Self-inverse Networks

Anonymous CVPR submission

Paper ID 2944

Abstract

We apply the self-inverse function for one to one paired image to image bidirectional translation with self-inverse networks.For paired images x_i and y_i from domain X and domain Y respectively,these networks not only learn the mapping from x_i to y_i, but also learn the inverse mapping from y_i to x_i with the same networks f.Our goal is to learn a mapping in term of a self-inverse function $f : x_i \rightleftharpoons y_i$.That is to say: the function $f : x_i \rightharpoonup y_i$ and its inverse function $f^{-1} : x_i \leftharpoondown y_i$ satisfies $f = f^{-1}$. This makes it possible to learn only one networks for bidirectional image to image translation.We proposed suitable neural network architectures and some principles for self-inverse networks. We also demonstrate that the advantages of applying self-inverse function for the one to one mapping supervised learning problem.For applications,these networks achieve a better result than the baseline pix2pix [12] network.

1. Introduction

When you go hiking along a route from a point A to a point B and get lost at the point B, the easiest way to back to where you start which is point A is to go back along the same route from point B to point A.This is called an inverse problem in science because it starts with the results and then calculate the causes.It is the inverse of a forward problem which starts with input and then calculates the output.The inverse problems are some of the most important mathematical problems in science and mathematics.It has wide applications in signal processing [18, 5, 6],computer vision [21, 20, 2], medical imaging [1, 24],machine learning [3, 19],natural language processing [22, 25], communication theory [7] and many other fields.

Many problems in computer vision,computer graphic and image processing and natural language processing can be seen as a inverse problems.Obviously, in language translation,if we treat the language A to the language B translation as a forward process, then the language B to the language A translation is its reverse problem.Similarly, In computer vision, there is a concept of image to image translation [12, 31].In medical imaging,there are image reconstruction problems.Traditionally, each of these tasks using two different functions for the forward and its inverse processes.Our goal in this paper is to prove that the same one function is able to do the two processes at the same time for the one to one mapping problem.

Before we go further, Let's review the definition of self-inverse function.

> The definition of self-inverse function
>
> A self inverse function is a function f, such that $y = f(x)$, with the special property that $f(f(x)) = x$, or written another way, $f(x) = f^{-1}(x)$.
>
> Such self-inverse functions are called involutions. Since the function $f : X \to Y$ and its inverse $f^{-1} : Y \to X$coincide for an involution, the domain and codomain must be the same $X = Y$. Moreover, since f must be a bijection (in order to have an inverse), the range must equal the codomain. So in the end, the domain of an involution f must equal its range $f(X) = X$.

The community has explored the power of CNNs in various of tasks in computer vision and many other fields.But so far by my best knowledge, no one has has explored the usage and applications of CNNs to learn a self-inverse function. Why should we learn a self-inverse function?

There are several advantages:

1.In the perspective of application, it has the effect of killing two birds with one stone and it is a novel way for multi-task learning.That is to say with this only one neural network, it can generate output given input and vice versa.It is able to do both tasks with itself simultaneously.

2.It automatically doubles the sample size which is a good fit for data-driven to avoid the over-fitting problem.It expands the domain range by combining each two domains into one codomain.In other word, It transform a one to one bidirectional mapping problem to a self mapping problem.

3.It implicitly shrinks the target function space.Given a neural network architecture, its function space is very huge with thousands of parameters. When it is trained to learn a

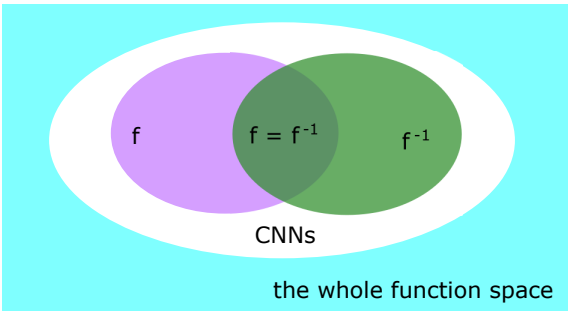

Figure 1. Functions space.Blue area:the whole function space;White area:the function space of a CNNs;Purple area:the function space of f;Green area:the function space of f^{-1}; Overlap area:the function space of $f = f^{-1}$

self-inverse function.As shown in Figure 1,the target function sapce is the overlapping area which is a subset of of the function space of f and f^{-1}.Function f and f^{-1} are the target function space when the CNNs is to learn without a self-inverse constrain.

