ECE 792 HW4

-Tanisha Khurana (200483139)

Problem 1

ECE 792 HW4

Problem 1

since we are training a normalizdd flow model with a prior distribution of x & generating samples to a z distribution.

complex data distribution $- X$

$z = f_\theta(x)$, parameterized by θ & $prior_z(z)$.

$$\log p_x(x) = \log\left(p_z(f_\theta(x)) \left|\det\left(\frac{\partial f_\theta(x)}{\partial x}\right)\right|\right)$$

$$= \log p_z(f_\theta(x)) + \log\left(\left|\det\left(\frac{\partial f_\theta(x)}{\partial x}\right)\right|\right)$$

where $\frac{\partial f_\theta(x)}{\partial x}$ is the determinant of the Jacobian matrix

since the normalized flow is an invertible transform

$f : X \to Z$

$$z_{1:d} = x_{1:d}$$

My $z_{d+1:D} = x_{d+1:D} \circ \exp\left(s(x_{1:d})\right) + t(x_{1:d})$

where $s(\cdot)$ & $t(\cdot)$ are my scaling & translation functions

And my reverse computation is:

$$x_{1:d} = z_{1:d}$$

$$x_{d+1:D} = (z_{d+1:D} - t(z_{1:d})) \odot \exp(-s(z_{1:d})).$$

Thus our Jacobian usually is

$$\frac{\partial z}{\partial x^T} = \begin{bmatrix} I_d & 0 \\ \frac{\partial z_{d+1:D}}{\partial x^T_{1:d}} & \mathrm{diag}(\exp[s(x_{1:D})]) \end{bmatrix}$$

My determinant is just the product of the diagonal since my first $x_{1:d}$ variables are unchanged. then its just the identity function.

other $x_{d+1:D}$ are scaled by $\exp(s(\cdot))$ so the gradient is just a diagonal matrix

In the computation for training the non-diagonal part is never used.

The modification using $z_{1:d} = Ax_{1:d}$ instead of $z_{1:d} = x_{1:d}$

Jacobian of modified $J = \begin{bmatrix} A, & 0 \\ B & I \end{bmatrix}$

A is jacobian of transformation $z_{1:d} = Ax_{1:d}$
B is Jacobian of $z_{d+1:D}$ = scaling & translation

The modified Jacobian has a block diagonal structure with first block to Jacobian of lower triangular matrix & second block of Jacobian of affine transformation.

1. Since A is block lower triangular, its determinant is simply the product of the diagonal elements, which are the non-zero elements in the blue triangles of the A matrix. This determinant factor gets included in the determinant of the full Jacobian, which gives the change in volume due to the transformation. Therefore, the modified Jacobian has a block diagonal structure, with the first

block corresponding to the Jacobian of the block-lower-triangular transformation and the second block corresponding to the Jacobian of the affine transformation for the remaining dimensions.

2. How does the determinant vary and what is the impact on the overall efficiency of the synthesis process (with the synthesis process starting right from an independent noise process as input and ending towards the final synthesis)?

   The impact of this modification on the efficiency of the synthesis process depends on the specific architecture of the Normalized Flow and the distribution of the input data. Generally, the determinant of the Jacobian matrix plays a key role in the efficiency of the synthesis process, as it affects the scale of the distribution of the transformed data. If the determinant is very small, the scale of the transformed data will be very large, which can lead to numerical instability in the synthesis process.

   However, by choosing a block-lower-triangular matrix A with diagonal entries close to 1, the determinant of the new Jacobian matrix can be kept close to 1, which can improve the stability and efficiency of the synthesis process. This is because a Jacobian matrix with a determinant close to 1 implies that the transformation preserves the scale of the distribution, which can make the optimization process more stable and efficient.

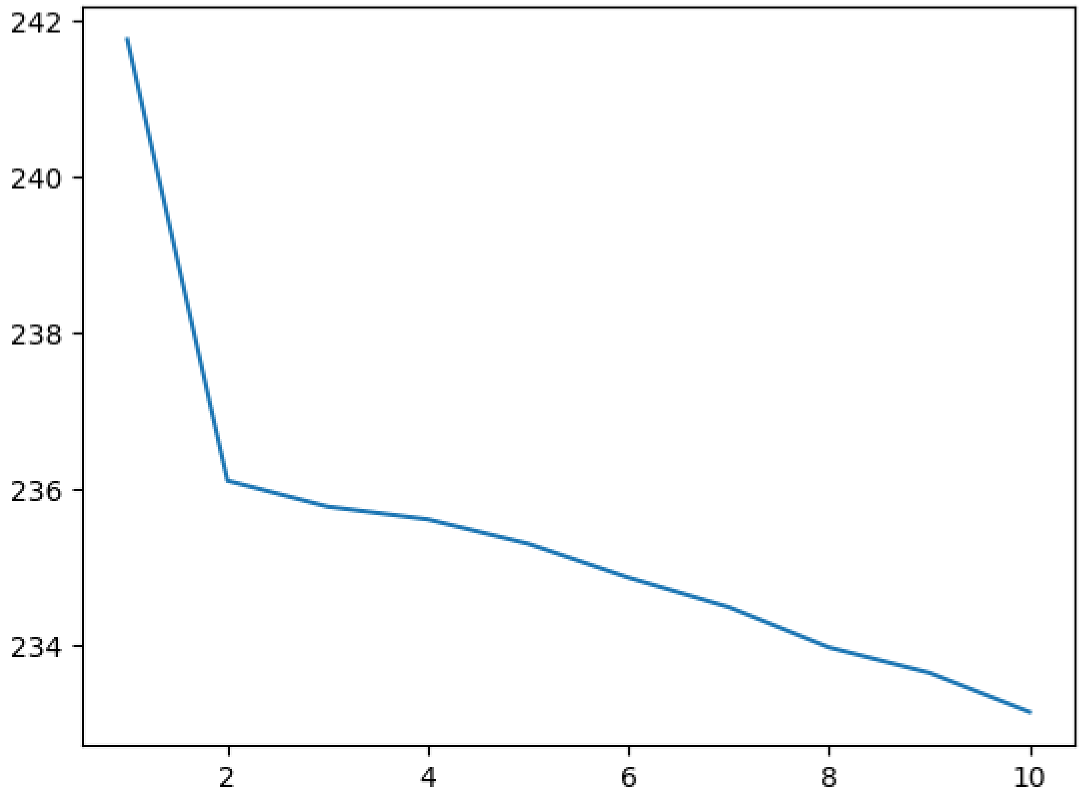
3. How does this impact the richness of synthesized data?
   The impact of the determinant on the richness of synthesized data can be understood from the fact that the determinant is related to the change in volume caused by the transformation from the input distribution to the output distribution. A higher determinant indicates that the transformation is expanding the volume, while a lower determinant indicates a contraction in volume.

   In the case of Normalizing Flows, the determinant of the Jacobian matrix determines how much the distribution of the transformed samples deviates from the prior distribution. A higher determinant indicates that the transformation is more complex and can better model the distribution of the data. A lower determinant indicates that the transformation is simpler and may not be able to capture the complexity of the data distribution. Therefore, a higher determinant is generally preferred as it indicates a more powerful transformation and a richer synthesis process.

