The Reading Report of Invertible Residual Networks

Yaqi Duan
South China University of Technology, Guangzhou, China
duanyaqiduanyaqi@gmail.com

Abstract

This article is a personal reading report[1] of the Invertible Residual Networks. Different from the existing network architecture on the flow-based model, this paper propose a new invertible network based on the ResNet. This invertible network retains the easy-to-handle density estimation and non-volume preserving which is the flow-based model's advantages while eliminating the limits of the dimension division. Invertible ResNet uses a Lipschitz constraint to ingeniously solve the three problems of making the network invertible, calculating the network's inverse function, and log-determinant approximation. The invertible ResNet not only performs well in the generation task, but also shows the potential in the discrimination task.

1. Information

This article is accepted by International Conference on Machine Learning, 2019. Invertible Residual Networks [2] is part of the innovation of Deep Learning Architectures system. The authors are Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, Jörn-Henrik Jacobsen from Universität Bremen, Germany, and Vector Institute, University of Toronto, Canada. By the way, Ricky T. Q. Chen, David Duvenaud are the authors of NeurIPS 2018 Best Paper-Neural Ordinary Differential Equations [3]. And the Vector Institute is created by Geoffrey Hiton, received the 2018 Turing Award, together with Yoshua Bengio and Yann LeCun, for their work on deep learning.They are sometimes referred to as the "Godfathers of AI" and "Godfathers of Deep Learning".

2. Background

First think about a question. Why do we need a invertible model? The answer is a invertible model means information lossless, and the model can be used to make a better classification network, we can also use the maximum likelihood directly as a generative model. The work done in this article is based on the very powerful ResNet [10], which means that it may have better performance than the previous flow-based model Glow [11] .

All in all, if a model is invertible , with low cost of reversibility and the strong ability of transform, then it can be used in discriminant tasks, density estimation, generation tasks and so on.

2.1. Invertible Network Architecture

Existing invertible network architectures, such as NICE [4], RealNVP [5], and Glow [11], achieve the reversibility of the model through clever dimensional decomposition of network architecture design. And by constraining the Jacobian matrix to be a triangular matrix to ensure the calculation cost of the determinant calculation, to further calculate the exactly log-likelihood. Though these model are elegant and beautiful in theory, there is a serious problem: because of the simple inverse transformation process, and the constraint of the Jacobian determinant, it will lead to weak nonlinear transformation capabilities in each layer.

The state-of-the-art flow-based model like Glow [11] , only transform a half of the variables in each layer. In order to ensure sufficient fitting capabilities, Glow [11] are higher requirements for the depth of the model (e.g. for 256-dimensional face generation tasks, the model uses about 600 convolutional layers, with a parameter amount of 200 M), and the calculation of this part is very large. However, invertible ResNet [2] doesn't require the deep number of network layers, it can still maintain sufficient transform capabilities with its architecture.

2.2. Neural Ordinary Differential Equations

The view of deep networks as dynamic over time provides two basic learning methods. One of them use discrete architecture to directly learn dynamics (Haber et al. [8]; Ruthotto et al. [13]); the other (Chen et al. [3]; Grathwohl et al. [7]) use neural network parameterizes ordinary differential equations to indirectly learn dynamics. They usually regard the reversible transformation as a dynamic system that is approximated by an ordinary differential equation (ODE)

[1]For more details about my own i-ResNet report. Please see: https://github.com/DuanYaQi/iResNet.

solver.

The state-of-the-art NFs model Continuous Normalizing Flows method (Chen et al. [3]) use the Jacobian trace instead of the Jacobian determinant. The specific method is to parameterize the linear transformation process through an ordinary differential equation. But compared with invertible ResNet [2], it does not have a fixed calculation budget. As shown in Figure 1, the difference between the continuous dynamics of linearly invertible ResNet and the continuous dynamics of standard ResNet is explained. Invertible ResNets are bijective along continuous paths, while regular ResNets may result in crossing and collapsing paths.

3. Methodology

The basic module of the ResNet model is

$$y = x + g(x) \triangleq F(x) \tag{1}$$

where x is the input feature, which is the identity mapping part, $g(x)$ is the feature transform, which is the residual part, and y is the output feature. We call the Equation (1) a ResNet Block. ResNet Block changes the original function y into $y - x$, which the neural network needs to fit, and the x, y are all vectors (tensors).

