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

**Team name**

|  |
| --- |
| iCAIRD |

**Submission filenames(s)**

|  |  |
| --- | --- |
| Hider |  |
| Seeker | X |

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

|  |  |
| --- | --- |
| Hider |  |
| Seeker | **Submission from Instassa:** an ensemble of a couple of GANs, KNN, and a random forest  **Submission from chaitanyakaul:** binary classification deep learning model |

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

|  |  |
| --- | --- |
| Hider |  |
| Seeker | **Submission from Instassa:** an ensemble of a random forest, two types of LOGANs, and starting kit version of KNN and binary predictor.  **Submission from chaitanyakaul:** attention over LSTM layers passing into a KNN classifier |

**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 |  |
| Seeker | **Submission from Instassa:** the result is combined from a few different re-id methods: the standard KNN and binary predictor (as came in the starter kit), a random forest classifier, and two types of black-box LOGAN (i.e. the class of model that trains a GAN on hider-generated data and then uses its discriminator for re-identifying the data used for training. LOGAN can use many types of GAN for that purpose - we built ours with DCGANs and Time-GANs.  All of these methods output binary re-id vectors, which are then combined as following: replace random forest re-id outcomes with 1, where the majority vote of the rest of the methods voted 1.  **Submission from chaitanyakaul:** For this solution, we treat the task as one of binary classification. We observed that the KNN algorithm in the baseline provided the best results, but we felt that it suffered from the drawback that the time information is flattened out before feeding it to the KNN classifier. To this end, we trained an LSTM based model to first process the data and create embeddings taking into account all the time series information in the data. We then got rid of the final classification layer and used the embedding from the last LSTM layers as the input to the KNN algorithm. We experimented with different values of the embedding nodes as well as distance functions for the model and found that the final embedding of 2048 length along with the default KNN distance function works best for this setting. |

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

|  |  |
| --- | --- |
| Hider |  |
| Seeker | **Submission from Instassa:** None, beyond the default loss functions of the corresponding GANs.  **Submission from chaitanyakaul:** None, apart from the binarycrossentropy loss to train the LSTM model. |

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

|  |  |
| --- | --- |
| Hider |  |
| Seeker | **Submission from Instassa:** as it’s an ensemble I will specify each method parameters separately:  KNN and binary predictor: as in the starting kit, except for KNN threshold distance parameter was moved replacing 0.5 by 0.45.  *DC-LOGAN* (a standard 4-layer DCGAN - Relu activations + tanh for the last layer in the generator and leakyRelu activations + linear for the last layer in the discriminator): trained for 3 epochs with batch-size 300, with default learning-rate 0.0002, and beta1 0.5.  *Time-LOGAN:* based on the starting kit time-gan, same parameters, other than reducing training to 20 iterations.  Random Forest: 1000 trees.  **Submission from chaitanyakaul:** For the LSTM model, the masking layer as used in the baseline, 2 LSTM layers of dimension 200 and 2048 respectively, dropout before the output layer with rate of 0.2, final dense layer has a sigmoid output.  Adam has default parameters as the original paper.  Train test split is 90%-10%.  Epochs = 10, batch size = 128, callbacks = save best model |

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

|  |  |
| --- | --- |
| Hider |  |
| Seeker | **Submission from Instassa:** None, everything trains real time. However, the way the ensemble re\_id votes are combined is based on the performance of the model on the training data provided to our team by the competition.  **Submission from chaitanyakaul:** None.  The embeddings feeding into the KNN classifier technically come from pretraining, but that pretraining is done in real time using the data provided for the competiton, and is done on the codalab server itself. |

**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 |  |
| Seeker | **Submission from Instassa:**  **Inputs:** hider-generated data *D*, and “enlarged dataset” *T* - real data used for training hiders model with some additional real data added.  **Hyperparameters:** *h* as described in the section above (there is quite a lot of them to list them again)  **Ensemble summary:**  for i, method in {knn, binary\_predictor, time-logan, dc-logan}:  re-id\_scores[i, :] = method(D, T, h[i])  majority\_re-id = sum(re-id\_scores)  majority\_re-id = [1, where(majority\_re-id >=3), 0 otherwise]  rf\_re-id\_scores = random\_forest(D, T, trees=1000)  final\_re-id = rf\_re-id\_scores  final\_re-id[where(majority\_re-id == 1)] = 1  **Output:** final\_re-id  **Submission from chaitanyakaul:**  **Inputs:** hider-generated data *D*, and “enlarged dataset” *T* - real data used for training hiders model with some additional real data added.  **Hyperparameters:** *h* as described in the section above  **Ensemble summary:**   1. Divide the data into a 90-10 split to train and test 2. Train LSTM model for the binary classification task 3. Save the best model by monitoring the validation loss 4. Pop() final classification layer from the model leaving a 2048 length embedding 5. Re-id = Feed 2048 length embedding into KNN classifier and compute re-id as shown in baseline   **Output:** re-id |

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)).