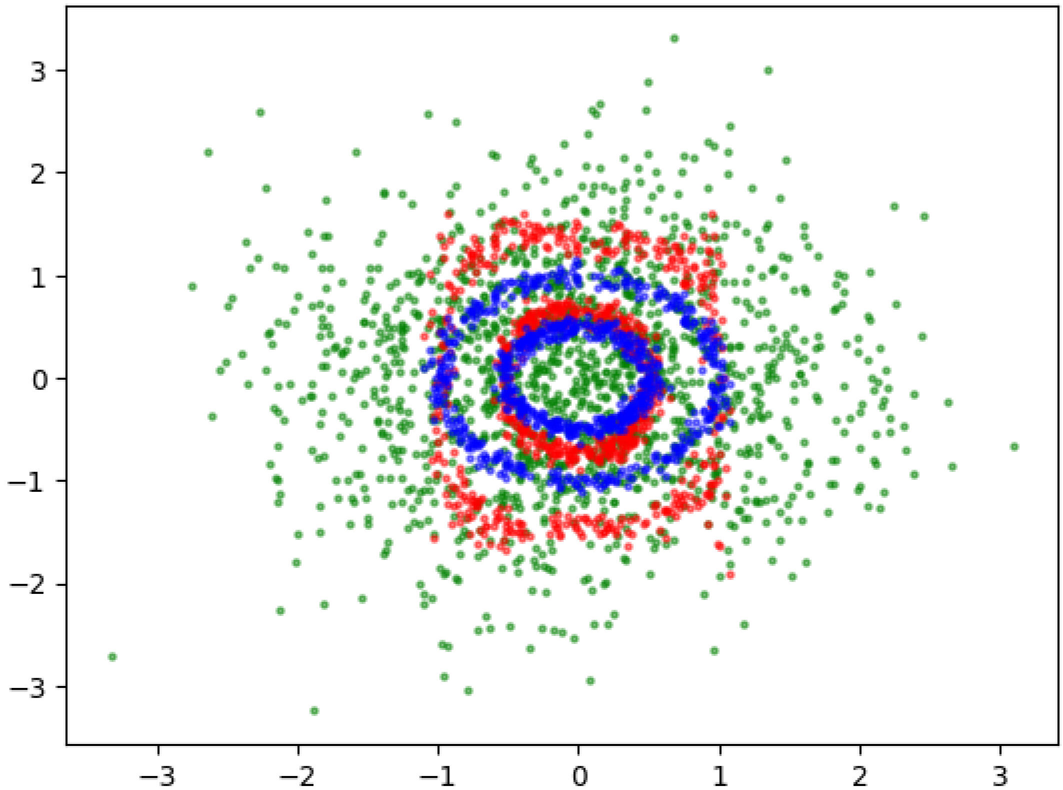
Problem 2

1 Affine Coupling layer model:

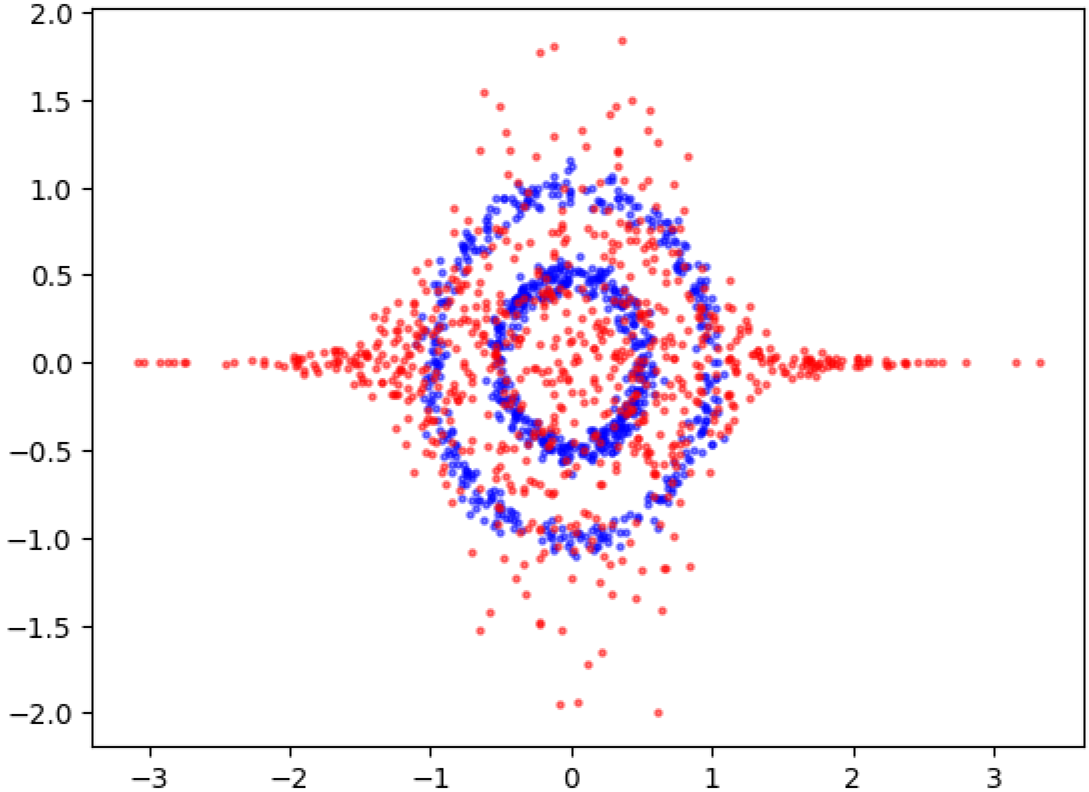
For number of hidden features: 512 and 10 epochs
Loss Plot:



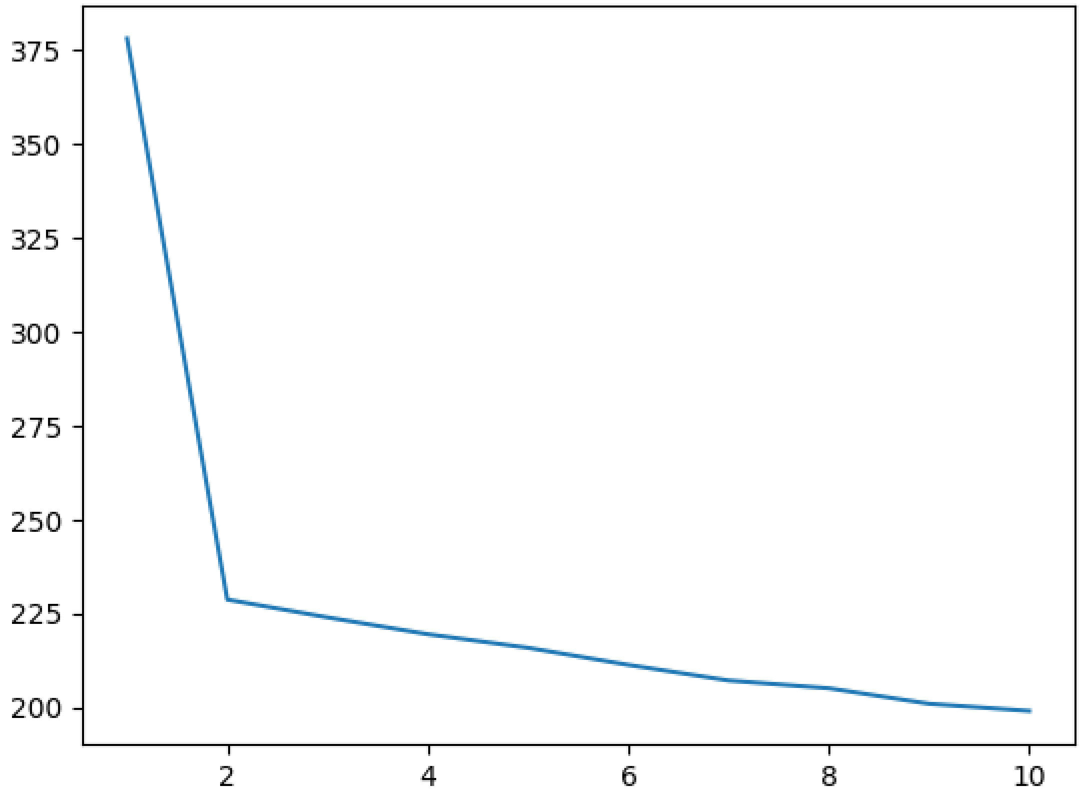
Forward transformation:
Green - Normal distribution z
Red - x_hat - samples from my model

