A. Tree Destructor Details

We give more details on how to compute the tree destructor transformation based on a density tree. Note that the tree destructor transformation is simply a piecewise independent linear transformation that depends on the leaf node of the sample.

A.1. Base case: One dimension and one split

The base case of density tree transformation is only acting on one dimension while leaving all the others fixed. Thus, we can simply consider a tree in one dimension with only one split. We denote $[a, b]$ to be the domain of a node, $t \in [a, b]$ is the splitting threshold, and $p \in [0, 1]$ is the relative probability of the left child (the relative probability of the right child is just $(1 - p)$). Thus, the parameters of a node are simply (a, b, t, p).

The transformation for the left $T_\ell(\cdot)$ and right $T_r(\cdot)$ of the split is as follows:

$$t_{\text{out}} \equiv (b - a)p + a \tag{9}$$

$$T_\ell(x) = \frac{t_{\text{out}} - a}{t - a}(x - a) + a \tag{10}$$

$$T_r(x) = \frac{b - t_{\text{out}}}{b - t}(x - t) + t_{\text{out}} \, , \tag{11}$$

where t_{out} can be viewed as the "output threshold" (i.e. where the threshold in the output domain for this node should be to yield a uniform density). This yields the following scales α and shifts β:

$$\alpha_\ell = \frac{t_{\text{out}} - a}{t - a} \tag{12}$$

$$\beta_\ell = a - a\alpha_\ell \tag{13}$$

$$\alpha_r = \frac{b - t_{\text{out}}}{b - t} \tag{14}$$

$$\beta_r = t_{\text{out}} - t\alpha_r \, . \tag{15}$$

A.2. Inductive Case

The full recursive details for a deeper tree is straightforward. We merely accumulate the linear transformation as we traverse the tree but only apply the transformation at the leaves. Composing any two linear operations can be simplified to a single linear operation:

$$T_1(T_2(x)) = \alpha_2(\alpha_1 x + \beta_1) + \beta_2 \tag{16}$$

$$\tilde{T}_{12}(x) = \alpha_2\alpha_1 x + \alpha_2\beta_1 + \beta_2 \tag{17}$$

$$= \tilde{\alpha} x + \tilde{\beta} \tag{18}$$

Thus accumulating transformations when traversing the tree only requires computing shift and scale values (as needed) for each node based on previous nodes and the current node.

B. Enlarged Figures

See next pages for enlarged figures.

