Appendix:
Invertible Generative Modeling using Linear Rational Splines

A MONOTONIC LINEAR RATIONAL SPLINES

A.1 Derivative Computation

Using the quotient rule for derivatives, the derivative of a linear rational spline function (as $g(\phi)$ in Eq. (5)) can be computed as:

$$\frac{\mathrm{d}g(\phi)}{\mathrm{d}\phi} = \begin{cases} \dfrac{\lambda^{(k)} w^{(k)} w^{(m)} \left(y^{(m)} - y^{(k)}\right)}{\left(w^{(k)}\left(\lambda^{(k)} - \phi\right) + w^{(m)}\phi\right)^2} & 0 \le \phi \le \lambda^{(k)} \\ \dfrac{\left(1-\lambda^{(k)}\right) w^{(m)} w^{(k+1)} \left(y^{(k+1)} - y^{(m)}\right)}{\left(w^{(m)}(1-\phi) + w^{(k+1)}\left(\phi - \lambda^{(k)}\right)\right)^2} & \lambda^{(k)} \le \phi \le 1 \end{cases} \tag{7}$$

To calculate the derivative with respect to x, we only need to divide Eq. (7) by $\delta^{(k)} = x^{(k+1)} - x^{(k)}$. As can be seen, the derivative of the function $g(x)$ never changes sign, even outside the interval $\left[x^{(k)}, x^{(k+1)}\right]$.

A.2 Inverse Computation

Unlike rational quadratic splines which require computing the root of a degree two polynomial, linear rational splines have a straightforward closed-form inverse. This function is again a linear rational spline, but with different parameters. The inverse of Eq. (5) can be computed as:

$$g^{-1}(y) = \begin{cases} \dfrac{\lambda^{(k)} w^{(k)} \left(y^{(k)} - y\right)}{w^{(k)}\left(y^{(k)} - y\right) + w^{(m)}\left(y - y^{(m)}\right)} & y^{(k)} \le y \le y^{(m)} \\ \dfrac{\lambda^{(k)} w^{(k+1)} \left(y^{(k+1)} - y\right) + w^{(m)}\left(y - y^{(m)}\right)}{w^{(k+1)}\left(y^{(k+1)} - y\right) + w^{(m)}\left(y - y^{(m)}\right)} & y^{(m)} \le y \le y^{(k+1)} \end{cases} \tag{8}$$

Again, this function gives us the value of ϕ in each interval. We should calculate $x = \delta^{(k)}\phi + x^{(k)}$ to translate this into the interval $\left[x^{(k)}, x^{(k+1)}\right]$.

A.3 Inverse Derivative Computation

The derivative of the inverse can be computed using the following relationship:

$$\frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y} = \begin{cases} \dfrac{\lambda^{(k)} w^{(k)} w^{(m)} \left(y^{(m)} - y^{(k)}\right)}{\left(w^{(k)}\left(y^{(k)} - y\right) + w^{(m)}\left(y - y^{(m)}\right)\right)^2} & y^{(k)} \le y \le y^{(m)} \\ \dfrac{\left(1-\lambda^{(k)}\right) w^{(m)} w^{(k+1)} \left(y^{(k+1)} - y^{(m)}\right)}{\left(w^{(k+1)}\left(y^{(k+1)} - y\right) + w^{(m)}\left(y - y^{(m)}\right)\right)^2} & y^{(m)} \le y \le y^{(k+1)} \end{cases} \tag{9}$$

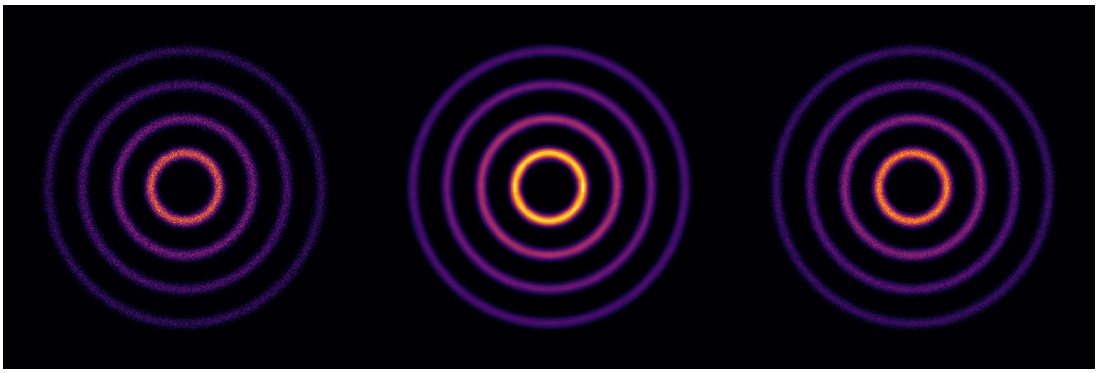
This function captures the change of inverse with respect to ϕ in each interval. To translate this into x, we should multiply this derivative by $\delta^{(k)}$. As in the forward pass, we can see that the derivative of the inverse does not change its sign even outside the interval $0 \le \phi \le 1$.