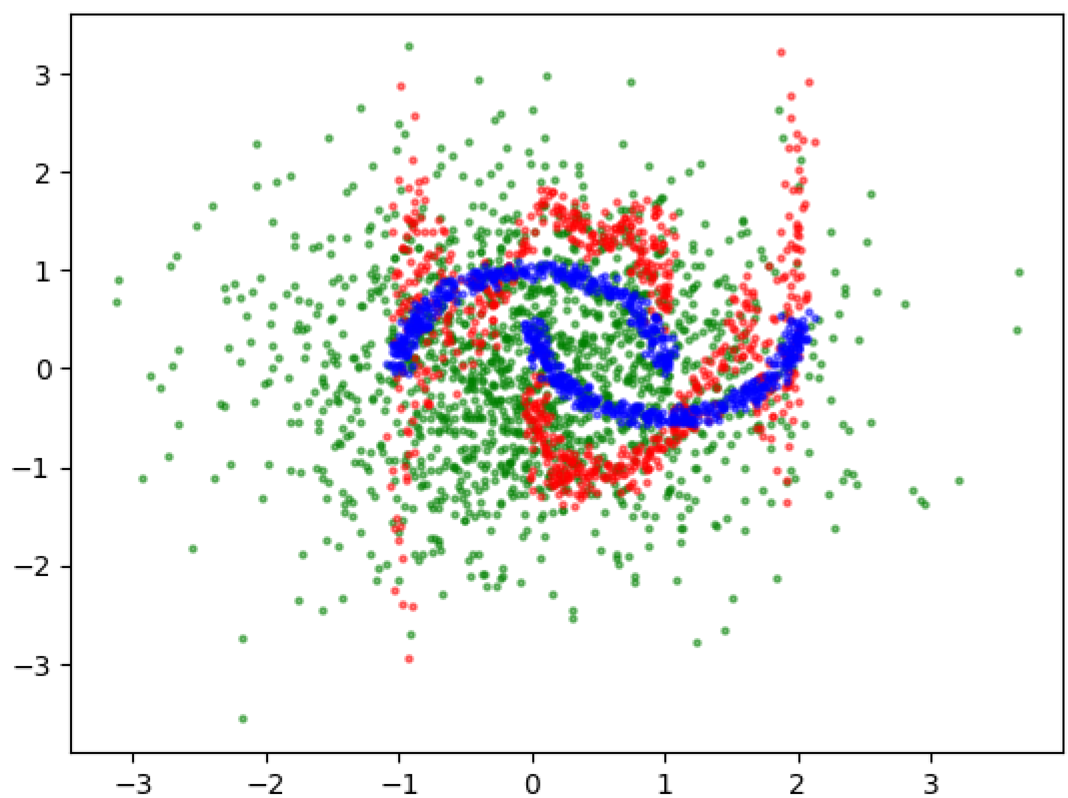


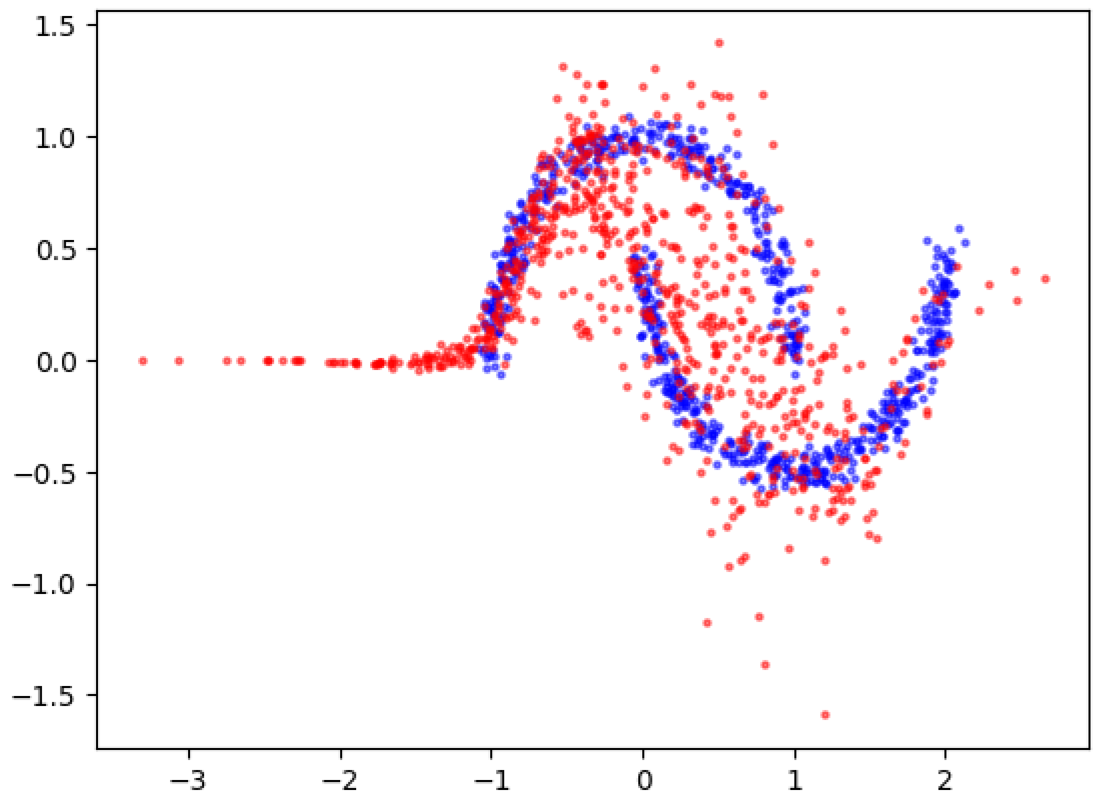
Inverse transformation:
Blue - data x
Red - x_hat - samples from my model