This operation can ensure that the depth of the network is as large as possible, but there is no need to worry about the gradient dispersion. In a broad sense, ResNet allows the dimensionality to be changed by 1×1 convolution, but here the ResNet does not change the dimensionality, means the input and output dimensions are the same, which is a necessary and sufficient condition for the bijective function.

For the classification problem, we only need to ensure that the network model is reversible, so as to achieve information lossless, see Section 3.1. For the reconstruction of the target, we need to calculate the inverse function of the network model for sampling and inference, see Section 3.2. For using maximum likelihood (the essence of the generation problem) models like flow-based model (e.g. NICE [4], RealNVP [5], and Glow [11]) to train the generative model, then we need to solve the problem of the calculation loss of the Jacobian in the network, see Section 3.3.

3.1. Invertible Condition

Equation 1 is the basic module of ResNet, it is only necessary to ensure that each module is invertible . The sufficient condition for reversibility of Equation 1 is

$$\text{Lip}(g) < 1 \tag{2}$$

where $\text{Lip}(g) < 1$ is the Lipschitz norm of the function g, and

$$\text{Lip}(g) \triangleq \max_{x_1 \neq x_2} \frac{\|g(x_1) - g(x_2)\|_2}{\|x_1 - x_2\|_2} \tag{3}$$

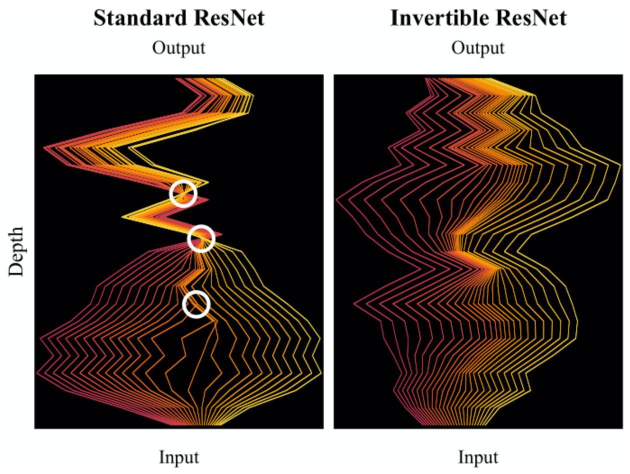

Figure 1. The dynamics of the standard ResNet network (left) and iResNet (right). Invertible ResNets describes a bijective continuous dynamics, while ResNets result in crossing and folding paths (circled in white) corresponding to non-bijective continuous dynamics. Due to the collapsing path, ResNets is not an effective density model.

The g is a neural network, the neural network is a combination of matrix operations and activation functions,

$$g(x) = \sigma(Wx + b) \tag{4}$$

It is parameterized by the non-linear activation function σ, and the affine transformation is represented by the weight W and the deviation b. According to Equation 3, we have:

$$\|\sigma(Wx_1 + b) - \sigma(Wx_2 + b)\| \leq C(W, b) \cdot \|x_1 - x_2\| \tag{5}$$

where $C(W, b)$ is an expression about W and b, and its value is Lipschitz constant. Let $\Delta x \to 0$ and use the first-order term approximation to get

$$\left\| \frac{\partial \sigma}{\partial x} W(x_1 - x_2) \right\| \leq C(W, b) \cdot \|x_1 - x_2\| \tag{6}$$

According to the chain rule, the sufficient condition for $\text{Lip}(g) < 1$ is that the Lipschitz-Norm of σ is not more than 1 and the Lipschitz-Norm of $Wx + b$ is less than 1, so we have Condition 7

$$\text{Lip}(\sigma) < 1 \quad \text{and} \quad \text{Lip}(Wx + b) < 1 \tag{7}$$

3.1.1 Lip(σ) < 1

To satisfies Condition 7. For the $\partial\sigma/\partial x$ term in Equation (6), currently commonly used activation functions (such as sigmoid, tanh, relu, elu, etc.) all satisfy the constraint that the derivative has upper and lower bounds.

