A. Universal approximation with ill-conditioned affine coupling networks

A.1. Simpler universality under zero-padding.

First (as a warmup), we give a much simpler proof than (Huang et al., 2020) that affine coupling networks are universal approximators in Wasserstein under zero-padding, which moreover shows that only a small number of affine coupling layers are required. For Q a probability measure over $\mathbb{R}^n$ satisfying weak regularity conditions (see Theorem 8 below), by Brenier's Theorem (Villani, 2003) there exists a W_2 optimal transport map

$$\varphi : \mathbb{R}^n \to \mathbb{R}^n$$

such that if $X \sim N(0, I_{n\times n})$, then the pushforward $\varphi_{\#}(X)$ is distributed according to Q, and a corresponding transport map in the opposite direction which we denote φ^{-1}. If we allow for arbitrary functions t in the affine coupling network, then we can implement the zero-padded transport map $(X, 0) \mapsto (\varphi(X), 0)$ as follows:

$$(X, 0) \mapsto (X, \varphi(X)) \mapsto (\varphi(X), \varphi(X)) \mapsto (\varphi(X), 0). \tag{2}$$

Explicitly, in the first layer the translation map is $t_1(x) = \varphi(x)$, in the second layer the translation map is $t_2(x) = x - \varphi^{-1}(x)$, and in the third layer the translation map is $t_3(x) = -x$. Note that no scaling maps are required: with zero-padding the basic NICE architecture can be universal, unlike in the unpadded case where NICE can only hope to implement volume preserving maps. This is because every map from zero-padded data to zero-padded data is volume preserving. Finally, if we are required to implement the translation maps using neural networks, we can use standard approximation-theoretic results for neural networks, combined with standard results from optimal transport, to show universality of affine coupling networks in Wasserstein. First, we recall the formal statement of Brenier's Theorem:

Theorem 8 (Brenier's Theorem, Theorem 2.12 of (Villani, 2003))**.** *Suppose that P and Q are probability measures on $\mathbb{R}^n$ with densities continuous with respect to the Lebesgue measure. Then $Q = (\nabla\psi)_{\#}P$ for ψ a convex function, and moreover $\nabla\psi$ is the unique W_2-optimal transport map from P to Q.*

It turns out that the transportation map $\varphi := \nabla\psi$ is not always a continuous function, however there are simple sufficient conditions for the distribution Q under which the map is continuous (see e.g. (Caffarelli, 1992)). From these results (or by directly smoothing the transport map), we know any distribution with bounded support can be approached in Wasserstein distance by smooth pushforwards of Gaussians. So for simplicity, we state the following Theorem for distributions which are the pushforward of smooth maps.

Theorem 9 (Universal approximation with zero-padding)**.** *Suppose that P is the standard Gaussian measure in $\mathbb{R}^n$ and $Q = \varphi_{\#}P$ is the pushforward of the Gaussian measure through φ and φ is a smooth map. Then for any $\epsilon > 0$ there exists a depth 3 affine coupling network g with no scaling and feedforward ReLU net translation maps such that $W_2(g_{\#}(P \times \delta_{0^n}), Q \times \delta_{0^n}) \leq \epsilon$.*

Proof. For any $M > 0$, let $f_M(x) = \min(M, \max(-M, x))$ be the 1-dimensional truncation map to $[-M, M]$ and for a vector $x \in \mathbb{R}^n$ let $f_M(x) \in [-M, M]^n$ be the result of applying f_M coordinate-wise. Note that f_M can be implemented as a ReLU network with two hidden units per input dimension. Also, any continuous function on $[-M, M]^n$ can be approximated arbitrarily well in L^∞ by a sufficiently large ReLU neural network with one hidden layer (Leshno et al., 1993). Finally, note that if $\|f - g\|_{L^\infty} \leq \epsilon$ then for any distribution P we have $W_2(f_{\#}P, g_{\#}P) \leq \epsilon$ by considering the natural coupling that feeds the same input into f and g.

Now we show how to approximate the construction of (2) using these tools. For any $\epsilon > 0$, if we choose M sufficiently large and then take $\tilde{\varphi}$ and $\widetilde{\varphi^{-1}}$ to be sufficiently good approximations of φ and φ^{-1} on $[-M, M]^n$, we can construct an affine coupling network with ReLU feedforward network translation maps $\tilde{t}_1(x) = f_M(\tilde{\varphi}(f_M(x)))$, $\tilde{t}_2(x) = x - \widetilde{\varphi^{-1}}(x)$, and $\tilde{t}_3(x) = -x$, such that the output has W_2 distance at most ϵ from Q. □

Universality without padding. We now show that universality in Wasserstein can be proved even if we don't have zero-padding, using a lattice-based encoding and decoding scheme. Let $\epsilon > 0$ be a small constant, to be taking sufficiently small. Let $\epsilon' \in (0, \epsilon)$ be a further constant, taken sufficiently small with respect to ϵ and similar for ϵ'' wrt ϵ'. Suppose the input dimension is $2n$, and let $X = (X_1, X_2)$ with independent $X_1 \sim N(0, I_{n\times n})$ and $X_2 \sim N(0, I_{n\times n})$ be the input the the affine coupling network. Let $f(x)$ be the map which rounds $x \in \mathbb{R}^n$ to the closest grid point in $\epsilon\mathbb{Z}^n$ and define

$g(x) = x - f(x)$. Note that for a point of the form $z = f(x) + \epsilon' y$ for y which is not too large, we have that $f(z) = f(x)$ and $g(z) = y$. Let φ_1, φ_2 be the desired transportation maps guaranteed by Brenier's theorem, so that the distribution of $\varphi_1(X)$ is the target distribution Q and $\varphi_2(X)$ is a standard Gaussian independent of $\varphi_1(X)$. (In other words, φ_1, φ_2 correspond to the first half and second half of the output coordinates of the transport map from the $2n$ dimensional standard Gaussian to the desired padded distribution.) Now we consider the following sequence of maps:

$$(X_1, X_2) \mapsto (X_1, \epsilon' X_2 + f(X_1)) \tag{3}$$
$$\mapsto (f(\varphi_1(f(X_1), X_2)) + \epsilon' \varphi_2(f(X_1), X_2) + O(\epsilon''), \epsilon' X_2 + f(X_1)) \tag{4}$$
$$\mapsto (f(\varphi_1(f(X_1), X_2)) + \epsilon' \varphi_2(f(X_1), X_2) + O(\epsilon''), \varphi_2(f(X_1), X_2) + O(\epsilon''/\epsilon')). \tag{5}$$

More explicitly, in the first step we take $s_1(x) = \log(\epsilon')\vec{1}$ and $t_1(x) = f(x)$. In the second step, we take $s_2(x) = \log(\epsilon'')\vec{1}$ and t_2 is defined by $t_2(x) = f(\varphi_1(f(x), g(x))) + \epsilon' \varphi_2(f(x), g(x))$. In the third step, we take $s_3(x) = \log(\epsilon'')\vec{1}$ and define $t_3(x) = \frac{g(x)}{\epsilon'}$.

Again, taking sufficiently good approximations to all of the maps allows us to approximate this map with neural networks, which we formalize below.

Proof of Theorem 1. Turning (3),(4), and (5) into a universal approximation theorem for ReLU-net based feedforward networks just requires us to modify the proof of Theorem 9 for this scenario.

Fix $\delta > 0$, the above argument shows we can choose $\epsilon, \epsilon', \epsilon'' > 0$ sufficiently small so that if h is map defined by composing (3),(4), and (5), then $W_2(h_\# P, Q) \leq \epsilon/4$. The layers defining h may not be continuous, since f is only continuous almost everywhere. Using that continuous functions are dense in L^2, we can find a function f_ϵ which is continuous and such that if we define h_ϵ by replacing each application of f by f_ϵ, then $W_2(h_\epsilon \# P, Q) \leq \epsilon/2$.

Finally, since f_ϵ is an affine coupling network with continuous s and t functions, we can use the same truncation-and-approximation argument from Theorem 9 to approximate it by an affine coupling network g with ReLU feedforward s and t functions such that $W_2(g \# P, Q) \leq \epsilon$, which proves the result. □

B. Missing proofs for Section 5

B.1. Upper Bound

First, we recall a folklore result about permutations. Let S_n denote the symmetric group on n elements, i.e. the set of permutations of $\{1, \ldots, n\}$ equipped with the multiplication operation of composition. Recall that the *order* of a permutation π is the smallest positive integer k such that π^k is the identity permutation.

Lemma 5. *For any permutation $\pi \in S_n$, there exists $\sigma_1, \sigma_2 \in S_n$ of order at most 2 such that*

$$\pi = \sigma_1 \sigma_2.$$

Proof. This result is folklore. We include a proof of it for completeness[6].

First, recall that every permutation π has a unique decomposition $\pi = c_1 \cdots c_k$ as a product of disjoint cycles. Therefore if we show the result for a single cycle, so $c_i = \sigma_{i1}\sigma_{i2}$ for every i, then taking $\sigma_1 = \prod_{i=1}^k \sigma_{i1}$ and $\sigma_2 = \prod_{i=1}^k \sigma_{i2}$ proves the desired result since $\pi = \sigma_1\sigma_2$ and σ_1, σ_2 are both of order at most 2.

It remains to prove the result for a single cycle c of length r. The cases $r \leq 2$ are trivial. Without loss of generality, we assume $c = (1 \cdots r)$. Let $\sigma_1(1) = 2$, $\sigma_1(2) = 1$, and otherwise $\sigma_1(s) = r + 3 - s$. Let $\sigma_2(1) = 3$, $\sigma_2(2) = 2$, $\sigma_2(3) = 1$, and otherwise $\sigma_2(s) = r + 4 - s$. It's easy to check from the definition that both of these elements are order at most 2.

We now claim $c = \sigma_2 \circ \sigma_1$. To see this, we consider the following cases:

1. $\sigma_2(\sigma_1(1)) = \sigma_2(2) = 2$.

[6]This proof, given by HH Rugh, and some other ways to prove this result can be found at https://math.stackexchange.com/questions/1871783/every-permutation-is-a-product-of-two-permutations-of-order-2.

2. $\sigma_2(\sigma_1(2)) = \sigma_2(1) = 3$.

3. $\sigma_2(\sigma_1(r)) = \sigma_2(3) = 1$.

4. For all other s, $\sigma_2(\sigma_1(s)) = \sigma_2(r + 3 - s) = s + 1$.

In all cases we see that $c(s) = \sigma_2(\sigma_1(s))$ which proves the result. □

Next, we supply the proof of Lemma 1

Proof of Lemma 1. It is easy to see that swapping two elements is possible in a fashion that doesn't affect other dimensions by the following 'signed swap' procedure requiring 3 matrices:

$$(x, y) \mapsto (x, y - x) \mapsto (y, y - x) \mapsto (y, -x). \tag{6}$$

Next, let $L = \{1, \ldots, d\}$ and $R = \{d + 1, \ldots, 2d\}$. There will be an equal number of elements which in a particular permutation will be permuted from L to R as those which will be permuted from R to L. We can choose an arbitrary bijection between the two sets of elements and perform these 'signed swaps' in parallel as they are disjoint, using a total of 3 matrices. The result of this will be the elements partitioned into L and R that would need to be mapped there.

We can also (up to sign) transpose elements within a given set L or R via the following computation using our previous 'signed swaps' that requires one 'storage component' in the other set:

$$([x, y], z) \mapsto ([z, y], -x) \mapsto ([z, x], y) \mapsto ([y, x], -z).$$

So, up to sign, we can in 9 matrices compute any transposition in L or R separately. In fact, since any permutation can be represented as the product of two order-2 permutations (Lemma 5) and any order-2 permutation is a disjoint union of transpositions, we can implement an order-2 permutation up to sign using 9 matrices and an arbitrary permutation up to sign using 18 matrices.

