Pune Vidyarthi Griha College of Engineering

and Technology Pune

DIGITAL SIGNAL PROCESSING

Application Assignment

Speech De-noising Using Deep Learning

Hybrid Group No: 5

Group Member 1: Mohammad Shahbaz Alam

Group Member 2: Devidas Ingale

Group Member 3: Avdhoot Patil

Guided By: Dr Vikram Gadre Sir

June 20, 2021

1. Title
2. Software/Platform used
3. Steps to install and run the code
4. Stepwise windows screenshots
5. Block Diagram
6. Algorithm used
7. Observation
8. Conclusion
9. Code Links

1) Title

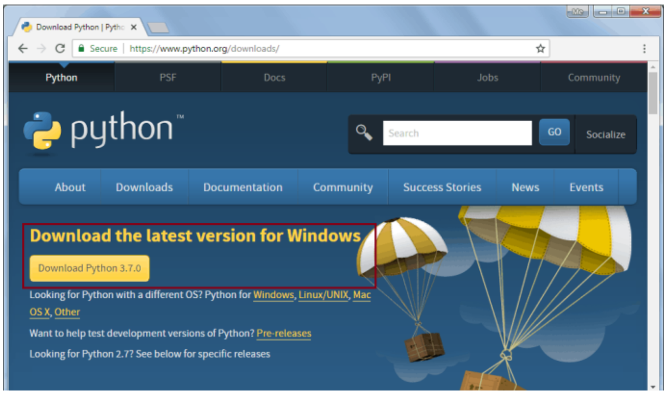
Speech Denoising Using Deep learning

2) Software/Platform used

Coding: Python in Jupyter Notebook IDE
Neural Network made by: TensorFlow 2
Libraries: NumPy, Librosa, Soundfile etc.

3) Steps to install & run the code

1) Download python in given link: https://www.python.org/downloads/



Download the latest version for Windows

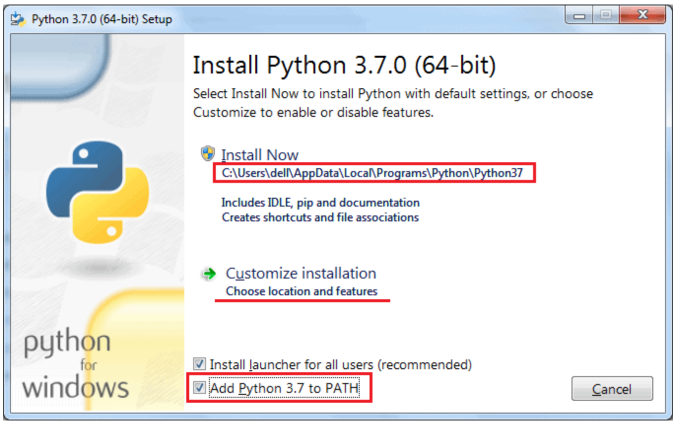
Download Python 3.7.0

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases

Looking for Python 2.7? See below for specific releases

- Install python



Install Python 3.7.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

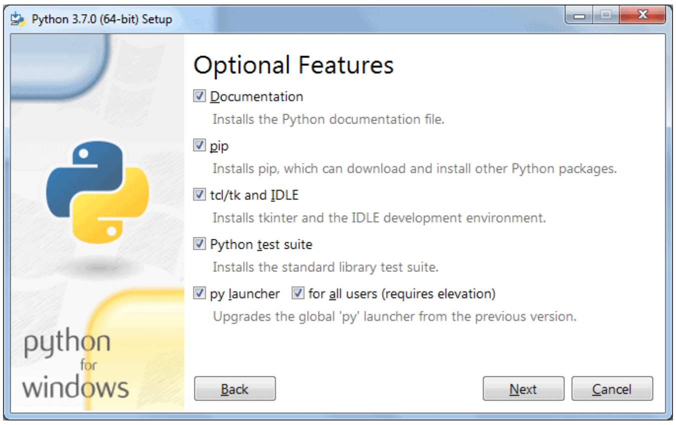
Install Now
C:\Users\dell\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

- Click all optional features including pip



Optional Features

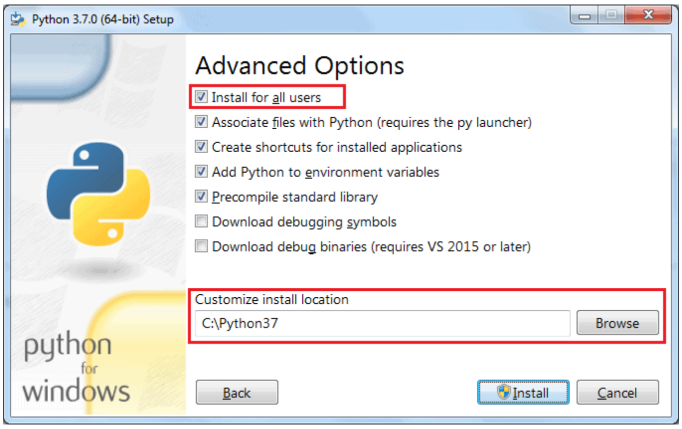
☑ Documentation
Installs the Python documentation file.

☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

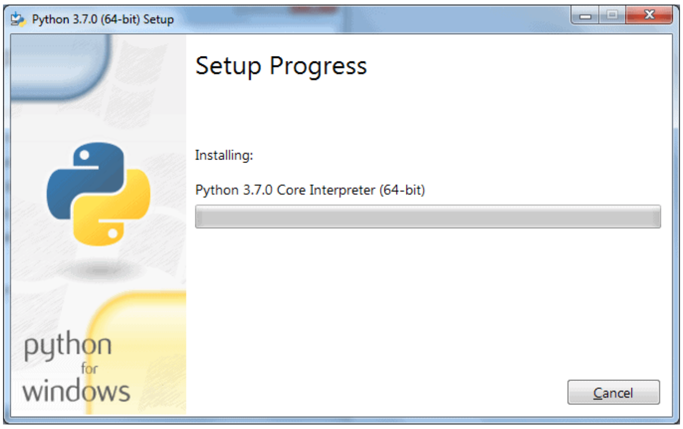
☑ py launcher ☑ for all users (requires elevation)
Upgrades the global 'py' launcher from the previous version.



Advanced Options

☑ Install for all users
☑ Associate files with Python (requires the py launcher)
☑ Create shortcuts for installed applications
☑ Add Python to environment variables
☑ Precompile standard library
□ Download debugging symbols
□ Download debug binaries (requires VS 2015 or later)

Customize install location
C:\Python37



Setup Progress

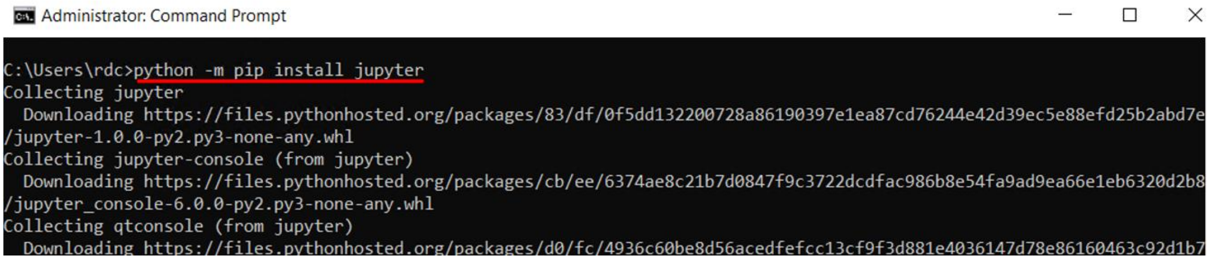
Installing:

Python 3.7.0 Core Interpreter (64-bit)

- Python is installed

2) Now installing Jupyter notebook using pip in command prompt

```
python -m pip install jupyter
```

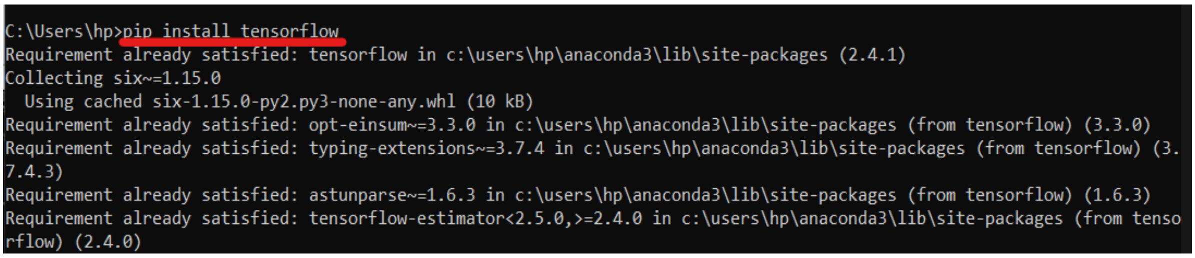


Administrator: Command Prompt

```
C:\Users\rdc>python -m pip install jupyter
Collecting jupyter
  Downloading https://files.pythonhosted.org/packages/83/df/0f5dd132200728a86190397e1ea87cd76244e42d39ec5e88efd25b2abd7e
/jupyter-1.0.0-py2.py3-none-any.whl
Collecting jupyter-console (from jupyter)
  Downloading https://files.pythonhosted.org/packages/cb/ee/6374ae8c21b7d0847f9c3722dcdfac986b8e54fa9ad9ea66e1eb6320d2b8
/jupyter_console-6.0.0-py2.py3-none-any.whl
Collecting qtconsole (from jupyter)
  Downloading https://files.pythonhosted.org/packages/d0/fc/4936c60be8d56acedfefcc13cf9f3d881e4036147d78e86160463c92d1b7
```

