1. **TEAM INFORMATION**

Name: Olivér Balogh, BSc

Email: [balogh.oliver@med.semmelweis-univ.hu](mailto:balogh.oliver@med.semmelweis-univ.hu)

Affiliation:

* Cardiometabolic and MTA-SE System Pharmacology Research Group, Department of Pharmacology and Pharmacotherapy, Semmelweis University, Budapest, Hungary

Contribution: Write the code; Analyze the results

Name: Bettina Benczik, MSc

Email: [benczik.bettina@med.semmelweis-univ.hu](mailto:benczik.bettina@med.semmelweis-univ.hu)

Affiliation:

* Cardiometabolic and MTA-SE System Pharmacology Research Group, Department of Pharmacology and Pharmacotherapy, Semmelweis University, Budapest, Hungary

Contribution: Write the code; Analyze the results

Name: Mátyás Pétervári, PharmD

Email: [petervari.matyas@med.semmelweis-univ.hu](mailto:petervari.matyas@med.semmelweis-univ.hu)

Affiliation:

* Cardiometabolic and MTA-SE System Pharmacology Research Group, Department of Pharmacology and Pharmacotherapy, Semmelweis University, Budapest, Hungary
* Sanovigado Kft., Budapest, Hungary

Contribution: Write the code; Analyze the results

Name: Bence Ágg, MD, PhD

Email: [agg.bence@med.semmelweis-univ.hu](mailto:agg.bence@med.semmelweis-univ.hu)

Affiliation:

* Cardiometabolic and MTA-SE System Pharmacology Research Group, Department of Pharmacology and Pharmacotherapy, Semmelweis University, Budapest, Hungary
* Heart and Vascular Center, Semmelweis University, Budapest, Hungary
* Pharmahungary Group, Szeged, Hungary

Contribution: Write the code; Analyze the results; Project coordination

Name: Péter Ferdinandy, MD, PhD, MBA

Email: [peter.ferdinandy@pharmahungary.com](mailto:peter.ferdinandy@pharmahungary.com)

Affiliation:

* Cardiometabolic and MTA-SE System Pharmacology Research Group, Department of Pharmacology and Pharmacotherapy, Semmelweis University, Budapest, Hungary
* Pharmahungary Group, Szeged, Hungary

Contribution: Scientific advisor, Institutional background and financing

1. **CODE DESCRIPTION/USAGE**

All source files are also available at <https://github.com/semmelweis-pharmacology/ppi_pred>.

Our source code includes 9 files written in Python (version 3.6.9), shell script for Bash and in JavaScript Object Notation (JSON):

1. preprocess.py

* **Description**: This code is used to perform the splitting of the original network and generation of induced subgraphs from the network provided in the form of an edge list CSV file as the first command line argument.
* **Usage**: Not intended to be called directly. It is run by batch\_preprocess.sh.
* **Hyperparameters**: N/A

1. batch\_preprocess.sh

* **Description**: This shell script performs all preprocessing steps necessary for running the applied generative adversarial network (GAN) training by calling the above preprocess.py script and the node2vec tool for all networks provided in the network\_list array. It also generates JSON files for the parametrization of gan\_90.py with the appropriate input paths of the edge list, embedding and induced subgraph files.
* **Usage**: batch\_preprocess.sh <input\_dir> <output\_dir> <node2vec\_path>
  + <input\_dir>: path to the directory where the adjacency list CSV files for the networks specified in the network\_list array are located.
  + <output\_dir>: path to the directory to which all preprocessing, training, prediction and post-processing outputs should be written.
  + <node2vec\_path>: path to the binary of the Stanford Network Analysis Platform (<https://github.com/snap-stanford/snap>) implementation of the node2vec algorithm.
* **Hyperparameters**: N/A

1. template\_input.json

* **Description**: This file is used by batch\_preprocess.sh as a template to generate JSON files for the parametrization of the gan\_90.py.

1. gan\_90.py

* **Description**: This code performs GAN training and prediction on the specified 90 percent downgraded network.
* **Usage**: Not intended to be called directly. It is called by batch\_gan\_90.sh, which provides the necessary parameters in the form of the JSON files generated by batch\_preprocess.sh.
* **Hyperparameters**: Optimal values are provided in the source code.

1. batch\_gan\_90.sh

* **Description**: This shell script runs gan\_90.py for all the JSON files generated by batch\_preprocess.sh.
* **Usage**: batch\_gan\_90.sh <output\_dir>
  + <output\_dir>: the same path that is given as the second command line argument (<output\_dir>) to batch\_preprocess.sh.
* **Hyperparameters**: N/A

1. postprocess\_90.py

* **Description**: This script performs the post-processing and statistical analysis of the predictions given by batch\_gan\_90.sh.
* **Usage**: Not intended to be called directly. It is called by batch\_postprocess\_90.sh.
* **Hyperparameters**: None.