In total, we used 3 matrices to move elements to the correct side and 18 matrices to move them to their correct position, for a total of 21 matrices. □

Lemma 6. *Suppose $A \in \mathbb{R}^{n\times n}$ is a matrix with n distinct real eigenvalues. Then there exists an invertible matrix $S \in \mathbb{R}^{n\times n}$ such that $A = SDS^{-1}$ where D is a diagonal matrix containing the eigenvalues of A.*

Proof. Observe that for every eigenvalue λ_i of A, the matrix $(A - \lambda_i I)$ has rank $n - 1$ by definition, hence there exists a corresponding real eigenvector v_i by taking a nonzero solution of the real linear system $(A - \lambda I)v = 0$. Taking S to be the linear operator which maps e_i to standard basis vector v_i, and $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$ proves the result. □

Next, we give the proof of Lemma 2

Proof of Lemma 2. Let

$$D = (M - I)E^{-1},$$

$$H = (M^{-1} - I)E^{-1},$$

$$E = -AM,$$

where A is an invertible matrix that will be specified later. We can multiply out with these values giving

$$\begin{aligned}
&\begin{bmatrix} I & 0 \\ A & I \end{bmatrix} \begin{bmatrix} I & D \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ E & I \end{bmatrix} \begin{bmatrix} I & H \\ 0 & I \end{bmatrix} \\
&= \begin{bmatrix} I & 0 \\ A & I \end{bmatrix} \begin{bmatrix} I & (I-M)M^{-1}A^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -AM & I \end{bmatrix} \begin{bmatrix} I & (I-M^{-1})M^{-1}A^{-1} \\ 0 & I \end{bmatrix} \\
&= \begin{bmatrix} I & (M^{-1}-I)A^{-1} \\ A & AM^{-1}A^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -AM & I \end{bmatrix} \begin{bmatrix} I & (I-M^{-1})M^{-1}A^{-1} \\ 0 & I \end{bmatrix} \\
&= \begin{bmatrix} M & (M^{-1}-I)A^{-1} \\ 0 & AM^{-1}A^{-1} \end{bmatrix} \begin{bmatrix} I & (I-M^{-1})M^{-1}A^{-1} \\ 0 & I \end{bmatrix} \\
&= \begin{bmatrix} M & 0 \\ 0 & AM^{-1}A^{-1} \end{bmatrix}
\end{aligned}$$

Here what remains is to guarantee $AM^{-1}A^{-1} = S$. Since S and M^{-1} have the same eigenvalues, by Lemma 6 there exist real matrices U, V such that $S = UXU^{-1}$ and $M^{-1} = VXV^{-1}$ for the same diagonal matrix X, hence $S = UV^{-1}M^{-1}VU^{-1}$. Therefore taking $A = UV^{-1}$ gives the result. □

Now that we have the Lemmas, we prove the upper bound.

Proof of Theorem 6. Recall that our goal is to show that $GL_+(2d, \mathbb{R}) \subset \mathcal{A}^K$ for an absolute constant $K > 0$. To show this, we consider an arbitrary matrix $T \in GL_+(2d, \mathbb{R})$, i.e. an arbitrary matrix $T : 2d \times 2d$ with positive determinant, and show how to build it as a product of a bounded number of elements from $\mathcal{A}$. As T is a square matrix, it admits an LUP decomposition (Horn & Johnson, 2012): i.e. a decomposition into the product of a lower triangular matrix L, an upper triangular matrix U, and a permutation matrix P. This proof proceeds essentially by showing how to construct the L, U, and P components in a constant number of our desired matrices.

By Lemma 1, we can produce a matrix $\tilde{P}$ with $\det \tilde{P} > 0$ which agrees with P up to the sign of its entries using $O(1)$ linear affine coupling layers. Then $T\tilde{P}^{-1}$ is a matrix which admits an LU decomposition: for example, given that we know TP^{-1} has an LU decomposition, we can modify flip the sign of some entries of U to get an LU decomposition of $T\tilde{P}^{-1}$. Furthermore, since $\det(T\tilde{P}^{-1}) > 0$, we can choose an LU decomposition $T\tilde{P}^{-1} = LU$ such that $\det(L), \det(U) > 0$ (for any decomposition which does not satisfy this, the two matrices L and U must both have negative determinant as $0 < \det(T\tilde{P}^{-1}) = \det(L)\det(U)$. In this case, we can flip the sign of column i in L and row i in U to make the two matrices positive determinant).

It remains to show how to construct a lower/upper triangular matrix with positive determinant out of our matrices. We show how to build such a lower triangular matrix L as building U is symmetrical.

At this point we have a matrix $\begin{bmatrix} A & 0 \\ B & C \end{bmatrix}$, where A and C are lower triangular. We can use column elimination to eliminate the bottom-left block:

$$\begin{bmatrix} A & 0 \\ B & C \end{bmatrix} \begin{bmatrix} I & 0 \\ -C^{-1}B & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & C \end{bmatrix},$$

where A and C are lower-triangular.

Recall from (6) that we can perform the signed swap operation in $\mathbb{R}^2$ of taking $(x, y) \mapsto (y, -x)$ for x using 3 affine coupling blocks. Therefore using 6 affine coupling blocks we can perform a sign flip map $(x, y) \mapsto (-x, -y)$. Note that because $\det(L) > 0$, the number of negative entries in the first d diagonal entries has the same parity as the number of negative entries in the second d diagonal entries. Therefore, using these sign flips in parallel, we can ensure using 6 affine coupling layers that that the first d and last d diagonal entries of L have the same number of negative elements. Now that the number of negative entries match, we can apply two diagonal rescalings to ensure that:

1. The first d diagonal entries of the matrix are distinct.
2. The last d diagonal entries contain the multiplicative inverses of the first d entries up to reordering. Here we use that the number of negative elements in the first d and last d elements are the same, which we ensured earlier.

At this point, we can apply Lemma 2 to construct this matrix from four of our desired matrices. Since this shows we can build L and U, this shows we can build any matrix with positive determinant.

Now, let's count the matrices we needed to accomplish this. In order to construct $\tilde{P}$, we needed 21 matrices. To construct L, we needed 1 for column elimination, 6 for the sign flip, 2 for the rescaling of diagonal elements, and 4 for Lemma 2 giving a total of 13. So, we need $21 + 13 + 13 = 47$ total matrices to construct the whole LUP decomposition. □

B.2. Lower Bound

Finally, we proceed to give the proof of Lemma 3.

Proof of Lemma 3. We explicitly solve the block matrix equations. Multiplying out the LHS gives

$$\begin{bmatrix} C & D \\ AC & AD+B \end{bmatrix} \begin{bmatrix} G & H \\ EG & EH+F \end{bmatrix} = \begin{bmatrix} CG+DEG & CH+DEH+DF \\ ACG+ADEG+BEG & ACH+ADEH+ADF+BEH+BF \end{bmatrix}.$$

Say

$$T = \begin{bmatrix} X & Y \\ Z & W \end{bmatrix}.$$

Starting with the top-left block gives that

$$X = (C+DE)G$$

$$D = (XG^{-1} - C)E^{-1} \tag{7}$$

Next, the top-right block gives that

$$Y = (C+DE)H + DF = XG^{-1}H + DF$$

$$H = GX^{-1}(Y - DF). \tag{8}$$

Equivalently,

$$D = (Y - XG^{-1}H)F^{-1} \tag{9}$$

Combining (8) and (7) gives

$$H = GX^{-1}(Y - (XG^{-1} - C)E^{-1}F)$$

$$H = GX^{-1}Y - (I - GX^{-1}C)E^{-1}F \tag{10}$$

The bottom-left and (7) gives

$$Z = ACG + ADEG + BEG$$
$$ZG^{-1} = AC + (AD+B)E$$

$$E = (AD+B)^{-1}(ZG^{-1} - AC) \tag{11}$$
$$E = (A(XG^{-1} - C)E^{-1} + B)^{-1}(ZG^{-1} - AC)$$
$$E^{-1} = (ZG^{-1} - AC)^{-1}(A(XG^{-1} - C)E^{-1} + B)$$
$$(ZG^{-1} - AC) = (A(XG^{-1} - C)E^{-1} + B)E = A(XG^{-1} - C) + BE$$
$$E = B^{-1}((ZG^{-1} - AC) - A(XG^{-1} - C))$$

$$E = B^{-1}(ZG^{-1} - AXG^{-1}) \tag{12}$$

Taking the bottom-right block and substituting (11) gives

$$W = ACH + (AD+B)(EH+F) = ACH + (ZG^{-1} - AC)H + (AD+B)F$$

$$W = ZG^{-1}H + ADF + BF. \tag{13}$$

Substituting (7) into (13) gives

$$W = ZG^{-1}H + A(Y - XG^{-1}H) + BF = (Z - AX)G^{-1}H + AY + BF.$$

Substituting (10) gives

$$\begin{aligned} W = &\quad (Z - AX)G^{-1}(GX^{-1}Y - (I - GX^{-1}C)E^{-1}F) + AY + BF \\ = &\quad (Z - AX)(X^{-1}Y - (G^{-1} - X^{-1}C)E^{-1}F) + AY + BF. \end{aligned}$$

Substituting (12) gives

$$W = (Z - AX)(X^{-1}Y - (G^{-1} - X^{-1}C)(ZG^{-1} - AXG^{-1})^{-1}BF) + AY + BF$$

$$\begin{aligned} W - ZX^{-1}Y - BF = &\quad (Z - AX)(X^{-1}C - G^{-1})((Z - AX)G^{-1})^{-1}BF \\ = &\quad (Z - AX)(X^{-1}C - G^{-1})G(Z - AX)^{-1}BF \\ = &\quad (Z - AX)X^{-1}C(Z - AX)^{-1} - BF \end{aligned}$$

$$W - ZX^{-1}Y = (Z - AX)X^{-1}C(Z - AX)^{-1} \tag{14}$$

Here we notice that $W - ZX^{-1}Y$ is similar to $X^{-1}C$, where we get to choose values along the diagonal of C. In particular, this means that $W - ZX^{-1}Y$ and $X^{-1}C$ must have the same eigenvalues. □

Proof of Theorem 7. First, note that element in $\mathcal{A}^4$ can be written in either the form $L_1R_1L_2R_2$ or $R_1L_1R_2L_2$ for $L_1, L_2 \in \mathcal{A}_\mathcal{L}$ and $R_1, R_2 \in \mathcal{A}_\mathcal{R}$. We construct an explicit matrix which cannot be written in either form.

Consider an invertible matrix of the form

$$T = \begin{bmatrix} X & 0 \\ 0 & W \end{bmatrix}$$

and observe that the Schur complement T/X is simply W. Therefore Lemma 3 says that this matrix can only be in $\mathcal{A}_L\mathcal{A}_R\mathcal{A}_L\mathcal{A}_R$ if W is similar to $X^{-1}C$ for some diagonal matrix C. Now consider the case where W is a permutation matrix encoding the permutation $(1\ 2\ \cdots\ d)$ and X is a diagonal matrix with nonzero entries. Then $X^{-1}C$ is a diagonal matrix as well, hence has real eigenvalues, while the eigenvalues of W are the d-roots of unity. (The latter claim follows because for any ζ with $\zeta^d = 1$, the vector $(1, \zeta, \cdots, \zeta^{d-1})$ is an eigenvector of W with eigenvalue ζ). Since similar matrices must have the same eigenvalues, it is impossible that $X^{-1}C$ and W are similar.

The remaining possibility we must consider is that this matrix is in $\mathcal{A}_\mathcal{R}\mathcal{A}_\mathcal{L}\mathcal{A}_\mathcal{R}\mathcal{A}_\mathcal{L}$. In this case by applying the symmetrical version of Lemma 3 (which follows by swapping the first n and last n coordinates), we see that $W^{-1}C$ and X must be similar. Since $\mathrm{Tr}(W^{-1}C) = 0$ and $\mathrm{Tr}(X) > 0$, this is impossible. □

B.3. Exact Representation of Nonlinear Functions

Finally, we give the proof of Corollary 4, which shows that there exist functions which cannot be exactly represented by 4 composed (nonlinear) affine couplings.

Proof of Corollary 4. Let $T \in \mathbb{R}^{2d\times 2d}$ be the matrix given by Theorem 3 for some d. Let $f(x) = Tx$. If g is any depth-4 affine coupling with nonlinear s and t functions its Jacobian at zero cannot be T, since its Jacobian can be calculated by precisely the matrix multiplication given in Theorem 3 with matrices which are the Jacobians of the affine coupling layers. So, g and f will not have matching Taylor expansions at zero. Hence, f cannot be exactly represented by any depth-4 affine coupling. □

B.4. Maximum Likelihood Estimation

In this section, we elaborate on the consequences of Theorem 2 for Maximum Likelihood Estimation with linear affine coupling networks.