3) Install TensorFlow 2 using pip in command prompt:

```
pip install tensorflow
```



```
C:\Users\hp>pip install tensorflow
Requirement already satisfied: tensorflow in c:\users\hp\anaconda3\lib\site-packages (2.4.1)
Collecting six~=1.15.0
  Using cached six-1.15.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: opt-einsum~=3.3.0 in c:\users\hp\anaconda3\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: typing-extensions~=3.7.4 in c:\users\hp\anaconda3\lib\site-packages (from tensorflow) (3.
7.4.3)
Requirement already satisfied: astunparse~=1.6.3 in c:\users\hp\anaconda3\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: tensorflow-estimator<2.5.0,>=2.4.0 in c:\users\hp\anaconda3\lib\site-packages (from tenso
rflow) (2.4.0)
```

4) Installing other libraries like librosa, NumPy, soundfile etc using pip in command prompt:

```
pip install librosa
```

```
pip install numpy
```

```
pip install soundfile
```

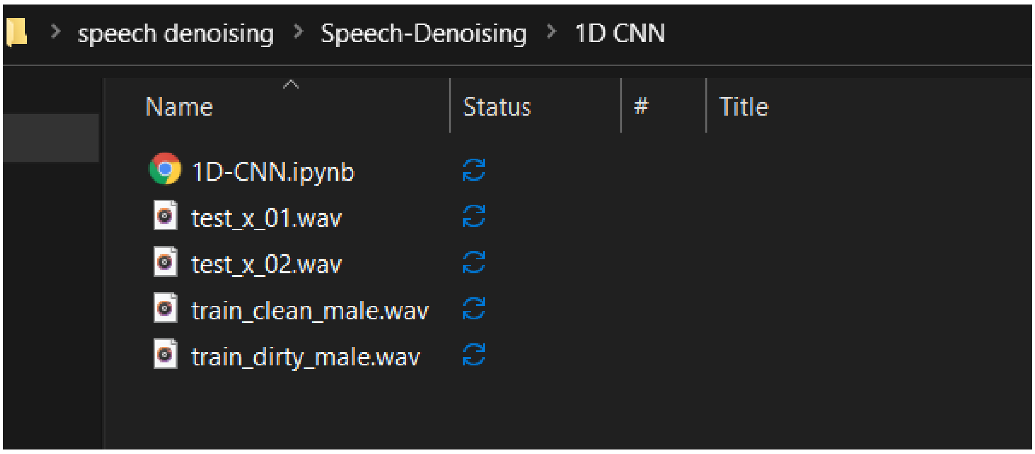
```
pip install matplotlib
```

```
pip install scipy
```

```
pip install wavefile
```

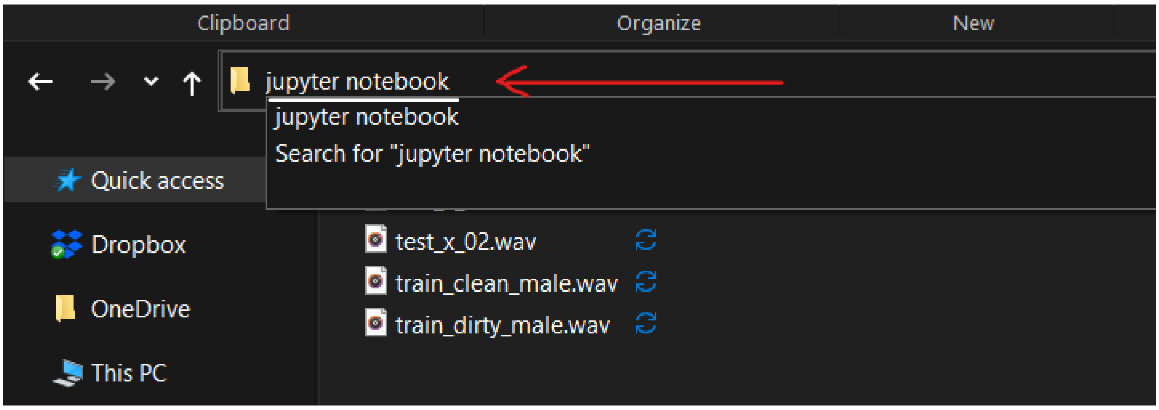
```
pip install ipython
```

Open File Containing 1D CNN jupyter with test & training files

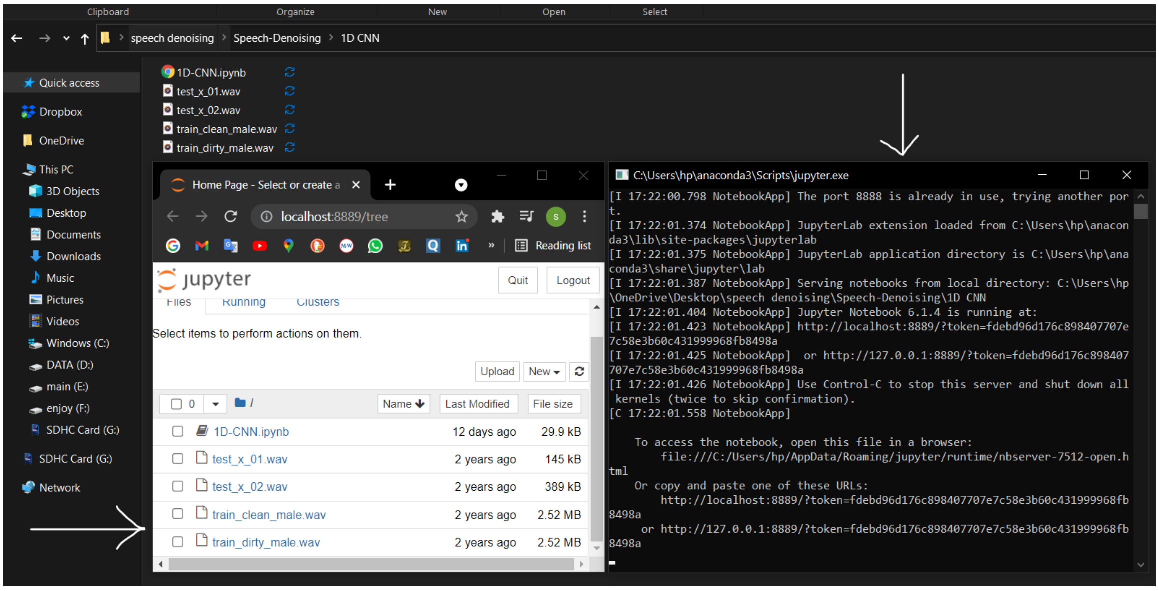


Name	Status	#	Title
1D-CNN.ipynb			
test_x_01.wav			
test_x_02.wav			
train_clean_male.wav			
train_dirty_male.wav			

In location bar, write jupyter notebook, then press “enter”. It opens terminal



This will open terminal and jupyter notebook



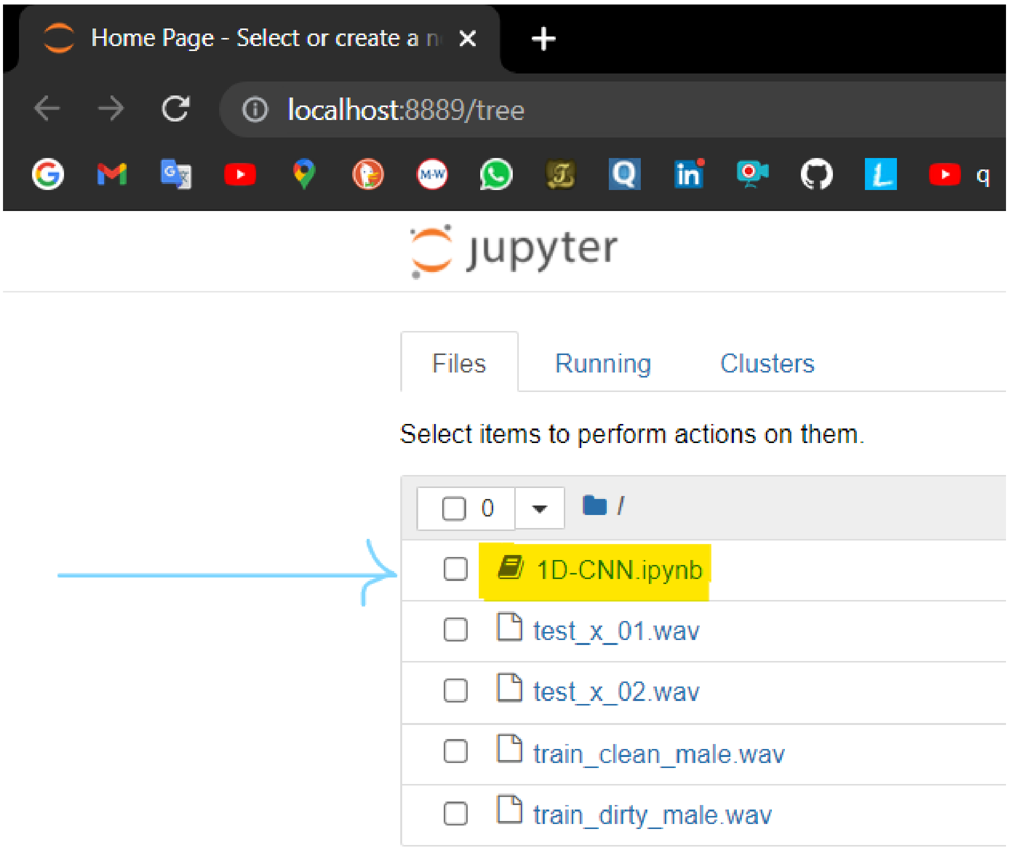
Select items to perform actions on them.

