1. Additional Experiments

Table 1 contains a summary of each of the 14 selected datasets. It also contains the AUC and APR achieved by the Target-Only model on each dataset which is used as the baseline in Fig. 1, where we report the results of training a multi-layer perceptron (MLP) instead of the logistic regression used in the main paper. As can be seen from the figure, the results are very similar for the MLP, with RadialGAN outperforming the two benchmarks in almost all datasets.

In Fig. 2 we compare the performance of training a predictive model when additional samples are generated by RadialGAN versus simply having additional true samples (i.e. from the target dataset), which would be the best thing we could possibly have (a model whose goal is to generate additional samples cannot hope to generate samples that are better than true samples). 100 samples are taken from the target dataset, and RadialGAN then generates additional ones using other datasets, whereas True Datasets simply uses additional samples from the target dataset. As can be seen from the figure, the performance improvement from RadialGAN is almost as good as from simply having more true samples, indicating that RadialGAN is able to generate additional samples that provide almost as much information to a predictive model as the true samples would. In the figure, Target-Only is the performance of a predictive model trained only on the initial 100 samples (i.e. it makes no use of any additional samples), and is given to indicate the effect that generating no further samples would have.

---

1University of California, Los Angeles, CA, USA 2University of Oxford, UK 3Alan Turing Institute, UK. Correspondence to: Jinsung Yoon <jsyoon0823@gmail.com>.

2. Hyper-parameter Optimization

In all experiments described in the main manuscript, the depth of the generator and discriminator in all benchmarks is set to 3. The depth of all encoders, decoders and discriminators in RadialGAN is also set to 3. For all the benchmarks and RadialGAN, the number of hidden nodes in the first, second and third hidden layers is \( d, d/2 \) and \( d \) (where \( d \) is the dimension of the input), respectively. We use \( \text{tanh} \) as the activation function for each layer except for the output layer where we use the sigmoid activation function and the size of all mini-batches is 64 (i.e. \( k_G = k_D = 64 \)). For RadialGAN, we use cross-validation to select \( \lambda, \beta \) among \( \{0.1, 0.5, 1, 2, 10\} \) and also to select the dimension of \( Z \) from \( \{\max_i(d_i), \max_i(d_i/16), \max_i(d_i/4)\} \) where \( d_i \) is the dimension of \( X^{(i)} \). For CycleGAN (Zhu et al., 2017), co-GAN (Mirza & Osindero, 2014), and StarGAN (Choi et al., 2017), we also use cross-validation to select \( \lambda \) among \( \{0.1, 0.5, 1, 2, 10\} \). (Wiens et al., 2014) does not have any hyper-parameters to be optimized. We use tensorflow to implement all the benchmarks and RadialGAN.
References


