaging*, pages 14–23. Springer, 2017.

[49] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 350–358, 2012.

[50] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *arXiv preprint arXiv:1809.07294*, 2018.

[51] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Trans. Computational Imaging*, 3(1):47–57, 2017.

[52] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

A Additional Results

Additional Cityscapes Mappings: photo→label

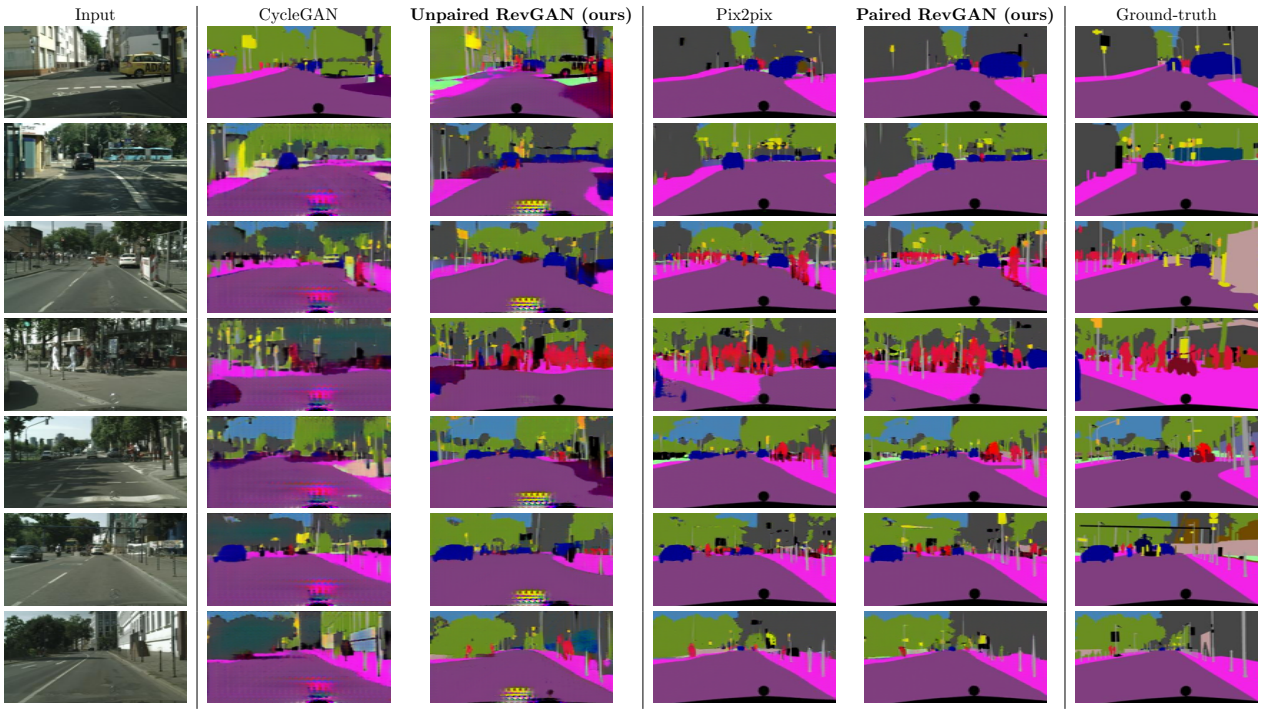


Figure 16: Additional image mappings for *photo→label* on the *Cityscapes* test set.

Additional Cityscapes Mappings: label→photo

Figure 17: Additional image mappings for *label→photo* on the *Cityscapes* test set.

Additional Maps Mappings: Satellite→Maps

Figure 18: Additional image mappings for *satellite→maps* on *Maps* test set.