Name	Last Modified	File size
1D-CNN.ipynb	12 days ago	29.9 kB
test_x_01.wav	2 years ago	145 kB
test_x_02.wav	2 years ago	389 kB
train_clean_male.wav	2 years ago	2.52 MB
train_dirty_male.wav	2 years ago	2.52 MB

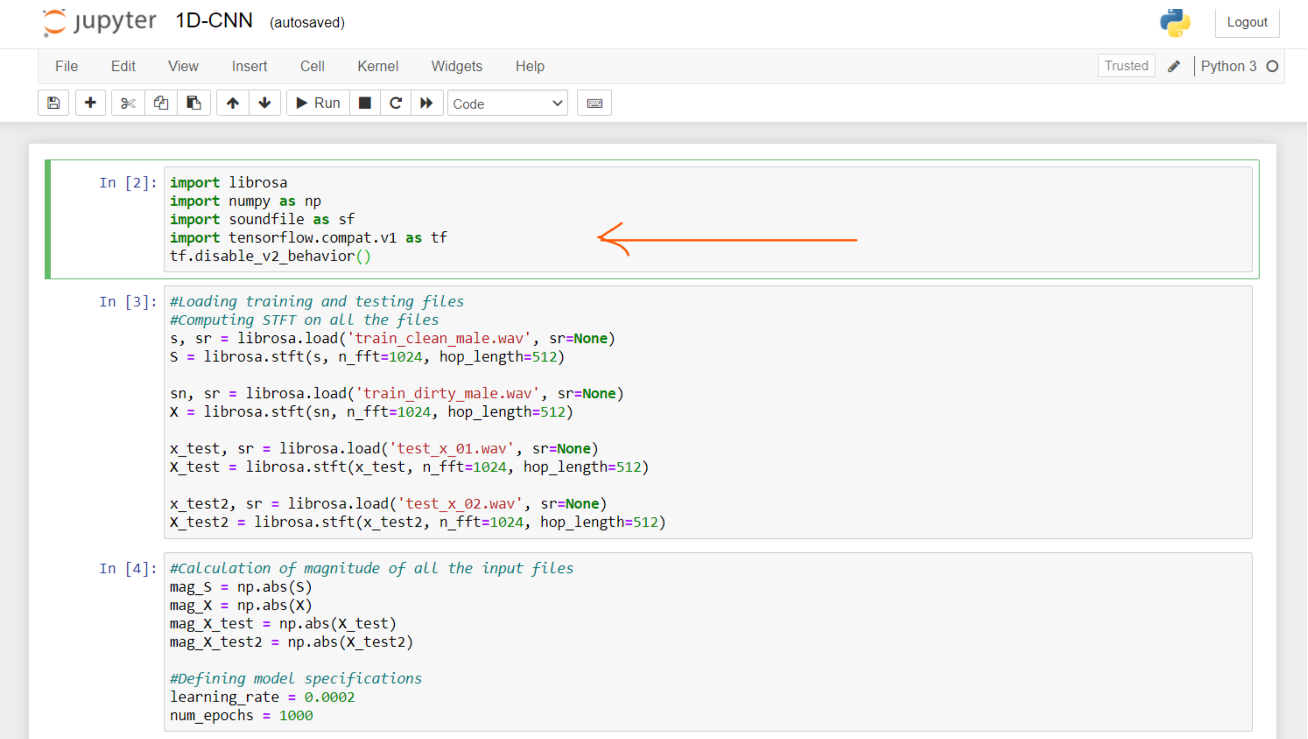
C:\Users\hp\anaconda3\Scripts\jupyter.exe

```
[I 17:22:00.798 NotebookApp] The port 8888 is already in use, trying another por
t.
[I 17:22:01.374 NotebookApp] JupyterLab extension loaded from C:\Users\hp\anacon
da3\lib\site-packages\jupyterlab
[I 17:22:01.375 NotebookApp] JupyterLab application directory is C:\Users\hp\ana
conda3\share\jupyter\lab
[I 17:22:01.387 NotebookApp] Serving notebooks from local directory: C:\Users\hp
\OneDrive\Desktop\speech denoising\Speech-Denoising\1D CNN
[I 17:22:01.404 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 17:22:01.423 NotebookApp] http://localhost:8889/?token=fdebd96d176c898407707e
7c58e3b60c431999968fb8498a
[I 17:22:01.425 NotebookApp]  or http://127.0.0.1:8889/?token=fdebd96d176c898407
707e7c58e3b60c431999968fb8498a
[I 17:22:01.426 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
[C 17:22:01.558 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/hp/AppData/Roaming/jupyter/runtime/nbserver-7512-open.h
tml
    Or copy and paste one of these URLs:
        http://localhost:8889/?token=fdebd96d176c898407707e7c58e3b60c431999968fb
8498a
     or http://127.0.0.1:8889/?token=fdebd96d176c898407707e7c58e3b60c431999968fb
8498a
```

Open 1D-CNN.pynb file



Click in this block and press "shift + Enter" to run this block and continue till training of file

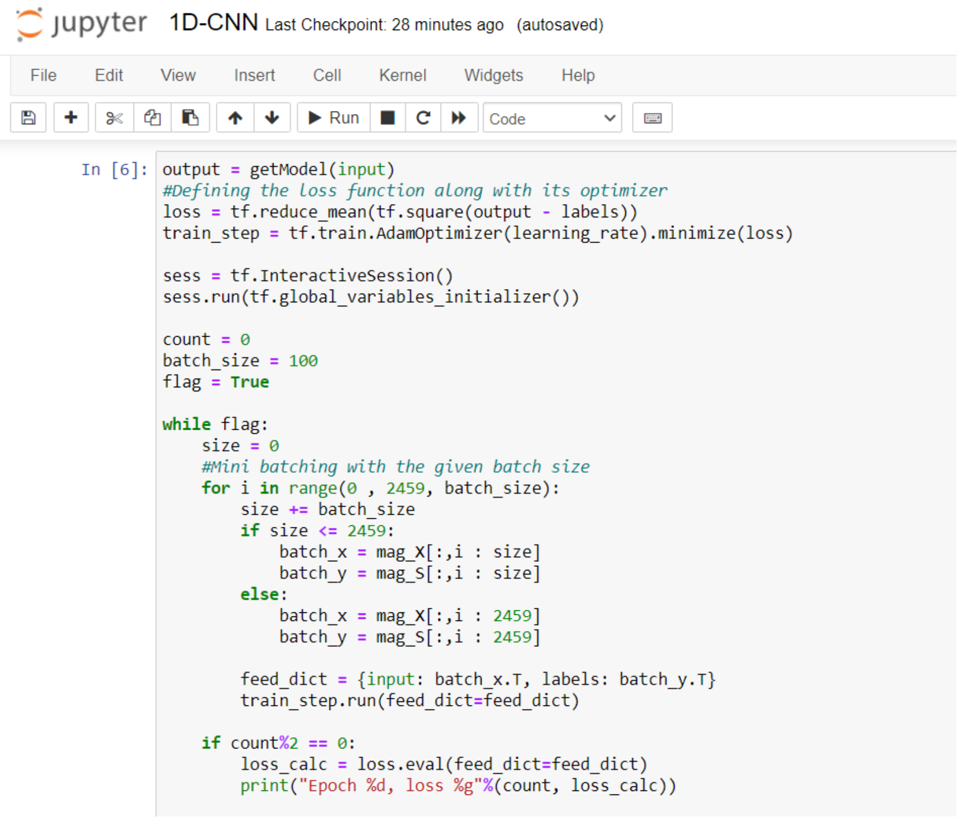


```
In [2]: import librosa
        import numpy as np
        import soundfile as sf
        import tensorflow.compat.v1 as tf
        tf.disable_v2_behavior()
```

```
In [3]: #Loading training and testing files
        #Computing STFT on all the files
        s, sr = librosa.load('train_clean_male.wav', sr=None)
        S = librosa.stft(s, n_fft=1024, hop_length=512)

        sn, sr = librosa.load('train_dirty_male.wav', sr=None)
        X = librosa.stft(sn, n_fft=1024, hop_length=512)

        x_test, sr = librosa.load('test_x_01.wav', sr=None)
        X_test = librosa.stft(x_test, n_fft=1024, hop_length=512)

        x_test2, sr = librosa.load('test_x_02.wav', sr=None)
        X_test2 = librosa.stft(x_test2, n_fft=1024, hop_length=512)
```

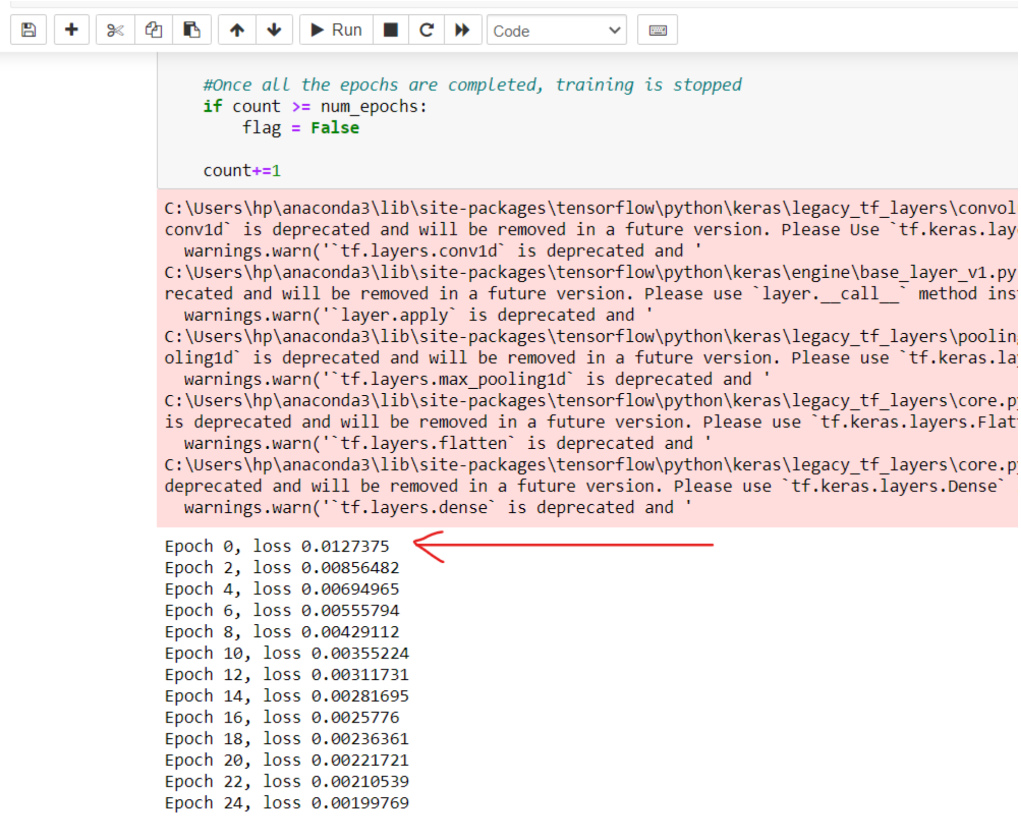
```
In [4]: #Calculation of magnitude of all the input files
        mag_S = np.abs(S)
        mag_X = np.abs(X)
        mag_X_test = np.abs(X_test)
        mag_X_test2 = np.abs(X_test2)

        #Defining model specifications
        learning_rate = 0.0002
        num_epochs = 1000
```

