3rd Symposium on Advances in Approximate Bayesian Inference, 2020 1–11

Invertible DenseNets

Yura Perugachi-Diaz Y.M.PERUGACHIDIAZ@VU.NL
Jakub M. Tomczak J.M.TOMCZAK@VU.NL
Sandjai Bhulai S.BHULAI@VU.NL
Vrije Universiteit Amsterdam, Netherlands

Abstract

We introduce Invertible Dense Networks (i-DenseNets), a more parameter efficient alternative to Residual Flows. The method relies on an analysis of the Lipschitz continuity of the concatenation in DenseNets, where we enforce the invertibility of the network by satisfying the Lipschitz constraint. Additionally, we extend this method by proposing a learnable concatenation, which not only improves the model performance but also indicates the importance of the concatenated representation. We demonstrate the performance of i-DenseNets and Residual Flows on toy, MNIST, and CIFAR10 data. Both i-DenseNets outperform Residual Flows evaluated in negative log-likelihood, on all considered datasets under an equal parameter budget.

1. Introduction

Neural networks are frequently used in supervised learning tasks such as classification, where models are trained to predict labels. However, they are also used to parameterize generative models that try to estimate the true distribution of the observed data. Generative models can be used to generate realistic-looking images that are hard to separate from real ones, detection of adversarial attacks (Fetaya et al., 2019; Jacobsen et al., 2018), and for hybrid modeling (Nalisnick et al., 2019) which have the property to both classify and generate.

The generative architecture come in different designs. A common approach to train generative models is using the likelihood objective. One kind of model that also uses this approach are flow-based models. Flow-based models consist of invertible transformations that allow them to compute the likelihood using the change of variable formula. The main difference that determines an exact computation or approximation of a flow-based model, lies in the design of the transformation layer. The design used to make this layer invertible can consist of the exact computation of the inverse or a numerical technique. For example, (Dinh et al., 2016) use coupling layers that consist of functions stacked on each other to make the flow invertible. This allows an exact computation while modeling complex convolutional neural networks that do not require the computation of the derivative.

Recently, Behrmann et al. (2018) have proposed deep-residual blocks as a transformation layer. The deep-residual networks (ResNets) of (He et al., 2016) are known for their successes in supervised learning approaches. In a ResNet block, each input of the block is added to the output, which forms the input for the next block. Since ResNets are not necessarily invertible, Behrmann et al. (2018) enforce the Lipschitz constraint in such a manner that the network becomes invertible. Furthermore, Chen et al. (2019) proposed Residual Flows, an improvement of i-ResNets, that uses an unbiased estimator of the log-determinant, which results in even better performance.

In supervised learning, an architecture that uses fewer parameters and is even more powerful than the deep-residual network, is the Densely Connected Convolution Network (DenseNet), which was first presented in (Huang et al., 2017). The network showed to improve significantly in recognition tasks on benchmark datasets such as CIFAR, SVHN, and ImageNet, by using fewer computations and having fewer parameters than ResNets while performing at a similar level. In contrary to a ResNet block, a DenseNet layer consists of a concatenation of the input with the output. In this work, we introduce invertible Dense Networks (i-DenseNets), and we show that we can enforce the Lipschitz constraint in a similar manner as in (Behrmann et al., 2018). Further, we show that this model can be efficiently trained as a generative model and outperforms Residual Flows under an equal parameter budget.

2. Background

Let us consider a vector of observable variables $x \in \mathbb{R}^d$ and a vector of latent variables $z \in \mathbb{R}^d$. We define a bijective function $f : \mathbb{R}^d \to \mathbb{R}^d$ which maps a latent variable to datapoint $x = f(z)$. If f is invertible, we define its inverse as $F = f^{-1}$. Further, we use the change of variable formula to compute the likelihood of a datapoint x by:

$$\ln p_X(x) = \ln p_Z(z) + \ln |\det J_F(x)|, \tag{1}$$

where $p_Z(z)$ is a base distribution (e.g., the standard Gaussian) and $J_F(x)$ is the Jacobian of F at x. The change of variable formula allows tractable evaluation of the data and the flows are trained using the maximum likelihood objective.