Additional Maps Mappings: Maps→Satellite

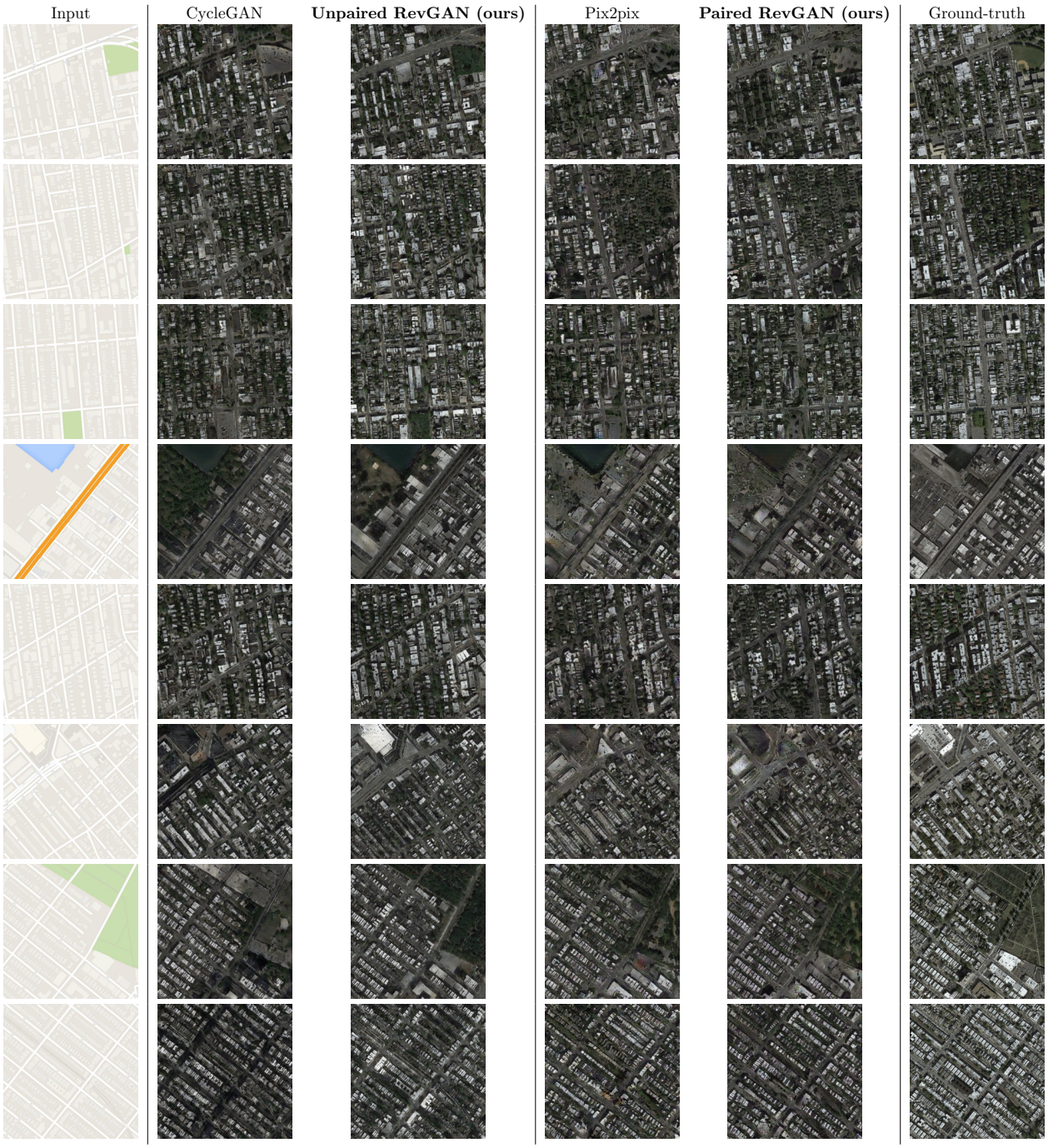


Figure 19: Additional image mappings for *maps→satellite* on *Maps* test set.