B DETAILS OF SIMULATION RESULTS

B.1 Synthetic Density Estimation

For the density estimation task on the Rings dataset in Figure 2, we generated a set of 350,000 data points. Then, we used batches of size 512 to train our model, which is a linear rational spline (LRS) flow in the coupling layer mode. The number of coupling layers is 2. For the LRS function of each layer, we used 64 bins and a tail bound of 5. For optimization, we used the Adam (Kingma and Ba, 2015) optimizer, with a learning rate of 0.0005 and cosine annealing (Loshchilov and Hutter, 2017). Finally, a 2-d standard normal was used as the starting probability distribution.

Note that sometimes, it is common to use an infinite data generator, which generates a different set of data at each iteration. We performed our simulation under this condition, too. The results of our method after only 50,000 iterations are depicted in Figure 4.



Data Samples Estimated Density Model Generated Samples

Figure 4: Density estimation on synthetic 2-d data samples using an infinite data generator.

For the results depicted in Figure 3, we used almost the same setup as for the Rings dataset. Here, however, we used a set of 1M data samples and 1.5M iterations. Also, we used a uniform random variable as the starting probability distribution.

B.2 Density Estimation of Real-world Data

In the density estimation of the UCI tabular datasets and BSDS300 (Martin et al., 2001), we used the configurations of Tables 4 and 5 to train our model. Note that here, we used a residual network (He et al., 2016) to determine the parameters of a rational linear spline. Like Durkan et al. (2019b), the Adam optimizer and cosine annealing of the learning rate were used in the optimization of all datasets, except for cases with a † superscript where we used RAdam (Liu et al., 2019) with no annealing. Unlike (Durkan et al., 2019b), which uses a fixed value to clip the norm of gradients, we considered changing it here to see if any of the results would be improved. This value has been shown in the tables under *Maximum Gradient Value* row. Also, batch normalization (Ioffe and Szegedy, 2015) and dropout (Srivastava et al., 2014) were found to be useful in some of the cases.

Also, in Table 6 we have included the results of our model under the hyper-parameters set for rational quadratic spline flows (Durkan et al., 2019b).

B.3 Generative Modeling of Image Datasets

For the generative modeling tasks, we used the Adam optimizer with cosine annealing of the learning rate. The initial learning rate was set to 0.0005. For all datasets, we used batches of size 256, and trained the model for 200k iterations. We followed the multi-scale architecture of Dinh et al. (2017) as used in rational quadratic splines (Durkan et al., 2019b) and Glow (Kingma and Dhariwal, 2018). As in Glow, each layer consists of multiple stacked steps of basic transformations, which are built by using an actnorm, a 1x1 convolution, and a coupling layer. Here, we used rational linear spline functions to build the coupling layer transformation of each

Table 4: Hyper-parameters used for simulations of coupling (C) layer transformations using linear rational splines (Table 1).

PARAMETER	POWER	GAS	HEPMASS	MINIBOONE	BSDS300†
Learning Rate	0.0005	0.0005	0.0005	0.0001	0.0005
Batch Size	512	512	256	128	512
Number of Learning Iterations	400k	400k	400k	200k	500k
Transformation Layers	10	10	20	10	20
Tail Bound	3	3	3	8	3
Number of Bins	8	8	8	4	8
ResNet Layers	2	2	1	1	1
ResNet Hidden Features	256	256	128	32	128
Maximum Gradient Value	5	5	5	25	5
Dropout Probability	0.0	0.1	0.2	0.5	0.5
Batch Normalization	N	N	Y	Y	Y

Table 5: Hyper-parameters used for simulations of autoregressive (AR) transformations using linear rational splines (Table 1).