Behrmann et al. (2018) construct an invertible ResNet layer which is only constraint in Lipschitz continuity. A ResNet is defined as: $F(x) = x + g(x)$, where g is modeled by a (convolutional) neural network and F represents a ResNet layer which is in general not invertible. However, they construct g in such way to satisfy Lip(g) < 1 by using spectral normalization of (Gouk et al., 2018; Miyato et al., 2018), such that:

$$\text{Lip}(g) < 1, \quad \text{if} \quad ||W_i||_2 < 1, \tag{2}$$

where $||\cdot||_2$ is the ℓ_2 norm. Since the Banach fixed-point theorem holds in this specific case, the ResNet layer F has a unique inverse, even though there does not need to be an analytical closed-form solution. Further, the log-determinant can be estimated using the Hutchinsons trace estimator (Skilling, 1989; Hutchinson, 1990), at a lower cost than to fully compute the trace of the Jacobian. Chen et al. (2019) propose Residual Flows that uses an improved method to estimate the log-determinant with an unbiased estimator.

3. Invertible Dense Networks

We introduce i-DenseNets, an invertible model based on DenseNets parametrizations. To formulate i-DenseNets, we define a DenseBlock as a function $F : \mathbb{R}^d \to \mathbb{R}^d$ with $F(x) = x + g(x)$, where g consists of Dense Layers $\{h_i\}_{i=1}^n$ that are expressed as:

$$g(x) = h_{n+1} \circ h_n \circ \cdots \circ h_1(x), \tag{3}$$

where h_{n+1} represents a 1×1 convolution to match the output size of $\mathbb{R}^d$. A layer h_i consist of two parts concatenated to each other. The upper part is a copy of the input signal. The lower part consist of the transformed input, where the transformation is a multiplication of (convolutional) weights W_i with the input signal, followed by a non-linearity ϕ having $\mathrm{Lip}(\phi) \leq 1$, such as ReLU, ELU, LipSwish, or tanh. As an example, a Dense Layer h_2 can be composed as follows:

$$h_1(x) = \begin{bmatrix} x \\ \phi(W_1 x) \end{bmatrix}, \quad h_2(h_1(x)) = \begin{bmatrix} h_1(x) \\ \phi(W_2 h_1(x)) \end{bmatrix}. \tag{4}$$

3.1. Enforcing Lipschitz constraint

If we enforce the function g to satisfy $\mathrm{Lip}(g) < 1$, the DenseBlock F is invertible and the Banach fixed point theorem holds. As a result, the inverse can be approximated in the same manner as in (Behrmann et al., 2018). To satisfy $\mathrm{Lip}(g) < 1$, we can enforce $\mathrm{Lip}(h_i) < 1$ for all n layers. Therefore, we first need to determine the Lipschitz constant for a Dense Layer h_i. We know that a function f is K-Lipschitz if for all points v and w the following holds (for the full derivation see Appendix A):

$$d_Y(f(v), f(w)) \leq \mathrm{K} d_X(v, w), \tag{5}$$

where we assume that the distance metrics $d_X = d_Y = d$ are chosen to be the ℓ_2-norm. Further, let two functions f_1 and f_2 be concatenated in h:

$$h_v = \begin{bmatrix} f_1(v) \\ f_2(v) \end{bmatrix}, \quad h_w = \begin{bmatrix} f_1(w) \\ f_2(w) \end{bmatrix}, \tag{6}$$

where function f_1 is the upper part and f_2 is the lower part. We can now find an analytical form to express a limit on K for the Dense Layer in the form of Equation (5):

$$\begin{aligned} d(h_v, h_w)^2 &= d(f_1(v), f_1(w))^2 + d(f_2(v), f_2(w))^2, \\ d(h_v, h_w)^2 &\leq (\mathrm{K}_1^2 + \mathrm{K}_2^2) d(v, w)^2, \end{aligned} \tag{7}$$

where we know that the Lipschitz constant of h consist of two parts, namely, $\mathrm{Lip}(f_1) = \mathrm{K}_1$ and $\mathrm{Lip}(f_2) = \mathrm{K}_2$. Therefore, the Lipschitz constant of layer h can be expressed as:

$$\mathrm{Lip}(h) = \sqrt{(\mathrm{K}_1^2 + \mathrm{K}_2^2)}. \tag{8}$$

With spectral normalization of Equation (2), we know that we can enforce (convolutional) weights W_i to be at most 1-Lipschitz. Hence, for all n Dense Layers we apply the spectral normalization on the lower part which locally enforces $\mathrm{Lip}(f_2) = \mathrm{K}_2 < 1$. Further, since we enforce each layer h_i to be at most 1-Lipschitz and we start with h_1, where $f_1(x) = x$, we know that $\mathrm{Lip}(f_1) = 1$. Therefore, the Lipschitz constant of an entire layer can be at most $\mathrm{Lip}(h) = \sqrt{1^2 + 1^2} = \sqrt{2}$, thus dividing by this limit enforces each layer to be at most 1-Lipschitz.