Additional Chest CT Domain-adaptation Mappings

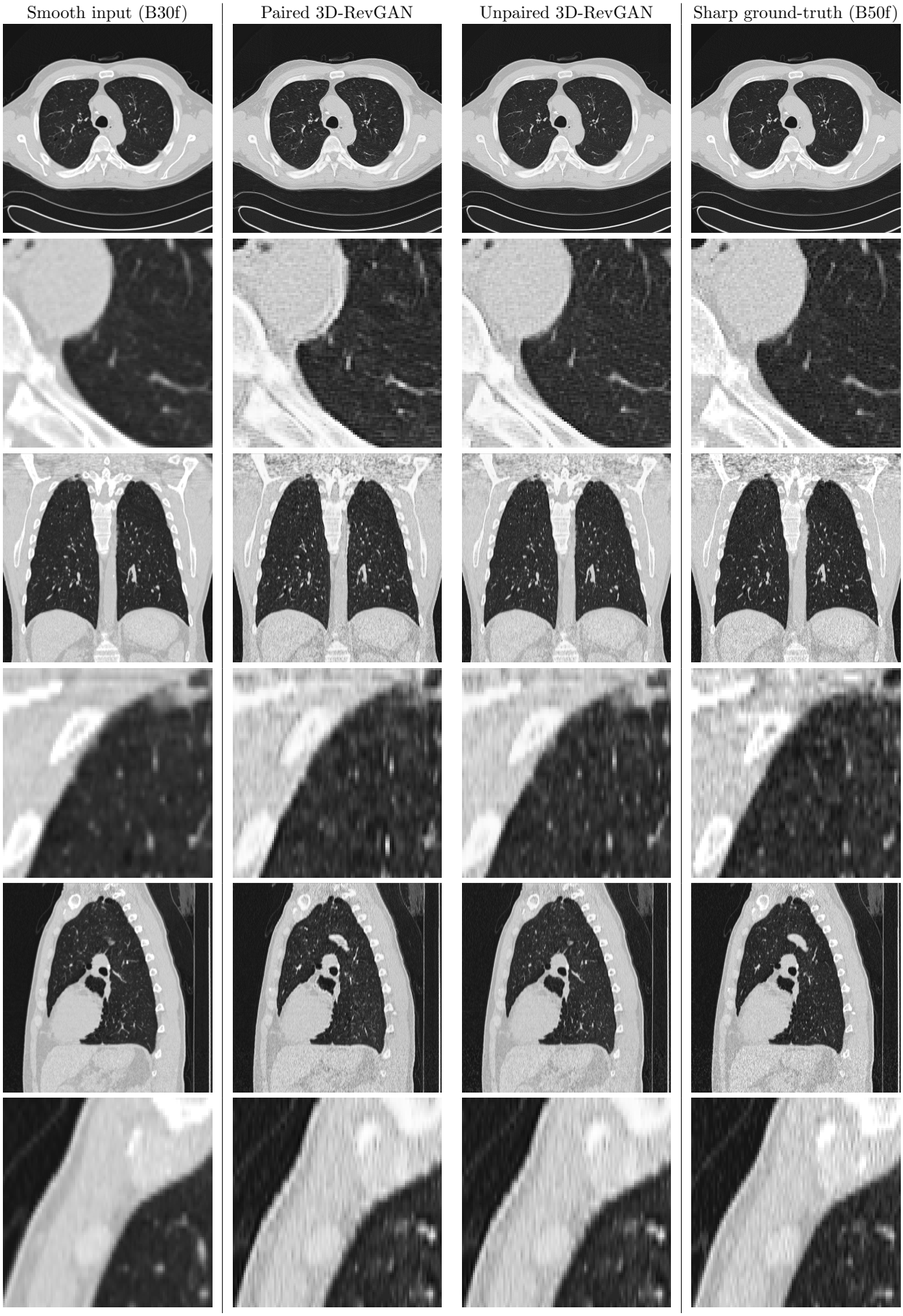


Figure 20: Smooth to sharp. Visualization of a coronal slice (top) a patch from a coronal slice slices (bottom) of the first volume in the *NLST* test set. From left to right: low-resolution input, paired 3D-RevGAN, unpaired 3D-RevGAN and high-resolution ground-truth.