Run by pressing “shift + Enter “till this block starts training of audio files



```
In [6]: output = getModel(input)
        #Defining the loss function along with its optimizer
        loss = tf.reduce_mean(tf.square(output - labels))
        train_step = tf.train.AdamOptimizer(learning_rate).minimize(loss)

        sess = tf.InteractiveSession()
        sess.run(tf.global_variables_initializer())

        count = 0
        batch_size = 100
        flag = True

        while flag:
            size = 0
            #Mini batching with the given batch size
            for i in range(0 , 2459, batch_size):
                size += batch_size
                if size <= 2459:
                    batch_x = mag_X[:,i : size]
                    batch_y = mag_S[:,i : size]
                else:
                    batch_x = mag_X[:,i : 2459]
                    batch_y = mag_S[:,i : 2459]

                feed_dict = {input: batch_x.T, labels: batch_y.T}
                train_step.run(feed_dict=feed_dict)

            if count%2 == 0:
                loss_calc = loss.eval(feed_dict=feed_dict)
                print("Epoch %d, loss %g"%(count, loss_calc))
```

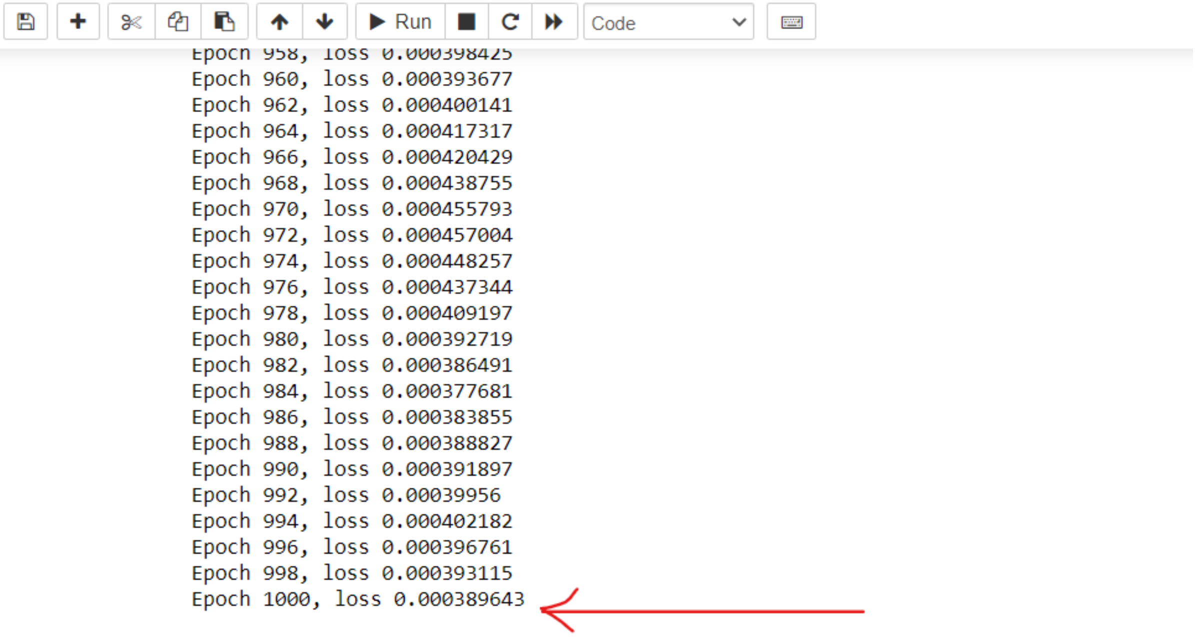
Now, training starts



```
    #Once all the epochs are completed, training is stopped
    if count >= num_epochs:
        flag = False

    count+=1
```

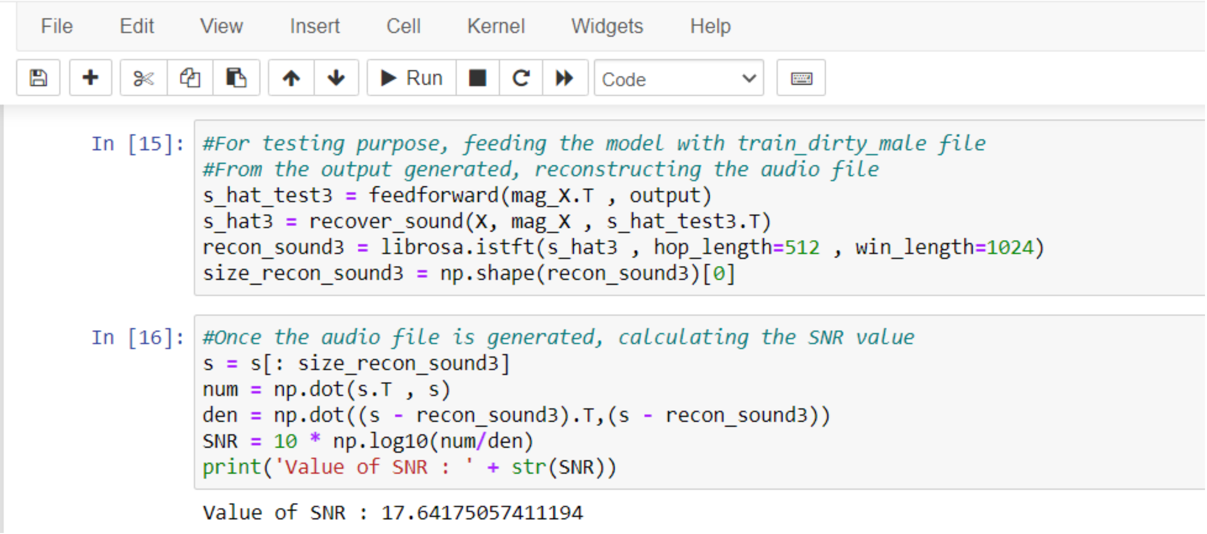
```
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\convol
conv1d` is deprecated and will be removed in a future version. Please Use `tf.keras.lay
  warnings.warn('`tf.layers.conv1d` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\engine\base_layer_v1.py
recated and will be removed in a future version. Please use `layer.__call__` method ins
  warnings.warn('`layer.apply` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\poolin
oling1d` is deprecated and will be removed in a future version. Please use `tf.keras.la
  warnings.warn('`tf.layers.max_pooling1d` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\core.p
is deprecated and will be removed in a future version. Please use `tf.keras.layers.Flat
  warnings.warn('`tf.layers.flatten` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\core.p
deprecated and will be removed in a future version. Please use `tf.keras.layers.Dense`
  warnings.warn('`tf.layers.dense` is deprecated and '

Epoch 0, loss 0.0127375
Epoch 2, loss 0.00856482
Epoch 4, loss 0.00694965
Epoch 6, loss 0.00555794
Epoch 8, loss 0.00429112
Epoch 10, loss 0.00355224
Epoch 12, loss 0.00311731
Epoch 14, loss 0.00281695
Epoch 16, loss 0.0025776
Epoch 18, loss 0.00236361
Epoch 20, loss 0.00221721
Epoch 22, loss 0.00210539
Epoch 24, loss 0.00199769
```

Here, training completes. So again, start running next blocks



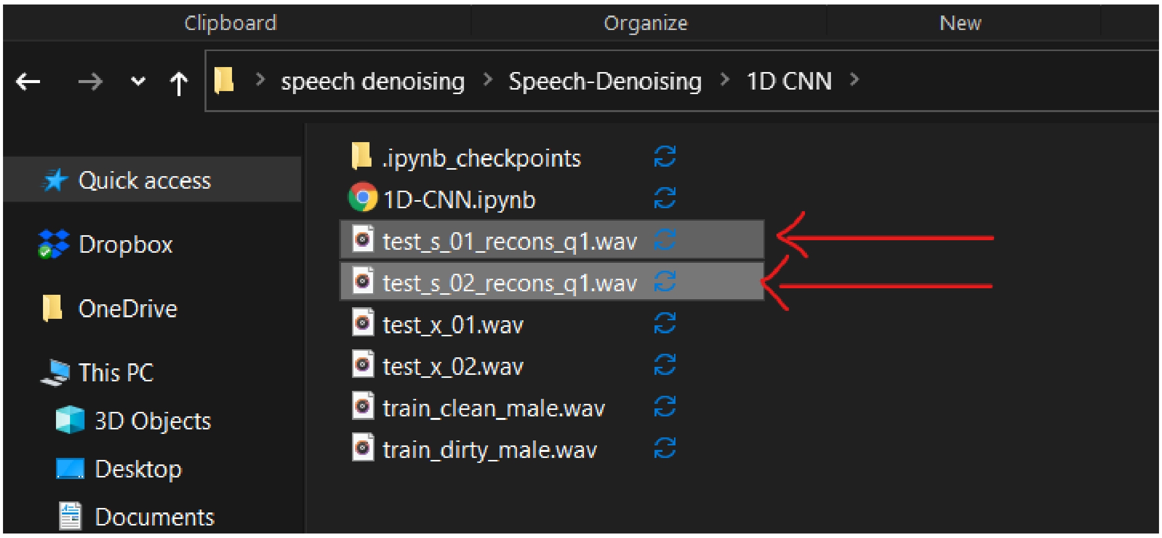
```
Epoch 958, loss 0.000398425
Epoch 960, loss 0.000393677
Epoch 962, loss 0.000400141
Epoch 964, loss 0.000417317
Epoch 966, loss 0.000420429
Epoch 968, loss 0.000438755
Epoch 970, loss 0.000455793
Epoch 972, loss 0.000457004
Epoch 974, loss 0.000448257
Epoch 976, loss 0.000437344
Epoch 978, loss 0.000409197
Epoch 980, loss 0.000392719
Epoch 982, loss 0.000386491
Epoch 984, loss 0.000377681
Epoch 986, loss 0.000383855
Epoch 988, loss 0.000388827
Epoch 990, loss 0.000391897
Epoch 992, loss 0.00039956
Epoch 994, loss 0.000402182
Epoch 996, loss 0.000396761
Epoch 998, loss 0.000393115
Epoch 1000, loss 0.000389643
```