3.2. Learnable concatenation

We have shown that we can enforce an entire Dense Layer to have $\mathrm{Lip}(h_i) < 1$ by applying a spectral norm on the (convolutional) weights W_i and then divide the layer h_i by $\sqrt{2}$. To optimize and learn the importance of the concatenated representations, we create learnable parameters η_1 and η_2 for, respectively, the upper and lower part of each layer h_i. Since the upper and lower part of the layer can be at most 1-Lipschitz, multiplication by these factors results in functions that are at most η_1-Lipschitz and η_2-Lipschitz. From Appendix A we know that the layer is then at most $\sqrt{\eta_1^2 + \eta_2^2}$−Lipschitz. Dividing by this factor results in a bound that is at most 1-Lipschitz.

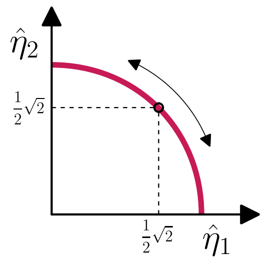


Figure 1: Range of the possible normalized parameters $\hat{\eta}_1$ and $\hat{\eta}_2$.

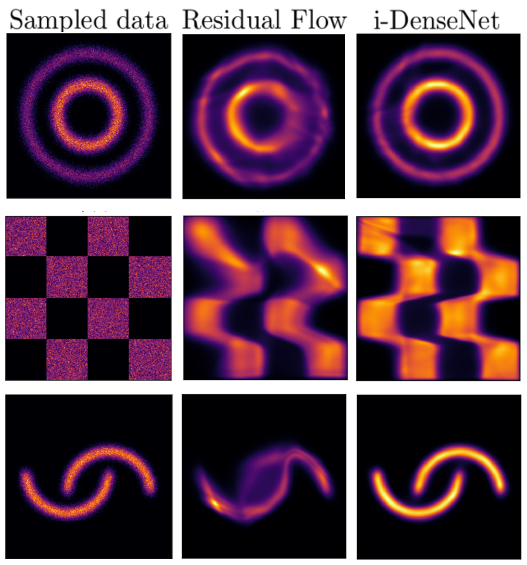
In practice, we initialize η_1 and η_2 at value 1 and during training use a softplus function to avoid them being negative. The range of the normalized parameters is between $0 \leq \hat{\eta}_1, \hat{\eta}_2 \leq 1$ and can be expressed on the unit circle as is shown in Figure 1. In the special case where $\eta_1 = \eta_2$, the normalized parameters are $\hat{\eta}_1 = \hat{\eta}_2 = \frac{1}{2}\sqrt{2}$. This case corresponds to the situation in Section 3.1 where the concatenation was not learned. An additional advantage is that the normalized $\hat{\eta}_1$ and $\hat{\eta}_2$ express the importance of the upper and lower signal. For example, when $\hat{\eta}_1 > \hat{\eta}_2$, the input signal is of more importance than the transformed signal.

4. Experiments

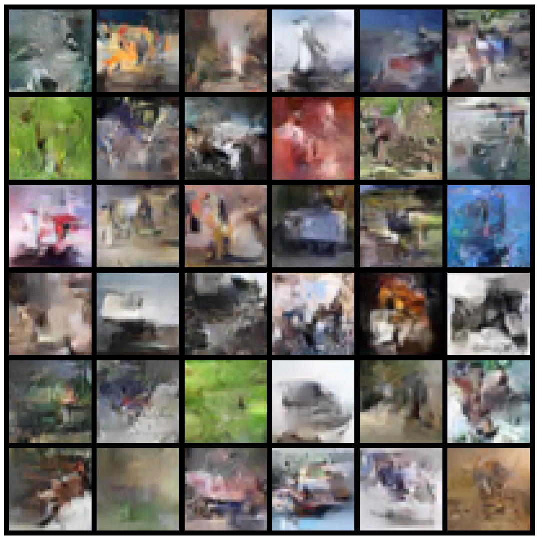
To make a clear comparison between the performance of Residual Flows and i-DenseNets, we train both models on 2-dimensional toy data and on high-dimensional image data, MNIST and CIFAR10. To benchmark the models, we use the architecture of Residual flow (Chen et al., 2019). Since we have a constrained computational budget, we use a smaller architecture of the model and choose number of scales for the toy data and image data set to, respectively, 10 blocks and 4 blocks per 3 scales instead of 100 blocks and 16 blocks per 3 scales. For the other arguments, default settings are used. To compare Residual Flows with i-DenseNets, we utilize an architecture that uses a similar number of parameters for each dataset trained on. A detailed description of this architecture can be found in Appendix B. Furthermore, we add the option to learn the parameters of the concatenation. The models trained on toy data were trained for 50,000 iterations (default setting) and on image data for 200 epochs.