1. batch\_postprocess\_90.sh

* **Description**: This shell scripts runs the postprocess\_90.py to perform the post-processing and statistical analysis of the predictions given by batch\_gan\_90.sh.
* **Usage**: batch\_postprocess\_90.sh <output\_dir>
  + <output\_dir>: the same path that is given as the second command line argument (<output\_dir>) to batch\_preprocess.sh.
* **Hyperparameters**: None.

1. gan\_100.py

* **Description**: This code performs GAN training and prediction on the original network.
* **Usage**: gan\_100.py <parameter\_json>
  + <parameter\_json>: Path to the JSON file with the following parameters:
    - oSourceList:
      * oN90\_Ad: The path to the edge list of a 90% reduced version of the original network.
      * oN90\_Em: The path to the embedding file obtained by the node2vec algorithm for the above 90% reduced network.
      * oN90\_Mod: The path to the file containing the induced subgraphs for the above 90% reduced network.
      * oN100\_Ad: The path to the edge list of the original network.
      * oN90\_Em: The path to the embedding file obtained by the node2vec algorithm for the original network.
      * oN90\_Mod: The path to the file containing the induced subgraphs for the original network.
    - oOutputPath: The path to the directory in which the predictions will be written.
    - oOutputName: The name prefix of the output files.
* **Hyperparameters**: Optimal values are provided in the source code.

1. postprocess\_100.py

* **Description**: This script performs the post-processing of predictions given by batch\_gan\_90.sh writing out the 500 predicted edges. Self-loops are ignored.
* **Usage**: postprocess\_100.py <output\_dir> <network\_name> <gan\_filename>
  + <output\_dir>: The same path that is given as the second command line argument (<output\_dir>) to batch\_preprocess.sh.
  + <network\_name>: Name of the full network for which prediction was made.
  + <gan\_filename>: Full path for the output of gan\_100.py.
* **Hyperparameters**: None.

1. **COMPUTING ENVIRONMENT**

All the programs were run on an ASUS RS720-E9-RS8-G server (CPU: 2x Intel Xeon Gold 5218, 16-core, 2.30GHz; RAM: 512GB; Tesla V100 PCIe 32GB) with Ubuntu Server 20.04.1 LTS operating system.

1. **EXTERNAL PACKAGES/LIBRARIES**

The following Python dependencies are required:

- python==3.6.9

- networkx==2.5

- numpy==1.18.5

- pandas==1.1.0

- scikit-learn==0.23.2

- tensorflow==2.3.0

- Stanford Network Analysis Platform (SNAP) - Release 6.0, Dec 4, 2020

1. **ADDITIONAL DATASET USED IN THE METHOD**

N/A

1. **METHOD DESCRIPTION**

Our protein-protein interaction prediction approach is based on the training of a generative adversarial network using Wasserstein distance-based loss improved with gradient penalty performing image-to-image translation conditioned on embedding information of the network topology [1–3]. This model consists of a generator and a discriminator trained by an adversarial learning process [4].

The input for the generator is the embedding of the 90% splitted versions of the original network generated by the node2vec algorithm [5] along with a set of adjacency matrices of induced subgraphs together covering most of the nodes and edges of the original network. These induced subgraphs are assessed by a modified version of the classical breadth first search algorithm [6] which is started from each node of the network and iterated over layers containing *ni* neighbor of the nodes of the previous layer until the traversed *k* layers altogether contain less nodes than a predefined limit *l* (in our case *l = 36*):

At the last step of this modified breadth-first search *l-sk* randomly selected nodes are added to the induced subgraph. Eventually 4 nodes are randomly removed from the induced subgraph to arrive at an induced subgraph of 32 nodes.

The input for the discriminator is the same as for the generator with the addition of either confidence matrices produced by the generator or the adjacency matrices with the full connectivity of the original network.

In case of multiple edge confidence values for the same node pair, the maximum of these values is chosen.

1. **TIME COMPLEXITY ANALYSIS (Optional)**

N/A

**Reference**

1. Arjovsky M, Chintala S, Bottou L. (2017) Wasserstein GAN. arXiv, .

2. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A. (2017) Improved Training of Wasserstein GANs. Adv Neural Inf Process Syst, 2017-December: 5768–5778.

3. Isola P, Zhu J-Y, Zhou T, Efros AA. (2016) Image-to-Image Translation with Conditional Adversarial Networks. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017, 2017-January: 5967–5976.

4. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. (2020) Generative adversarial networks. Commun ACM, 63: 139–144. doi:10.1145/3422622.

5. Grover A, Leskovec J. (2016) node2vec: Scalable Feature Learning for Networks. Proc ACM SIGKDD Int Conf Knowl Discov Data Min, 13-17-August-2016: 855–864.

6. Lee CY. (1961) An Algorithm for Path Connections and Its Applications. IRE Trans Electron Comput, EC-10: 346–365. doi:10.1109/TEC.1961.5219222.