Testing a pre-trained SinGAN model:

This document serves as a guide to running a pre-trained SinGAN model to use style-transfer methods for data augmentation purposes. The model attempts to apply style transfer from the style image (training spectrogram) to the content image (test spectrogram).

Since SinGAN uses a multi-scale architecture, it is possible to generate output at different trained levels. To successfully run the model and to test out your own files, follow the instructions carefully:

Setting up the environment:

To begin, install the python requirements mentioned in the requirements.txt file, you may do so by running the following command:

pip install -r requirements.txt *(Note that SinGAN uses an older version of torch)*

from the SinGAN directory set as the working directory.

Prerequisites:

1. Python 3
2. CPU or Nvidia GPU + CUDA CuDNN

Setting up the files for testing:

The model weights and parameters are set already under the "**TrainedModels**" directory, and all that is left to do is set up your files (files you want to add noise to), to do so follow the steps given below:

1. Create a new folder "**content**" in the "**datasets**" folder.
2. Inside "**content**", place the files you want the noise added to.
3. Now, with everything set up, run the following command to begin the generation process: (Note: the training file used is placed in the root directory)

   python test.py --name fe_03_1007-02235-A-056935-057917-A.wav

4. The results generated would be placed in a new folder inside the "Output" folder.

The 'name' argument above is the name of the training file used to train the model. SinGAN uses only 1 image/spectrogram for training.

CAUTION:

Make sure you use **.wav** files and **mono(single)** channel.

Explanation:

We are using the paint2image task of the model to apply the required style transfer. As previously mentioned, SinGAN uses a multi-scale architecture, and we can utilize it to generate samples at different levels. This allows us to observe different level of detail and clarity in sound of the outputs produced.

The scale-wise generation is abstracted for the sake of the user, and by running the python script *test.py* as mentioned above, the code will automatically generate data for each scale and place it accordingly in the output directories for observation.