4.1. Toy data

We trained the models on different types of 2-dimensional toy data distributions, namely, two circles, a checkerboard, and two moons. The results of the learned density distributions are presented in Figure 2(*a*). We observe that Residual Flows are capable to capture high-probability areas. However, they have trouble with learning low probability regions for two circles and moons. i-DenseNets are capable in capturing all regions of the datasets. Table 1, where the negative log-likelihood for the models are presented, also shows that i-DenseNets with and without learnable concatenation (LC) outperform Residual Flows. The biggest difference in performance is for two moons where i-DenseNets with LC obtain 2.39 nats compared to 2.60 nats for Residual Flows. This is consistent with the density estimation plots where i-DenseNets are better in capturing the data distribution than Residual Flows.



(*a*) Density estimation results after 50,000 iterations of the Residual Flow and i-DenseNet. Trained on 2-dimensional toy data.



(*b*) Samples of i-DenseNet with learnable concatenation.

Figure 2: Results of density estimation for 2-dimensional toy data (a), and samples of the i-DenseNet trained on CIFAR10 (b).

4.2. Image Data

The results of the models trained on MNIST and CIFAR10 data are presented in Table 1. We notice that i-DenseNets outperform Residual Flows in bits per dimension (bpd) on CIFAR10 with 3.41 bpd without LC and 3.39 bpd with LC, against 3.42 bpd for the Residual Flow. Figure 2(*b*) presents samples of the i-DenseNet with LC trained on CIFAR10, in Appendix D more samples of the models can be found. Additionally, in Appendix C a heatmap of the normalized parameters $\hat{\eta}_1$ and $\hat{\eta}_2$ of the learnable concatenation is presented.

During training on MNIST the original Residual Flow suffered from unstable results. This might be due to the coefficient for the spectral normalization, which controls the Lipschitz constraint. In return, this leads to an unstable Jacobian determinant estimation. We adjusted the Lipschitz coefficient for the spectral normalization by setting it to 0.93 for all models. Additionally, the concatenation in DenseNets is multiplied by 0.98. Due to slight fluctuations, the results are averaged over the last 5 epochs, which are presented in Table 1. We observe that i-DeseNets without and with LC outperform the Residual Flow with respectively 1.05 bpd and 1.04 bpd against 1.08 bpd of the Residual Flow. In general we observe that i-DenseNets with LC outperform Residual Flows and i-DenseNets without LC. On two moons, the performance of i-DenseNets with and without LC are tied.

Model	2 circles	Checkerboard	2 moons	MNIST	CIFAR10
Residual Flow	3.44	3.81	2.60	1.08	3.42
Invertible DenseNet	3.32	3.68	**2.39**	1.05	3.41
Invertible DenseNet+LC	**3.30**	**3.66**	**2.39**	**1.04**	**3.39**

Table 1: Negative log-likelihood results on test data in nats (toy data) and bits per dimension (MNIST and CIFAR10). i-DenseNets with and without learnable concatenation are compared with the Residual Flow.

5. Conclusion

We introduced i-DenseNets, a parameter efficient alternative to Invertible ResNets. Our method enforces invertibility by satisfying the Lipschitz continuity in Dense Layers. In addition, we proposed a version where the concatenation is learned during training, which also indicates which representations are used. We used a smaller architecture under an equal parameter budget, where we demonstrated the performance of i-DenseNets and compared these models to Residual Flows on toy, MNIST, and CIFAR10 data. In conclusion, both i-DenseNets with fixed and learnable concatenation outperformed Residual Flows in negative log-likelihood.

References

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks, 2018.

Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2016.

Ethan Fetaya, Jörn-Henrik Jacobsen, Will Grathwohl, and Richard Zemel. Understanding the limitations of conditional generative models, 2019.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael Cree. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.

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.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid Models with Deep and Invertible Features. In *International Conference on Machine Learning*, pages 4723–4732, 2019.

