**IMPORTANT:**

The majority of the code used in this project was from the Github repository: <https://github.com/sangwoomo/FreezeD>.

This is the authors implementation of their paper, “Freeze the Discriminator: a Simple Baseline for Fine-Tuning GANs”, written by Sangwoo Mo, Minsu Cho and Jinwoo Shin. The original readme file is README.md and a copy of the license is found in LICENSE.md, which permits use of their software. It is important to follow these instructions and download the specified pre-trained GAN models in order to run the code. The inceptionv3 model is included with this software submission and is not necessary to redownload.

In this document, the changes made to the original software and new files are indicated and explained, followed by instructions on how to repeat the the experiments performed for the dissertation “Investigating the use of generative adversarial networks to augment small datasets”.

The CUDA version used was cuda-10.0, with CUDNN version cudnn-v7.6.5.32-forcuda10.0. It is recommended that these are used.

The file PYTHON\_ENV.txt shows all python libraries installed while running the software. While some may no longer be necessary, refer to the versions used in this file if having issues.

**MAIN MODIFICATIONS TO ORIGINAL SOFTWARE:**

Note that the modifications were made specifically for use with the GestATional dataset. If not using this dataset, it is recommended that the original FreezeD repository be used instead.

1. The datasets/*make\_image\_list*.*py* file is run to generate a text file containing the image paths and labels of all files to be used in training the GANs. This was modified to:
   1. Convert the folder names (classes) into integer labels.
   2. Take an extra parameter to specify the degree of augmentation to use (i.e. the number of rotations to keep in the dataset), and prepare the data as such.
   3. An additional file, *make\_image\_list\_cv\_split*.*py*, was created to use when preparing the image lists for training the GANs used in the classification experiments. This partitions the dataset using stratified kfold sampling, and produces three image lists of training data for the dataset specified – one for each fold.
2. The *evaluation.py* file was extended to:
   1. The original *get\_mean\_cov* function gets the mean and covariance of the inceptionv3 activations of the input dataset. This was changed to *get\_mean\_cov\_acts*, to return the activations too, as it is necessary for KID calculation.
   2. *calc\_FID*  was changed to allow FID to be averaged over a number of evaluations, and allows it to be run when not used as a chainer extension.
   3. *calc\_KID* was added as a chainer extension, and operates similarly to *calc\_FID* but instead computes KID. Note that the actual computation of KID was provided from code in the repository (*metric* and *kid\_score* files) but had to be adapted for use with chainer.
3. The evaluations*/calc\_ref\_stats.py* file is used to precalculate FID/KID statistics on the training data used. It imports the *get\_mean\_cov\_acts* function previously mentioned to do this.
   1. Previously only FID statistics were saved. This was updated to save KID statistics for use during training.
   2. Also takes augmentation strategy as a parameter for specifying the image list to load and directory to save in.
4. The *finetune\_v2.py* file is used to train the GANs. The chainer extensions used were modified:
   1. A chainer extension was added to calculate the KID every evaluation step.
   2. Extra report keys added to accommodate extra results from the FID/KID calculated (mean/std/min).
   3. The best model is saved according to the best KID instead of FID, and a snapshot is saved every evaluation step.
5. A script, *evaluate\_trained\_gans.py* was added. For an input directory of trained GAN outputs (after *finetune\_v2.py* is run), calculates the KID/FID class-wise and overall for each experiment.
   1. Takes the results directory as input as location to save .csv results.
   2. The trained model directories must be placed in the *models\_to\_eval* folder, and the associated config file must be named *config.yml.*
6. A *train\_resnet.py* file was added. This trains a resnet-18 model for each fold using stratified kfold sampling for the input dataset, and the results are aggregated to calculate the confusion matrix and accuracy/f-1score.
   1. The *gen\_image\_with\_condition­* function from the FreezeD repo was adapted to generate a single image only and allow the use of the truncation trick. The *sample\_continious­* function in *source/miscs/random\_samples.py* was also modified to this end.
   2. Uses the associated generator to generate the required images during runtime, rather than loading from a directory of saved images.
7. The *resnet\_experiments.py* file is an extension of *train\_resnet.py.* 
   1. The main change is to specify the experiments to run in a python dictionary, rather than via command line. This allows the results to be aggregated together, rather than individually.
   2. Each experiment is also run 5 times to calculate an average of the performance with standard deviation.

**HOW TO RUN EXPERIMENTS:**

To repeat the experiments, all scripts are prepared in the *shell\_scripts* folder. If not using slurm and instead running on your own machine, only run the python commands. Note that the config files are included for all these experiments and no changes need to be made.

1. The GestATional dataset must be placed in the *datasets* folder, with the folder names *gestational\_face, gestational\_ear, gestational\_foot.* The subdirectories must be the class names, *extremely, very, moderate, term, late.*
2. To repeat the GAN experiments on the entire dataset, the *freezeD\_full/none/partial.sh* scripts can be run.
   1. The image lists are prepared, followed by precalculating the KID/FID statistics, then the FreezeD/Finetuning experiments.
3. Move the output of the previous experiments to the *models\_to\_*eval folder and run evaluate\_shell.sh to calculate average FID/KID of the best generator.
4. For training the GANs used in classification experiments, first run *cv\_split\_GAN*.sh to prepare the image lists needed.
5. Next, run *train\_cv\_split\_GANs.sh* to calculate KID/FID statistics and train all models (face/ear/foot) for each training data split (0/1/2).
6. To run the individual classification experiments (the *train\_resnet.py* script), run *resnet\_face/ear/foot/\_experiments.sh*.
7. To run the classification experiments together with averaged aggregated output (*resnet\_experiments.py)*, then run *resnet\_scratch.sh* or *resnet\_pretrained.sh*.
8. To run classification experiments based on number of generated images used in augmentation, run *resnet\_ims.sh,* and *resnet\_ims\_trunc.sh* to run with the truncation trick.