**NeurIPS Hide-and-seek Privacy Challenge documentation questionnaire**

**Team name**

|  |
| --- |
| UniMelb Peekaboo |

**Submission filenames(s)**

|  |  |
| --- | --- |
| Hider | scoring-eval.zip  scoring-eval/utils/timegan.py contains commented code |
| Seeker |  |

**What class of algorithms does your solution belong to?** (e.g. GANs, VAEs, noise-injection, nearest neighbor, etc.)

|  |  |
| --- | --- |
| Hider | TimeGAN |
| Seeker |  |

**Describe your algorithm in one sentence** (e.g. “Noise is added to the original data and then this data is returned.”)

|  |  |
| --- | --- |
| Hider | Differentially private (DP) training of TimeGan. |
| Seeker |  |

**Describe your algorithm in words** (e.g. “Noise is drawn from a Gaussian distribution, with mean 0 and variance s, where the dimension is determined by the size of the dataset. This noise is added to the original data to produce a noisy version of the dataset and this noisy dataset is then returned as the synthetic data.”)

|  |  |
| --- | --- |
| Hider | \*Team name: UniMelb Peekaboo  \*Method name: DPTimeGAN  \*Method description:  ====================  Our approach is inspired by differential privacy (DP) [1,2] to obtain a privacy-preserving version of TimeGan.  Differential privacy, at a high level, provides record-level privacy for a result of a function f: anyone who observes f(D) on some dataset D, cannot tell whether some record r was in D or not. In our case, f is a gradient computation and D is a batch of records.  A common approach to obtain DP when training a machine learning model is by using a Gaussian mechanism as follows: clip the contribution that any record in a batch makes to a gradient to a range [-C,C] and then add noise drawn from a Normal distribution with standard deviation of sigma [3] to the gradient. Parameter sigma determines the level of privacy with DP parameters (epsilon,delta): higher sigma adds more noise and gives higher privacy, however, it also then decreases the utility of the model.  TimeGan consists of several models: embedder, discriminator and generator. However, only embedder and discriminator models in timegan.py are directly trained based on original data. We choose to train D using a differentially private approach based on the Gaussian mechanism mentioned above. Based on our experiments we choose to set C=1 and sigma=0.1 to get the balance between the reidentification score (for knn and binary) to be around 0.5 (i.e., close to a random guess) and feature similarity (one\_step\_ahead and half of the feature predictions).  ===============  [1] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. Foundation sand Trends in Theoretical Computer Science 2014.  [2] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, Jiayu Zhou Differentially Private Generative Adversarial Network. Arxiv 2018.  [3] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. ACM CCS 2016 |
| Seeker |  |

**Specify any loss functions used** (e.g. “No loss functions used.”)

|  |  |
| --- | --- |
| Hider | Original loss function of TimeGan. However, we did try to optimize “balanced reidentification accuracy” as opposed to original reidentification accuracy scores, in order to have a better comparison with a random guess strategy. |
| Seeker |  |

**Specify any hyperparameters and how they are optimized (or preset values)** (e.g. “The noise size, s, is set to 0.1.”)

|  |  |
| --- | --- |
| Hider | We experimented with   1. DP parameter noise magnitude, sigma, in the range of [0.1,10] 2. DP parameter norm clipping, C, in the range of [1,3] 3. number of iterations in range of 1000 to 10000 4. batch size (16, 32, 64, 128) [this parameter is important as the larger the batch the more averaging happens, so accuracy can be better at the cost of privacy due to the analysis of sampling] 5. though not a hyperparameter: TensorFlowPrivacy library vs. clipping gradients manually; we chose latter for easier integration with the submission server due to less dependencies. 6. though not a hyperparameter: we tried SGD optimizer but AdamOptimizer performed better.   We used the following metrics to determine best parameters that we use in the code:   1. balanced reidentification accuracies 2. one step prediction 3. #predictions outside of the allowed threshold   As a result we chose sigma = 0.1, C =1, #iterations = 4000 (based on loss), original batch size. |
| Seeker |  |

**Specify any pre-trained models used by your algorithm** (e.g. “None.”)

|  |  |
| --- | --- |
| Hider | None |
| Seeker |  |

**Pseudo-code for your algorithm**

e.g. **Inputs:** Dataset, D, random seed

**Hyperparameters:** s (default 0.1)

1. Determine dataset dimension: n x d x T

2. Draw N ~ N(0, s), an n x d x T dimensional Gaussian

3. Return D + N

|  |  |
| --- | --- |
| Hider | [Description of the code in timegan.py]  During training:  Extract gradients from discriminator D  C = 1 *# hyperparameter for clipping norm*  sigma = 0.1 *# hyperparameter for noise scale*  Clip gradients to C and normalize them  Draw noise from Normal distribution with parameter sigma  Add noise to clipped gradients  Apply noisy gradients  [timegan.py]    optimizer = tf.train.AdamOptimizer() *# choose the optimiser*  D\_grad\_vars = optimizer.compute\_gradients(D\_loss, var\_list = d\_vars) *# take out the gradient*  C = 1 *# hyperparameter control clipping norm*  sigma = 0.1 *# hyperparameter control noise level*  D\_grad\_vars = [(tf.divide(grad, tf.maximum(tf.constant(1.), tf.divide(tf.norm(grad), tf.constant(C, TF\_FLOAT)))) + tf.random\_normal(grad.shape, 0, sigma \* C), var) **for** grad, var **in** D\_grad\_vars] *# Clipping, normalising and noising the gradient*  D\_solver = optimizer.apply\_gradients(D\_grad\_vars) *# put the processed gradient back* |
| Seeker |  |

Finally, alongside this document **please also submit a commented version of your code**. Please include:

- Docstrings for each new class/function defined

- Inline comments for your main function/class

The goal of these comments is to tie the code to the description you have provided here. Please do not alter the actual content of your code - only add comments/docstrings.

**Submitting your documentation and commented code**

Please submit your commented code within a .zip or equivalent file type (1 file per solution), and share it with us as an attachment alongside this Word doc.

You can send these via email (to [nm736@cam.ac.uk](mailto:nm736@cam.ac.uk); [james.jordon@wolfson.ox.ac.uk](mailto:james.jordon@wolfson.ox.ac.uk); [es583@cam.ac.uk](mailto:es583@cam.ac.uk)) or DM James Jordon/Evgeny Saveliev on Slack (you can join the workspace [with this URL](https://join.slack.com/t/hideandseekpr-fbc8582/shared_invite/zt-k2h9xye8-RQNen128uXIG2TRsLa_ppA)).