John Skilling. The eigenvalues of mega-dimensional matrices. In *Maximum Entropy and Bayesian Methods*, pages 455–466. Springer, 1989.

Appendix A. Derivation of Lipschitz constant K for the concatenation

We know that a function f is K-Lipschitz if for all points v and w the following holds:

$$d_Y(f(v), f(w)) \leq \mathrm{K} d_X(v, w), \tag{9}$$

where d_Y and d_X are distance metrics and K is the Lipschitz constant.

Consider the case where we assume to have the same distance metric $d_Y = d_X = d$ and where the distance metric is assumed to be chosen as any p-norm, where $p \geq 1$, for vectors: $||\delta||_p = \sqrt[p]{\sum_{i=1}^{len(\delta)} |\delta_i|^p}$. Further, we assume a DenseBlock to be a function h where the output for each data point v and w is expressed as follows:

$$h_v = \begin{bmatrix} f_1(v) \\ f_2(v) \end{bmatrix} = \begin{bmatrix} a_v \\ b_v \end{bmatrix}, \quad h_w = \begin{bmatrix} f_1(w) \\ f_2(w) \end{bmatrix} = \begin{bmatrix} a_w \\ b_w \end{bmatrix}, \tag{10}$$

where in this paper for a Dense Layer and for a data point x the function $f_1(x) = x$ and f_2 expresses a linear combination of (convolutional) weights with x followed by a non-linearity, for example $\phi(W_1 x)$. We can re-write Equation (9) for the DenseNet function as:

$$d(h_v, h_w) \leq \mathrm{K} d(v, w), \tag{11}$$

where K is the unknown Lipschitz constant for the entire DenseBlock. However, we can find an analytical form to express a limit on K. To solve this, we know that the distance between h_v and h_w can be expressed by the p-norm as:

$$d(h_v, h_w) = \sqrt[p]{\sum_{i=1}^{len(h_v)} |h_{v,i} - h_{w,i}|^p}, \tag{12}$$

where we can simplify the equation by taking the p-th power:

$$d(h_v, h_w)^p = \sum_{i=1}^{len(a_v)} |a_{v,i} - a_{w,i}|^p + \sum_{i=1}^{len(b_v)} |b_{v,i} - b_{w,i}|^p. \tag{13}$$

Since we know that the distance of a can be expressed as:

$$d(a_v, a_w) = \sqrt[p]{\sum_{i=1}^{len(a_v)} |a_{v,i} - a_{w,i}|^p}, \tag{14}$$

which is similar for the distance of b, re-writing the second term of Equation (13) in the form of Equation (11) is assumed to be of form:

$$d(a_v, a_w)^p \leq \mathrm{K}_1^p d(v, w)^p, \tag{15}$$

which is similar for b, $d(b_v, b_w)^p \leq \mathrm{K}_2^p d(v, w)^p$. Assuming this, we can find a form of Equation (11) by substituting with Equation (13) and Equation (15):

$$d(h_v, h_w)^p = \sum_{i}^{len(h_v)} |h_{v,i} - h_{w,i}|^p \le \mathrm{K}_1^p d(a_v, a_w)^p + \mathrm{K}_2^p d(b_v, b_w)^p \\ = (\mathrm{K}_1^p + \mathrm{K}_2^p) d(v, w)^p. \tag{16}$$

Now, taking the p-th root we have:

$$d(h_v, h_w) \le \sqrt[p]{(\mathrm{K}_1^p + \mathrm{K}_2^p)} d(v, w), \tag{17}$$

where we have derived the form of Equation (11) and where $\mathrm{Lip}(h) = \mathrm{K}$ is expressed as:

$$\mathrm{Lip}(h) = \sqrt[p]{(\mathrm{K}_1^p + \mathrm{K}_2^p)}, \tag{18}$$

where $\mathrm{Lip}(f_1) = \mathrm{K}_1$ and $\mathrm{Lip}(f_2) = \mathrm{K}_2$, which are assumed to be known Lipschitz constants.