Finally, we get SNR of 17.6417 by 1D CNN



```
In [15]: #For testing purpose, feeding the model with train_dirty_male file
         #From the output generated, reconstructing the audio file
         s_hat_test3 = feedforward(mag_X.T , output)
         s_hat3 = recover_sound(X, mag_X , s_hat_test3.T)
         recon_sound3 = librosa.istft(s_hat3 , hop_length=512 , win_length=1024)
         size_recon_sound3 = np.shape(recon_sound3)[0]

In [16]: #Once the audio file is generated, calculating the SNR value
         s = s[: size_recon_sound3]
         num = np.dot(s.T , s)
         den = np.dot((s - recon_sound3).T,(s - recon_sound3))
         SNR = 10 * np.log10(num/den)
         print('Value of SNR : ' + str(SNR))

         Value of SNR : 17.64175057411194
```

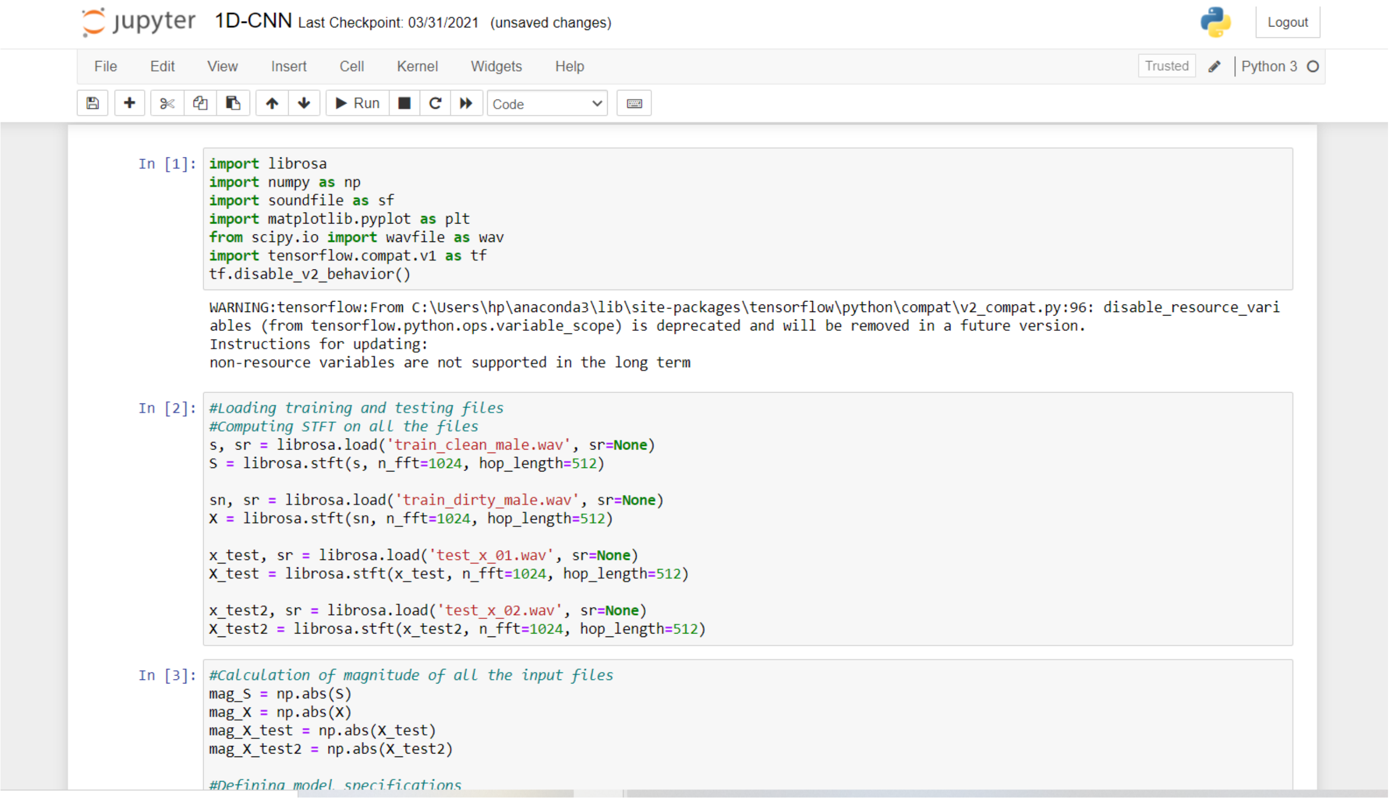
After running jupyter notebook we get our output 1D CNN denoised audio wav file in our given location



.ipynb_checkpoints
1D-CNN.ipynb
test_s_01_recons_q1.wav
test_s_02_recons_q1.wav
test_x_01.wav
test_x_02.wav
train_clean_male.wav
train_dirty_male.wav

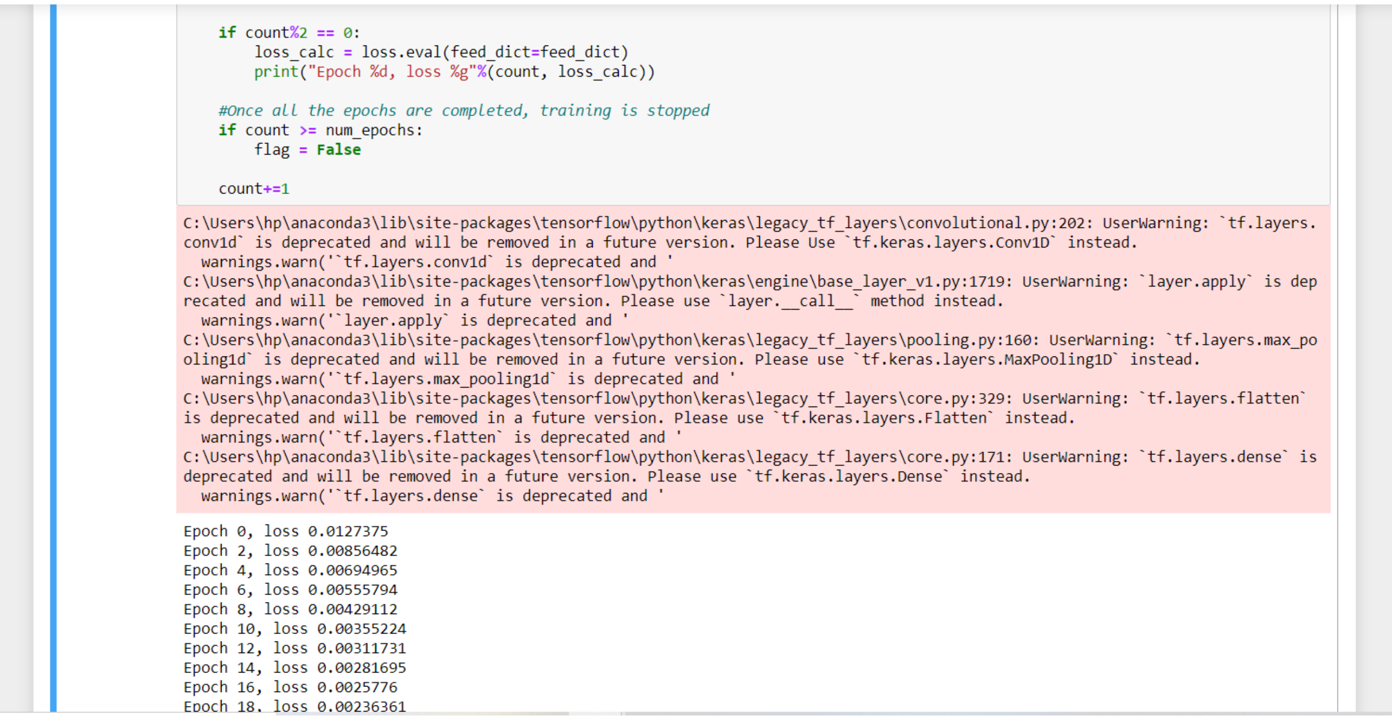
4) Stepwise windows screenshot

Working with jupyter notebook for speech denoising using deep learning.