Recall that if $\{P_\theta\}_\theta$ is a class of densities parameterized by θ, then the *Maximum Likelihood Estimate* of θ given data points $x_1, \ldots, x_n$ is any maximizer of the joint log likelihood of the data points x_i, namely

$$\sum_{i=1}^{n} \log P_\theta(x_i).$$

By Theorem 2, $\exists k \leq 47$ such that a linear affine coupling model with this many layers can implement an arbitrary orientation-preserving invertible linear map. This means that for this choice of k, the class of distributions writeable as the pushforward of $N(0, I)$ through a linear affine coupling layer with k layers is exactly the set of distributions $N(0, \Sigma)$ with Σ invertible. This follows since:
(1) The pushforward of $N(0, I)$ through a linear affine coupling network is a Gaussian distribution of the form $N(0, A^T A)$ where A is the linear map that the network computes, and
(2) Any distribution $N(0, \Sigma)$ can be sampled from by outputting $\Sigma^{1/2} Z$ where $Z \sim N(0, I)$, and by the simulation result a linear affine coupling model can exactly represent $\Sigma^{1/2}$.

As is well known (e.g. Chapter 8 of (Rao et al., 1973)), for the class of Gaussian distributions with zero-mean the maximum-likelihood estimate is the sample covariance matrix, i.e.

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T.$$

Since the MLE does not depend on the parameterization of the class of distributions, this also holds for the class of pushforwards of linear affine coupling models which we proved is equivalent. This leads to the following result giving consistency with Gaussian data.

Corollary 10. *Let $k \leq 47$ be as in Theorem 2, and suppose that $x_1, \ldots, x_n \sim N(0, \Sigma)$ i.i.d. with $\Sigma : d \times d$ invertible. Suppose that $\hat{P}$ is the Maximum Likelihood Estimator given data $x_1, \ldots, x_n$ among the class of distributions which are pushforwards of $N(0, I)$ through a linear affine coupling network with k layers. Then $\hat{P} \to N(0, \Sigma)$ as the number of samples $n \to \infty$, i.e. maximum likelihood estimation is consistent. Moreover, for any $\epsilon > 0$, $n = poly(d, 1/\epsilon)$ samples suffice to ensure $d_{TV}(\hat{P}, N(0, \Sigma)) \leq \epsilon$ with high probability.*

Proof. By the Law of Large Numbers, the sample covariance matrix is a consistent estimator for the true covariance matrix so we also obtain convergence of distributions, showing the first claim. The total variation distance guarantee follows from concentration of the sample covariance matrix about its mean, and from standard bounds on total variation distance between Gaussians in terms of their covariance matrices (Devroye et al., 2018). □

Note that this result is for the true MLE, i.e. the *global maximum* of the likelihood objective. When the log likelihood is maximized using e.g. gradient descent, this may not necessarily reach the global maximum.

C. Missing proofs for Section 6

In this section we prove Theorem 5; for the reader's convenience we repeat all details from the sketch here along with providing the omitted proofs. The intuition behind the k/p bound on the depth relies on parameter counting: a depth k/p invertible network will have k parameters in total (p per layer)—which is the size of the network we are trying to represent. Of course, the difficulty is that we need more than f_θ, g simply not being identical: we need a quantitative bound in various probability metrics.

Proof of Theorem 5. The proof will proceed as follows. First, we will exhibit a large family of distributions (of size $\exp(kd)$), s.t. each pair of these distributions has a large pairwise Wasserstein distance between them. Moreover, each distribution in this family will be approximately expressible as the pushforward of the Gaussian through a small neural network. Since the family of distributions will have a large pairwise Wasserstein distance, by the triangle inequality, no other distribution can be close to two distinct members of the family.

Second, we can count the number of "approximately distinct" invertible networks of depth l: each layer is described by p weights, hence there are lp parameters in total. The Lipschitzness of the neural network in terms of its parameters then allows to argue about discretizations of the weights.

Formally, we show the following lemma:

Lemma 7 (Large family of well-separated distributions)**.** *For every $k = o(exp(d))$, for d sufficiently large and $\gamma > 0$ there exists a family $\mathcal{D}$ of distributions, s.t. $|\mathcal{D}| \geq \exp(kd/20)$ and:*

1. *Each distribution $p \in \mathcal{D}$ is a mixture of k Gaussians with means $\{\mu_i\}_{i=1}^k$, $\|\mu_i\|^2 = 20\gamma^2 d$ and covariance $\gamma^2 I_d$.*
2. *$\forall p \in \mathcal{D}$ and $\forall \epsilon > 0$, we have $W_1(p, g_{\#\mathcal{N}}) \leq \epsilon$ for a neural network g with at most $O(k)$ parameters.*[7]
3. *For any $p, p' \in \mathcal{D}, W_1(p, p') \geq 20\gamma^2 d$.*

Proof of Lemma 7. The proof of this lemma will rely on two ideas: first, we will show that there is a family of distributions consisting of mixtures of Gaussians with k components – s.t. each pair of members of this family is far in W_1 distance, and each member in the family can be approximated by the pushforward of a network of size $O(k)$.

The reason for choosing mixtures is that it's easy to lower bound the Wasserstein distance between two mixtures with equal weights and covariance matrices in terms of the distances between the means. This is done in Lemma 8 below.

Lemma 8. *Let μ and ν be two mixtures of k spherical Gaussians in d dimensions with mixing weights $1/k$, means $(\mu_1, \mu_2, \ldots, \mu_k)$ and $(\nu_1, \nu_2, \ldots, \nu_k)$ respectively, and with all of the Gaussians having spherical covariance matrix $\gamma^2 I$ for some $\gamma > 0$. Suppose that there exists a set $S \subseteq [k]$ with $|S| \geq k/10$ such that for every $i \in S$,*

$$\min_{1 \leq j \leq k} \|\mu_i - \nu_j\|^2 \geq 20\gamma^2 d.$$

Then $W_1(\mu, \nu) = \Omega(\gamma\sqrt{d})$.

Proof. By the dual formulation of Wasserstein distance (Kantorovich-Rubinstein Theorem) (Villani, 2003), we have $W_1(\mu, \nu) = \sup_\varphi \left[\int \varphi d\mu - \int \varphi d\nu\right]$ where the supremum is taken over all 1-Lipschitz functions φ. Towards lower bounding this, consider $\varphi(x) = \max(0, 2\gamma\sqrt{d} - \min_{i \in S} \|x_i - \mu_i\|)$ and note that this function is 1-Lipschitz and always valued in $[0, 2\gamma\sqrt{d}]$. For a single Gaussian $Z \sim \mathcal{N}(0, \gamma^2 I_{d \times d})$, observe that

$$\mathrm{E}_{Z \sim \mathcal{N}(0, \gamma^2 I)}[\max(0, 2\gamma\sqrt{d} - \|Z\|)] \geq 2\gamma\sqrt{d} - \mathrm{E}_{Z)}[\|Z\|] \geq 2\gamma\sqrt{d} - \sqrt{\mathrm{E}_{Z \sim \mathcal{N}}[\|Z\|^2]} \geq \gamma\sqrt{d}.$$

Therefore, we see that $\int \varphi d\mu = \Omega(\gamma\sqrt{d})$ by combining the above calculation with the fact that at least $1/10$ of the centers for μ are in S. On the other hand, for $Z \sim \mathcal{N}(0, \gamma^2 I_{d \times d})$ we have

$$\Pr(\|Z\|^2 \geq 10\gamma^2 d) \leq 2e^{-10d}$$

(e.g. by Bernstein's inequality (Vershynin, 2018), as $\|Z\|^2$ is a sum of squares of Gaussians, i.e. a χ^2-random variable). In particular, since the points in S do not have a close point in $\{\nu_i\}_{i=1}^k$, we similarly have $\int \varphi d\nu = O(e^{-10d}\gamma\sqrt{d}) = o(\gamma\sqrt{d})$, since very little mass from each Gaussian in ν_i lands in the support of φ by the separation assumption. Combining the bounds gives the result. □

Given this, to design a family of mixtures of Gaussians with large pairwise Wasserstein distance, it suffices to construct a large family of k-tuples for the means, s.t. for each pair of k-tuples $(\{\mu_i\}_{i=1}^k, \{\nu_i\}_{i=1}^k)$, there exists a set $S \subseteq [k], |S| \geq k/10$, s.t. $\forall i \in S, \min_{1 \leq j \leq k} \|\mu_i - \nu_j\|^2 \geq 20\gamma^2 d$. We do this by leveraging ideas from coding theory (the Gilbert-Varshamov bound (Gilbert, 1952; Varshamov, 1957)). Namely, we first pick a set of $\exp(\Omega(d))$ vectors of norm $20\gamma^2 d$, each pair of which has a large distance; second, we pick a large number $(\exp(\Omega(kd)))$ of k-tuples from this set at random, and show with high probability, no pair of tuples intersect in more than $k/10$ elements.

Concretely, first, by elementary Chernoff bounds, we show there exists a large family of well-separated points in Lemma 9 below.

[7] The size of g doesn't indeed depend on ϵ. The weights in the networks will simply grow as ϵ becomes small.

Lemma 9 (Large family of well-separated points)**.** *Let $\epsilon > 0$. There exists a set $\{v_1, v_2, \ldots, v_N\}$ of vectors $v_i \in \mathbb{R}^d$, $\|v_i\| = 1$ with $N = \exp(d\epsilon^2/4)$, s.t. $\|v_i - v_j\|^2 \geq 2(1-\epsilon)$ for all $i \neq j$.*

Proof. Recall that for a random unit vector v on the sphere in d dimensions, $\Pr(v_i > t/\sqrt{d}) \leq e^{-t^2/2}$. (This is a basic fact about spherical caps, see e.g. (Rao, 2011)). By spherical symmetry and the union bound, this means for two unit vectors v, w sampled uniformly at random $\Pr(|\langle v, w\rangle| > t/\sqrt{d}) \leq 2e^{-t^2/2}$. Taking $t = \epsilon\sqrt{d}$ gives that the probability is $2e^{-d\epsilon^2/2}$; therefore if draw N i.i.d. vectors, the probability that two have inner product larger than ϵ in absolute value is at most $N^2 e^{-d\epsilon^2/2} < 1$ if $N = e^{d\epsilon^2/4}$, which in particular implies such a collection of vectors exists. □

From this, we construct a large set of k-sized subsets of this family which have small overlap, essentially by choosing such subsets uniformly at random. More precisely we use the following result:

Lemma 10 ((Rödl & Thoma, 1996))**.** *There exists a set consisting of $(\frac{N}{2k})^{k/10}$ subsets of size k of $[N]$, s.t. no pair of subsets intersect in more than $k/10$ elements.*

To handle part 2 of Lemma 7, we also show that a mixture of k Gaussians can be approximated as the pushforward of a Gaussian through a network of size $O(k)$. The idea is rather simple: the network will use a sample from a standard Gaussian in $\mathbb{R}^{d+1}$. We will subsequently use the first coordinate to implement a "mask" that most of the time masks all but one randomly chosen coordinate in $[k]$. The remaining coordinates are used to produce a sample from each of the components in the Gaussian, and the mask is used to select only one of them. This is done in Lemma 12 below. □

With Lemma 7 in hand, we finish the Wasserstein lower bound with a standard epsilon-net argument, using the parameter Lipschitzness of the invertible networks. Namely, the following lemma is immediate from standard covering number bounds for the Euclidean ball (Vershynin, 2018):

Lemma 11. *Suppose that $\Theta \subset \mathbb{R}^{d'}$ is contained in a ball of radius $R > 0$ and f_θ is a family of invertible layerwise networks which is L-Lipschitz with respect to its parameters. Then there exists a set of neural networks $S_\epsilon = \{f_i\}$, s.t. $|S_\epsilon| = O\left((\frac{LR}{\epsilon})^{d'}\right)$ and for every $\theta \in \Theta$ there exists a $f_i \in S_\epsilon$, s.t. $\mathrm{E}_{x\sim N(0,I_{d\times d})} \|f_\theta(x) - f_i(x)\|_\infty \leq \epsilon$.*

The proof of Theorem 5 can then be finished by triangle inequality: since the family of distributions has large Wasserstein distance, by the triangle inequality, no other distribution can be close to two distinct members of the family. Finally, KL divergence bounds can be derived from the Bobkov-Götze inequality (Bobkov & Götze, 1999), which lower bounds KL divergence by the squared Wasserstein distance. Concretely:

Theorem 11 ((Bobkov & Götze, 1999))**.** *Let $p, q : \mathbb{R}^d \to \mathbb{R}^+$ be two distributions s.t. for every 1-Lipschitz $f : \mathbb{R}^d \to \mathbb{R}^+$ and $X \sim p$, $f(X)$ is c^2-subgaussian. Then, we have $KL(q, p) \geq \frac{1}{2c^2} W_1(p, q)^2$.*

Then, to finish the two inequalities in the statement of the main theorem, we will show that:

- For any mixture of k Gaussians where the component means μ_i satisfy $\|\mu_i\| \leq M$, the condition of Theorem 11 is satisfied with $c^2 = O(\gamma^2 + M^2)$. (In fact, we show this for the pushforward through g, the neural network which approximates the mixture, which poses some non-trivial technical challenges). This is done in Appendix C, Lemma 12.
- A pushforward of the standard Gaussian through a L-Lipschitz generator f satisfies the conditions of Theorem 11 with $c^2 = L^2$, which implies the second part of the claim. This follows from Theorem 5.2.2 in (Vershynin, 2018).