Appendix B. Implementation

We used a smaller architecture of Residual Flows (Chen et al., 2019), with an adjustment of number of blocks per scale set to 4 instead of 16. For training we ensured an equal parameter budget for i-DenseNets. The architecture of i-DenseNets for image data are presented in Table 2. A DenseBlock consist of several Dense Layers. The last Dense Layer h_n is followed by a 1×1 convolution to match the output of size $\mathbb{R}^d$ after which a squeezing layer is applied. The final part of the network consist of a Fully Connected (FC) layer with number of blocks set to 4. Before the concatenation in the FC layer, a Linear layer of input $\mathbb{R}^d$ to output dimension 64 is applied, followed by the Dense Layer with DenseNet growth 32 and activation LipSwish. The DenseNet depth is set to 3. The final part consist of a Linear layer to match the output of size $\mathbb{R}^d$.

Nr. of scales	Nr. of blocks per scale	DenseNet Depth	DenseNet Growth	Dense Layer	Output
3	4	3	108 (MNIST) 124 (CIFAR10)	$\begin{bmatrix} 3\times3 \quad \text{conv} \\ \text{LipSwish} \\ \text{concat} \end{bmatrix}$	$\begin{bmatrix} 1\times1 \quad \text{conv} \end{bmatrix}$

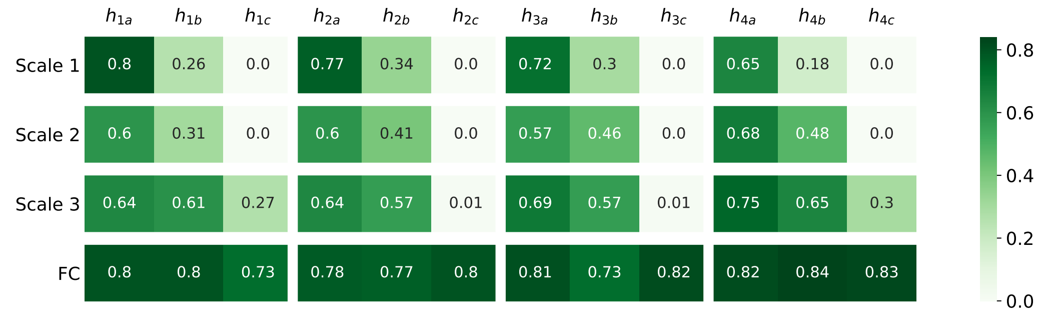
Table 2: The architecture for function g for image data.

Toy data We used 10 scale blocks for all models. Furthermore, we used default settings of Residual Flows. For i-DenseNets, we choose a DenseNet -depth and -growth of, respectively, 4 and 90 with 504K parameters and Residual Flows utilize 501K parameters.

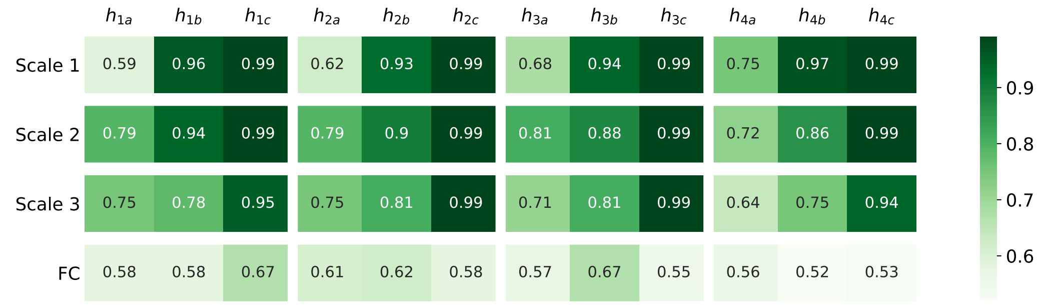
MNIST All models used 3 scales where the number of blocks per scale is set to 4. Due to instability of Residual Flows, we set our coefficient that controls the Lipschitz constraint from 0.98 to 0.93. Furthermore, default settings of Residual Flows are used. For i-DenseNets, we used a coefficient controlling the Lipschitz of the concatenated blocks set to 0.98. i-DenseNets use a DenseNet -depth and -growth of, respectively, 3 and 108 with 5.0M parameters and Residual Flows utilize 5.0M parameters.

CIFAR10 All models used 3 scales where the number of blocks per scale is set to 4. Furthermore, default settings of Residual Flows are used. i-DenseNets use a DenseNet -depth and -growth of, respectively, 3 and 124 with 8.7M parameters and Residual Flows utilize the 8.7M parameters.

Appendix C. Visualization of learnable concatenation



(a) Heatmap of $\hat{\eta}_1$



(b) Heatmap of $\hat{\eta}_2$.