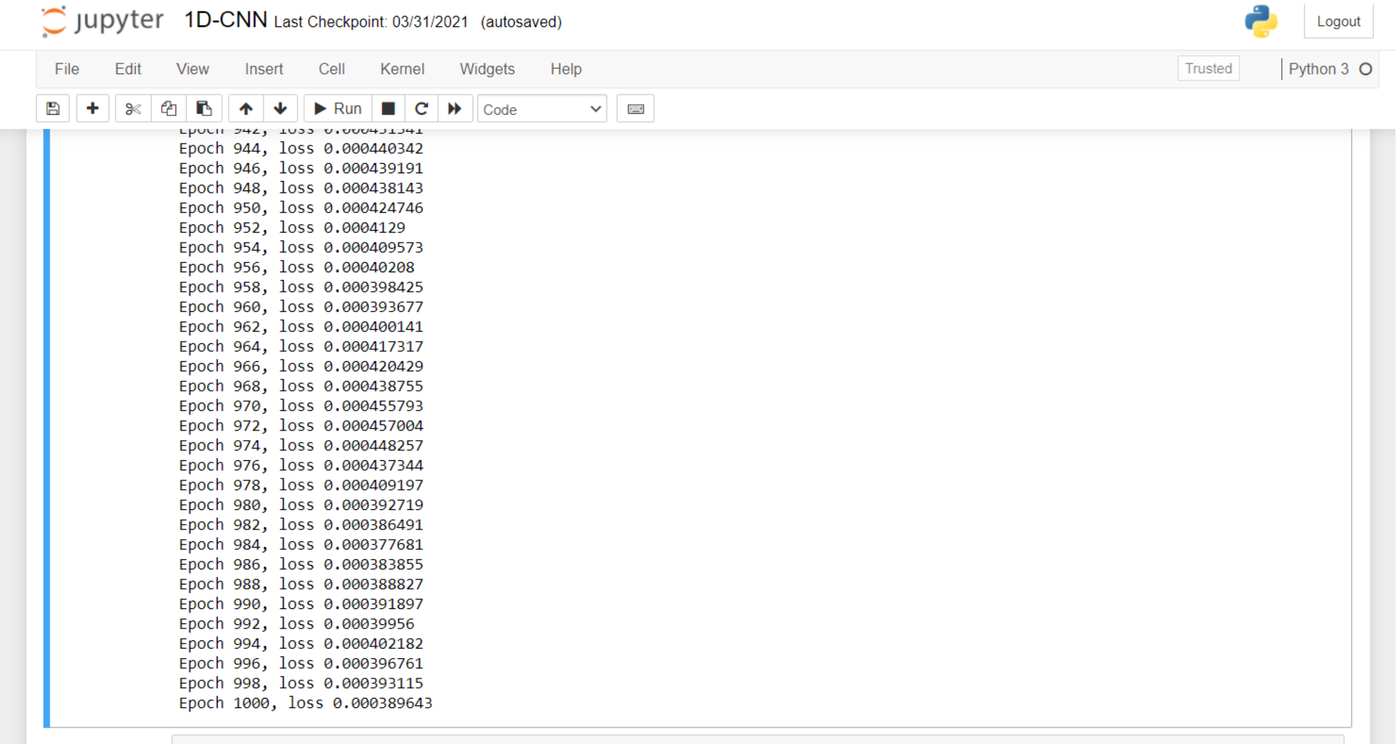
```
In [1]: import librosa
        import numpy as np
        import soundfile as sf
        import matplotlib.pyplot as plt
        from scipy.io import wavfile as wav
        import tensorflow.compat.v1 as tf
        tf.disable_v2_behavior()

        WARNING:tensorflow:From C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\compat\v2_compat.py:96: disable_resource_vari
        ables (from tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
        Instructions for updating:
        non-resource variables are not supported in the long term

In [2]: #Loading training and testing files
        #Computing STFT on all the files
        s, sr = librosa.load('train_clean_male.wav', sr=None)
        S = librosa.stft(s, n_fft=1024, hop_length=512)

        sn, sr = librosa.load('train_dirty_male.wav', sr=None)
        X = librosa.stft(sn, n_fft=1024, hop_length=512)

        x_test, sr = librosa.load('test_x_01.wav', sr=None)
        X_test = librosa.stft(x_test, n_fft=1024, hop_length=512)

        x_test2, sr = librosa.load('test_x_02.wav', sr=None)
        X_test2 = librosa.stft(x_test2, n_fft=1024, hop_length=512)

In [3]: #Calculation of magnitude of all the input files
        mag_S = np.abs(S)
        mag_X = np.abs(X)
        mag_X_test = np.abs(X_test)
        mag_X_test2 = np.abs(X_test2)

        #Defining model specifications
```

Training files:



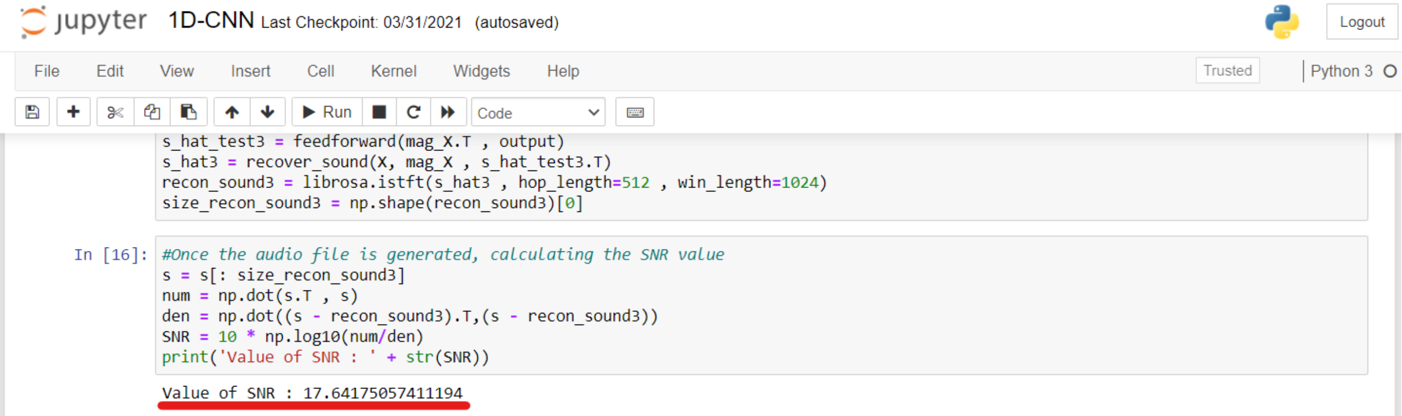
```
    if count%2 == 0:
        loss_calc = loss.eval(feed_dict=feed_dict)
        print("Epoch %d, loss %g"%(count, loss_calc))

    #Once all the epochs are completed, training is stopped
    if count >= num_epochs:
        flag = False

    count+=1

C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\convolutional.py:202: UserWarning: `tf.layers.
conv1d` is deprecated and will be removed in a future version. Please Use `tf.keras.layers.Conv1D` instead.
  warnings.warn('`tf.layers.conv1d` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\engine\base_layer_v1.py:1719: UserWarning: `layer.apply` is dep
recated and will be removed in a future version. Please use `layer.__call__` method instead.
  warnings.warn('`layer.apply` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\pooling.py:160: UserWarning: `tf.layers.max_po
oling1d` is deprecated and will be removed in a future version. Please use `tf.keras.layers.MaxPooling1D` instead.
  warnings.warn('`tf.layers.max_pooling1d` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\core.py:329: UserWarning: `tf.layers.flatten`
is deprecated and will be removed in a future version. Please use `tf.keras.layers.Flatten` instead.
  warnings.warn('`tf.layers.flatten` is deprecated and '
C:\Users\hp\anaconda3\lib\site-packages\tensorflow\python\keras\legacy_tf_layers\core.py:171: UserWarning: `tf.layers.dense` is
deprecated and will be removed in a future version. Please use `tf.keras.layers.Dense` instead.
  warnings.warn('`tf.layers.dense` is deprecated and '

Epoch 0, loss 0.0127375
Epoch 2, loss 0.00856482
Epoch 4, loss 0.00694965
Epoch 6, loss 0.00555794
Epoch 8, loss 0.00429112
Epoch 10, loss 0.00355224
Epoch 12, loss 0.00311731
Epoch 14, loss 0.00281695
Epoch 16, loss 0.0025776
Epoch 18, loss 0.00236361
```

Training completes with 1000 epochs



```
Epoch 942, loss 0.000431341
Epoch 944, loss 0.000440342
Epoch 946, loss 0.000439191
Epoch 948, loss 0.000438143
Epoch 950, loss 0.000424746
Epoch 952, loss 0.0004129
Epoch 954, loss 0.000409573
Epoch 956, loss 0.00040208
Epoch 958, loss 0.000398425
Epoch 960, loss 0.000393677
Epoch 962, loss 0.000400141
Epoch 964, loss 0.000417317
Epoch 966, loss 0.000420429
Epoch 968, loss 0.000438755
Epoch 970, loss 0.000455793
Epoch 972, loss 0.000457004
Epoch 974, loss 0.000448257
Epoch 976, loss 0.000437344
Epoch 978, loss 0.000409197
Epoch 980, loss 0.000392719
Epoch 982, loss 0.000386491
Epoch 984, loss 0.000377681
Epoch 986, loss 0.000383855
Epoch 988, loss 0.000388827
Epoch 990, loss 0.000391897
Epoch 992, loss 0.00039956
Epoch 994, loss 0.000402182
Epoch 996, loss 0.000396761
Epoch 998, loss 0.000393115
Epoch 1000, loss 0.000389643
```

SNR calculated after loading, training, processing



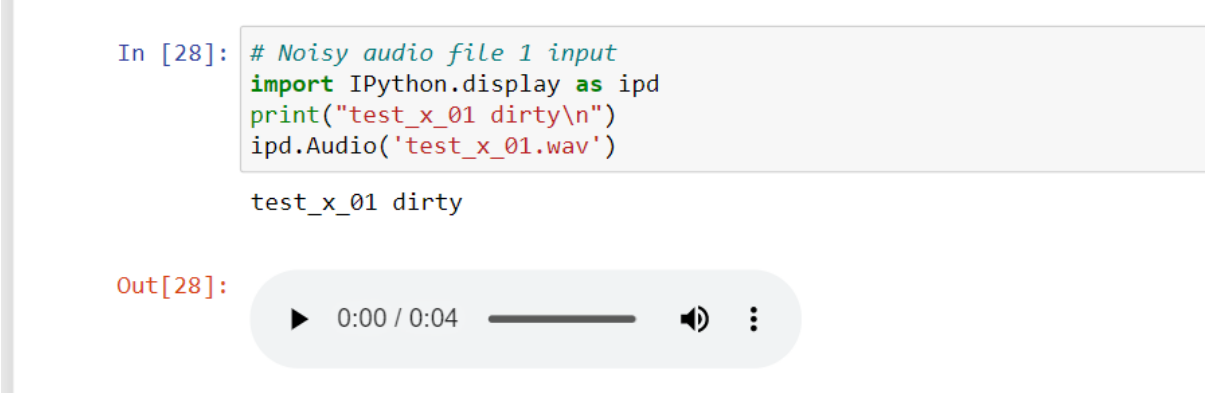
```
s_hat_test3 = feedforward(mag_X.T , output)
s_hat3 = recover_sound(X, mag_X , s_hat_test3.T)
recon_sound3 = librosa.istft(s_hat3 , hop_length=512 , win_length=1024)
size_recon_sound3 = np.shape(recon_sound3)[0]
```

```
In [16]: #Once the audio file is generated, calculating the SNR value
         s = s[: size_recon_sound3]
         num = np.dot(s.T , s)
         den = np.dot((s - recon_sound3).T,(s - recon_sound3))
         SNR = 10 * np.log10(num/den)
         print('Value of SNR : ' + str(SNR))

         Value of SNR : 17.64175057411194
```

