Reversible GANs for Memory-efficient Image-to-Image Translation

Tycho F.A. van der Ouderaa
University of Amsterdam
tychovdo@gmail.com

Daniel E. Worrall
University of Amsterdam
d.e.worrall@uva.nl

Abstract

The Pix2pix [15] and CycleGAN [31] 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 at near constant memory budget.

1. Introduction

Computer vision was once considered to span a great many disparate problems, such as superresolution [11], colorization [7], denoising and inpainting [29], or style transfer [12]. 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 [28]). With the new advances of modern machine learning, many of these tasks have been unified under the term of *image-to-image translation* [15].

Mathematically, given two image domains X and Y, the task is to find or learn a mapping $F : X \rightarrow Y$, 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 $G : Y \rightarrow X$. 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* [31]. 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 paper, 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 $X \rightarrow Y$ and $Y \rightarrow X$ mapping, but just a single $X \rightarrow Y$ model, which we can run in reverse to approximate $Y \rightarrow X$. Furthermore, note by explicitly weight-tying the $X \rightarrow Y$ and $Y \rightarrow X$ models, we can see that training in the $X \rightarrow Y$ direction will also train the reverse $Y \rightarrow X$ direction, which does not necessarily occur with separate models. Lastly, there is also a computational benefit that invertible networks are very memory-efficient [13]; 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 networks performing better in image generation tasks [4]. Furthermore, this enables dense pixel-wise translation models to be shifted to memory-intensive arenas, such as 3D (see Section 5.3 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 [15] or CycleGAN [31], 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 5).

2. Background and Related Work

In this section, we recap the basics behind Generative Adversarial Networks (GANs), cycle-consistency, and reversible/invertible networks.

2.1. Generative Adversarial Networks (GANs)

Generative adversarial networks (GANs) [14] enjoy huge success in tasks such as image generation [4], image interpolation [17], and image re-editing [26]. 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:

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

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 [14] is

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

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* [24].

2.2. Image-to-Image Translation

In a standard (paired) image-to-image translation problem [15], 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

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

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 [30], for instance, L_1-losses treat each pixel as independent. Instead, since GANs do not apply the loss per-pixel, they can account for these 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 [15], 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; although, in practice, we usually ignore the additional noise input [31]. 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

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

where

$$L_{L1}(F) = \mathbb{E}_{x,y} \|y - F(x)\|_1 \quad (5)$$
$$L_{\text{cGAN}}(F, D) = \mathbb{E}_x \left[\log D(x) + \log(1 - D(F(x)))\right]. \quad (6)$$

λ is a tuneable hyperparameter typically set in the range $10 - 100$ [15].

2.3. Cycle-consistency

The CycleGAN model was proposed as an alternative to Pix2pix for unpaired domains [31]. The model 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

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

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

2.4. Invertible Neural Networks (INNs)

In recent years, several studies have proposed invertible neural networks (INNs) in the context of normalizing flow-based methods [27] [20]. It has been shown that INNs are capable of generating high quality images [19], perform image classification without information loss in the hidden layers [16] and analyzing inverse problems [1]. Most of the work on INNs, including this study, heavily relies upon the transformations introduced in NICE [9] later extended in RealNVP [10]. Although INNs share interesting properties they remain relatively unexplored.
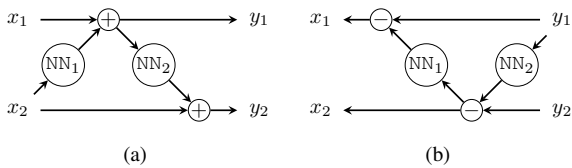
Figure 1. Illustration of forward (a) and backward (b) pass of a residual block in a reversible residual layer. Note, functions NN_1 and NN_2 need not be invertible, hence the layer is very flexible and in practical terms very easy to implement.

Additive Coupling In our model, we obtain an invertible residual layer, as used in [13], using a technique called *additive coupling* [9]: first we split an input x (typically over the channel dimension) into (x_1, x_2) and then transform them using arbitrary complex functions NN_1 and NN_2 (such as a ReLU-MLPs) in the form (left):

$$y_1 = x_1 + \text{NN}_1(x_2) \qquad x_1 = y_1 - \text{NN}_1(x_2) \qquad (8)$$
$$y_2 = x_2 + \text{NN}_2(y_1) \qquad x_2 = y_2 - \text{NN}_2(y_1). \qquad (9)$$

The inverse mappings can be seen on the right. Figure 1 shows a schematic of these equations.