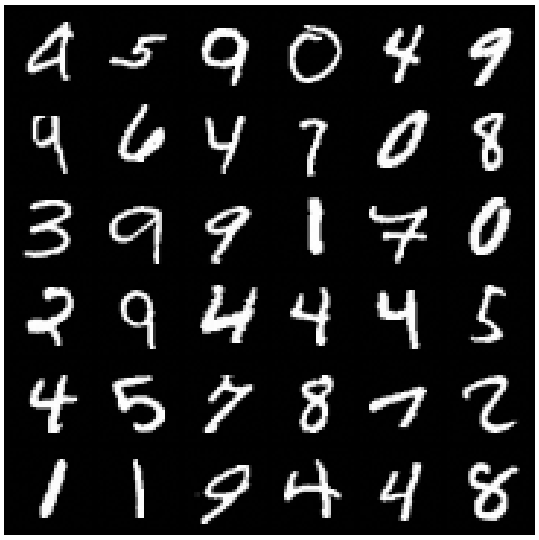
Figure 3: Heatmaps of the normalized η_1 and η_2 after training for 200 epochs on CIFAR10.

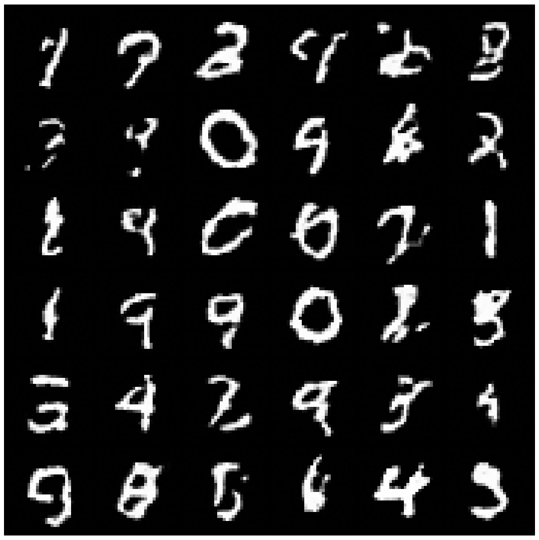
Figure 3 shows the heatmap for (a) the normalized parameter $\hat{\eta}_1$ and (b) normalized parameter $\hat{\eta}_2$ after 200 epochs, trained on CIFAR10. Every scale level 1, 2 and 3 contain 4 DenseBlocks, that each contain 3 Dense Layers with convolutional layers. The final level FC indicates that fully connected layers are used. The letters 'a', 'b', and 'c' index the Dense Layers per block. Remarkably, all scale levels for the last layers h_{ic} give little importance to the input signal. The input signals for these layers are in most cases multiplied with $\hat{\eta}_1$ (close to) zero, while the transformed signal uses almost all the information when multiplied with $\hat{\eta}_2$ which is close to one. This indicates that the transformed signal is of more importance for the network than the input signal. For the fully connected part, this difference is not that pronounced.

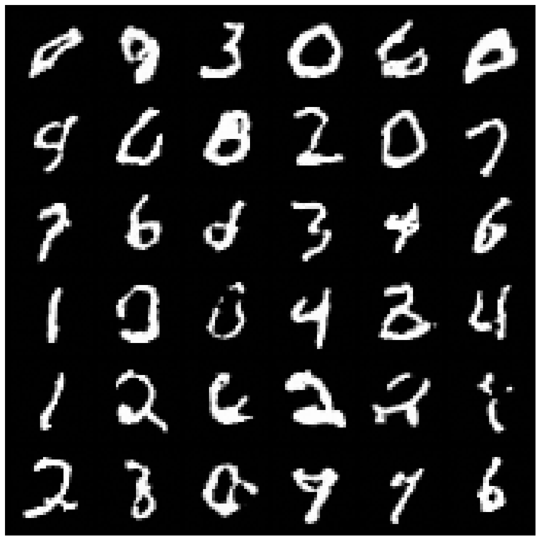
Appendix D. Model samples

This appendix contains real images and samples of the models trained on MNIST (Figure 4). *(a)* shows real images of MNIST, *(b)* shows samples of the Residual Flow trained on MNIST, as well as samples of *(c)* i-DenseNet without LC and *(d)* i-DenseNet with LC.

Figure 5 contains real images and samples of the models trained on images of CIFAR10. *(a)* shows real images of CIFAR10, *(b)* shows samples of the Residual Flow trained on CIFAR10, as well as samples of *(c)* i-DenseNet without LC and *(d)* i-DenseNet with LC.


(*a*) Real images.