4.Training a neural network in a self-inverse way will have an advantage of implicit deep supervision and U-Net is one of the perfect architecture for self-inverse networks.Let's take the U-Net network architecture as an example. Each layer or a block of convlution-BatchNorm-ReLU can be treated as a function

$$f_i$$

, the i is the index of the i-th layer or the i-th block.Then the function f is

$$f : y_i = f_n(...(f_2(f_1(f_0(x_i)...) - - - (1)$$

And its inverse function f^{-1}

$$f^{-1} : x_i = f_n(...(f_2(f_1(f_0(y_i)...) - - - (2)$$

According to the property of inverse fucntion, from equation(1), we can derive its inverse function f^{-1}

$$f^{-1} : x_i = f_0^{-1}(...(f_{n-2}^{-1}(f_{n-1}^{-1}(f_n^{-1}(y_i)...) - - - (3)$$

. Compare eqution (2) and (3), we can assume

$$f_i = f_{n-i}^{-1}$$

This means f_i and f_{n-i} are each others' inverse function.According to the property of the inverse function, the output of f_i should be the input of f_{n-i}. As the U-Net skip-connection between the i-th block and (n-i)-th block,the U-Net skip-connection can feed the output of f_i to f_{n-i} as input.This makes U-Net is one of the good architecture for self-inverse networks.The implicit deep supervision here means:

(i) the block i and the block n-i are each others' inverse function.

(ii) the feature maps at block i and block n-i has a fixed input-output correlation because of the U-Net skip connection.

In this paper, we apply the self-inverse networks for image to image bidirectional translation.We used parts of the datasets and the same evaluation metrics in "pix2pix" [12].Our code is available at https://github.com/ALISCIFP/Self-Inverse-Network.

2. Related work

Image to Image Translation

The concept of image to image translation is a broad concept.It includes image style transfer, translation between image and semantic labels,greyscale to color, edge-map to photograph,super resolution [16] and many image manipulations.It dates back to image analogies by Herzmann et al [11],which employs a nonparametric texture model [8] from a single input-output training image pair. More recent approaches use a dataset of input-output examples to learn a parametric translation function using CNNs [17]. Our approach builds on the pix2pix framework of Isola et al [12], which uses a conditional generative adversarial network [9] to learn a mapping from input to output images. Similar ideas have been applied to various tasks such as generating photographs from sketches [23] or from attribute and semantic layouts [13]. However, unlike these prior works,we learn this translation with one network in a bidirectional way instead of using two separate networks for each direction.

3. Method

Our goal is to learn a self-inverse mapping function or bidirectional mapping function f for pairs (x_i, y_i). This means

$$f : x_i \rightleftharpoons y_i$$

It also can illustrated in this way: the function f

$$f : x_i \rightharpoonup y_i$$

and its inverse function f^{-1}

$$f^{-1} : y_i \rightharpoonup x_i$$

satisfies

$$f = f^{-1}$$

and where samples $\{x_i\}_{i=1}^N \in X$ and $\{y_i\}_{i=1}^N \in Y$. The symbol $\rightleftharpoons$ means bidirectional mapping; The symbol $\rightharpoonup$ means one directional mapping; The $=$ means the two functions on both sides are exactly the same function.

3.1. Objectives and network architectures

We adapt the architecture for our self-inverse networks from "pix2pix" [12] which have shown massive paired image to image translation.This network contains two stride-2 convolutions, several residual blocks [10], and two fractionally strided convolutions with stride $1/2$. We use 6 blocks for 128×128 images, and 9 blocks for 256×256 and higher resolution training images. Similar to "pix2pix" [12], we use instance normalization [27]. For the discriminator networks we use 70×70 PatchGANs [12, 14, 15], which aim to classify whether 70×70 overlapping image patches are real or fake. Such a patch-level discriminator architecture has fewer parameters than a full-image discriminator, and can be applied to arbitrarily-sized images in a fully convolutional fashion [12].