2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−3
−2
−1
0
1
2
3


375
350
325
300
275
250
225
200
2
4
6
8
10

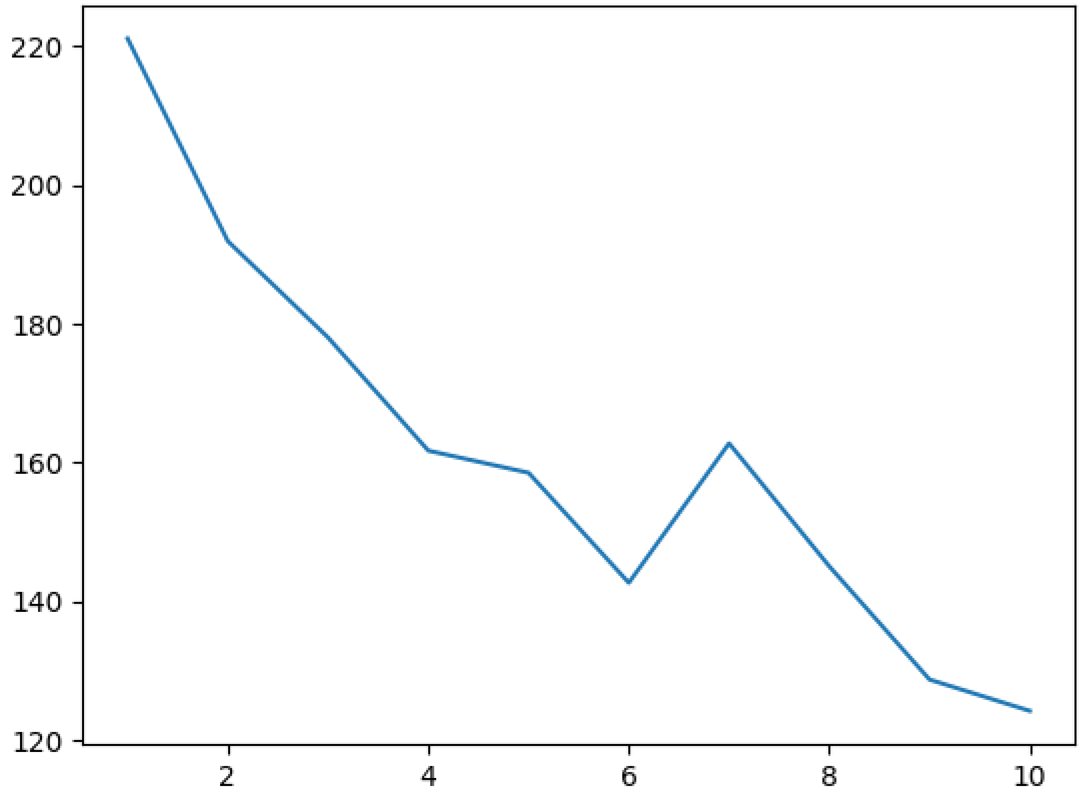

3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

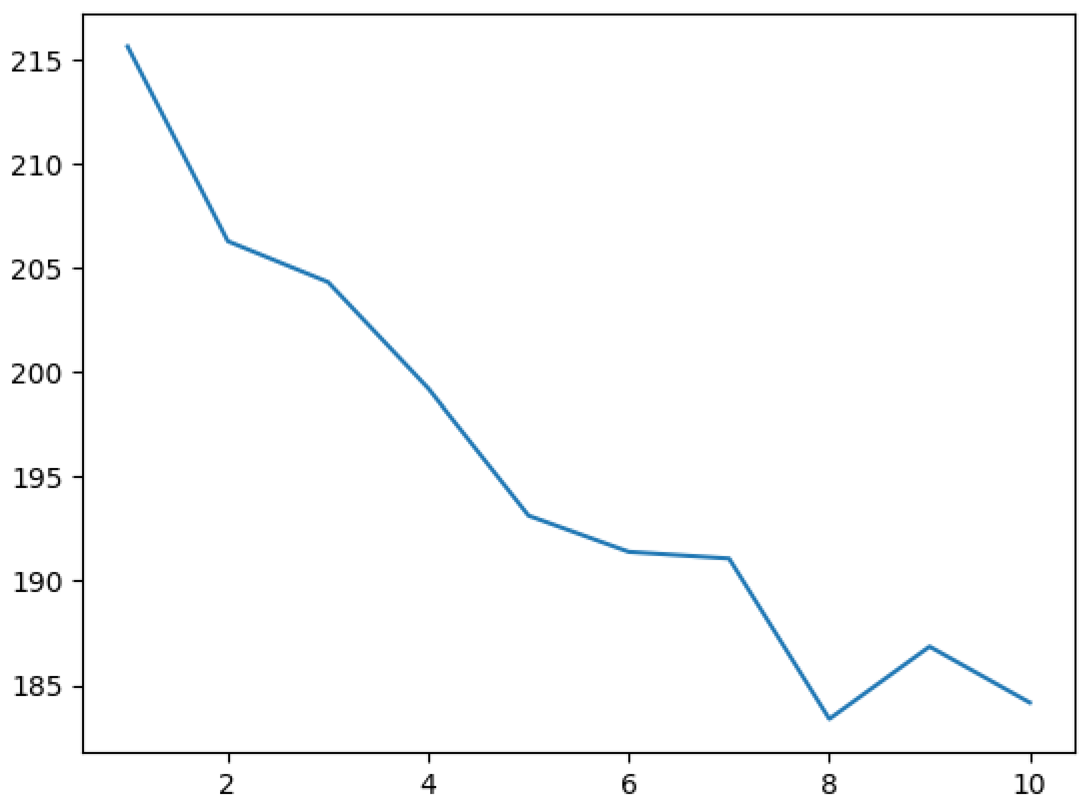


3 Affine Coupling layer model:

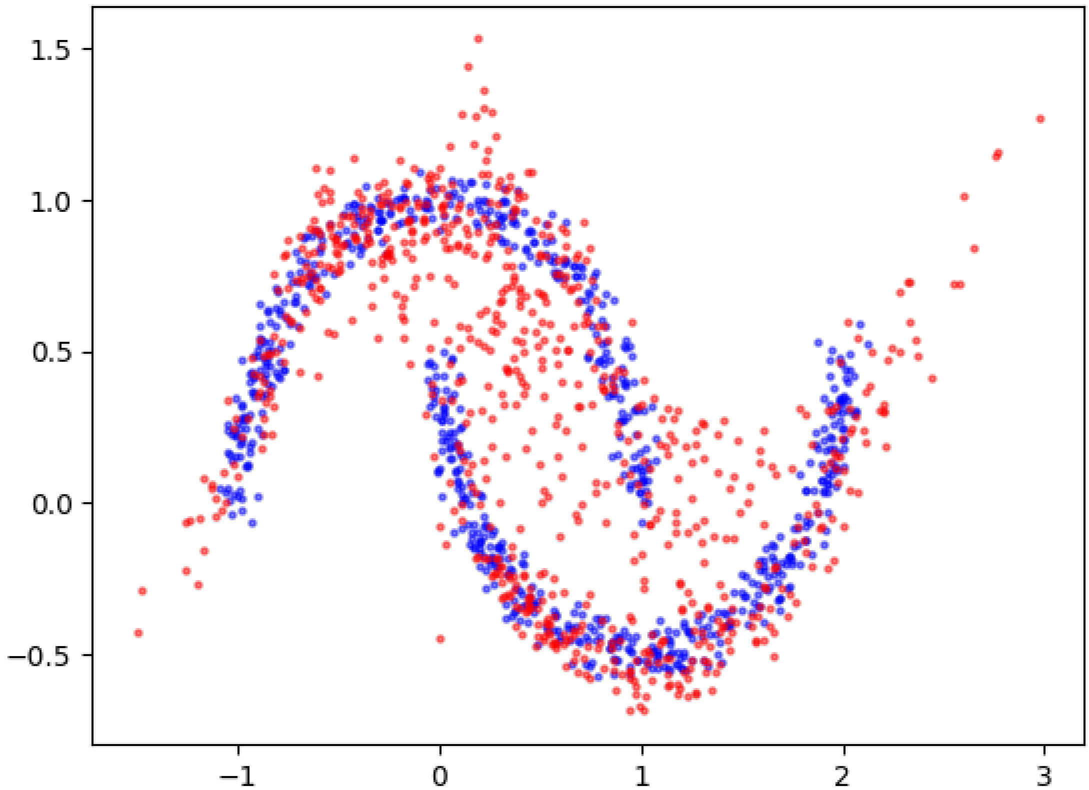
For number of hidden features: 512 and 10 epochs

