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

**Team name**

|  |
| --- |
| realHIder |

**Submission filenames(s)**

|  |  |
| --- | --- |
| Hider | Seedsa |
| Seeker | barbod\_pj |

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

|  |  |
| --- | --- |
| Hider | GANs |
| Seeker | Binary Classifiers |

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

|  |  |
| --- | --- |
| Hider | Synthetic data is generated using a GAN which was trained on the training data in a self-supervised manner. |
| Seeker | A binary predictor is trained to differentiate between the synthetic data and enlarged data and used to find the training data. |

**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 | The Generator takes noise as input from a uniform distribution and generates the synthetic data. It consists of RNN and self-attention modules. The discriminator takes the synthetic data and real data as input and tries to distinguish between them. It consists of RNN layers. The loss to train this GAN is the WGAN-GP loss. We also add a self-supervised loss to train the generator. To add self-supervised information to our training, we use a transformer based on attention layers. This transformer tries to predict the next time step of the time series data, given the data up to that time step. The transformer is trained on the real data to capture the dynamics of the data across time. In the training of the generator, we pass the synthetic data through this transformer and force the output to be the input data shifted one step forward in time. In this way the synthetic data is forced to capture the dynamics of the real data. At test time, we return the synthetic data generated by the generator. |
| Seeker | It’s the baseline binary classifier that has gone through hyper-parameter tuning. |

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

|  |  |
| --- | --- |
| Hider | WGAN-GP loss + self-supervised MSE loss + two moments loss |
| Seeker | Binary corss entropy loss |

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

|  |  |
| --- | --- |
| Hider | The d\_model, dff, and num\_heads of the attention modules are set to 24, 128, and 2. The batch size, batch\_size, and number of iterations, iterations, are set to 128 and 20000. The hidden dimension, hidden\_dim, of the RNNs is set to 24. The self-supervised loss coefficient is set to 100. The gradient penalty coefficient is set to 10. |
| Seeker | The number of layers (n\_layer), batch size (batch\_size), number of epochs (epoch), the RNNs’ hidden dimension (h\_dim), and the learning rate (learning\_rate) are set to 3, 128, 400, dim (the number of features for each time step), and 0.001. |

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

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

**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 | **Inputs:** Dataset D  **Hyperparameters: hidden\_dim=24, d\_model=24, iterations=20000, batch\_size=128, num\_heads=2, dff=128, supervised loss coefficient=100, gradient penalty coefficient=10**   1. Scale the data between 0 and 1 using MinMaxScaler   2. Train the transformer using D for iterations=20000 to predict the next time step data (D shifted one step forward)  **For iterations=20000 steps:**  3. Train the generator using the WGAN-GP loss and the self-supervised loss for 2 steps  4. Train the discriminator using the WGAN-GP loss for 1 step  5. Return the final generated data |
| Seeker | The same as the baseline binary classifier. |

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)).