SNR: 17.641750

Noisy Input Files

Noisy audio input file 1



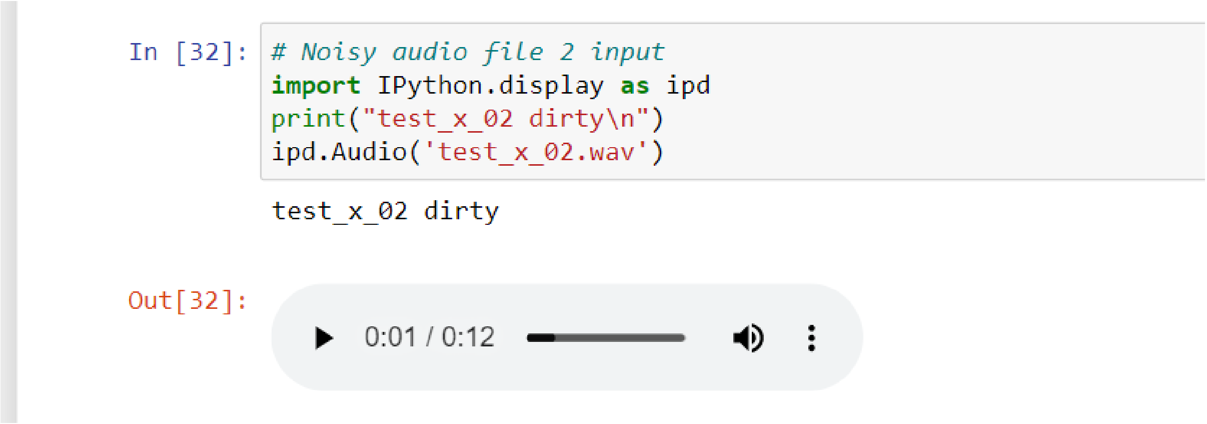
In [28]:

```
# Noisy audio file 1 input
import IPython.display as ipd
print("test_x_01 dirty\n")
ipd.Audio('test_x_01.wav')
```

```
test_x_01 dirty
```

Out[28]:

Noisy audio input file 2



In [32]:

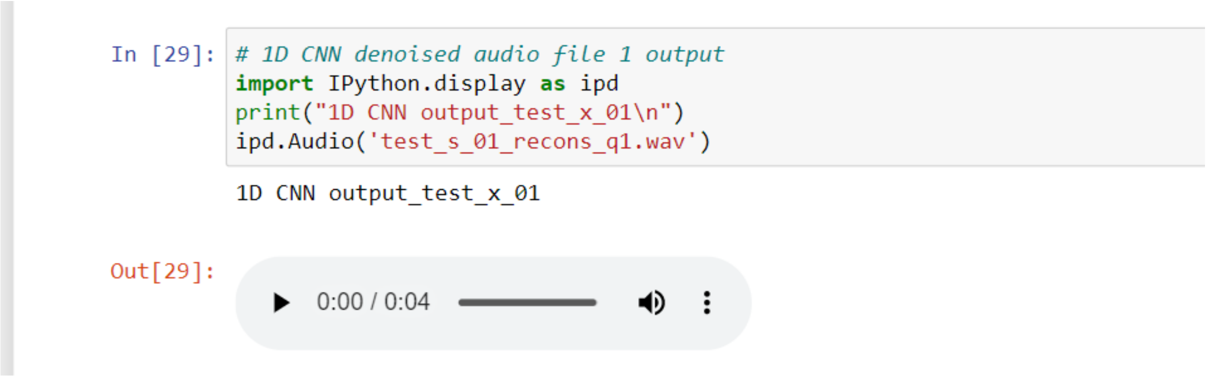
```
# Noisy audio file 2 input
import IPython.display as ipd
print("test_x_02 dirty\n")
ipd.Audio('test_x_02.wav')
```

```
test_x_02 dirty
```

Out[32]:

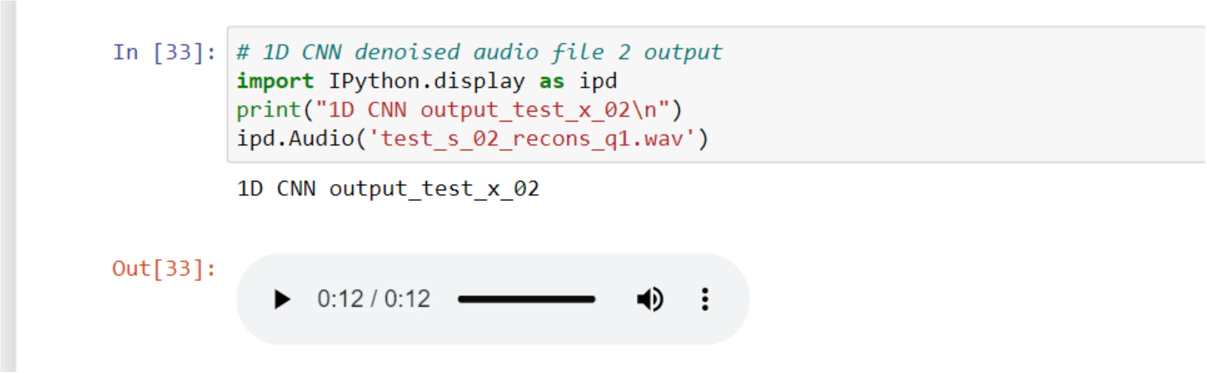
Denoised audio output files by 1D CNN

Denoised audio output file 1 by 1D CNN



```
In [29]: # 1D CNN denoised audio file 1 output
         import IPython.display as ipd
         print("1D CNN output_test_x_01\n")
         ipd.Audio('test_s_01_recons_q1.wav')

         1D CNN output_test_x_01

Out[29]:
```

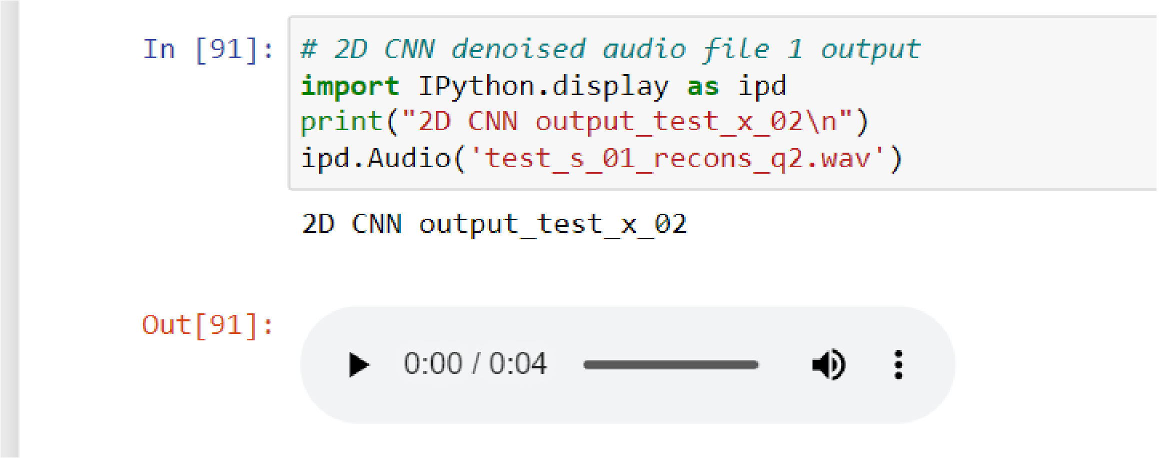
Denoised audio output file 2 by 1D CNN



```
In [33]: # 1D CNN denoised audio file 2 output
         import IPython.display as ipd
         print("1D CNN output_test_x_02\n")
         ipd.Audio('test_s_02_recons_q1.wav')

         1D CNN output_test_x_02

Out[33]:
```

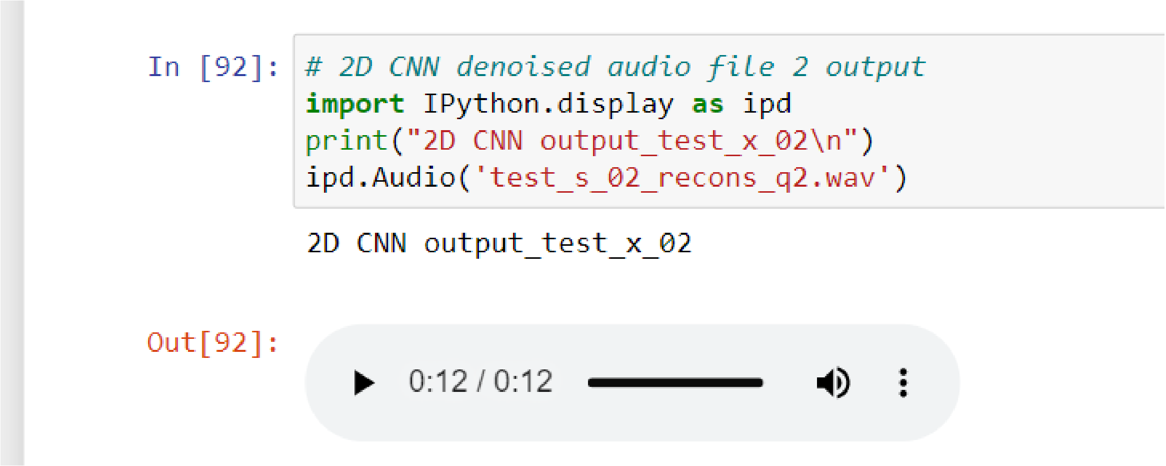
Denoised audio output files by 1D CNN

Denoised audio file 1 by 2D CNN



```
In [91]: # 2D CNN denoised audio file 1 output
         import IPython.display as ipd
         print("2D CNN output_test_x_02\n")
         ipd.Audio('test_s_01_recons_q2.wav')

         2D CNN output_test_x_02

Out[91]:
```