We used pix2pix GAN [12] and Cycle GAN [31] as our baseline. For fair comparison with the baseline, we adapted the same network structure and and Loss function including L1 loss and adversarial loss. The differences compared to pix2pix [12] are as follows:

In pix2pix [12] "U-net" archtecure, All ReLUs in the encoder are leaky, with slope 0.2, while ReLUs in the decoder are not leaky.In the architecture of self-inverse networks, there are two new designs to make the network architecture more symmetric,since self-inverse function is a self- symmetric function.

- "U-Net" architecture with all ReLUs in both encoder and decoder.
- "U-Net" architecture with all ReLUs are leaky with slope 0.2 in both encoder and decoder.

As shown in figure 1. The self-inverse network has smaller target function space compared to both function f and its inverse function f^{-1}. If the CNNs function space is not large enough, there might no overlap for function space of f and the function space of its inverse function f^{-1}.One way to solve this problem is to design a wider CNNs,that is to say, increase the channel number of the CNNs architecture. In pix2pix [12] "U-net" architecture,

- "U-Net" architecture with all convolutions kernels' input and output channels number doubled.
- "U-Net" architecture with all convolutions kernels' input and output channels number halved.

3.2. Training details

To train a CNNs as a self-inverse netowrk, randomly sample batch size pairs (x_i, y_i) and (y_i, x_i) alternatively and iteratively. In the bracket, the first item and the second item feed as the input and the ground truth respectively.We refer this as the alternative training. For fair comparsion with the "pix2pix" [12], we adapt the same training technique as the "pix2pix" [12]. For the same dataset, we use the same number of epoch and batch size. In othere words, except for the alternative training, everythin is the same as the baseline "pix2pix" [12]. Random jitter was applied by resizing the 256×256 input images to 286×286 and then randomly cropping back to size 256×256.

All netowrks were trained from scratch.The weights were initialized from a Gaussian distribution with mean 0 and standard deviation 0.02.

4. Experiments

The self-inverse function is a general mathematical function. Implementing self-inverse function with CNNs is also a general CNNs for many applcation. Due to time and computation resources limitation,we mainly explore the generality of self-inverse network in paired image to image translation. We test the method on a variety of tasks and datasets, including both graphics tasks, like photo generation, and vision tasks, like semantic segmentation:

- Semantic labels $\leftrightarrow$ photo, trained on the Cityscapes dataset [4].
- Architectural labels $\leftrightarrow$ photo, trained on the CMP Facades dataset [26].
- Map $\leftrightarrow$ aerial photo, trained on data scraped from Google Maps.
- Edges $\leftrightarrow$ photo, trained on data from [30] and [28]; binary edges generated using the HED edge detector [29] plus postprocessing.

5. Results and analysis

Note: Here only shows quantive result for the cityscape datasets. More results for other dataset will be in the supplementary

5.1. Comparison between network architectures

- Semantic labels $\leftrightarrow$ photo, trained on the Cityscapes dataset [4].

  For this experiment, we compared two architectures for self-inverse networks: U-Net and Encoder-Decoder.For fair comparison between these two architectures, The only difference between these two architecture are the skip connection between each layer i in the encoder and layer n-i in the decoder. where n is the total number of layers. In Table 1 and 2, En-Decoder is the Encoder-Decoder architecture.En-DecoderS is the Encoder-Decoder architecture with self-inverse.Ours is the "pix2pix" [12] with self inverse. From the two quantitative FCN score and Semantic segmentation metrics, The performance for "pix2pix" with