□

C.1. Simulating a mixture with a neural network

Lemma 12. *Let $p : \mathbb{R}^d \to \mathbb{R}^+$ be a mixture of k Gaussians with means $\{\mu_i\}_{i=1}^k$, $\|\mu_i\|^2 = 20\gamma^2 d$ and covariance $\gamma^2 I_d$. Then, $\forall \epsilon > 0$, we have $W_1(p, g_{\#\mathcal{N}}) \leq \epsilon$ for a neural network g with $O(k)$ parameters.*[8]
Moreover, for every 1-Lipschitz $\phi : \mathbb{R}^d \to \mathbb{R}^+$ and $X \sim g_{\#\mathcal{N}}$, $\phi(X)$ is $O(\gamma^2 d)$-subgaussian.

[8] The size of g doesn't indeed depend on ϵ. The weights in the networks will simply grow with ϵ.

Proof. We will use a construction similar to (Arora et al., 2017). Since the latent variable dimension is $d+1$, the idea is to use the first variable, say h as input to a "selector" circuit which picks one of the components of the mixture with approximately the right probability, then use the remaning dimensions—say variable z, to output a sample from the appropriate component.

For notational convenience, let $M = \sqrt{20\gamma^2 d}$. Let $\{h_i\}_{i=1}^{k-1}$ be real values that partition $\mathbb{R}$ into k intervals that have equal probability under the Gaussian measure. Then, the map

$$\tilde{f}(h, z) = \gamma z + \sum_{i=1}^{k} 1(h \in (h_{i-1}, h_i])\mu_i \tag{15}$$

exactly generates the desired mixture, where h_0 is understood to be $-\infty$ and $h_k = +\infty$.

To construct g, first we approximate the indicators using two ReLUs, s.t. we design for each interval $(h_{i-1}, h_i]$ a function $\tilde{1}_i$, s.t.:
(1) $\tilde{1}_i(h) = 1(h \in (h_{i-1}, h_i])$ unless $h \in [h_{i-1}, h_{i-1} + \delta_{i-1}^+] \cup [h_i - \delta_i^-, h_i]$, and the Gaussian measure of the union of the above two intervals is δ.
(2) $\sum_i \tilde{1}_i(h) = 1$.
The constructions of the functions $\tilde{1}_i$ above can be found in (Arora et al., 2017), Lemma 3. We subsequently construct the neural network $f(h, z)$ using ReLUs defined as

$$f(h, z) = \gamma z + \sum_{i=1}^{k} \left(\text{ReLU}(-M(1 - \tilde{1}_i(h)) + \mu_i) - \text{ReLU}(-M(1 - \tilde{1}_i(h)) - \mu_i)\right). \tag{16}$$

Denoting

$$B := \bigcup_{i=1}^{k-1} [h_i - \delta_i^-, h_i + \delta_i^+]$$

note that if $h \notin B$, $\forall z$, $f(h, z) = \tilde{f}(h, z)$, as desired. If $h \in [h_i - \delta_i^-, h_i + \delta_i^+]$, $f(h, z)$ by construction will be $\gamma z + \sum_{i=1}^k w_i(h)\mu_i$ for some $w_i(h) \in [0, 1]$ s.t. $\sum_i w_i(h) = 1$.

Denoting by $\phi(h, z)$ the joint pdf of h, z, by the coupling definition of W_1, we have

$$\begin{aligned}
W_1(f_{\#\mathcal{N}}, \mu) \leq & \int_{h \in \mathbb{R}, z \in \mathbb{R}^d} \left|\tilde{f}(h, z) - f(h, z)\right|_1 d\phi(h, z) \\
= & \int_{h \in \mathbb{R}} \Bigg| \sum_{i=1}^{k} 1(h \in (h_{i-1}, h_i])\mu_i - \sum_{i=1}^{k} \big(\text{ReLU}(-M(1 - \tilde{1}_i(h)) + \mu_i) - \\
& \text{ReLU}(-M(1 - \tilde{1}_i(h)) - \mu_i)\big) \Bigg|_1 d\phi(h) \\
= & \int_{h \in B} \left| \sum_{i=1}^{k} 1(h \in (h_{i-1}, h_i])\mu_i - \sum_i w_i(h)\mu_i \right|_1 d\phi(h) \\
\leq & \int_{h \in B} \max_{i,j} |\mu_i - \mu_j|_1 d\phi(h) \\
= & \int_{h \in B} 2M\sqrt{d} d\phi(h) \\
= & \, 2M\sqrt{d} \Pr[h \in B] \\
= & \, 2M\sqrt{d} k\delta
\end{aligned}$$

So if we choose $\delta = \frac{\epsilon}{2M\sqrt{dk}}$, we have the desired bound in W_1. (We note, making δ small only manifests in the size of the weights of the functions $\tilde{1}$, and not in the size of the network itself. This is obvious from the construction in Lemma 3 in (Arora et al., 2017).)

Proceeding to subgaussianity, consider a 1-Lipschitz function φ centered such that $\mathrm{E}[(\varphi \circ f)_{\#\mathcal{N}}] = 0$. Next, we'll show

that $(\varphi \circ f)_{\#\mathcal{N}}$ is subgaussian with an appropriate constant. We can view $f_{\#\mathcal{N}}$ as the sum of two random variables: γz and

$$\sum_{i=1}^{k} \left(\text{ReLU}(-M(1-\tilde{1}_i(h)) + \mu_i) - \text{ReLU}(-M(1-\tilde{1}_i(h)) - \mu_i)\right).$$

γz is a Gaussian with covariance $\gamma^2 I$. The other term is contained in an l_2 ball of radius M. Using the Lipschitz property and Lipschitz concentration for Gaussians (Theorem 5.2.2 of (Vershynin, 2018)), we see that $\Pr[|(\varphi \circ f)| \geq t] \leq \exp\left(-\frac{(t-M)^2}{2\gamma^2}\right)$. By considering separately the cases $|t| \leq 2M$ and $|t| > 2M$, we immediately see this implies that the pushforward is $O(\gamma^2 + M^2)$-subgaussian. Since $M^2 = O(\gamma^2 d)$, the claim follows. □

D. Experimental verification

D.1. Partitioned Linear Networks

In this section, we will provide empirical support for Theorems 2 and 3. More precisely, empirically, the number of required linear affine coupling layers at least for random matrices seems closer to the lower bound – so it's even better than the upper bound we provide.

Setup We consider the following synthetic setup. We train n layers of affine coupling layers, namely networks of the form

$$f_n(z) = \prod_{i=1}^{n} E_i \begin{bmatrix} C_i & D_i \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ A_i & B_i \end{bmatrix}$$

with E_i, B_i, C_i diagonal. Notice the latter two follow the statement of Theorem 2 and the alternating order of upper vs lower triangular matrices can be assumed without loss of generality, as a product of upper/lower triangular matrices results in an upper/lower triangular matrix. The matrices E_i turn out to be necessary for training – they enable "renormalizing" the units in the network (in fact, Glow uses these and calls them actnorm layers; in older models like RealNVP, batchnorm layers are used instead).

The training data is of the form Az, where $z \sim \mathcal{N}(0, I)$ for a fixed $d \times d$ square matrix A with either random standard Gaussian entries in Figures 4 to 8 or random standard Gaussian entries that are diagonal-constant (that latter giving a natural random ensemble of Toeplitz matrices) in Figures 9 to 13. This ensures that there is a "ground" truth linear model that fits the data well. [9] We then train the affine coupling network by minimizing the loss $\mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[(f_n(z) - Az)^2\right]$ and trained on a variety of values for n and d in order to investigate how the depth of linear networks affects the ability to fit linear functions of varying dimension.

Note, we are not training via maximum likelihood, but rather we are minimizing a "supervised" loss, wherein the network f_n "knows" which point x a latent z is mapped to. This is intentional and is meant to separate the representational vs training aspect of different architectures. Namely, this objective is easier to train, and our results address the representational aspects of different architectures of flow networks – so we wish our experiments to be confounded as little as possible by aspects of training dynamics.

We chose $n = 1, 2, 4, 8, 16$ layers and $d = 4, 8, 16, 32, 64$ dimensions (here a layer is one matrix and not a flipped pair as in our theoretical results). We present the standard L2 training loss and the squared Frobenius error of the recovered matrix $\hat{A}$ obtained by multiplying out the linear layers $||\hat{A} - A||_F^2$, both normalized by $1/d^2$ so that they are independent of dimensionality. We shade the standard error regions of these losses across the seeds tried. All these plots are log-scale, so the noise seen lower in the charts is very small.

We initialize the E, C, B matrices with 1s on the diagonal and A, D with random Gaussian elements with $\sigma = 10^{-5}$ and train with Adam with learning rate 10^{-4}. We train on 5 random seeds which affect the matrix A generated and the datapoints z sampled.

Finally, we also train similar RealNVP models on the same datasets, using a regression objective as done with the PLNs but ~~s and t networks~~ with two hidden layers with 128 units and the same numbers of couplings as with the PNN experiments.

[9] As a side remark, this ground truth is only specified up to orthogonal matrices U, as AUz is identically distributed to Az, due to the rotational invariance of the standard Gaussian.

Results The results demonstrate that the 1- and 2- layer networks fail to fit even coarsely any of the linear functions we tried. Furthermore, the 4-layer networks consistently under-perform compared to the 8- and 16-layer networks. The 8- and 16-layer networks seem to perform comparably, though we note the larger mean error for d=64, which suggests that the performance can potentially be further improved (either by adding more layers, or improving the training by better choice of hyperparameters; even on this synthetic setup, we found training of very deep networks to be non-trivial).

These experimental results suggest that at least for random linear transformations T, the number of required linear layers is closer to the lower bound. Moreover, the error for the Toeplitz ensemble is slightly larger, indicating this distribution is slightly harder. Closing this gap (both in a worst-case and distributional sense) is an interesting question for further work.

In our experiments with the RealNVP architecture, we observe more difficulty in fitting these linear maps, as they seem to need more training data to reach similar levels or error. We hypothesize this is due to the larger model class that comes with allowing nonlinear functions in the couplings.

D.2. Additional Padding Results on Synthetic Datasets

We provide further results on the performance of Real NVP models on datasets with different kinds of padding (no padding, zero-padding and Gaussian padding) on standard synthetic datasets–Swissroll, 2 Moons and Checkerboard.

The results are consistent with the performance on the mixture of 4 Gaussians: in Figures 24, 25, and 26, we see that the zero padding greatly degrades the conditioning and somewhat degrades the visual quality of the learned distribution. On the other hand, Gaussian padding consistently performs best, both in terms of conditioning of the Jacobian, and in terms of the quality of the recovered distribution.

E. Approximating entrywise nonlinearity with affine couplings

To show how surprisingly hard it may be to represent even simple functions using affine couplings, we show an example of a very simple function—an entrywise application of hyperbolic tangent, s.t. an arbitrary depth/width sequence of affine coupling blocks with $\tanh$ nonlinearities cannot exactly represent it. Thus, even for simple functions, the affine-coupling structure imposes nontrivial restrictions. Note that in contrast to Theorem 1, we are considering exact representation here.

Precisely, we show:

Theorem 12. *Let $d \geq 2$ and denote $g : \mathbb{R}^d \to \mathbb{R}^d$, $g(z) := (\tanh z_1, \ldots, \tanh z_d)$. Then, for any $W, D, N \in \mathbb{N}$ and norm $\|\cdot\|$, there exists an $\varepsilon(W, D, N) > 0$, s.t. for any network f consisting of a sequence of at most N affine coupling layers of the form:*

$$(y_S, y_{\bar{S}}) \to (y_S, y_{\bar{S}} \odot a(y_S) + b(y_S))$$

for in each layer an arbitrary set $S \subsetneq [d]$ and a, b arbitrary feed-forward $\tanh$ neural networks of width at most W, depth at most D, and weight norm into each unit of at most R, it holds that

$$\mathrm{E}_{x \in [-1,1]^d} \|f(x) - g(x)\| > \varepsilon(W, D, N, R).$$

The proof of the theorem is fairly unusual, as it uses some tools from complex analysis in several variables (see (Grauert & Fritzsche, 2012) for a reference) — though it's so short that we include it here. The result also generalizes to other neural networks with analytic activations.