PARAMETER	POWER	GAS	HEPMASS†	MINIBOONE†	BSDS300†
Learning Rate	0.0005	0.0005	0.0005	0.0001	0.0005
Batch Size	512	512	512	128	512
Number of Learning Iterations	400k	400k	500k	150k	500k
Transformation Layers	10	10	10	10	10
Tail Bound	3	3	5	5	3
Number of Bins	8	8	8	8	8
ResNet Layers	2	2	1	2	2
ResNet Hidden Features	256	256	128	64	512
Maximum Gradient Value	5	5	20	20	5
Dropout Probability	0.0	0.1	0.5	0.5	0.5
Batch Normalization	N	Y	N	Y	Y

Table 6: Test set log-likelihood in nats (higher is better) for four UCI datasets plus BSDS300 (Martin et al., 2001) under parameters set for rational quadratic splines in (Durkan et al., 2019b)

MODEL	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
LRS Flows (C)	0.65 ± 0.01	12.99 ± 0.02	-14.88 ± 0.03	-9.91 ± 0.53	157.56 ± 0.28
LRS Flows (AR)	0.66 ± 0.01	13.05 ± 0.02	-14.37 ± 0.03	-10.63 ± 0.47	157.50 ± 0.28

layer. Moreover, a ResNet with batch normalization was used to determine the parameters of each layer's linear rational spline functions. The detailed configuration used for the simulation of each dataset is given in Table 7.

B.4 Variational Auto-Encoders

For variational auto-encoders, we follow the same procedure as neural spline flows (Durkan et al., 2019b). First, a linear warm-up multiplier is used for the KL-divergence term in the cost function. This multiplier starts at the value 0.5, and then linearly increases to 1 as 10% of the training set passes. A ResNet with 2 blocks determines the parameters of the linear rational splines used in either coupling (C) or autoregressive (AR) transformations. The dimension of the latent space is set to 32, and 64 context features are computed by the encoder.

As before, the Adam optimizer with cosine annealing of an initial 0.0005 learning rate is used for optimization.

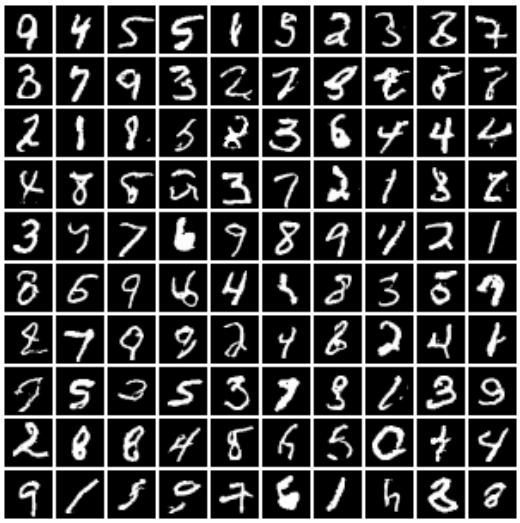
Table 7: Hyper-parameters used for invertible generative modeling simulations of Section 4.3 (Table 2).

PARAMETER	MNIST	CIFAR-10	IMAGENET 32	IMAGENET 64
Learning Rate	0.0005	0.0005	0.0005	0.0005
Batch Size	256	256	256	256
Number of Learning Iterations	200k	200k	200k	200k
Validation Frac. of Train. Set	2%	1%	2%	1%
Multi-scale Transform Levels	2	3	4	4
Num. of Trans. per Layer	7	7	7	7
ResNet Blocks	3	3	3	3
ResNet Hidden Channels	128	96	128	96
Tail Bound	3	3	3	3
Number of Bins	32	4	32	8
Dropout Probability	0.2	0.1	0.2	0.0

We use batches of size 256, and train the model for 150k iterations. Model selection is made using a validation set of 10k and 20k samples for MNIST and EMNIST, respectively.