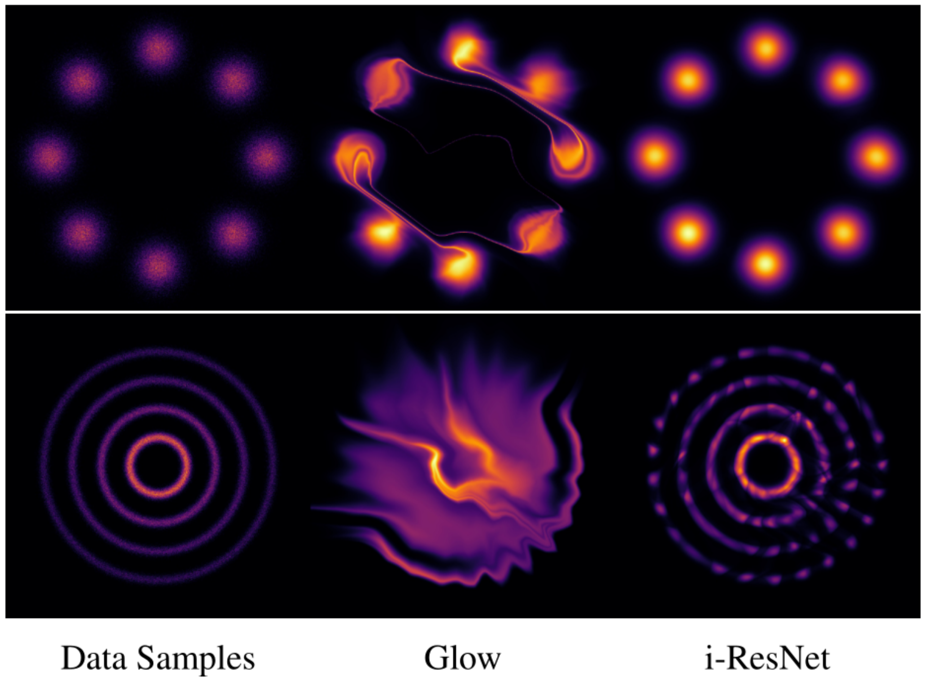

Figure 2. Visual comparison of i-ResNet flow and Glow [11].

3.1.2 Lip$(Wx + b) < 1$

This term means that the Lipschitz-Norm of the matrix W is required to be less than 1. Where the Lipschitz-Norm of the matrix W is the Spectral-Norm, denoted as Lip(W) or $\|W\|_2$.

To satisfies Condition 7. The Lipschitz-Norm of $Wx+b$ is less than 1, for the $W(x_1 - x_2)$ term in Equation (6), then we have

$$\|W(x_1 - x_2)\| \leq C\|x_1 - x_2\| \tag{8}$$

Now, the problem has been transformed into a matrix norm problem. The matrix norm is equivalent to the magnitude of the vector, which is defined as

$$\|W\|_2 = \max_{x \neq 0} \frac{\|Wx\|}{\|x\|} \tag{9}$$

Equation (9) also known as spectral norm, where $\|Wx\|$ and $\|x\|$ are the norms of vectors, the magnitude of the normal vector. According to Equation (9), we convert Equation (8) into

$$\|W(x_1 - x_2)\| \leq \|W\|_2 \cdot \|x_1 - x_2\| \tag{10}$$

Note: I also find that Equation (10) can also be proved by Cauchy's inequality transformation of Frobenius norm. And we can get that $\|W\|_F$ provides an upper bound of $\|W\|_2$. If accuracy is not required, you can also directly use $\|W\|_2 \rightarrow \|W\|_F$ to approximate and simplify calculation.

Theorem 1

The spectral norm $\|W\|_2$ is equal to the square root of the max eigenvalue (primary eigenvalue) of $W^\top W$. If W is a square matrix, then $\|W\|_2$ is equal to The absolute value of the max eigenvalue of W. The proofs are given in Appendix [A].

According to Theorem 1, the problem is to find the max eigenvalue of $W^\top W$. invertible ResNet use the power iteration algorithm (see Appendix [B]) to calculate the max eigenvalue of $W^\top W$. It's also the spectral norm $\|W\|_2$, which takes the square of the spectral norm $\|W\|_2$ as an additional regular term

$$loss = loss(y, g(x)) + \lambda\|W\|_2^2 \tag{11}$$

The above Equation (11) want to reduce to the smallest possible term $C = \|W\|_2$, so that the neural network satisfies the constraint of Lip$(Wx + b) < 1$. Invertible Resnet adds this regular term to all the kernel weights W in the model g to make the entire network invertible.