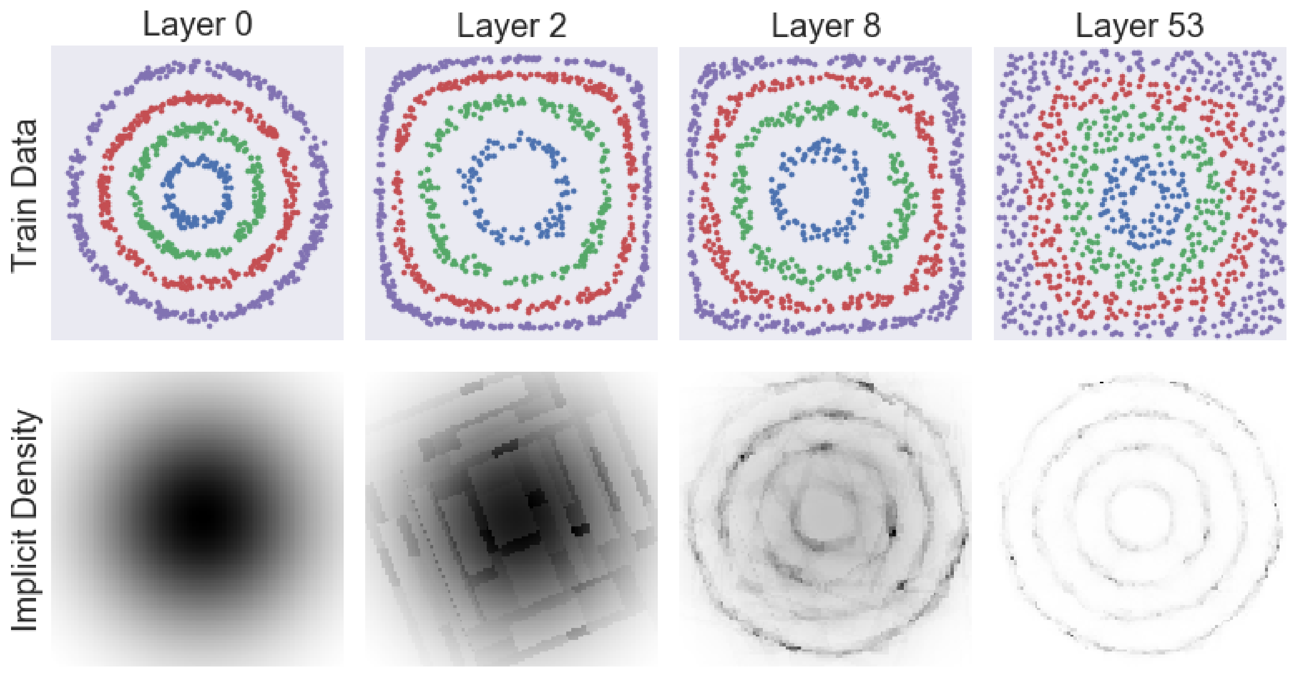

Figure 5. The transformed samples (top) and implicit density (bottom) at different layers of the DensityTree (100) model described in Sec. 4. While the initial layers do not provide a good estimate of the underlying density, after many layers, the underlying density begins to model the true underlying patterns.