Memory efficiency Interestingly, invertible residual layers are very memory-efficient because intermediate activations do not have to be stored to perform backpropagation [13]. 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).

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*.

Technique	Spatial Complexity (Activations)	Computational Complexity
Naive	$\mathcal{O}(L)$	$\mathcal{O}(L)$
Checkpointing [22]	$\mathcal{O}(\sqrt{L})$	$\mathcal{O}(L)$
Recursive [6]	$\mathcal{O}(\log L)$	$\mathcal{O}(L \log L)$
Additive Coupling [13]	$\mathcal{O}(1)$	$\mathcal{O}(L)$

Table 1. Comparison of Spatial and Computational Complexity copied from [13]. 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.

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 [25] and better all-round performance [4]. 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.

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

$$F = \text{Dec}_Y \circ C \circ \text{Enc}_X \qquad (10)$$
$$G = \text{Dec}_X \circ C^{-1} \circ \text{Enc}_Y. \qquad (11)$$

For the invertible cores we use invertible residual networks based on additive coupling as in [13]. 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-
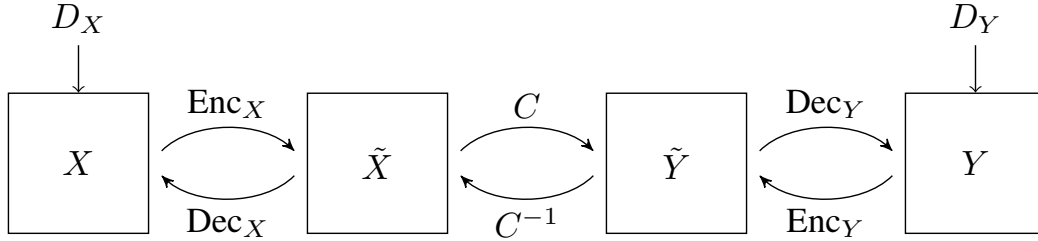
Figure 2. Schematic of our RevGAN model. Between the low-dimensional image spaces X and Y and their corresponding high-dimensional feature spaces $\tilde{X}$ and $\tilde{Y}$ we place non-invertible encoder and decoder networks $\text{Enc}_X, \text{Dec}_X, \text{Enc}_Y$ and Dec_Y. The feature spaces $\tilde{X}$ 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.

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.

Paired RevGAN   We train our paired, reversible, image-to-image translation model, using the standard Pix2pix loss functions of Equation 4 from [15], applied in the $X \to Y$ and $Y \to X$ directions:

$$L_{\text{RevGANpaired}} = \lambda(L_{\text{L1}}(F) + L_{\text{L1}}(G)) \\ + L_{\text{cGAN}}(F, D_Y) + L_{\text{cGAN}}(G, D_X) \quad (12)$$

We also experimented with extra input noise for the conditional GAN, but found it not to help.

Unpaired RevGAN   For unpaired RevGAN, we adapt the loss functions of the CycleGAN model [31], by replacing the L_1 loss with a cycle-consistency loss, so the total objective is:

$$L_{\text{RevGANunpaired}} = L_{\text{cGAN}}(F, D_Y) + L_{\text{cGAN}}(G, D_X) \\ + \mathbb{E}_x L_{\text{cycle}}(G, F, x) + \mathbb{E}_y L_{\text{cycle}}(F, G, y). \quad (13)$$

4. Implementation and datasets

The model we describe is very general and so below we explain in more detail the specifics of how to implement our paired and unpaired RevGAN models. We present 2D and 3D versions of the reversible models.

4.1. Implementation

Network Architectures   We use two main varieties of architecture. On the 2D problems, we modify the ResNet

from [31], by replacing the inner convolutions with a reversible core. The core consists of 6 or 9 reversible residual layers, dependent on the dataset—we use 6 reversible residual layers for the core on 128×128 (*Cityscapes*) data and 9 reversible residual layers on 256×256 (*Maps*) data. A more detailed description of the network architectures can be found in the supplementary material. In 3D, we use an architecture based on the SRCNN of [11] (more details in supplementary material).