C IMAGE SAMPLES

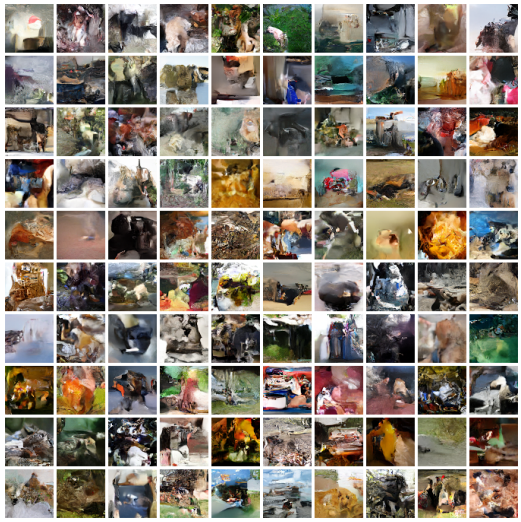
C.1 Random Image Samples Generated by Models Trained for Section 4.3



MNIST

CIFAR-10

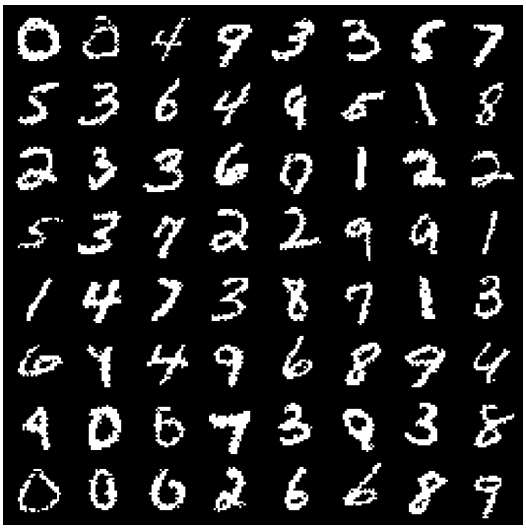
ImageNet 32x32



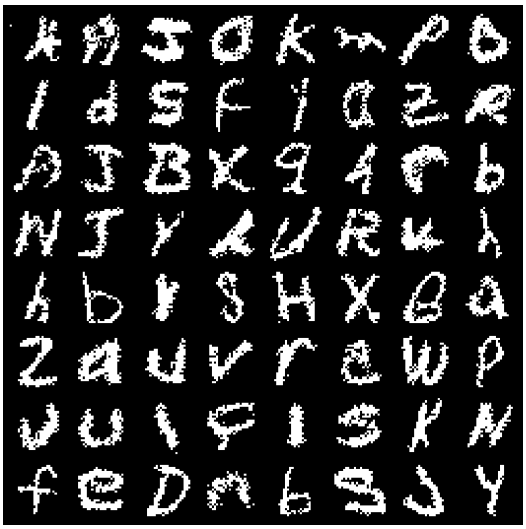
ImageNet 64x64

Figure 5: Random image samples generated by trained linear rational spline flows.

C.2 Randomly Generated VAE Samples



MNIST



EMNIST

Figure 6: Randomly generated VAE image samples. Linear rational spline flows were used as the prior and approximate posterior in these models.

D Further Simulation Results

To highlight the improvements in the current work, we perform a new set of image generation experiments on the MNIST (LeCun, 1998) dataset. Other than using a different family of splines, all of the hyperparameters of the models (summarized in Table 8) are fixed to be the same. For a given depth, the experiment is performed for 8 different seeds. We then train the model and repeat the same procedure for 5 various depths. In each of the experiments we pick the best flow model using a validation set. Finally, we measure the log-likelihood on the test set in BPD.

Table 8: Hyper-parameters used for invertible generative modeling simulations of Fig. 7.

PARAMETER	**MNIST**
Learning Rate	0.001
Batch Size	256
Number of Learning Iterations	50k
Validation Frac. of Train. Set	2%
Multi-scale Transform Levels	1
Num. of Trans. per Layer	$2, 4, 8, 16, 32$
ResNet Blocks	2
ResNet Hidden Channels	96
Tail Bound	3
Number of Bins	4
Dropout Probability	0.2

Figure 7 shows the simulation results. The top-left figure shows the average of log-likelihood on the 8 seeds. As shown, linear rational splines consistently perform better than rational quadratic splines despite using lower degree polynomials. The top-right figure shows the standard deviation of the results across different seeds. As the figure shows, the standard deviation of linear rational splines consistently decreases as the depth increases. However, the results of rational quadratic splines show fluctuations, and for the depth of 32 their standard deviation gets worse. This might be an indication of the fact that since they are using higher degree polynomials, they require more numerical accuracy as the depth increases. In contrast, as a composition of linear rational splines is still a linear rational spline, the standard deviation of our method's results consistently decreases. Finally, you can see the number of parameters, and its relative change in percentages for these simulations in the bottom figures. As the figures show, the increase in number of parameters is only 0.23% for this set of simulations which is negligible.