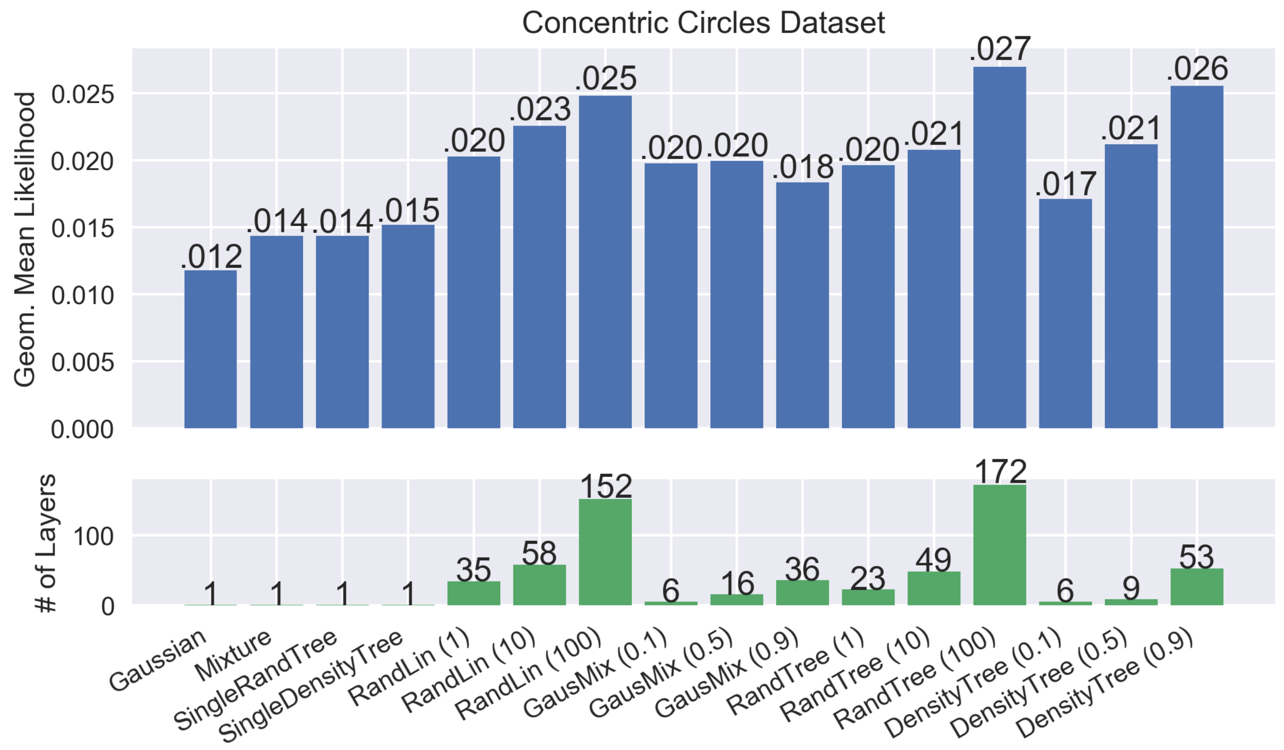

Figure 6. Deep models clearly outperform baseline models, and random trees seem to perform the best in terms of test likelihood. As expected, the selected number of layers increases as the amount of regularization on each layer increases.

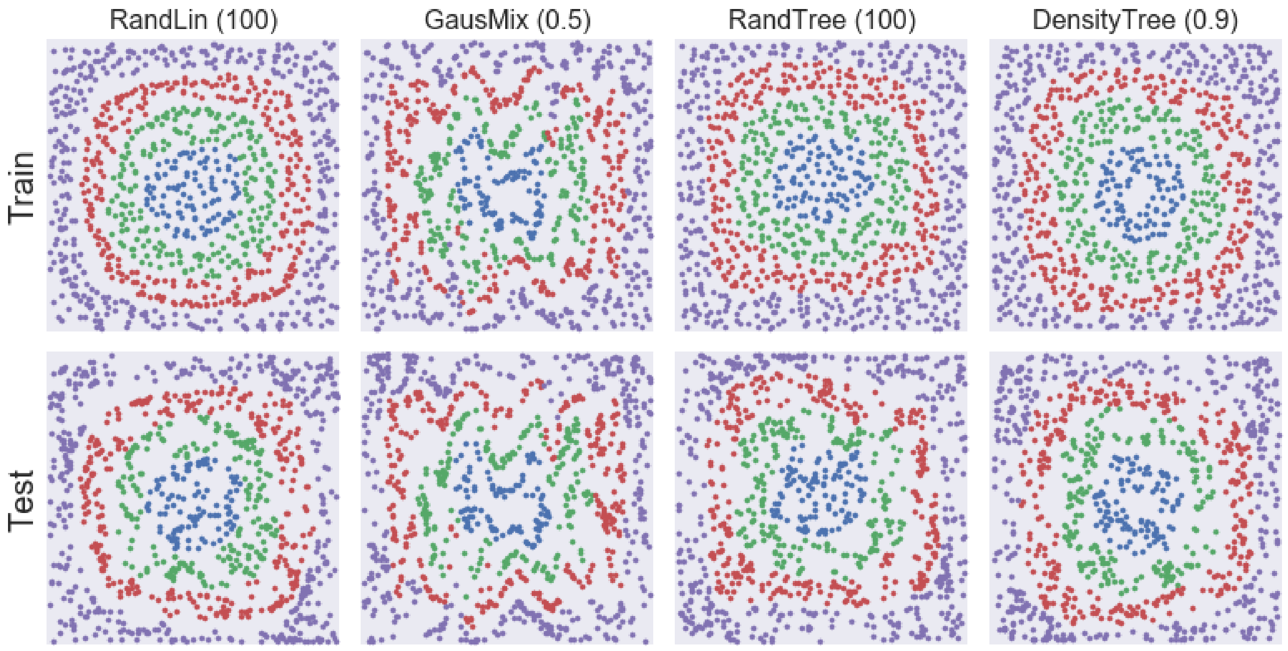

Figure 7. Each deep model transforms the data differently with DensityTree seeming to be the most regular. Notice that while the train data may be evenly spaced on the unit square, this does not mean the test data is also uniform. This is why highly regularized destructors are required to build usable deep models via a greedy algorithm.