(*b*) Samples of the Residual Flow.


(*c*) Samples of i-DenseNet without LC.

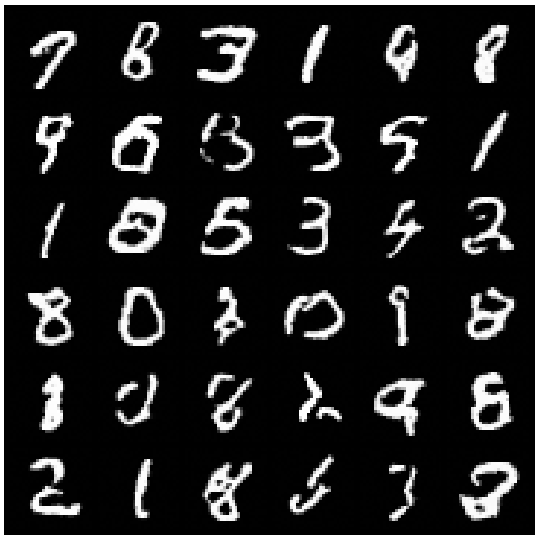
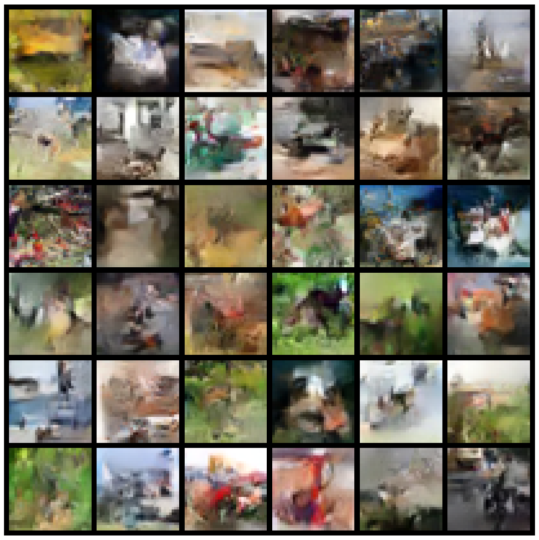
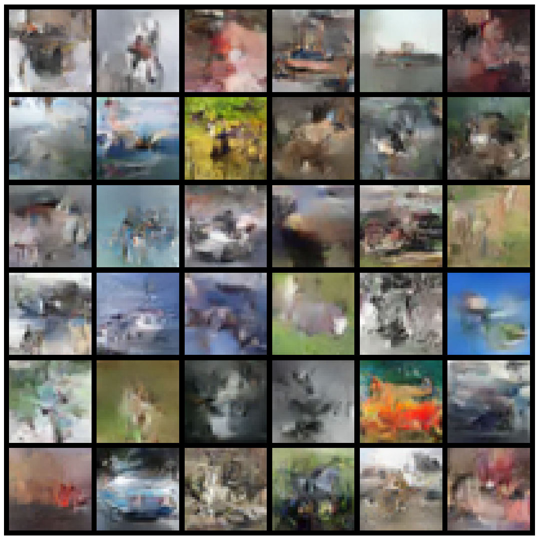

(*d*) Samples of i-DenseNet with LC.

Figure 4: Results on MNIST.

(a) Real images.


(b) Samples of the Residual Flow.


(c) Samples of i-DenseNet without LC.

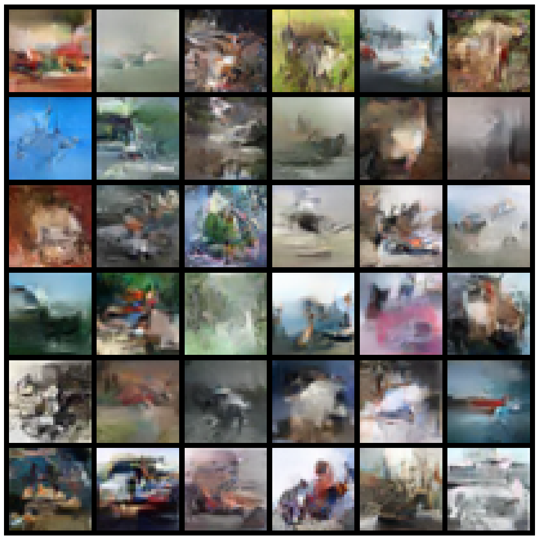

(d) Samples of i-DenseNet with LC.

Figure 5: Results on CIFAR10