Training details   All model parameters were initialized from a Gaussian distribution with mean 0 and standard deviation 0.02. For training we used the Adam optimizer [18] with a learning rate of 0.0002 (and $\beta_1 = 0.5, \beta_2 = 0.999$). We keep the learning rate fixed for the first 100 epochs and then linearly decay the learning rate to zero over the next 100 epochs, for the 2D models. The 3D models are trained with a fixed learning rate for 20 epochs. We used a λ factor of 10 for the unpaired models and a λ factor of 100 for the paired models.

4.2. Datasets

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

Cityscapes   The Cityscapes dataset [8] 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 [31] 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* [31], by first passing our generated images through a FCN-8s semantic segmentation model [21] separately trained on the same segmentation task. We then measure the quality of the

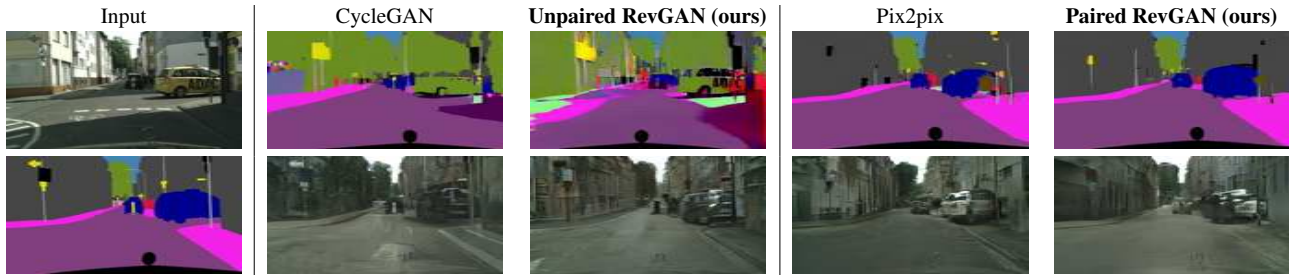| Input | CycleGAN | **Unpaired RevGAN (ours)** | Pix2pix | **Paired RevGAN (ours)** |

Figure 3. 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 Cycle-GAN 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.

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.

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 [15]. 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).

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 up-sampled 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 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 5×5 cube within the full brain mask, to stay in line with prior literature [28] [3].

5. Results

In this section, we evaluate the performance of our paired and unpaired RevGAN model, both quantitatively and qualitatively, against Pix2pix and CycleGAN baselines. Additionally, we study the scalability of our method in terms of memory-efficiency and model depth. For easy comparison, we aim to use the same metrics as used in related literature.

5.1. Qualitative Evaluation

We present qualitative results of the RevGAN model on the *Maps* dataset in Figure 4 and on the *Cityscapes* dataset in Figure 3. We picked the first images in the dataset 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 4, Table 2).

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

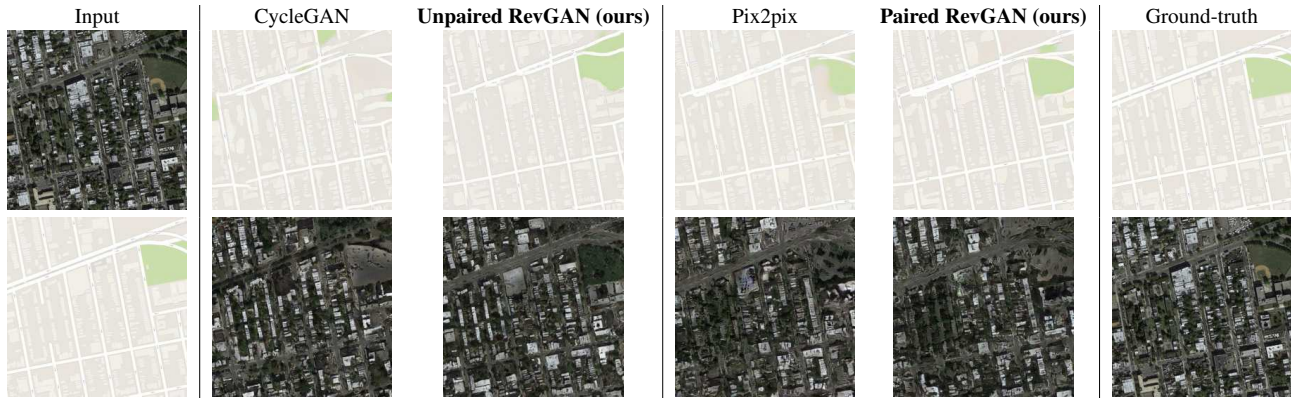| Input | CycleGAN | **Unpaired RevGAN (ours)** | Pix2pix | **Paired RevGAN (ours)** | Ground-truth |

Figure 4. 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.