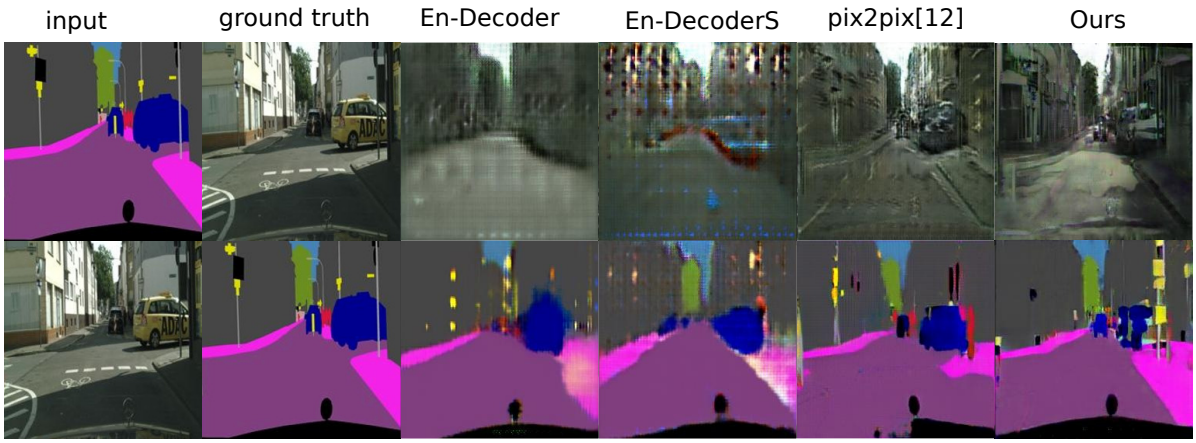

Figure 2. Comparison against baselines on Semantic labels ↔ photo task of Cityscapes dataset [4]

Method	Per-pixel acc.	Per-class acc.	Class IOU
En-Decoder	0.53	0.15	0.12
En-DecoderS	0.50	0.14	0.11
pix2pix [12]	0.63	0.21	0.16
Ours	**0.67**	**0.22**	**0.17**
Ground truth	0.8	0.26	0.21

Table 1. FCN-scores for different architectures, evaluated on cityscapes labels↔photos.

Method	Per-pixel acc.	Per-class acc.	Class IOU
En-Decoder	0.62	0.20	0.15
En-DecoderS	0.58	0.13	0.13
pix2pix [12]	0.83	0.36	0.22
Ours	**0.84**	**0.38**	**0.26**

Table 2. Performance of photo→labels on cityscapes with different architectures

U-Net is increased with self-inverse alternative training.While the performance for Encoder-Decoder is decreased with self-inverse alternative training. This result fits for the theoretical analysis of U-Net architecture in the introdcution section.

5.2. Comparison with all RelU and all LReLU

In the Ours method, All ReLUs in the encoder are leaky with slope 0.2 while ReLUs in the decoder.All ReLU is the "U-Net" architecture with all ReLUs in encoder and decoder.All LReLU is the "U-Net" architecture with all leaky ReLUs with slope 0.2 in encoder and decoder.As mentioned in the introduction for the second advantage,the domain range of the datasets for self inverse networks is a combination of the samples from X and Y.Obviously and intuitively according to the property of the self inverse function and the fourth advantage in the introduction section, the U-Net should have a symmetric architecture with all ReLUs and Leaky ReLUs in both the encoder and the decoder. As shown in Table 3 and Table 4, The symmetric architecture with all ReLUs or all Leaky ReLUs are better than the ours which has asymmetric architecture.

Method	Per-pixel acc.	Per-class acc.	Class IOU
All ReLU	0.68	0.22	0.17
All LReLU	**0.70**	**0.23**	**0.18**
Ours	0.67	0.22	0.17
Ground truth	0.8	0.26	0.21

Table 3. FCN-scores for different ReLUs, evaluated on cityscapes labels↔photos.

Method	Per-pixel acc.	Per-class acc.	Class IOU
All ReLu	0.85	0.39	0.27
All LReLU	**0.87**	**0.40**	**0.28**
Ours	0.84	0.38	0.26

Table 4. Performance of photo→labels on cityscapes for different ReLUs.