3.2. Invertible Function Calculation

In order to further sample and inference for reconstructing the target. We also need to calculate the inverse function of the network.

Table 1. Comparison of i-ResNet to a ResNet-164 baseline architecture of similar depth and width with varying Lipschitz constraints via coefficients c. Vanilla shares the same architecture as i-ResNet, without the Lipschitz constraint.

		ResNet-164	Vanila	c=0.9	c=0.8	c=0.7	c=0.6	c=0.8
Classfication Error %	MNIST	-	0.38	0.40	0.42	0.40	0.42	0.86
	CIFAR10	5.50	6.69	6.78	6.86	6.93	7.72	8.71
	CIFAR100	24.30	23.97	24.58	24.99	25.99	27.30	29.45
Guaranteed Inverse		No	No	Yes	Yes	Yes	Yes	Yes

Assume $y = x + g(x)$ is reversible, its implicit inverse function is $x = h(y)$, which is essentially to solve a system of nonlinear equations. The following iterative form is used in invertible Resnet

$$x_{n+1} = y - g(x_n) \tag{12}$$

Obviously, the iterative sequence $\{x_n\}$ is related to y, and once $\{x_n\}$ converges to a fixed function

$$\lim_{n\to\infty} x_n(y) = \hat{h}(y) \tag{13}$$

we will have $\hat{h}(y) = yg\left(\hat{h}(y)\right)$, which means that $\hat{h}(y)$ is the requested $x = h(y)$.

So if Equation (12) converges, then the result of convergence is the inverse function of $x + g(x)$. We have

$$\forall x_1, x_2, \quad \|g(x_1) - g(x_2)\|_2 \leq \text{Lip}(g)\|x_1 - x_2\|_2 \tag{14}$$

So according to Equation (2), we have

$$\begin{aligned} \|x_{n+1} - x_n\|_2 &= \|g(x_n) - g(x_{n-1})\|_2 \\ &\leq \text{Lip}(g)\|x_n - x_{n-1}\|_2 \\ &= \text{Lip}(g)\|g(x_{n-1}) - g(x_{n-2})\|_2 \\ &\leq \text{Lip}(g)^2\|x_{n-1} - x_{n-2}\|_2 \\ &\dots \\ &\leq \text{Lip}(g)^n\|x_1 - x_0\|_2 \end{aligned} \tag{15}$$

According to Banach's fixed point theorem, we have a sufficient condition of $\|x_{n+1} - x_n\|_2 \to 0$ is $\text{Lip}(g) < 1$. The content of Cauchylie is involved, and the proofs are given in Appendix [C].

When the normalization operation is done to make $x + g(x)$ reversible, its inverse function is the fixed point of $x_{n+1} = y - g(x_n)$. In the numerical calculation, only need to use the Equation (12) iterative format to iterate a certain number of steps, so that the accuracy satisfy the requirements, and the convergence result can be obtained, which is the inverse function of $x + g(x)$.

3.3. Jacobian Determinant Calculation

In order to use the maximum likelihood (the essence of the generation problem) as the loss function to train the generative model, for learning the powerful generative ability. It is necessary to calculate the Jacobian determinant. First, calculate the Jacobian matrix:

$$J_F \triangleq \frac{\partial}{\partial x}(x + g(x)) = I + \frac{\partial g}{\partial x} \triangleq I + J_g \tag{16}$$

The Jacobian determinant is $\det(J_F) = \det(I + J_g)$, but in fact, when training generative models, we alway calculate the logarithm of the absolute value of the Jacobian determinant,which is

$$\ln|\det(J_F)| = \ln|\det(I + J_g)| \equiv \ln\det(I + J_g) \tag{17}$$

The reason for eliminating the absolute value here is that the disturbance of the Lipschitz constraint identity makes $x+g(x)$ produce a constant positive determinant.

Theorem 2

$$\ln\det(\boldsymbol{A}) = \text{Tr}(\ln(\boldsymbol{A})) \tag{18}$$

The proofs are given in Appendix [A].