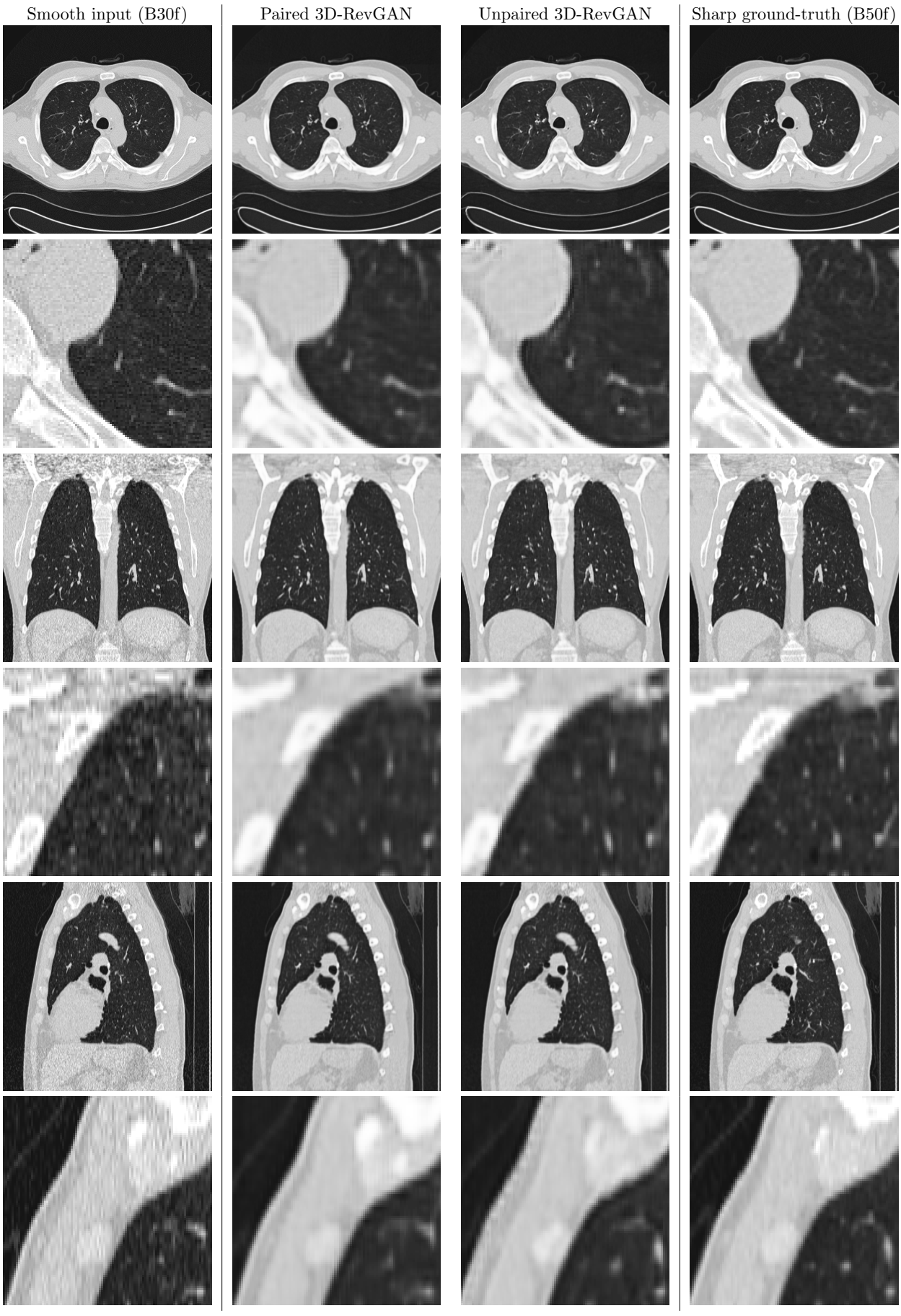


Figure 21: Sharp to smooth. Visualization of a coronal slice (top) a patch from a coronal slice slices (bottom) of the first volume in the *NLST* test set. From left to right: low-resolution input, paired 3D-RevGAN, unpaired 3D-RevGAN and high-resolution ground-truth.