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 the supplementary material.

5.2. Quantitative Evaluation

Cityscapes We provide quantitative evaluations of the performance of our RevGAN model on the *Cityscapes* dataset. 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 . In Table 2 the perfor-

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}	$\mathbf{6.68} \pm 0.30$	$\mathbf{11.85} \pm 0.56$
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	$\mathbf{7.82} \pm 0.35$	$\mathbf{13.76} \pm 0.60$
Unpaired (3D-CycleGAN)	17.23 ± 0.73	26.94 ± 1.20
Unpaired+2R	$\mathbf{11.05} \pm 0.51$	$\mathbf{17.76} \pm 1.38$
Unpaired+4R	18.98 ± 1.22	28.06 ± 1.44
Unpaired+8R	18.96 ± 0.85	27.94 ± 1.09

Table 3. Mean and standard deviation of RMSE scores measured on the 8 brains in the *HPC 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.

mance on 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 right-hand columns.

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 5.4), 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 4, 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.

5.3. 3D Volumes

We also evaluate the performance of our RevGAN model on a 3D super-resolution problem, using the *HTC Brains*

Model	Width	Params	maps→satellite			satellite→maps		
			MAE	PSNR	SSIM	MAE	PSNR	SSIM
CycleGAN [†]	32	5.7 M	139.85 ± 15.52	$\mathbf{14.62} \pm 1.16$	0.31 ± 0.05	138.86 ± 20.57	$\mathbf{26.25} \pm 3.64$	0.81 ± 0.06
Unpaired RevGAN	32	1.7 M	$\mathbf{133.57} \pm 18.09$	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	$\mathbf{133.12} \pm 17.18$	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	$\mathbf{139.23} \pm 12.76$	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	$\mathbf{14.91} \pm 1.13$	0.31 ± 0.05	$\mathbf{128.55} \pm 12.71$	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	$\mathbf{27.37} \pm 3.06$	0.82 ± 0.04

Table 4. Image quality on *Maps* dataset. Notice how in most of the experiments that the RevGAN performs better than the baseline. †
Parameter matched architectures

dataset of [28]. As baseline, we use a simple SRCNN model [11] (see supplementary material for architectural details) consisting of a $3 \times 3 \times 3$ convolutional layer as encoder, followed by a $1 \times 1 \times 1$ convolutional layer as decoder. For the discriminator, we use a 3D variant of the PatchGAN also used in [31]. The RevGAN model, extends the architecture by inserting an invertible core between the encoder and the decoder.

As can be seen in Figure 5, we obtain higher quality results 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.

With and without adversarial loss We performed the experiments on paired models with and without the adverarial loss L_{GAN}. We found that models without such loss generally perform better in terms of pixel-distance, 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.

5.4. Introspection

Memory usage In this experiment, we evaluate the GPU memory consumption of our RevGAN model for increasing depths and compare it with a CycleGAN baseline. We kept the widths of both models 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 5, 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 activa-
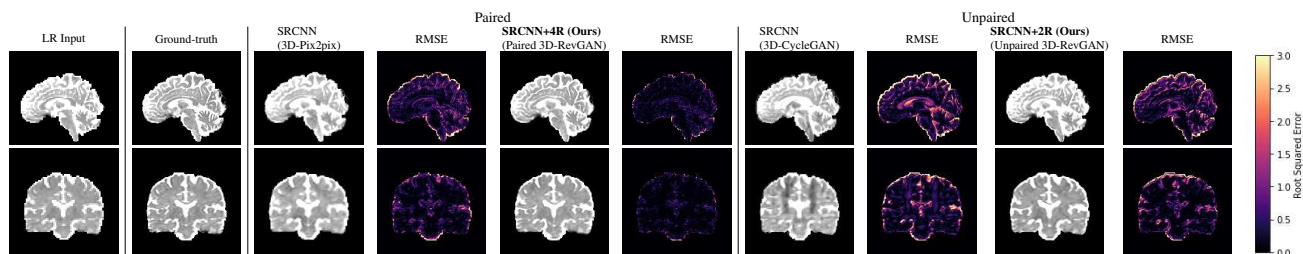