Proof of Theorem 12. By compactness of the class of models bounded by W, D, N, R, it suffices to prove that there is no way to exactly represent the function.

Suppose for contradiction that $f = g$ on the entirety of $[-1, 1]^d$. Let $z_1, \ldots, z_d$ denote the d inputs to the function: we now consider the behavior of f and g when we extend their definition to $\mathbb{C}^d$. From the definition, g extends to a holomorphic function (of several variables) on all of $\mathbb{C}^d \setminus \{z : \exists j, z_j = i\pi(k + 1/2) : k \in \mathbb{Z}\}$, i.e. everywhere where $\tanh$ doesn't have a pole. Similarly, there exists an dense open subset $D \subset \mathbb{C}^d$ on which the affine coupling network f is holomorphic, because it is formed by the addition, multiplication, and composition of holomorphic functions.

We next prove that $f = g$ on their complex extensions by the Identity Theorem (Theorem 4.1 of (Grauert & Fritzsche, 2012)). We must first show that $f = g$ on an open subset of $\mathbb{C}^d$. To prove this, observe that f is analytic at zero and its

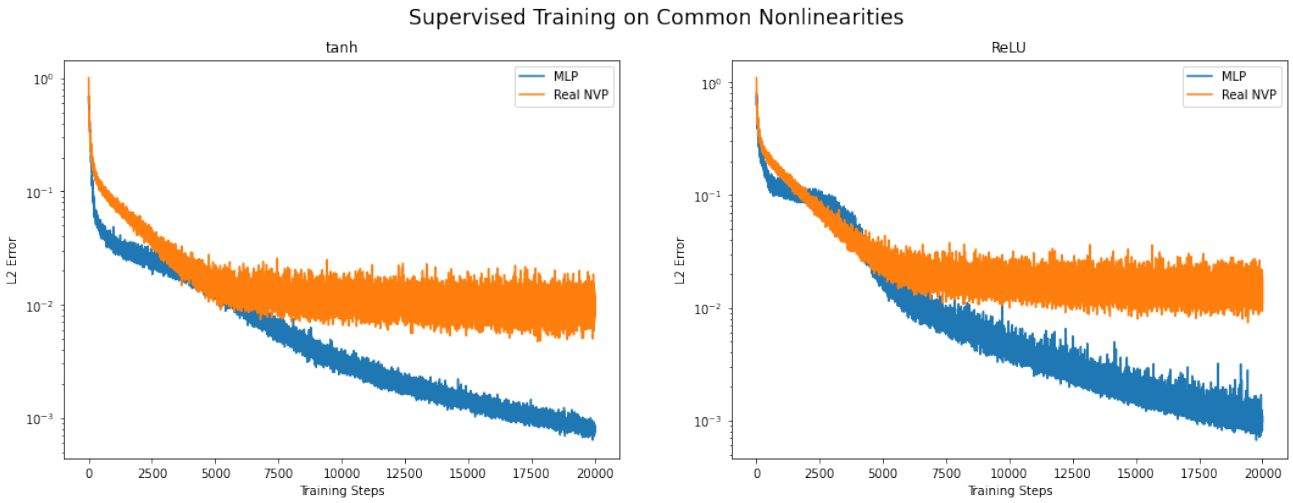

Figure 3. The smaller MLPs are much better able to fit simple elementwise nonlinearities than the affine couplings.

power series expansion is uniquely defined in terms of the values of f on $\mathbb{R}^d$ (for example, we can compute the coefficients by taking partial derivatives). It follows that the power series expansions of f and g are both equal at zero and convergent in an open neighborhood of 0 in $\mathbb{C}^d$, so we can indeed apply the Identity Theorem; this shows that $f = g$ wherever they are both defined.

From the definition $\tanh(z) = \frac{e^{2z}-1}{e^{2z}+1}$ we can see that g is periodic in the sense that $g(z + \pi i k) = g(z)$ for any $k \in \mathbb{Z}^d$. However, by construction the affine coupling network f is invertible whenever, at every layer, the output of the function a is not equal to zero. By the identity theorem, the set of inputs where each a vanishes is nowhere dense — otherwise, by continuity a vanishes on the open neighborhood of some point, so $a = 0$ by the Identity Theorem which contradicts the assumption. Therefore the union of inputs where a at any layer vanishes is also nowhere dense. Consider the behavior of f on an open neighborhood of 0 and of $i\pi$: we have shown that f is invertible except on a nowhere dense set, and also that $g = f$ wherever f is defined, but $g(z) = g(z + i\pi)$ so it's impossible for f to be invertible on these neighborhoods except on a nowhere dense subset. By contradiction, $f \neq g$ on $[-1, 1]^d$. □

Finally, to give empirical evidence that the above is not merely a theoretical artifact, we regress an affine coupling architecture to fit entrywise $\tanh$.

Specifically, we sample 10-dimensional vectors from a standard Gaussian distribution and train networks as in the padding section on a squared error objective such that each input is regressed on its elementwise $\tanh$. We train an affine coupling network with 5 pairs of alternating couplings with g and h networks consisting of 2 hidden layers with 128 units each. For comparison, we also regress a simple MLP with 2 hidden layers with 128 units in each layer, exactly one of the g or h subnetworks from the coupling architecture, which contains 20 such subnetworks. For another comparison, we also try this on the elementwise ReLU function, using affine couplings with $\tanh$ activations and the same small MLP.

As we see in Figure 3, the affine couplings fit the function substantially worse than a much smaller MLP – corroborating our theoretical result.

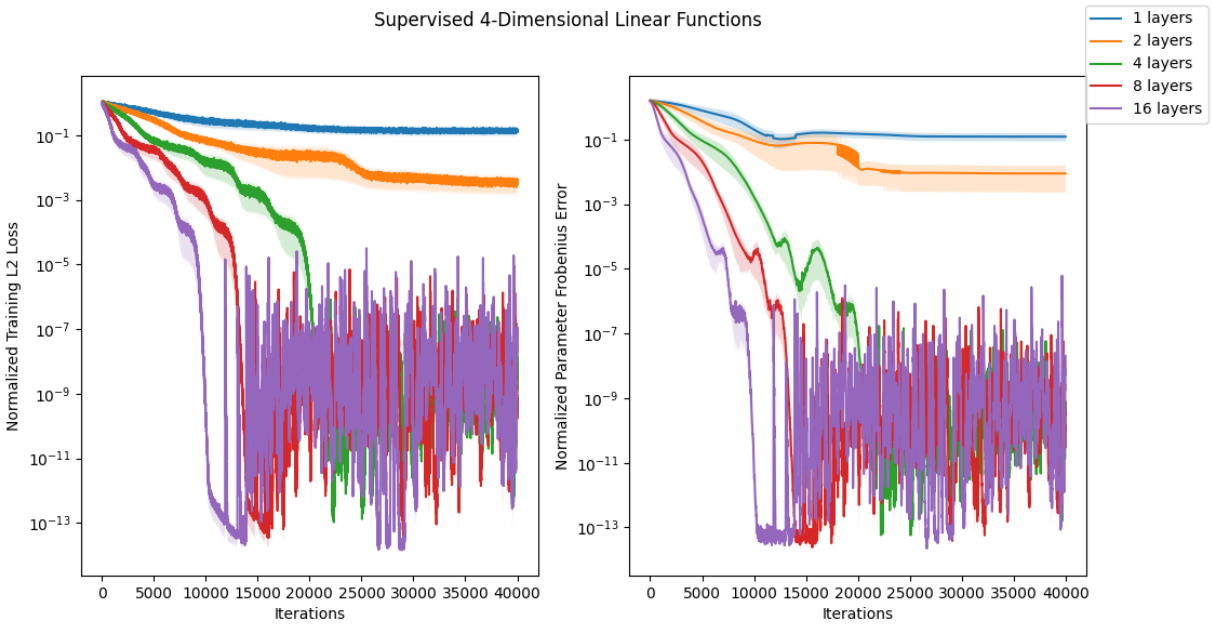

Figure 4. Learning Partitioned Linear Networks on 4-D linear functions.

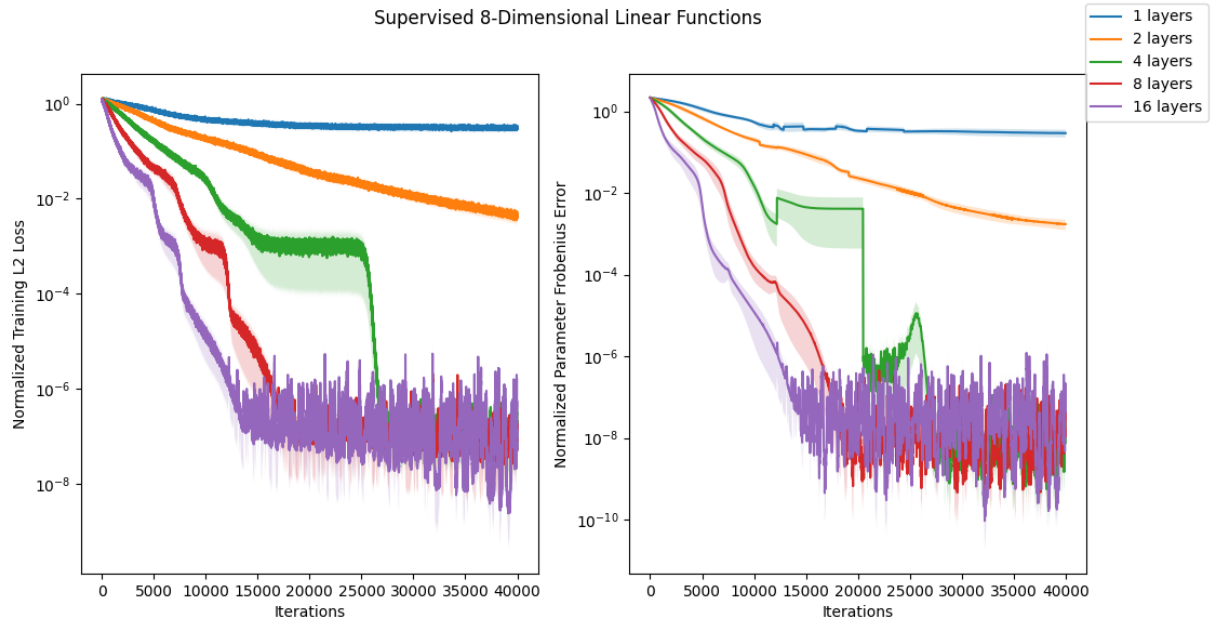

Figure 5. Learning Partitioned Linear Networks on 8-D linear functions.

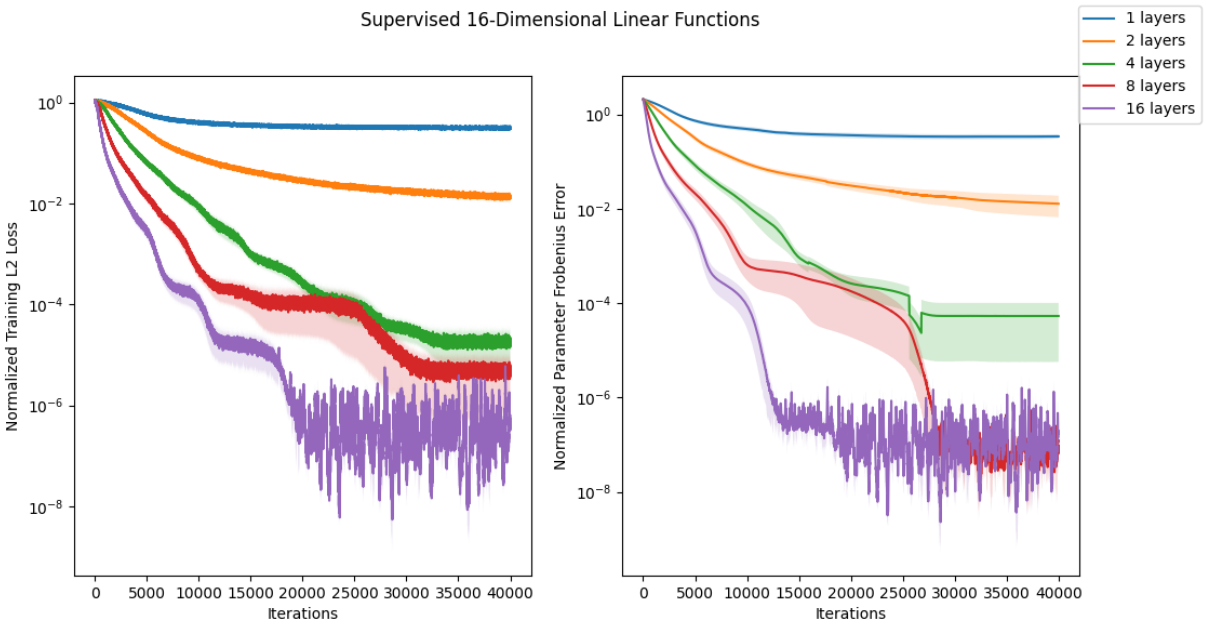

Figure 6. Learning Partitioned Linear Networks on 16-D linear functions.