Loss Plot

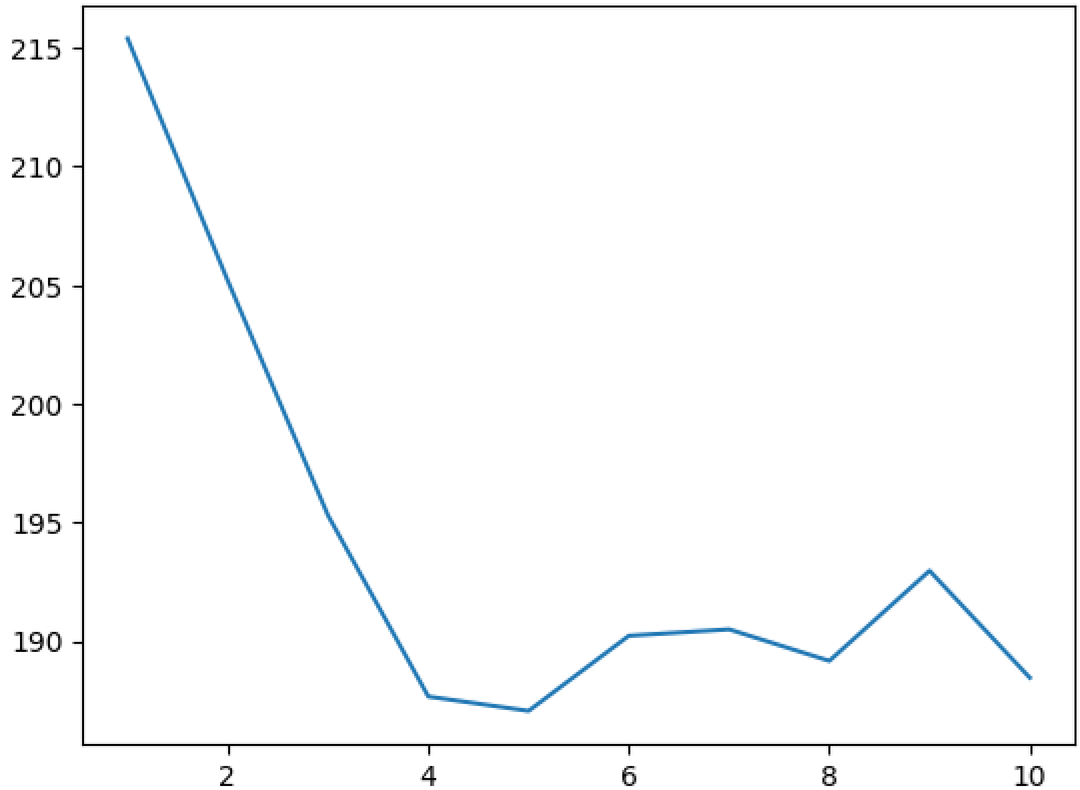

220
200
180
160
140
120
2
4
6
8
10

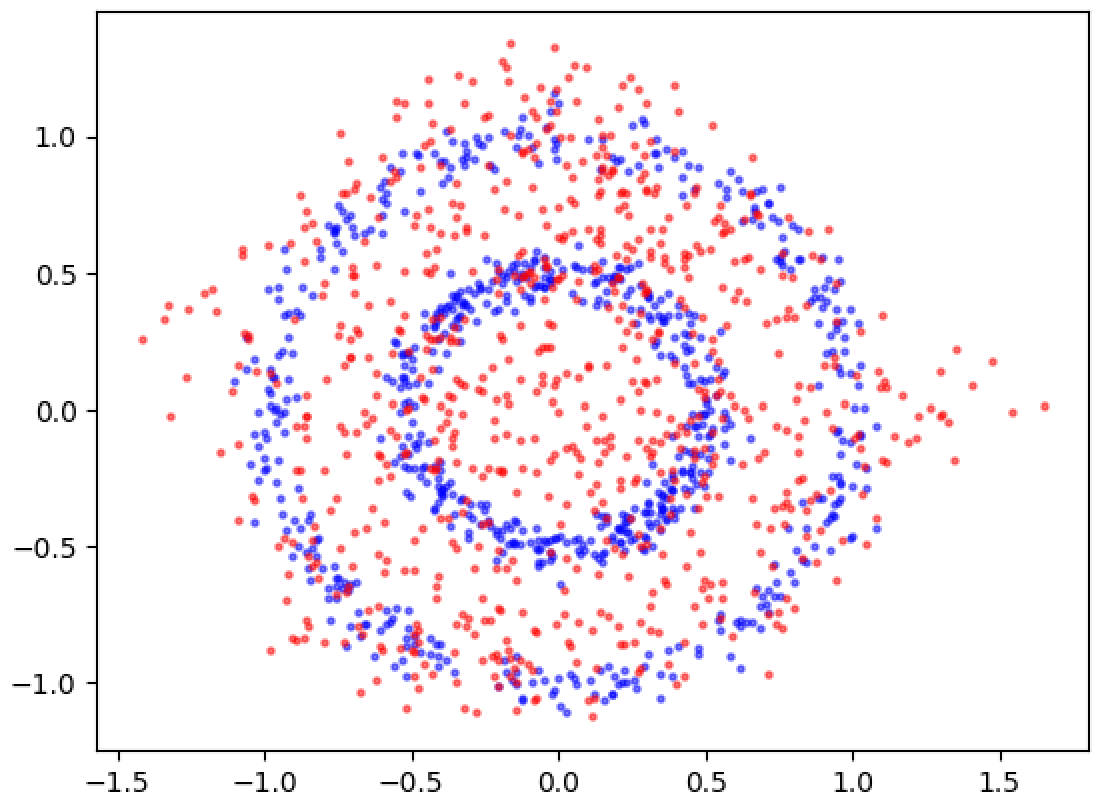

215
210
205
200
195
190
185
2
4
6
8
10

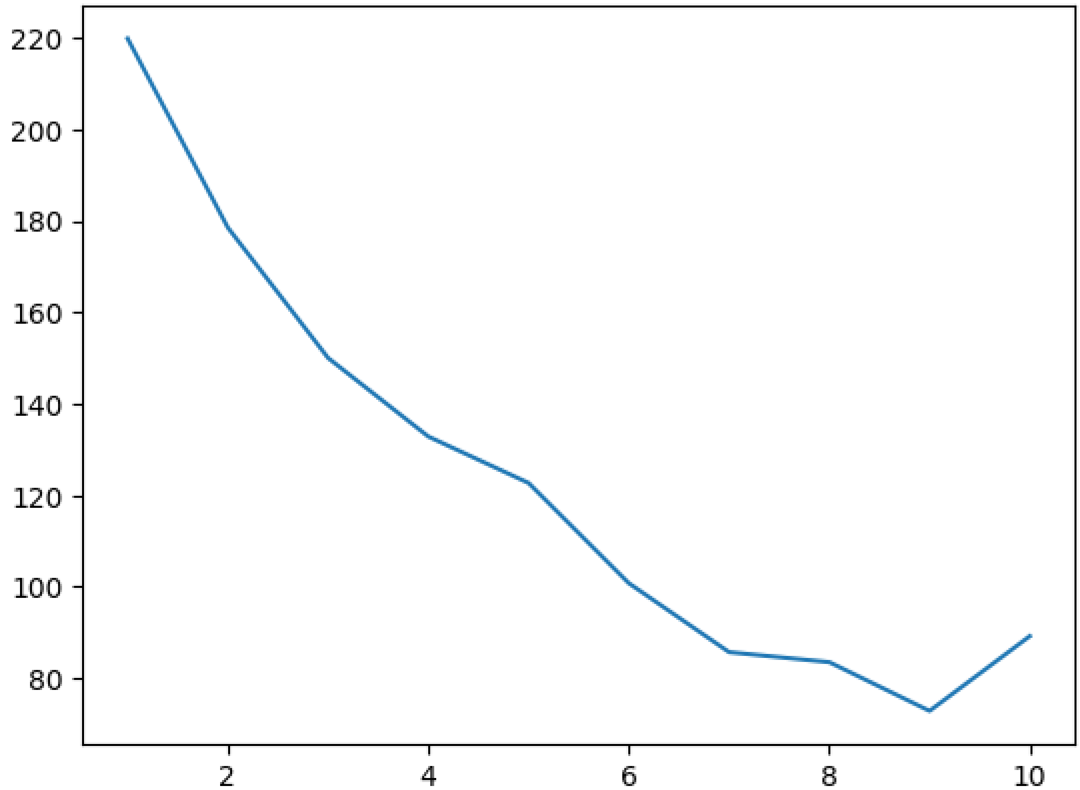