5.3. Comparison with double wider and half thinner architecture

In the Ours method, All ReLUs in the encoder are leaky with slope 0.2 while ReLUs in the decoder.As mentioned in the third advantage in the introduction section,For a fixed dadaset, If the model is too small or that is to say the function space of CNNs is not large enough, the chance that the function space of the function f and the function space of its inverse function f^{-1} have no overlap.In this case, there is no target function or self inverse function exist in this CNNs function space.In the end, the trained model will be biased.From table 5 and table 6,the double wider architecture increase the performance in both directions. But this

Method	Per-pixel acc.	Per-class acc.	Class IOU
Half	0.65	0.20	0.15
Double	**0.68**	**0.23**	**0.17**
Ours	0.67	0.22	0.17
Ground truth	0.8	0.26	0.21

Table 5. FCN-scores for networks double wider and half thinner, evaluated on cityscapes labels↔photos.

Method	Per-pixel acc.	Per-class acc.	Class IOU
half	0.82	0.35	0.25
double	**0.85**	**0.39**	**0.26**
Ours	0.84	0.38	0.26

Table 6. Performance of photo→labels on cityscapes with double wider and half thinner architecures

won't always necessarily be true. If the sample size is relatively small, the error of over-fitting from the wider networks might overwhelm the benefit from it for self-inverse networks.

5.4. Evaluation

We use the same evaluation datasets metrics as "pix2pix" [12]. we compare our method against several baseline.we also compare different architectures for our method.The tasks include semantic labels↔photo on the Cityscapes dataset [4], and map↔aerial photo on data scraped from Google Maps.

5.4.1 Metrics

AMT perceptual studies On the mapaerial photo task, we run real vs fake perceptual studies on Amazon Mechanical Turk (AMT) to assess the realism of our outputs. We follow the same perceptual study protocol from Isola et al. [12], except we only gather data from 25 participants per algorithm we tested. Participants were shown a sequence of pairs of images, one a real photo or map and one fake (generated by our algorithm or a baseline), and asked to click on the image they thought was real. The first 10 trials of each session were practice and feedback was given as to whether the participants response was correct or incorrect. The remaining 40 trials were used to assess the rate at which each algorithm fooled participants. Each session only tested a single algorithm, and participants were only allowed to complete a single session. Note that the numbers we report here are not directly comparable to those in [12] as our ground truth images were processed slightly differently and the participant pool we tested may be differently distributed from those tested in [12] (due to running the experiment at a different date and time). Therefore, our numbers should only be used to compare our current method against the baselines (which were run under identical conditions), rather than against [12].

FCN score Although perceptual studies may be the gold standard for assessing graphical realism, we also seek an automatic quantitative measure that does not require human experiments. For this we adopt the FCN score from [12], and use it to evaluate the Cityscapes labelsphoto task. The FCN metric evaluates how interpretable the generated photos are according to an off-the-shelf semantic segmentation algorithm (the fully-convolutional network, FCN, from [17]). The FCN predicts a label map for a generated photo. This label map can then be compared against the input ground truth labels using standard semantic segmentation metrics described below. The intuition is that if we generate a photo from a label map of car on road, then we have succeeded if the FCN applied to the generated photo detects car on road.

Semantic segmentation metrics To evaluate the performance of photolabels, we use the standard metrics from the Cityscapes benchmark, including per-pixel accuracy, per-class accuracy, and mean class Intersection-Over-Union (Class IOU) [4]

6. Discussion and conclusion

The results in this paper suggest that self-inverse networks is a good method for paired image to image bidirectional translation.After a neural network architecture is defined for image to image translation in one direction, The simplest way in practice to apply self-inverse network is to apply the alternative training method mentioned above..But theoretically, self-inverse function is a general function in mathematics and not any CNNs architecure is suitable for self-inverse networks.In this paper we domenstated in theroy and experiments that the U-Net is one of the perfect choice to realize self-inverse function with CNNs. Since the large number of parameters in CNNs provide enough function space to satisfy the strict requirement of a self -inverse function.The scalability and and huge function space of CNNs is a good choice for solving bidirectional translation problem not just image to image translation by using self-inverse function.