We can't calculate the Jacobian by Equation (17), because the amount of calculation is too large, and when backpropagating, the derivative of the determinant must be calculated, which is more complicated. Invertible Resnet came up with a cumbersome but effective solution, and uses Theorem 2 to get the following formula

$$\ln\det(I + J_g) = \text{Tr}(\ln(I + J_g)) \tag{19}$$

Expand $\ln(I + J_g)$ by the logarithmic expansion of the real series as:

$$\ln(I + J_g) = \sum_{n=1}^{\infty}(-1)^{n-1}\frac{J_g^n}{n} \tag{20}$$

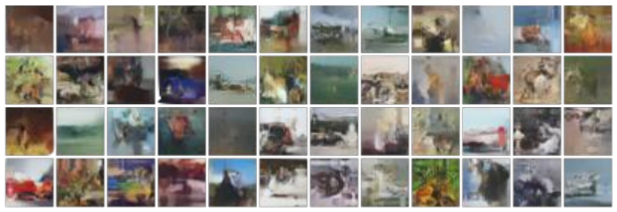

Figure 3. CIFAR10 samples from our i-ResNet flow.

The condition for the convergence of the series is $\|J_g\|_2 < 1$, $\text{Lip}(g) < 1$, which is exactly the constraint of invertible Resnet.

Now $\ln(I + J_g)$ is an infinite series. If the n term is truncated, then the error is proportional to $\text{Lip}(g)^n$, so invertible Resnet is based on $\text{Lip}(g)$ to determine the number of truncation. we can further write:

$$\text{Tr}(\ln(I + J_g)) = \sum_{n=1}^{N} (-1)^{n-1} \frac{\text{Tr}(J_g^n)}{n} + O\left(\text{Lip}(g)^N\right) \tag{21}$$

The Equation (21) needs to calculate J_g^n. But the J_g is a matrix, and the cost of calculating the matrix to the power of n is too huge. Therefore, invertible Resnet assumes that for any matrix A, there is

$$\text{Tr}(A) = \mathbb{E}_{u \sim p(u)}\left[u^\top A u\right] \tag{22}$$

where $p(u)$ is a multivariate probability distribution, its mean is 0, and the covariance is the identity matrix. Invertible Resnet randomly selects a vector u from $p(u)$ for each iteration, and then make that $u^\top Au$ is $\text{Tr}(A)$, i.e. transforms Equation (21) to Equation (23)

$$\text{Tr}(\ln(I + J_g)) \approx \sum_{n=1}^{N} (-1)^{n-1} \frac{u^\top J_g^n u}{n}, \quad u \sim p(u) \tag{23}$$

where

$$u^\top J_g^n u = u^\top J_g(\dots(J_g(J_g u))) \tag{24}$$

Therefore, we don't need to calculate J_g^n, and each step we only need to calculate the multiplication of a matrix and a vector, and each step of the calculation can be reused, so the cost of calculation is greatly reduced. Invertible Resnet first perform logarithmic expansion of the Jacobian matrix with Equation (20), and then convert the determinant calculation into the trace calculation with Equation (21), and use the probability sampling method with Equation (22), the Jacobian can be calculated most efficiently.

Table 2. MNIST and CIFAR10 bits/dim results. † Uses ZCA preprocessing making results not directly comparable.

Method	MNIST	CIFAR10
NICE (Dinh et al. [4])	4.36	4.48†
MADE (Germain et al. [6])	2.04	5.67
MAF (Papamakarios et al. [12])	1.89	4.31
Real NVP (Dinh et al. [5])	1.06	3.49
Glow (Kingma & Dhariwal [11])	1.05	3.35
FFJORD (Grathwohl et al. [7])	0.99	3.40
i-ResNet [2]	1.06	3.45

4. Results

4.1. Toy Dataset Experiment

The Toy data set constructs some regular random points. The purpose is to use a generative model to fit its distribution. Such experiments are often done in GAN. From the ablFigure 2, you can see that the effect of invertible Resnet [2] is much better than Glow [11]. I think it's because invertible Resnet is a very symmetrical model with no bias in dimension, while Glow is biased in dimension. It requires us to shuffle the input in some way, and then split it in half. Different, which brings about the problem of asymmetry.