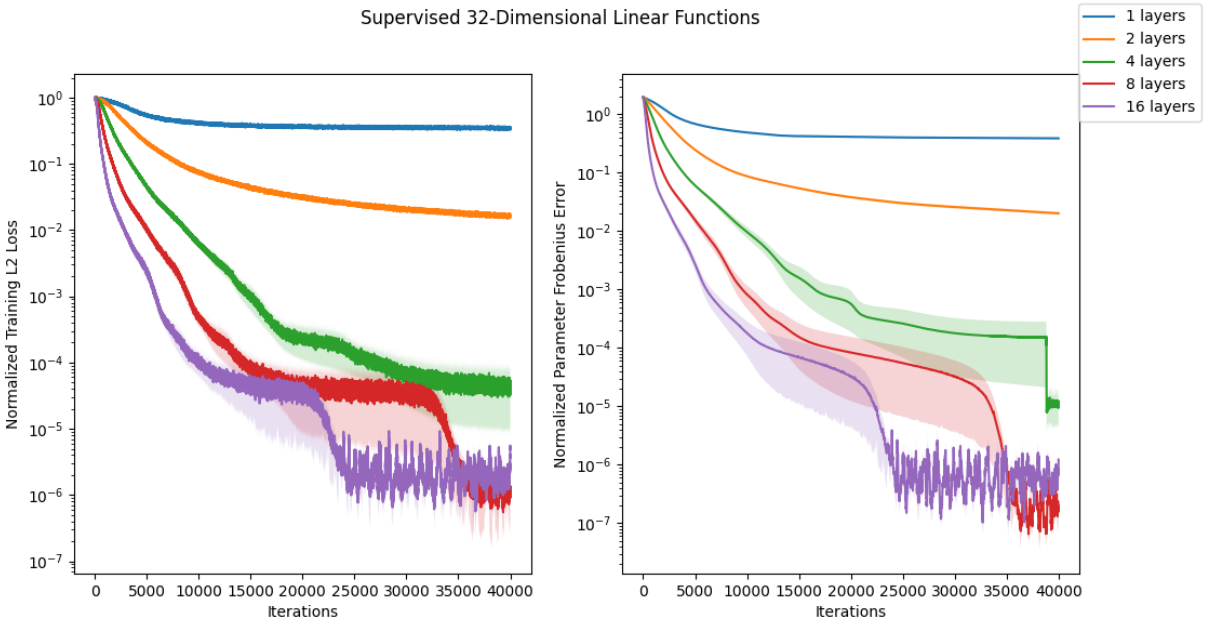

Figure 7. Learning Partitioned Linear Networks on 32-D linear functions.

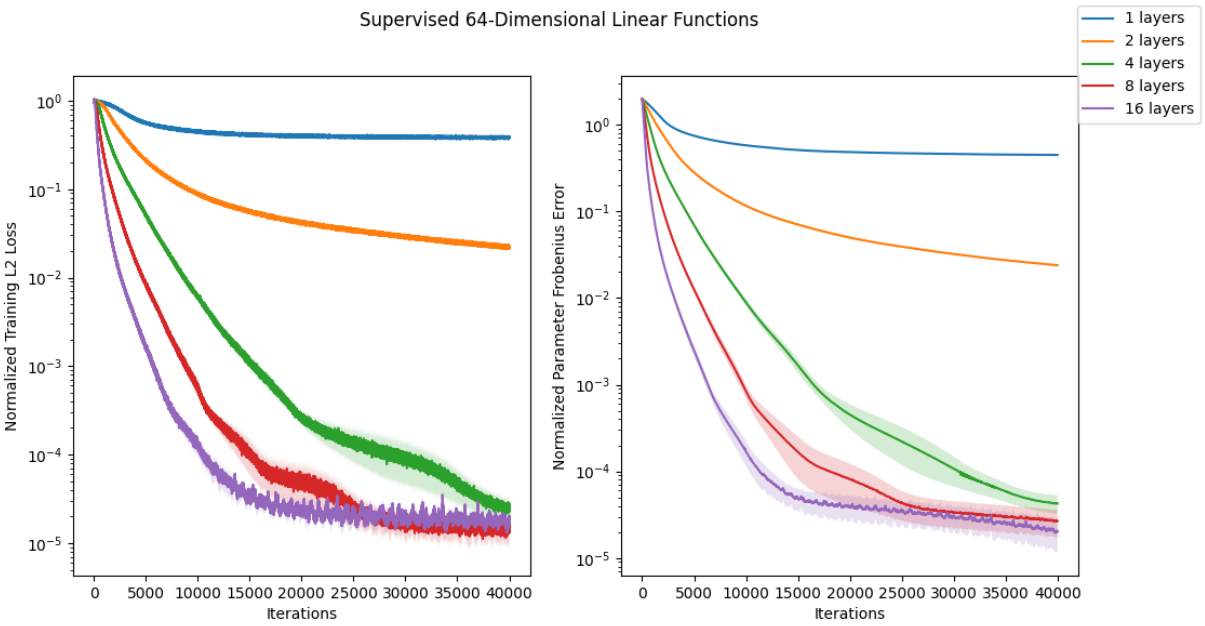

Figure 8. Learning Partitioned Linear Networks on 64-D linear functions.

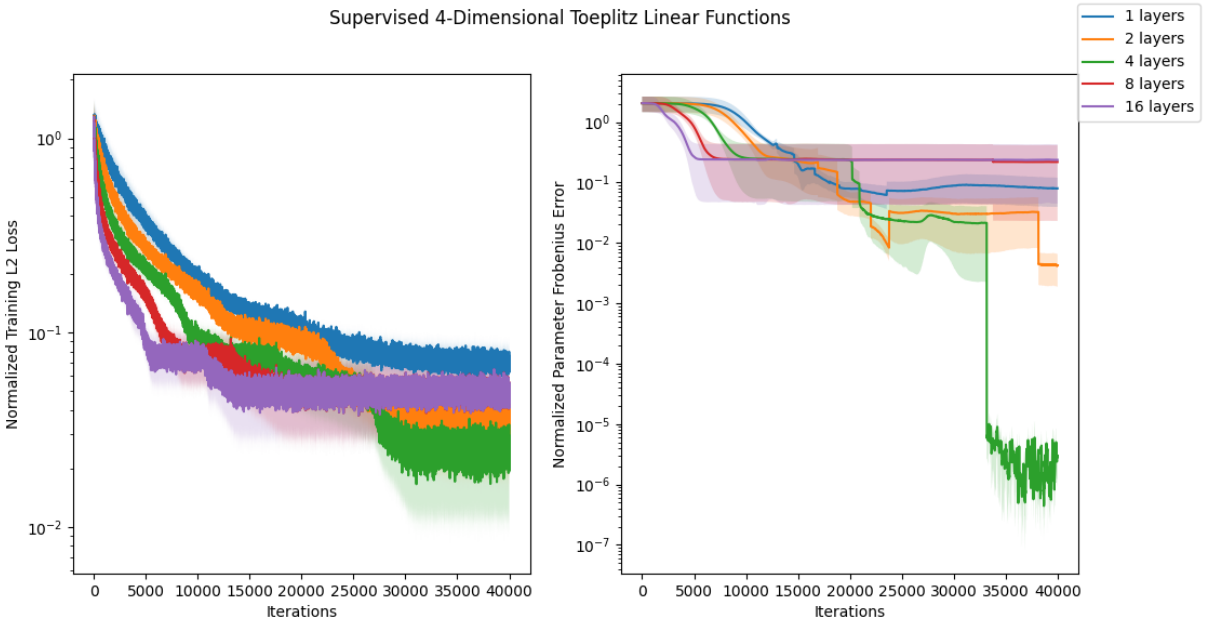

Figure 9. Learning Partitioned Linear Networks on 4-D Toeplitz functions.

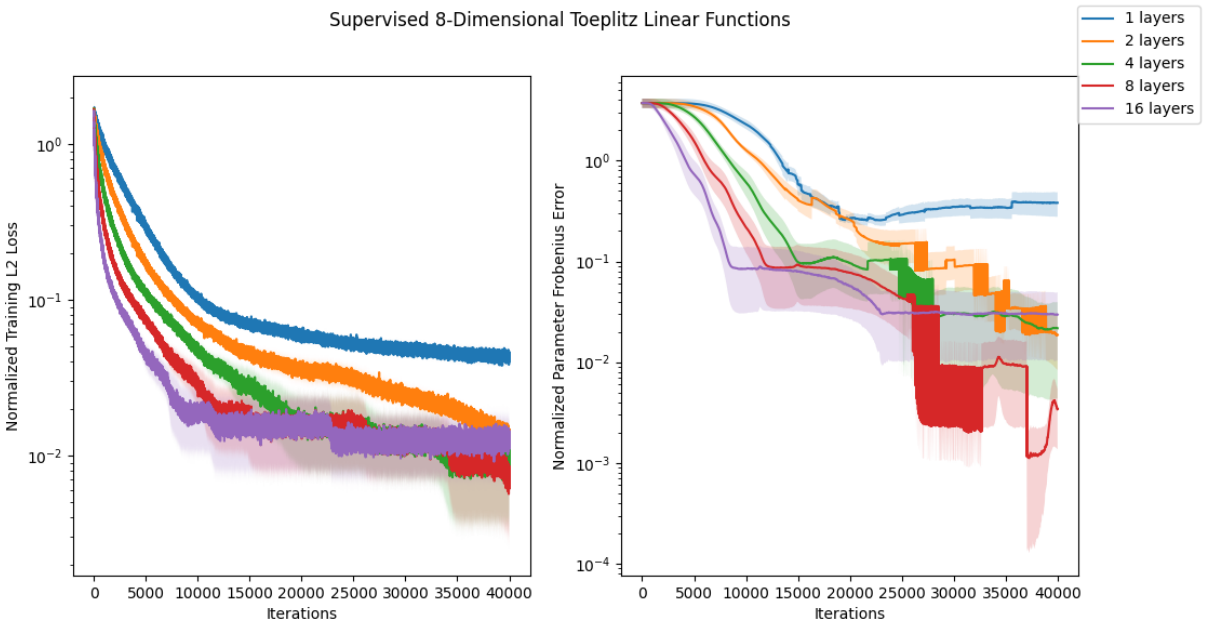

Figure 10. Learning Partitioned Linear Networks on 8-D Toeplitz functions.