7. Potential limitation and future work

Since one to one mapping is a requirement for inverse function to exist, theoretically, for image to image translation, the self-inverse network can only apply to paired image to image translation.But if you want to apply self-inverse network to unpaired image to image translation,there are two ways:

1.Instead of treating the samples from the two domains as image-wise unpaired, treat patches from two domains as paired.That is to say,train the self-inverse networks with

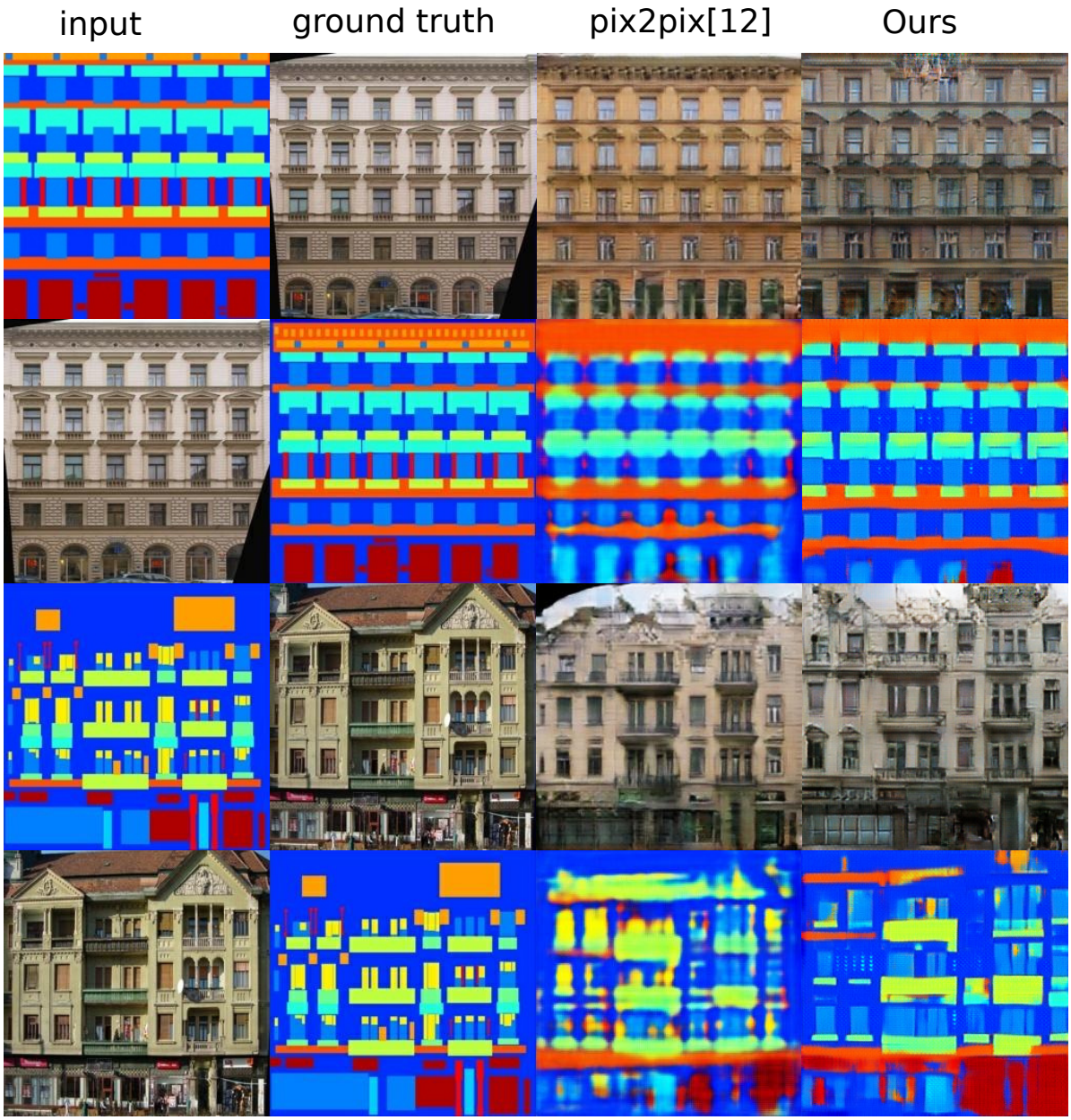

Figure 3. Comparison against baselines on Semantic labels ↔ photo task of facades dataset [26]