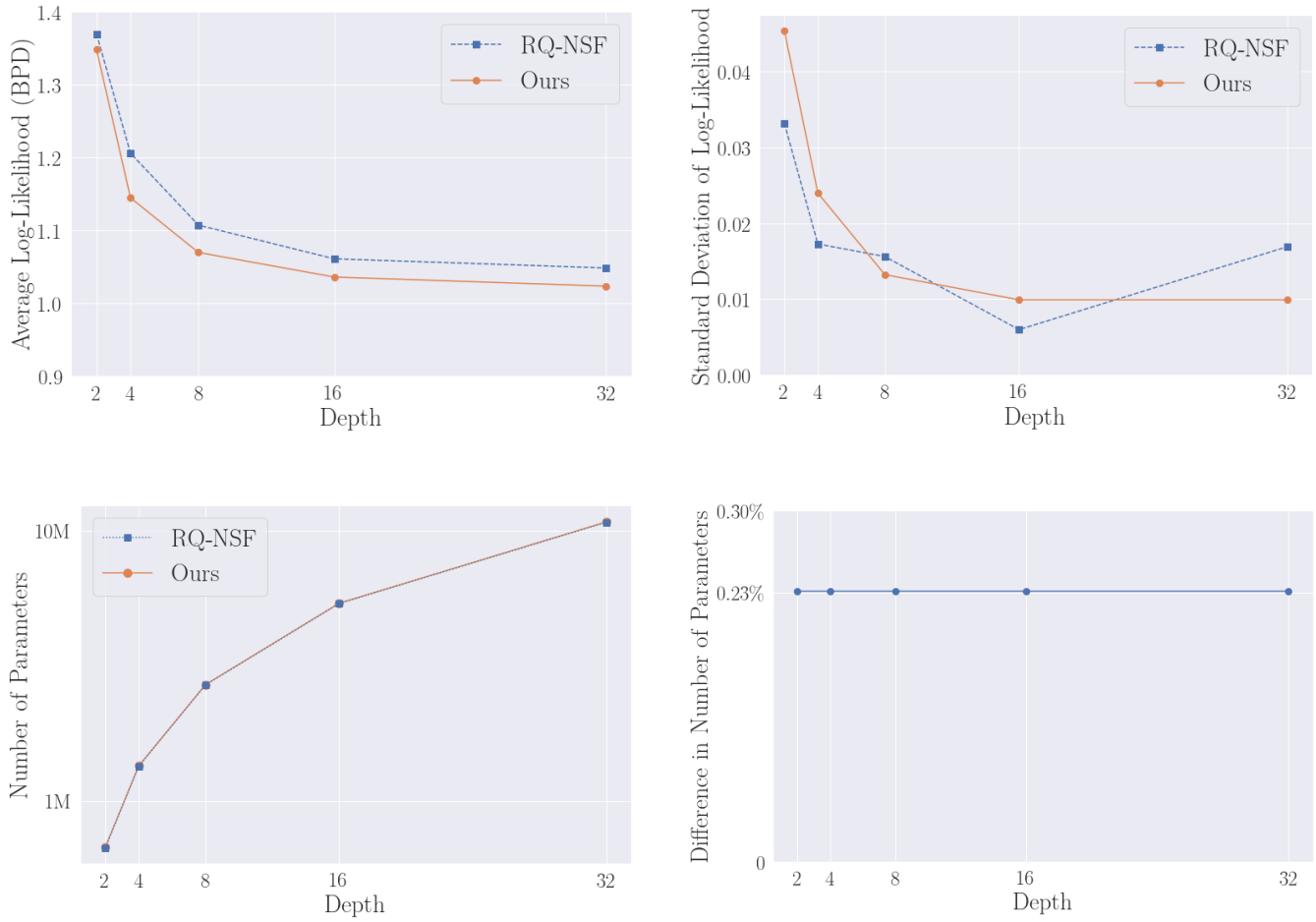


Figure 7: Comparison of linear rational and rational quadratic splines for image generation task on MNIST.