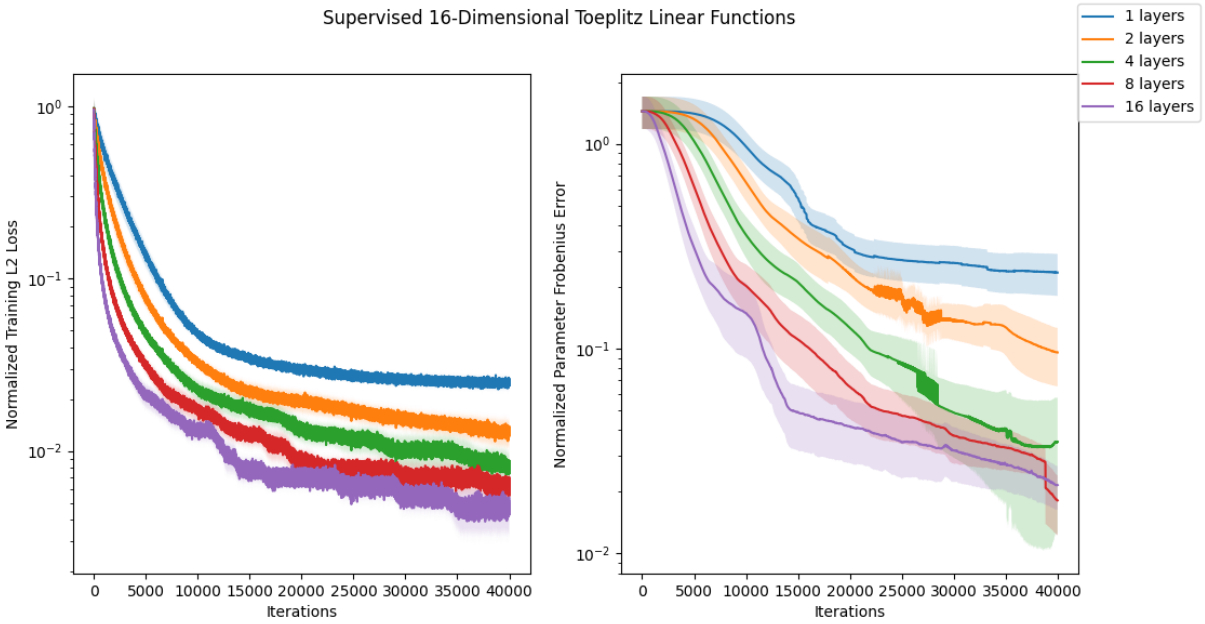

Figure 11. Learning Partitioned Linear Networks on 16-D Toeplitz functions.

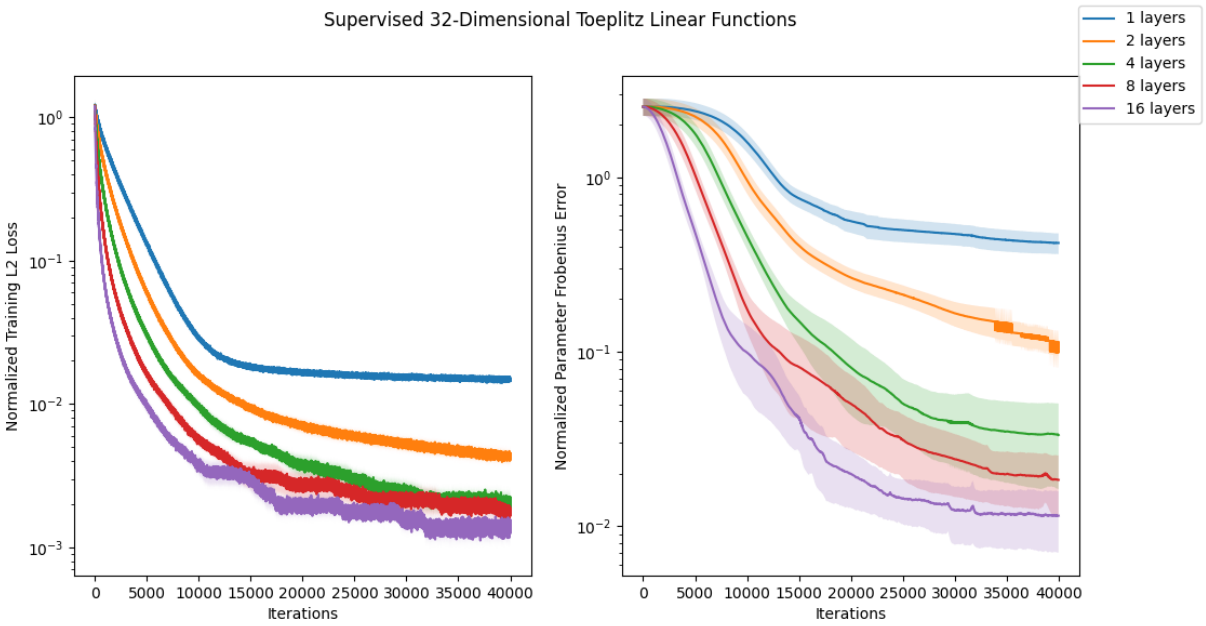

Figure 12. Learning Partitioned Linear Networks on 32-D Toeplitz functions.

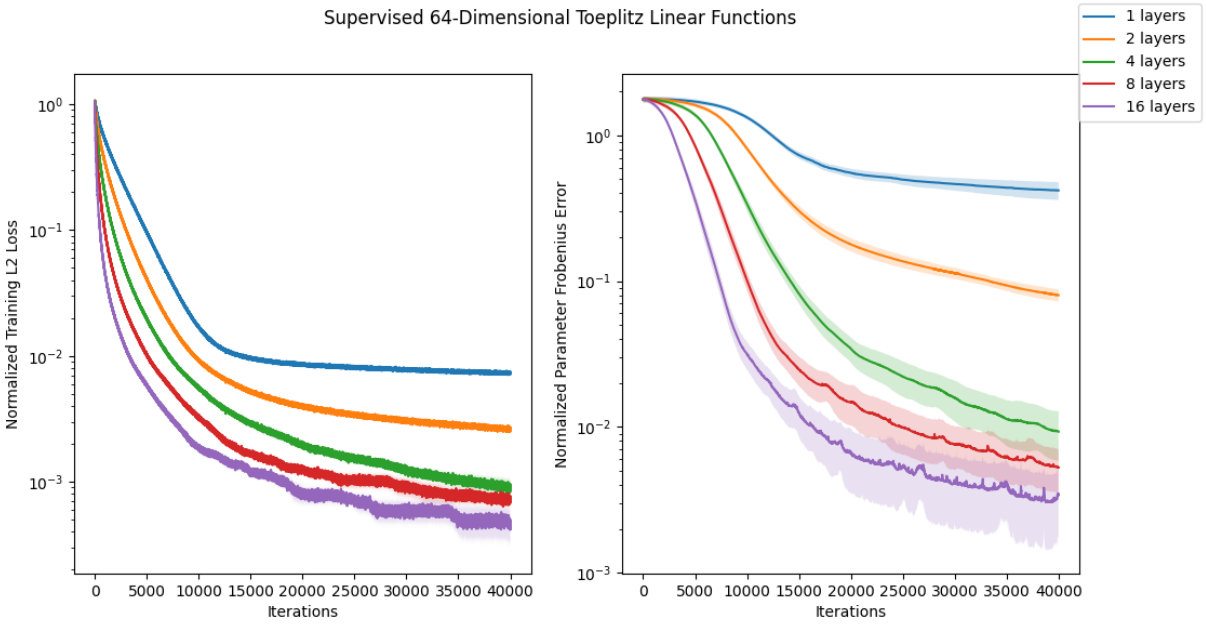

Figure 13. Learning Partitioned Linear Networks on 64-D Toeplitz functions.