patch-wise for unpaired image to image translation problem. 2.Use the dropout in self-inverse networks. With dropout the self-inverse network can works out as one to many mapping for unpaired image to image translation.Becuase dropping cna make the neural networks as a integration of many self-inverse networks.

Another missing import work is the architecture improvement for self-inverse networks.As mentioned in the fourth advantage in the introduction section,f_i and f_{n-i} are each others' inverse function, and the output of f_i should feed as input of f_{n-i}.This is realized by the skip connection in the U-Net.The skip connections concatenate activations from layer i to layer n-i. But there is a problem, the output of layer n-i doesn't feed as input to layer i. Due to time limitation. This architecture design hasn't been implemented yet.

In all, We are the first to use CNNs to realize self-inverse function and apply it to the paired image to image translation.There are still much more to explore the power and beauty of self-inverse networks.The neural networks archi-

tecture,like U-Net and its improvement version I proposed here is two example of the best architectures for self-inverse networks, but there is definitely more to explore.

References

[1] S. Arridge. Optical tomography in medical imaging. *Inverse problems*, 1999. 1

[2] A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. *Pattern Recognition, 1996.,*, 1996. 1

[3] E. Candes and J. Romberg. Sparsity and incoherence in compressive sampling. *Inverse problems*, 2007. 1

[4] 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. 3, 4, 5

[5] D. Donoho. Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data. *In Proceedings of Symposia in Applied Mathematics*, 1993. 1

[6] M. Ebrahimi and E. Vrscay. Solving the inverse problem of image zooming using self-examples. *International Conference Image Analysis and*, 2007. 1

[7] S. Efromovich and V. Koltchinskii. On inverse problems with unknown operators. *IEEE Transactions on Information*, 2001. 1

[8] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999. 2

[9] 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. 2

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3

[11] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001. 2

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 2, 3, 4, 5

[13] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016. 2

[14] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 3

[15] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016. 3

[16] D. Liu, Z. Wang, B. Wen, J. Yang, W. Han, and T. S. Huang. Robust single image super-resolution via deep networks with sparse prior. *IEEE Transactions on Image Processing*, 25(7):3194–3207, 2016. 2

[17] 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. 2, 5

[18] S. Mallat. *A wavelet tour of signal processing*. 1999. 1

[19] K. Mosegaard and M. Sambridge. Monte Carlo analysis of inverse problems. *Inverse problems*, 2002. 1

[20] G. Peyré, S. Bougleux, and L. Cohen. Non-local regularization of inverse problems. *Computer VisionECCV 2008*, 2008. 1

[21] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *nature*, 1985. 1

[22] J. Pollack. Implications of recursive distributed representations. *Advances in neural information processing systems*, 1989. 1

[23] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. *arXiv preprint arXiv:1612.00835*, 2016. 2

[24] A. Sarvazyan, O. Rudenko, and S. Swanson. Shear wave elasticity imaging: a new ultrasonic technology of medical diagnostics. *Ultrasound in medicine*, 1998. 1

[25] M. Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. *Proceedings of the second international conference on*, 1993. 1

[26] R. Tyleček and R. Šára. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, pages 364–374. Springer, 2013. 3, 6

[27] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 3

[28] S. "Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015. 3

[29] A. Yu and K. Grauman. Fine-Grained Visual Comparisons with Local Learning. In *Computer Vision and Pattern Recognition (CVPR)*, June 2014. 3

[30] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 3

[31] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017. 1, 3