Figure 5. Visualization of mean diffusivity maps on an sagittal slices (top) and axial slices (bottom) of the first brain in the *HCP Brain* test set. From left to right: low-resolution input, high-resolution ground-truth, paired model without reversible layers (SRCNN-3D-Pix2pix), paired model with reversible layers (Paired 3D-RevGAN), unpaired model without reversible layers (SRCNN-3D-CycleGAN) and an unpaired model with reversible layers (Unpaired 3D-RevGAN).
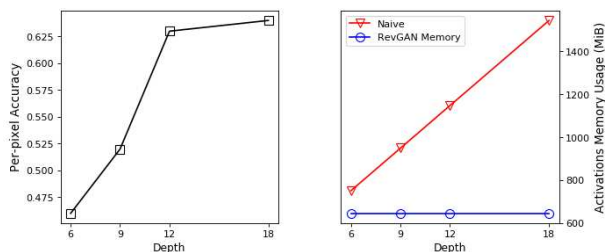
Depth	CycleGAN		Unpaired RevGAN	
	Model	Activations	Model	Activations
6	434.3	+ 752.0	374.4	**+ 646.1**
9	482.3	+ 949.0	385.4	**+ 646.1**
12	530.3	+ 1148.1	398.5	**+ 646.1**
18	626.3	+ 1543.9	423.4	**+ 646.1**
30	818.7	+ 2335.8	626.3	**+ 646.1**

Table 5. 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.

tions becomes particularly important when training models on larger data sizes (e.g. higher image resolutions or increased batch sizes).

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

As shown in Figure 6, 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 [5]. We leave the exploration of alternative reversible architectures to future work.

(a) Per-pixel accuracy against depth.

(b) Memory usage against depth.

Figure 6. Comparison of per-pixel accuracy for a width 64 RevGAN evaluated after 75 epochs and memory usage on *Cityscapes* dataset.

6. Limitations and Discussion

Our results indicate that we can train image-to-image translation models with close to constant memory requirements in depth (see Table 5). 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.

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. We accredit two mechanisms for this observation.

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. A possible way to circumvent using the the cycle-consistency loss is to design the encoders and decoders to be analytically pseudo-invertible. We in fact did experiments on this, by formulating the (strided-)convolutions as Toeplitz matrixvector products [23]. Unfortunately, we found that exact pseudo-invertibility is computationally too slow to run. 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. Conclusion

In this paper 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 simultaneously trains the mapping from Y to X. We demonstrate equivalent or improved performance in terms of image quality, compared to similar non-reversible methods. Additionally, we show that our model is more memory efficient, because activations of reversible residual layers do not have to be stored to perform backpropagation.

In future work we plan to explore techniques to get rid of the cycle-consistency loss, so that the network is automatically cycle-consistent to begin with.

References

[1] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

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

[3] S. B. Blumberg, R. Tanno, I. Kokkinos, and D. 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.

[4] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[5] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. Reversible architectures for arbitrarily deep residual neural networks. *arXiv preprint arXiv:1709.03698*, 2017.

[6] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

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

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. 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.

[9] L. Dinh, D. Krueger, and Y. Bengio. Nice: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[10] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[11] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[12] L. A. Gatys, A. S. Ecker, and M. 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.

[13] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, pages 2214–2224, 2017.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[16] J.-H. Jacobsen, A. Smeulders, and E. Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[17] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[20] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

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

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

[24] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[25] P. Ochs, T. Meinhardt, L. Leal-Taixé, and M. 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.

[26] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible conditional gans for image editing. *CoRR*, abs/1611.06355, 2016.

[27] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[28] R. Tanno, D. E. Worrall, A. Ghosh, E. Kaden, S. N. Sotiropoulos, A. Criminisi, and D. 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.

[29] J. Xie, L. Xu, and E. 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.

[30] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Trans. Computational Imaging*, 3(1):47–57, 2017.

[31] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.