4.2. Discrimination Task Experiment

Since invertible Resnet is a network based on ResNet, its most basic use is for classification task. Table 1 shows that using invertible Resnet [2] for classification tasks, the effect is also very good, the existence of Lipschitz constraints will not significantly affect the classification results (c equals $\text{Lip}(g)$).

Where ResNet-164 is a network with an architecture similar to invertible Resnet, and vanilla represents ResNet without Lipschitz constraints. It is observed that when a very conservative normalization is applied (when c is very small), the classification error of all tested data sets will be higher, but when c is large, espically it approaches 1, which is compared with ordinary ResNet-64 is extremely competitive.

4.3. Generative model experiment

The state-of-the-art flow-based model series is poorer than GAN in terms of generating complex pictures, but the quantitative comparison between them has no obvious advantages and disadvantages. We compare invertible Resnet with it, replace the coupling layer in the flow-based model with reversible residual blocks that have the same number of parameters, and replace the reversible linear transformation with Actnorm (Kingma et al. [11]). Figure 3 shows that the invertible Resnet is also excellent as a flow-based generative model.

The comparison with the results of other generative models can be found in Table 2. Although the invertible Resnet is not as good as Glow [11] and FFJORD [7], it is surprising that ResNets can be created to compete with these highly engineered generative models with very few modifications.

5. Conclusion

Before invertible Resnet, one of the most successful feed-forward architectures for discriminative learning was the deep residual network [10], but this architecture is very different from the corresponding generative model. This division complicates the selection or design of the appropriate architecture for a given task. And invertible Resnet proposes a new architecture with good performance that can solve this problem and bridges this gap. And it is not limited to specific network structure design. The whole work is very violent. Through a Lipschitz constant, the three problems of the reversible maximum likelihood generation model are solved. There is a whole sense of beauty in it.

However, there is still a gap between the results of individual networks in the discrimination task and the generation task. I think this is because the author is calculating the matrix trace Equation [21] using the method of probability sampling Equation [22], only taking a vector u randomly from the multivariate probability distribution $p(u)$, which leads to a biased estimate of the logarithmic determinant [9]. I personally think that if this part of the estimate can be processed by an unbiased method [1], it may improve certain performance.

References

[1] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019. 6

[2] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019. 1, 2, 5

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018. 1, 2

[4] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 1, 2, 5

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 1, 2, 5

[6] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015. 5

[7] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. 1, 5, 6

[8] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017. 1

[9] Insu Han, Haim Avron, and Jinwoo Shin. Stochastic chebyshev gradient descent for spectral optimization. *arXiv preprint arXiv:1802.06355*, 2018. 6

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[11] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018. 1, 2, 3, 5, 6

[12] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017. 5

[13] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2019. 1

A. Additional Lemmas and Proofs

Proof. (**Theorem 1**)

According to Equation 9, we have

$$\|W\|_2^2 = \max_{x\neq 0} \frac{x^\top W^\top W x}{x^\top x} = \max_{\|x\ Vert=1} x^\top W^\top W x \quad (25)$$

Suppose $W^\top W$ is diagonalized to diag$(\lambda_1, \dots, \lambda_n)$, that is, $W^\top W = U^\top \text{diag}(\lambda_1, \dots, \lambda_n)U$, where λ_i are all its characteristic roots and non-negative, where U is an orthogonal matrix, due to the product of orthogonal matrix and unit vector Or unit vector, then

$$\begin{aligned}\|W\|_2^2 &= \max_{\|x\|=1} x^\top \text{diag}(\lambda_1, \dots, \lambda_n)x \\ &= \max_{\|x\|=1} \lambda_1 x_1^2 + \dots + \lambda_n x_n^2 \\ &\leq \max\{\lambda_1, \dots, \lambda_n\}(x_1^2 + \dots + x_n^2) \quad (\text{Note:}\|x\| = 1) \\ &= \max\{\lambda_1, \dots, \lambda_n\}\end{aligned} \quad (26)$$

Thus $\|W\|_2^2$ is equal to the largest characteristic root of $W^\top W$.