Inverse transformation:
Blue - data x
Red - x_hat - samples from my model

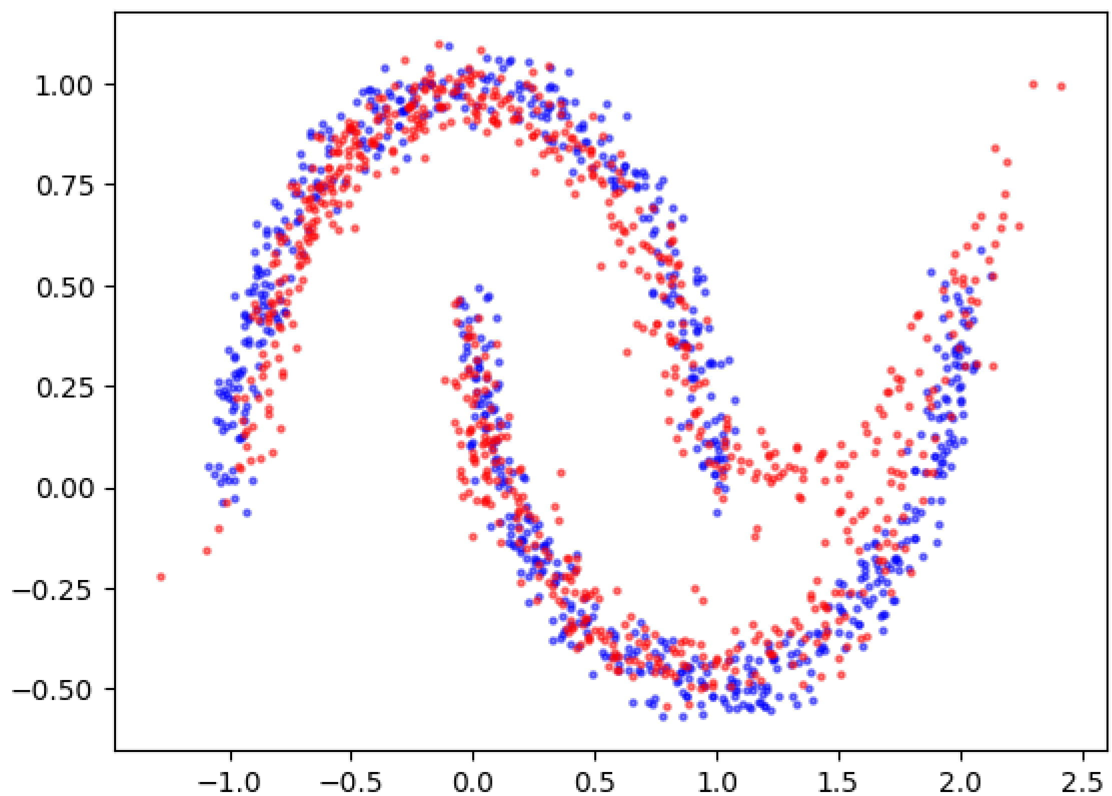


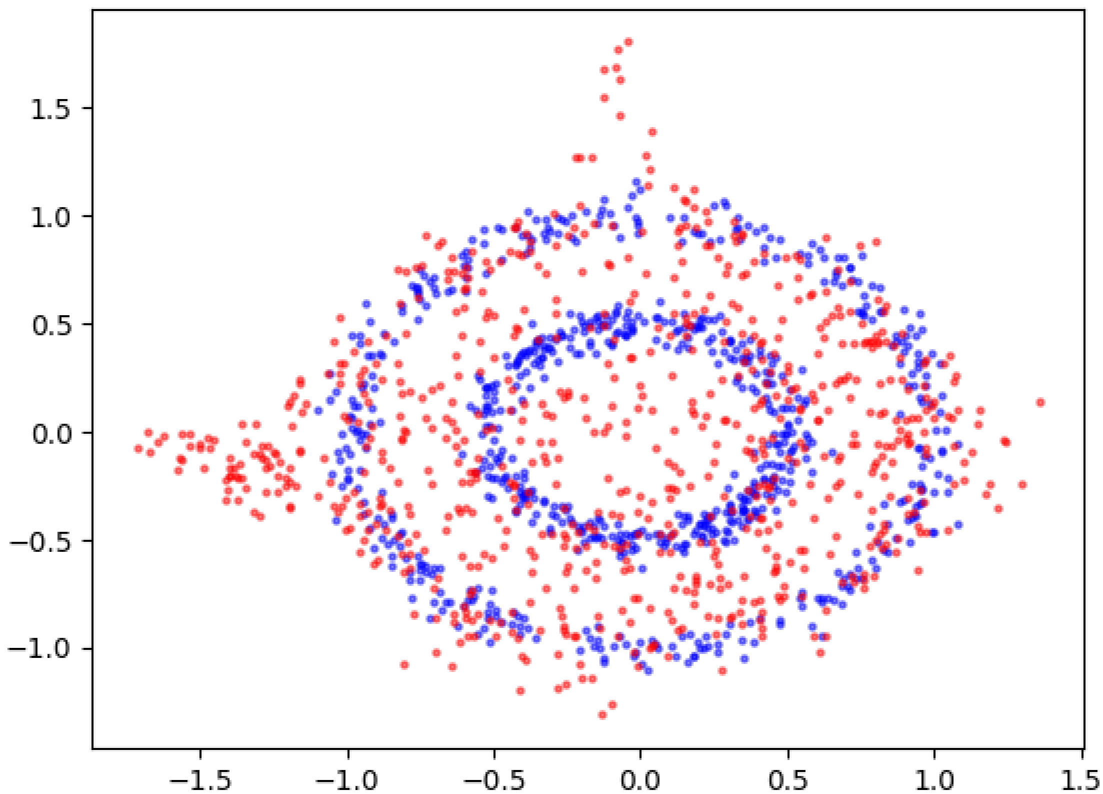
5 Affine Coupling layer model:


215
210
205
200
195
190
2
4
6
8
10

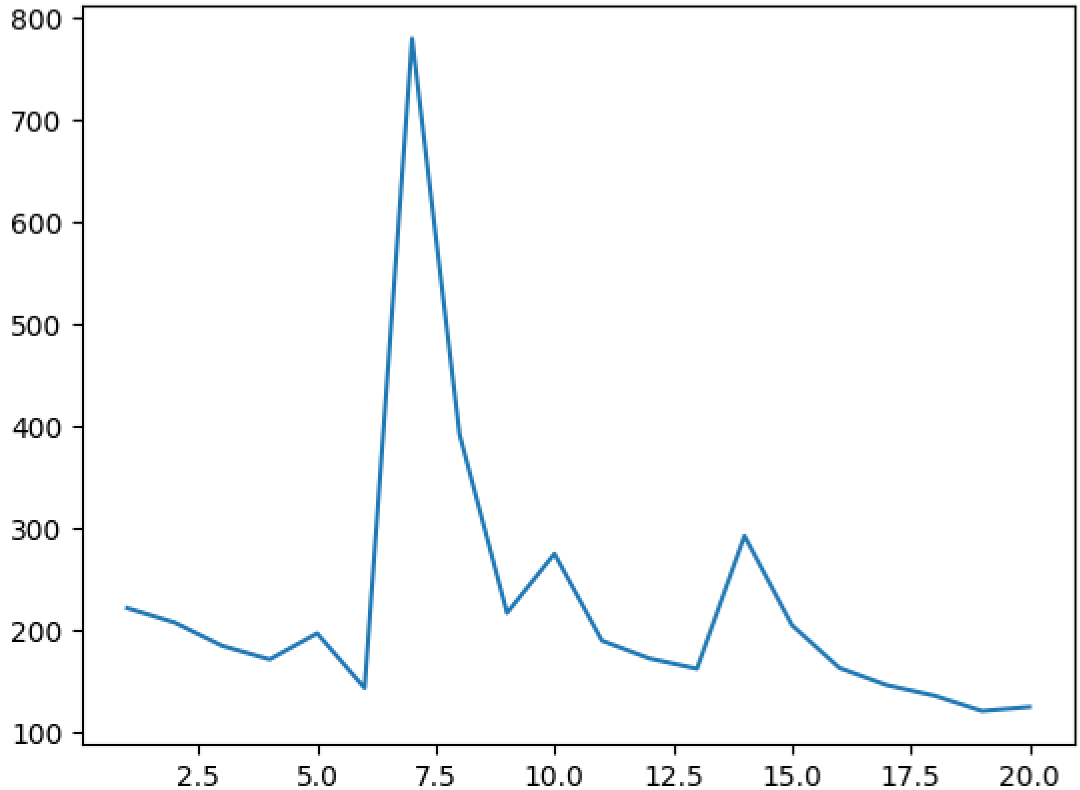

1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

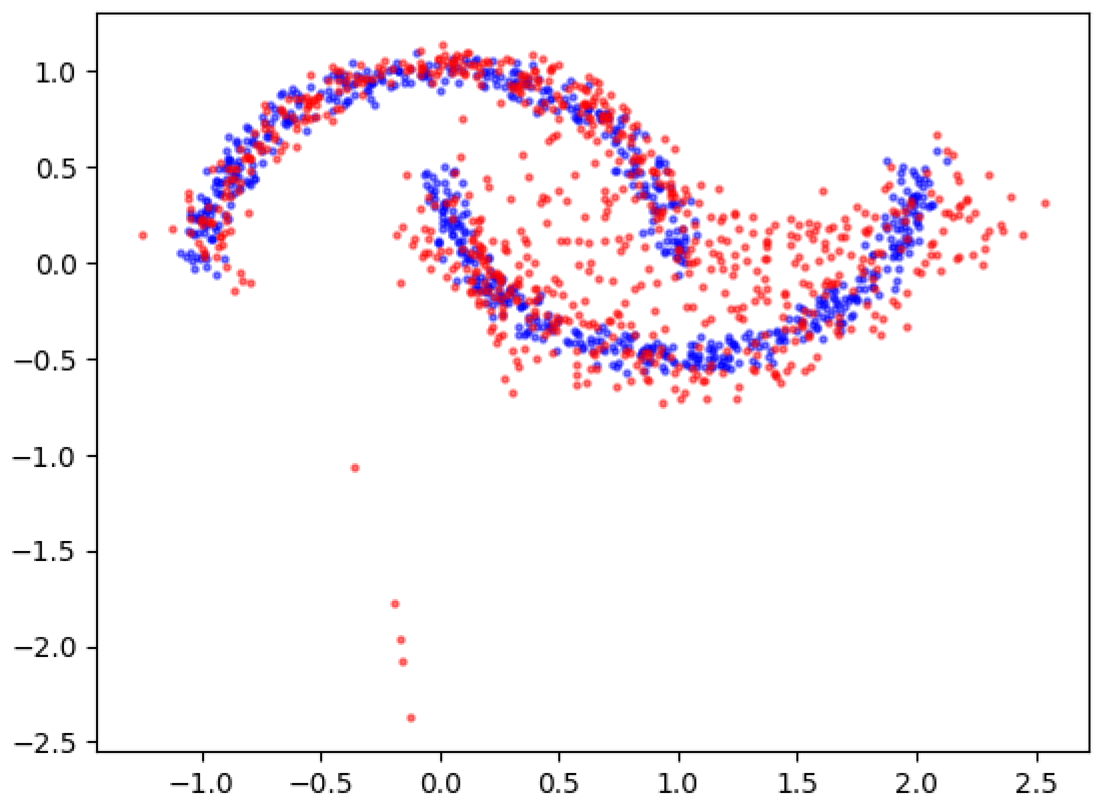

220
200
180
160
140
120
100
80
2
4
6
8
10

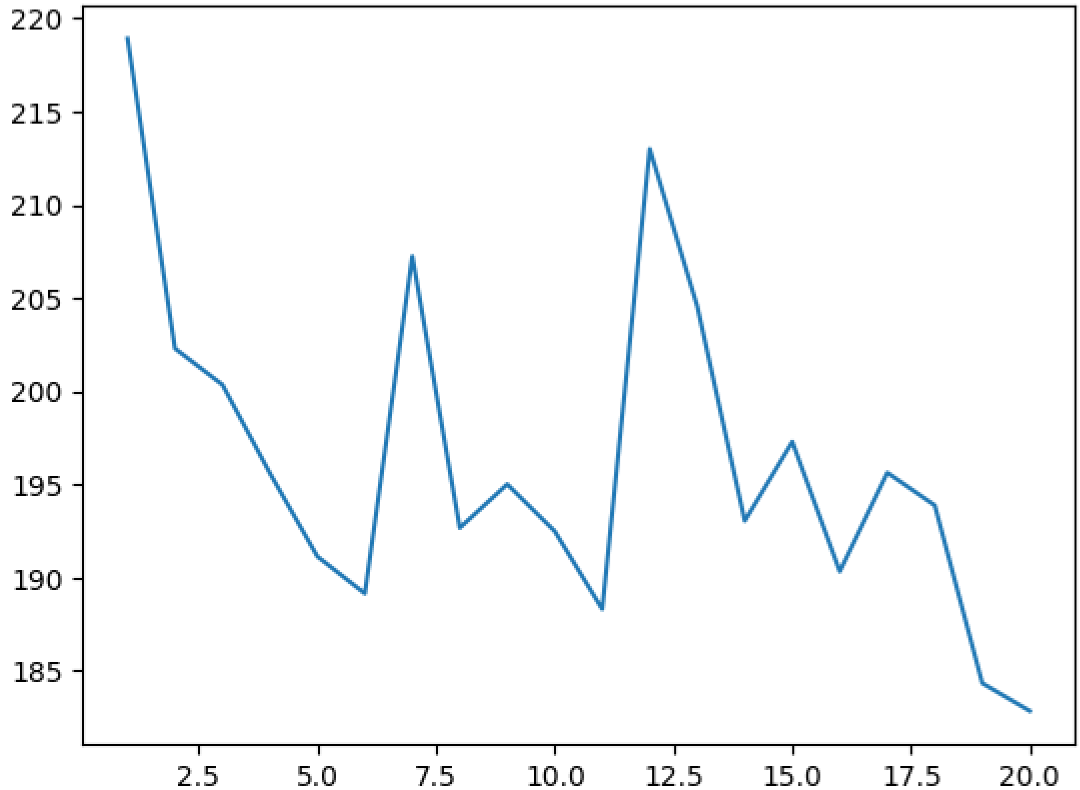

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

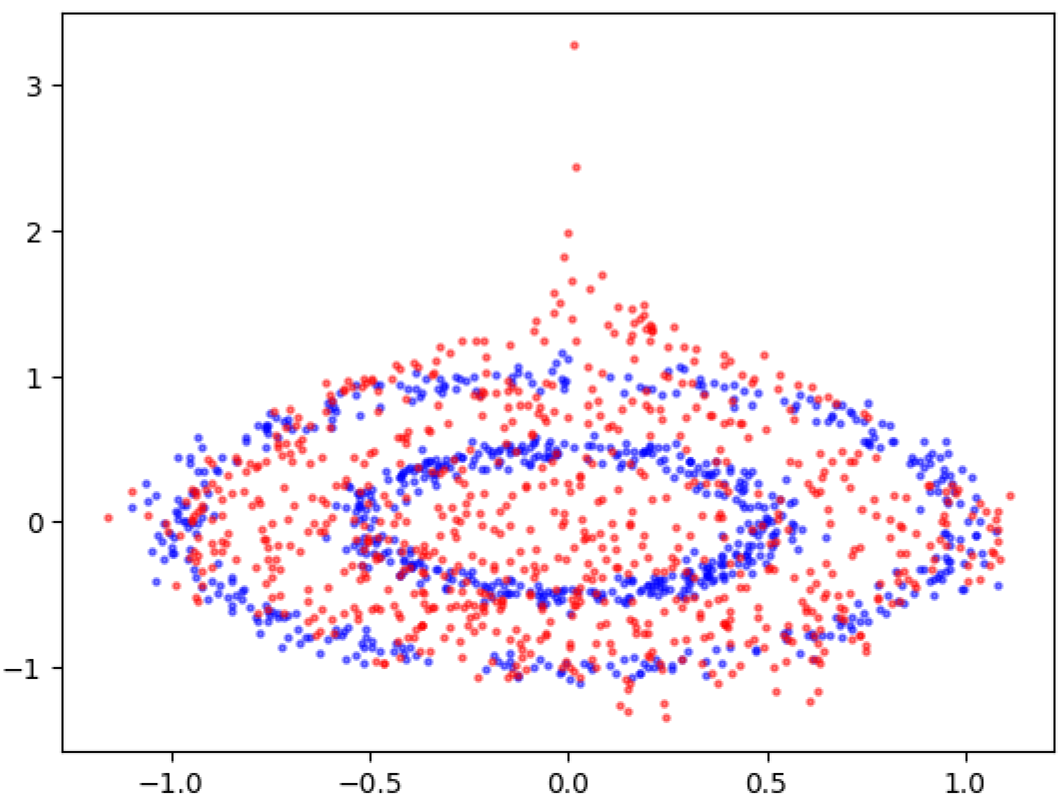


Changing num_features in hidden layer to 1024 and num epochs to 20 in 5 layer coupling


800
700
600
500
400
300
200
100
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5


220
215
210
205
200
195
190
185
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


3
2
1
0
−1
−1.0
−0.5
0.0
0.5
1.0

1. As we increase the number of layers in the normalizing flow, the distribution matching between the generated samples (z) and the data samples (x) becomes more accurate. This is because, with each layer added to the normalizing flow, the transformation applied to the data distribution becomes more complex, allowing it to better approximate the target distribution (i.e.,my original coordinates). This results in the generated samples becoming more similar to the target distribution.

   When using a one-layer normalizing flow, the generated samples may still retain some characteristics of the original distribution, and may not be perfectly matched to the target distribution. However, as we increase the number of layers to three and five, the generated samples become more like the target distribution.

   We saw a drastic change in loss after changing the number of layers from 1 to 5 `318.0093994140625` To `71.65155792236328`

2. The moons dataset was easier to train and the model was able to approximate the distribution better. This is because the moons dataset has a clearer separation between the two classes, while the circles dataset has a more complex and overlapping structure. In the moons dataset, the decision boundary can be easily approximated by a simple curve, while in the circles dataset, the decision boundary is more complex and requires a more sophisticated model to capture the subtle variations in the data. Additionally, the circles dataset has a larger intra-class variance, which can make it more difficult for the model to learn the underlying distribution. The moons dataset, on the other hand, has a smaller intra-class variance, which can make it easier for the model to learn the distribution.

3. As we saw in the last plot after changing my num_features from 512 to 1024, it was able to better match the target distribution. Increasing the hidden layer size helps the model learn more complex representations of the data, which can potentially improve the performance of the model.