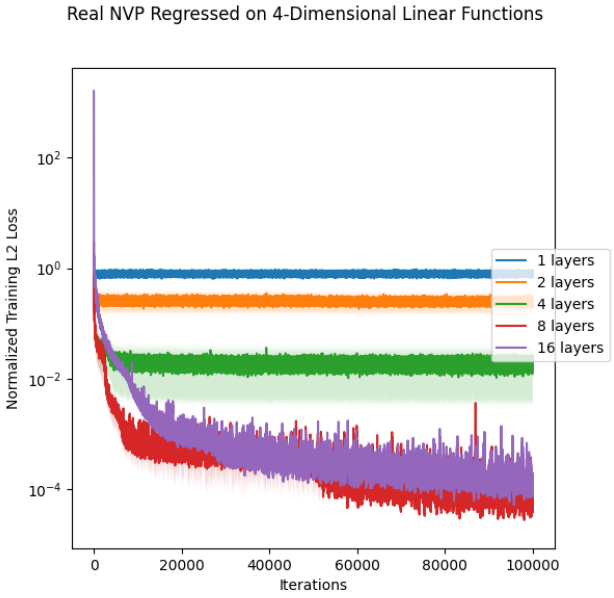

Figure 14. Real NVP Regressed on 4-D Linear Functions

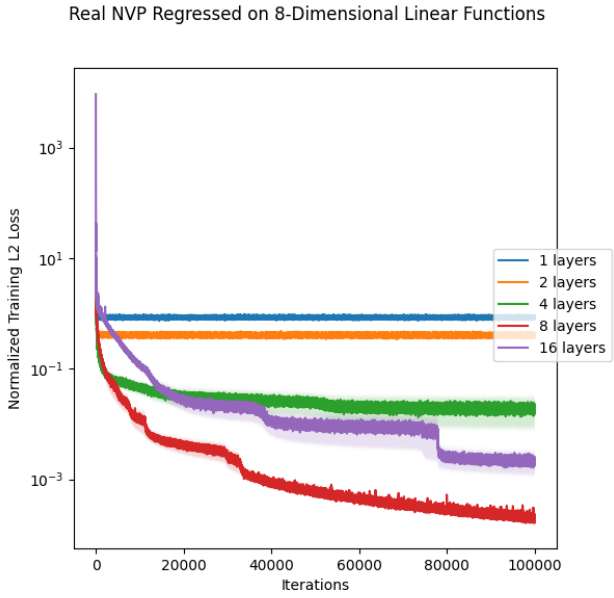

Figure 15. Real NVP Regressed on 8-D Linear Functions

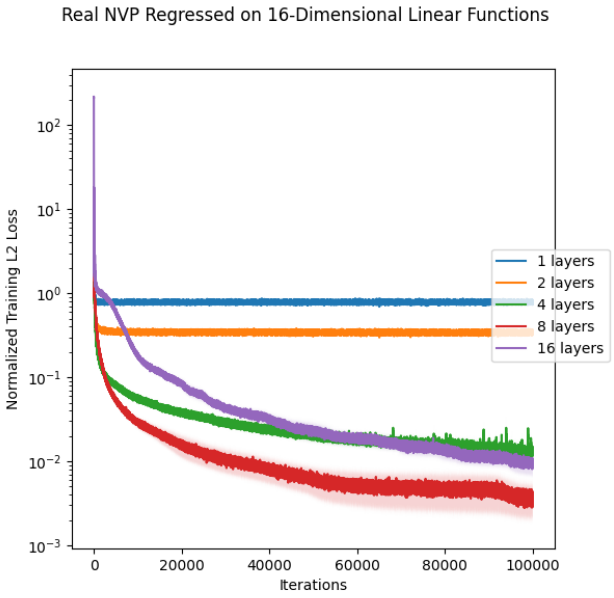

Figure 16. Real NVP Regressed on 16-D Linear Functions

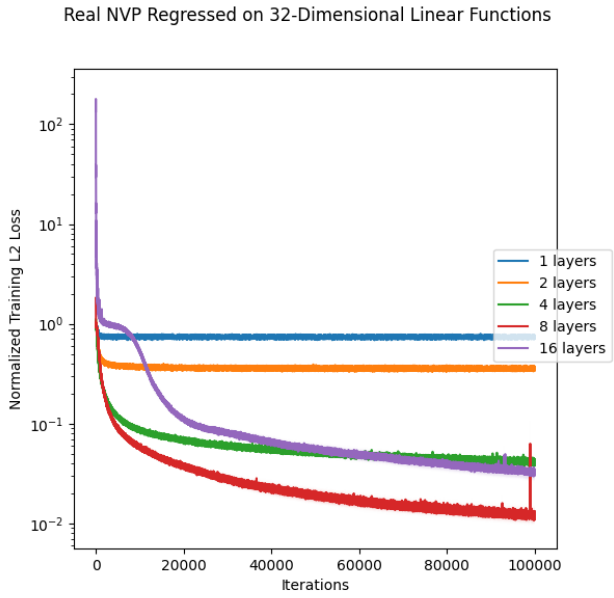

Figure 17. Real NVP Regressed on 32-D Linear Functions

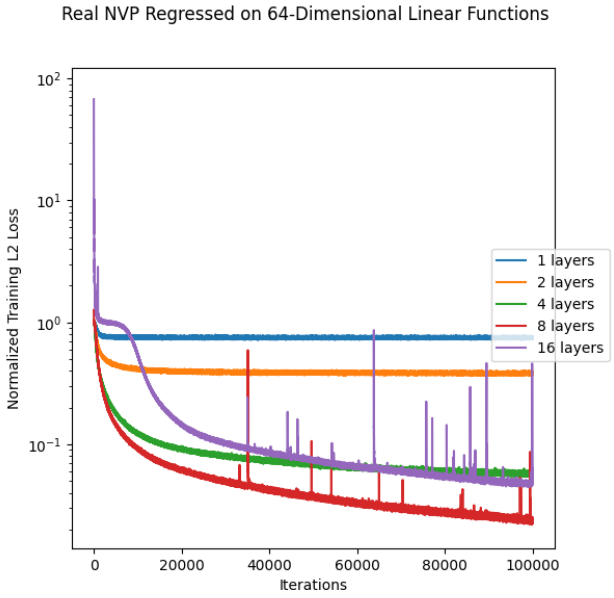

Figure 18. Real NVP Regressed on 64-D Linear Functions

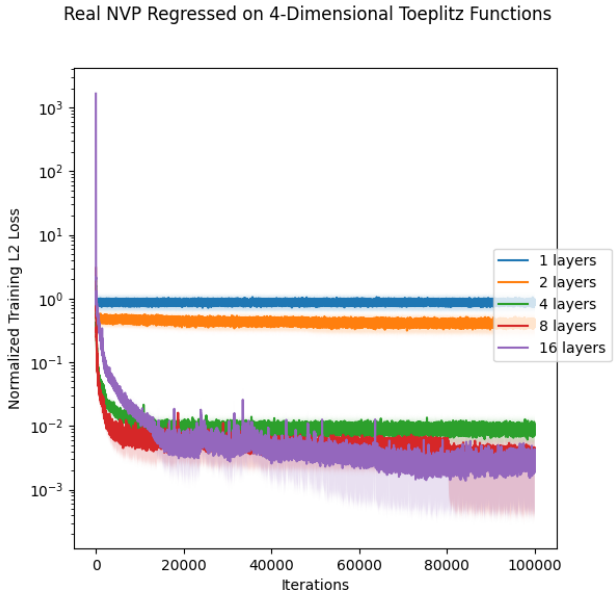

Figure 19. Real NVP Regressed on 4-D Toeplitz Functions

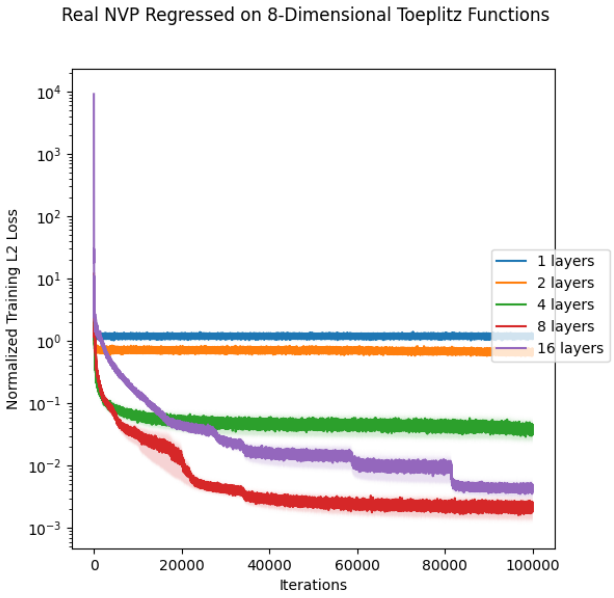

Figure 20. Real NVP Regressed on 8-D Toeplitz Functions

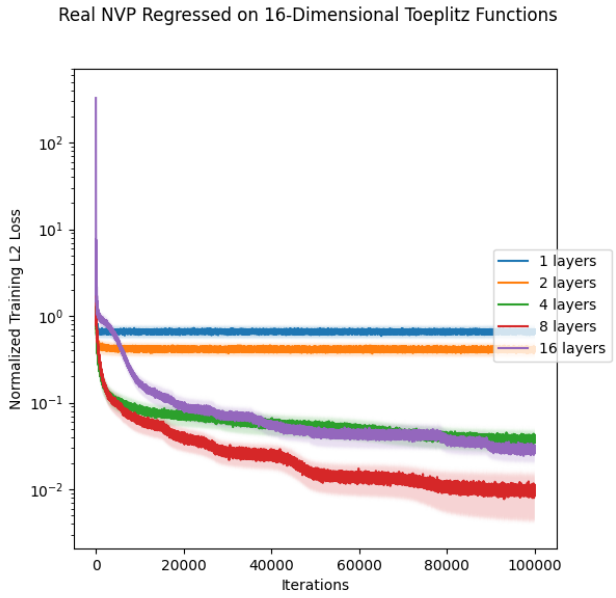

Figure 21. Real NVP Regressed on 16-D Toeplitz Functions

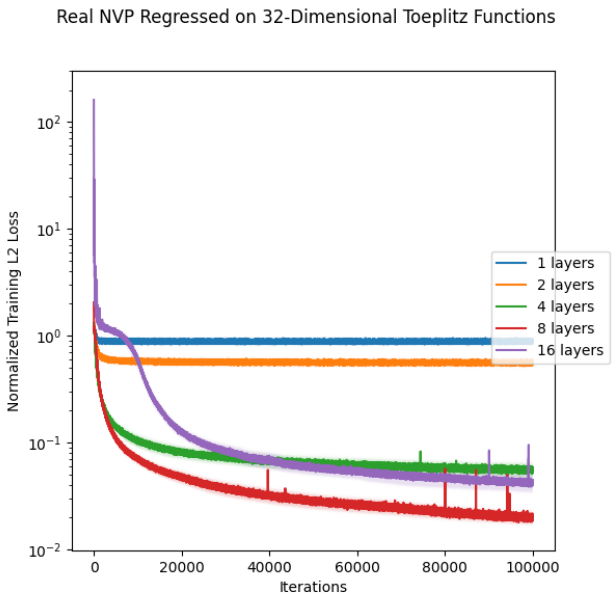

Figure 22. Real NVP Regressed on 32-D Toeplitz Functions

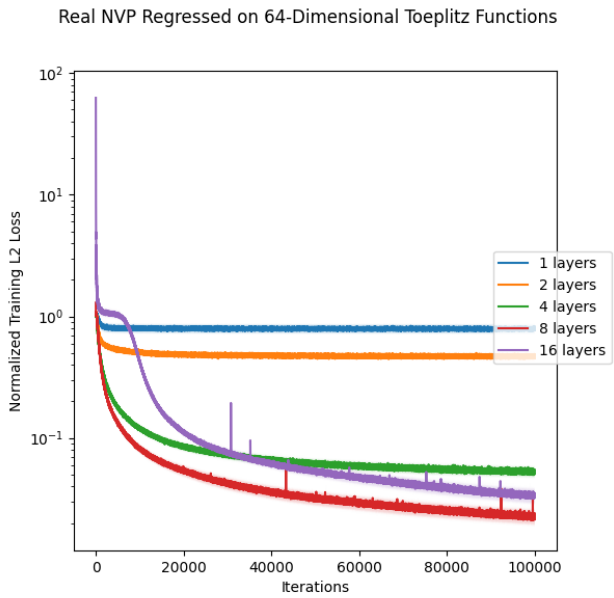

Figure 23. Real NVP Regressed on 164-D Toeplitz Functions

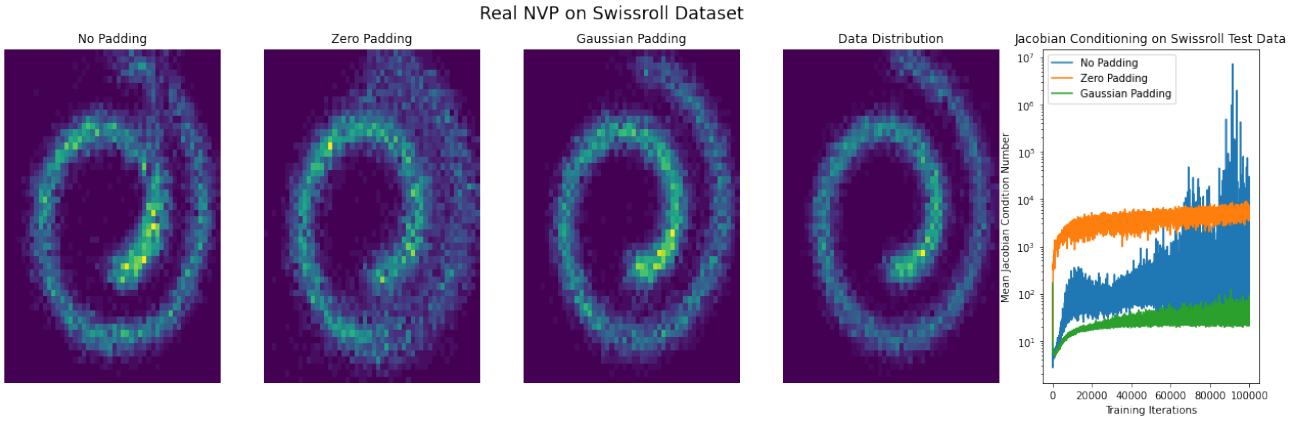

Figure 24. Real NVP on Swissroll Dataset

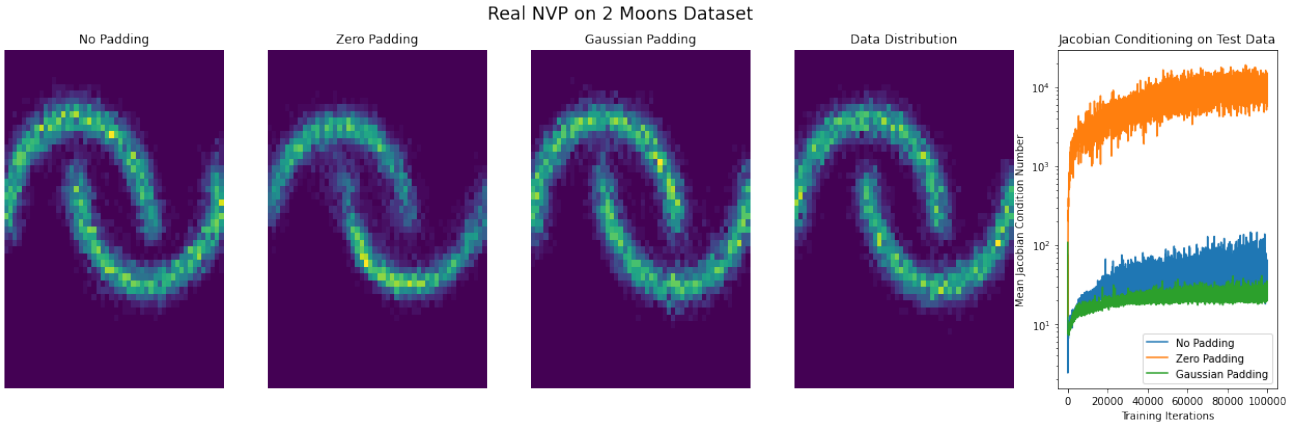

Figure 25. Real NVP on 2 Moons Dataset

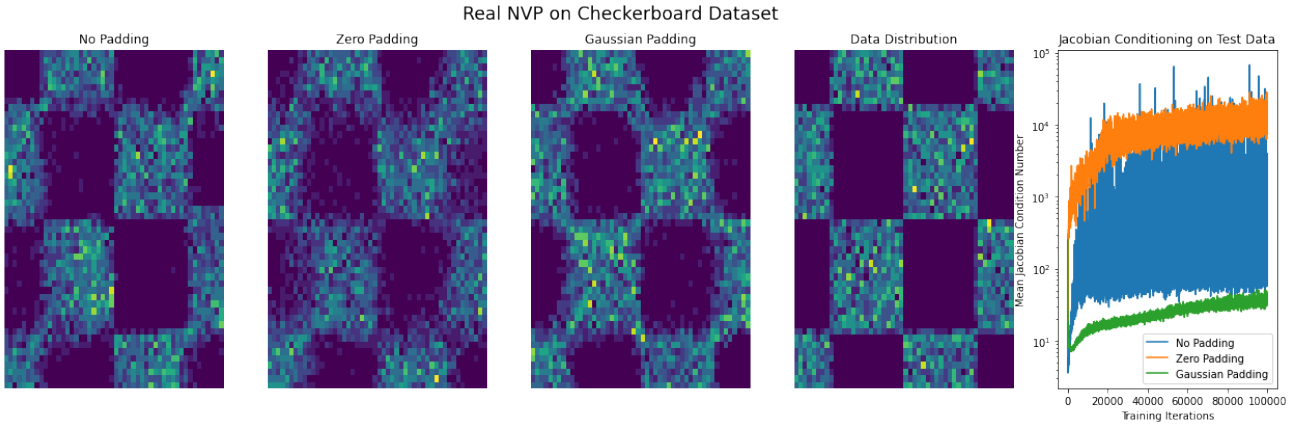

Figure 26. Real NVP on Checkerboard Dataset