Proof. (**Theorem 2**)

We consider the scalar parameter t

$$f(t) = \det(\exp(t\boldsymbol{A})) \quad (27)$$

Then find its derivative to get

$$\begin{aligned}\frac{d}{dt} f(t) =& f(t)\text{Tr}\left(\exp(-t\boldsymbol{A}) \underbrace{fracddt \exp(t\boldsymbol{A})}_{=\exp(t\boldsymbol{A})\boldsymbol{A}}\right) \\ =& f(t)\text{Tr}(\boldsymbol{A})\end{aligned} \quad (28)$$

Where Tr$(\boldsymbol{A})$ is a constant, we get a first-order homogeneous differential equation. Its analytical solution is

$$f(t) = C \exp(t\, \text{Tr}(\boldsymbol{A})) \quad (29)$$

When $t = 0$, we have $f(0) = 1$, and further calculations get $C = 1$, that is, $f(t) = \exp(t\, \text{Tr}(\boldsymbol{A}))$, which proves

$$\det(\exp(t\boldsymbol{A})) = \exp(t\, \text{Tr}(\boldsymbol{A})) \quad (30)$$

When $t = 1$, the identity $\det(\exp(\boldsymbol{A})) = \exp(\text{Tr}(\boldsymbol{A}))$ is obtained. Take the logarithm on both sides, you can get another common form of it

$$\ln \det(\boldsymbol{B}) = \text{Tr}(\ln(\boldsymbol{B})) \quad (31)$$

B. Power iterations Algorithm

Power iteration through the above iteration format

$$u \leftarrow \frac{(W^\top W)u}{\|(W^\top W)u\|} \quad (32)$$

After several iterations, the following formula is finally passed

$$\|W\|_2^2 \approx u^\top W^\top W u \quad (33)$$

Get the norm, which is the approximate value of the largest characteristic root. It can also be equivalently rewritten as

$$v \leftarrow \frac{W^\top u}{\|W^\top u\|},\ u \leftarrow \frac{Wv}{\|Wv\|}, \quad \|W\|_2 \approx u^\top W v \quad (34)$$

In this way, after initializing u, v (it can be initialized with a vector of all 1s), you can iterate several times to get u, v, and then substituting $u^\top W v$ into $\|W\|_2$ approximation.

C. Banach's Fixed Point Theorem

For any positive integer k, we continue to consider $\|x_{n+k} - x_n\|_2$:

$$\begin{aligned}\|x_{n+k} - x_n\|_2 &\leq \|x_{n+k} - x_{n+k-1}\|_2 + \dots \\ &\quad + \|x_{n+2} - x_{n+1}\|_2 + \|x_{n+1} - x_n\|_2 \\ &\leq \left(\text{Lip}(g)^{n+k-1} + \dots + \text{Lip}(g)^n\right) \|x_1 - x_0\|_2 \\ &= \frac{1 - \text{Lip}(g)^k}{1 - \text{Lip}(g)} \cdot \text{Lip}(g)^n \|x_1 - x_0\|_2 \\ &\leq \frac{\text{Lip}(g)^n}{1 - \text{Lip}(g)} \|x_1 - x_0\|_2\end{aligned} \quad (35)$$

It can be seen that we have got an upper bound of $\|x_{n+k} - x_n\|_2$, which is only related to n and can be arbitrarily small. In other words, for any $\varepsilon > 0$, we can find a n such that for any positive integer k there is $\|x_{n+k} - x_n\|_2 < varepsilon$. We call this sequence of numbers the Cauchy sequence, and it is bound to converge. At this point, we finally proved the convergence.

By the way, in the above formula, taking $k \to \infty$, we get:

$$\|x^* - x_n\|_2 \leq \frac{\text{Lip}(g)^n}{1 - \text{Lip}(g)}\ Vert x_1 - x_0\|_2 \quad (36)$$

In other words, the convergence speed of this iterative algorithm is proportional to Lip$(g)^n$, so naturally, the smaller the Lip(g), the faster the convergence, but

The smaller $\mathrm{Lip}(g)$, the weaker the fitting ability of the model. In the original paper, its range is 0.5□0.9.

To be more ambitious, this is actually the Banach fixed point theorem in functional analysis, also known as the compression mapping theorem (because $\mathrm{Lip}(g)$ is less than 1, so g is called Is a compression map.).