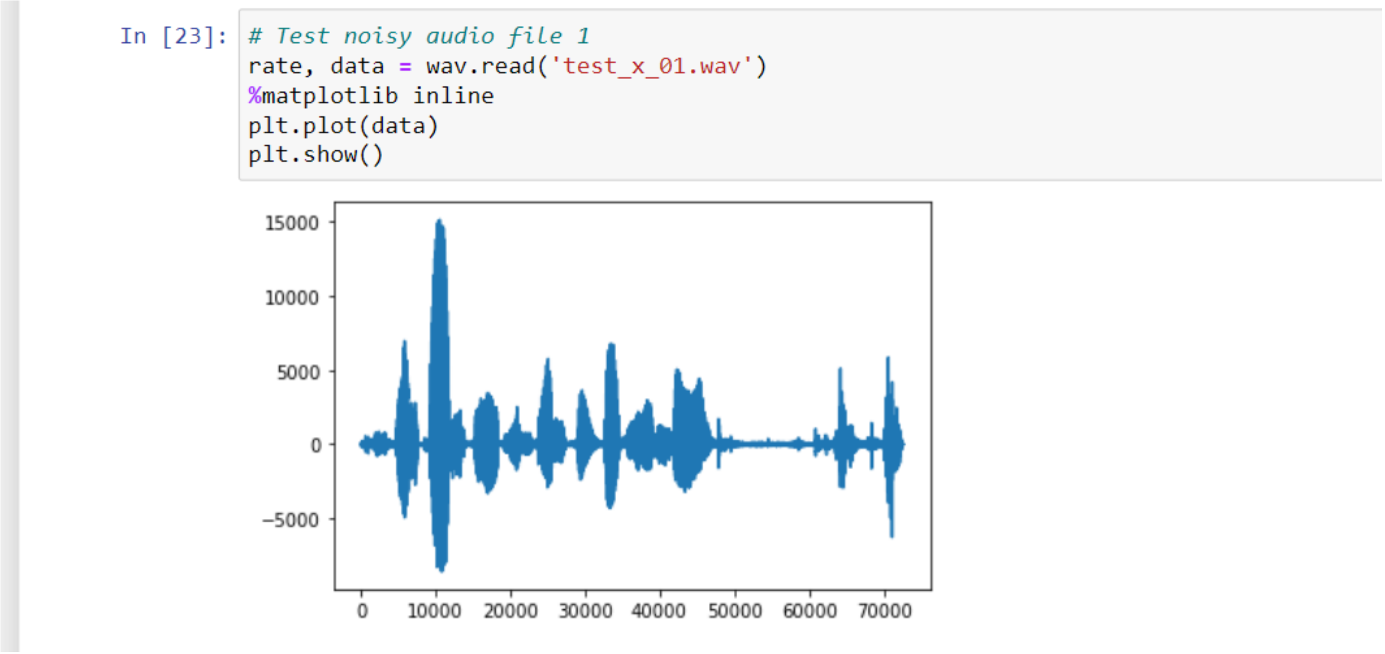
Denoised audio file 2 by 2D CNN



```
In [92]: # 2D CNN denoised audio file 2 output
         import IPython.display as ipd
         print("2D CNN output_test_x_02\n")
         ipd.Audio('test_s_02_recons_q2.wav')

         2D CNN output_test_x_02

Out[92]:
```

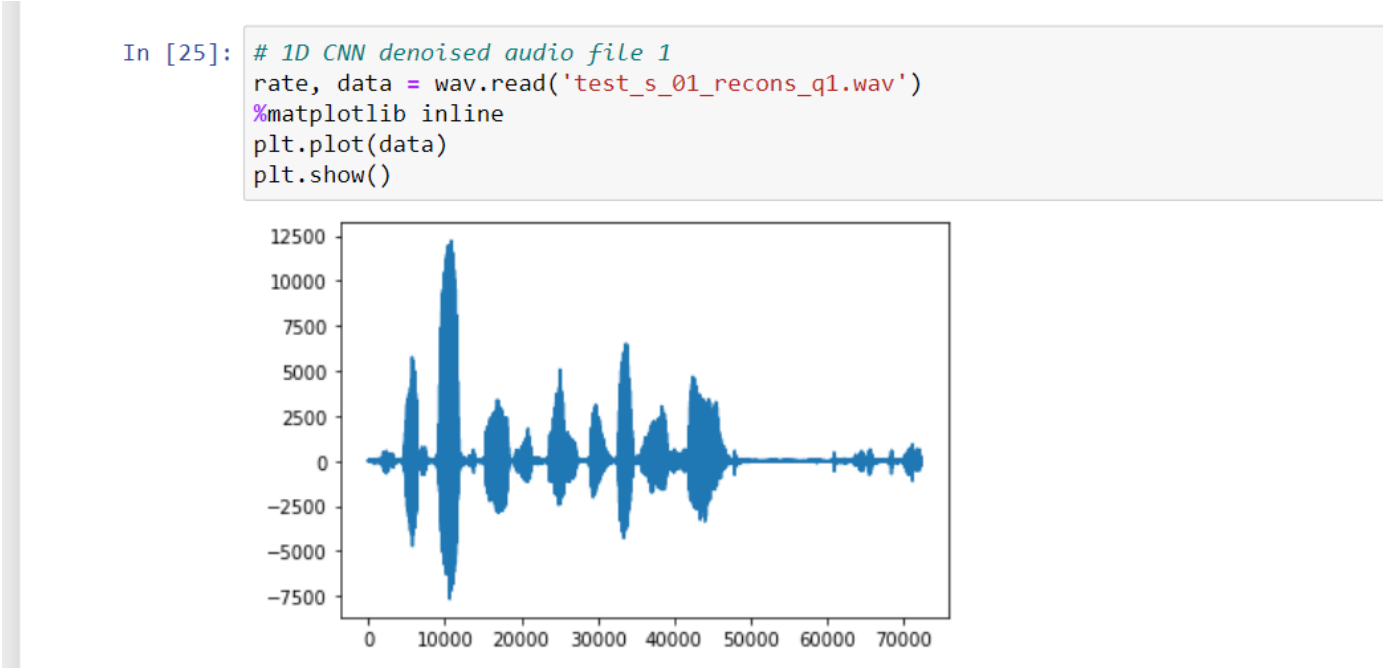
Waveforms of audio files (1D CNN)

Waveform of input noisy audio file 1



```
In [23]: # Test noisy audio file 1
         rate, data = wav.read('test_x_01.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

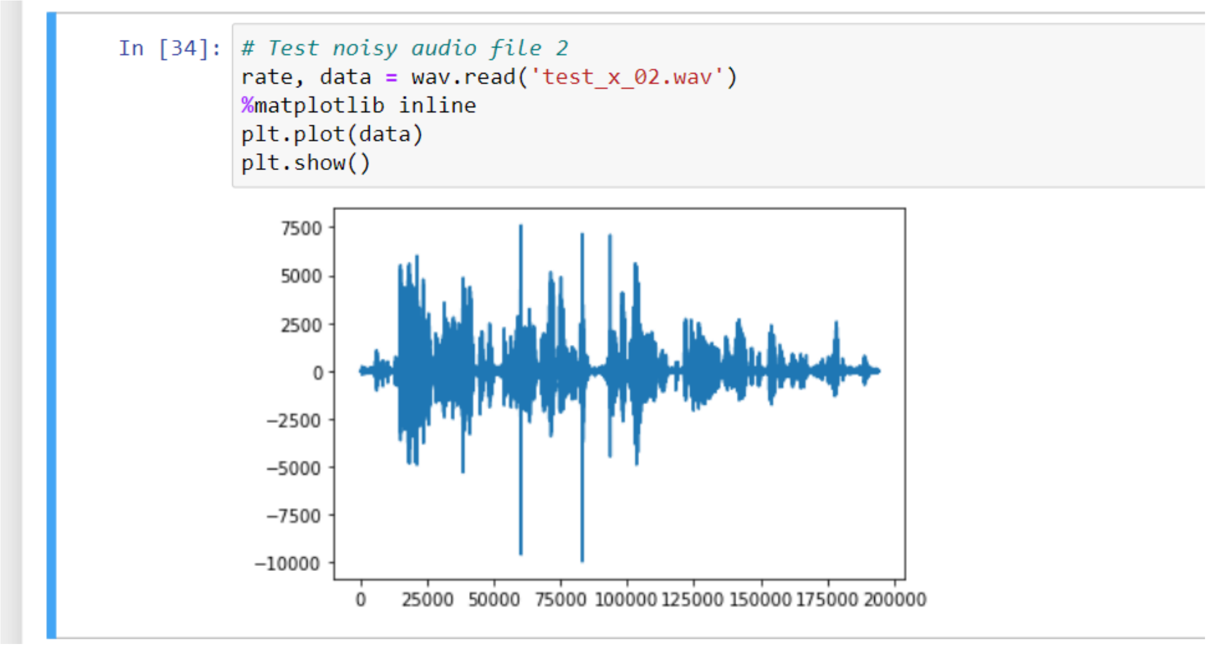
Waveform of output denoised audio file 1 by 1D CNN



```
In [25]: # 1D CNN denoised audio file 1
         rate, data = wav.read('test_s_01_recons_q1.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

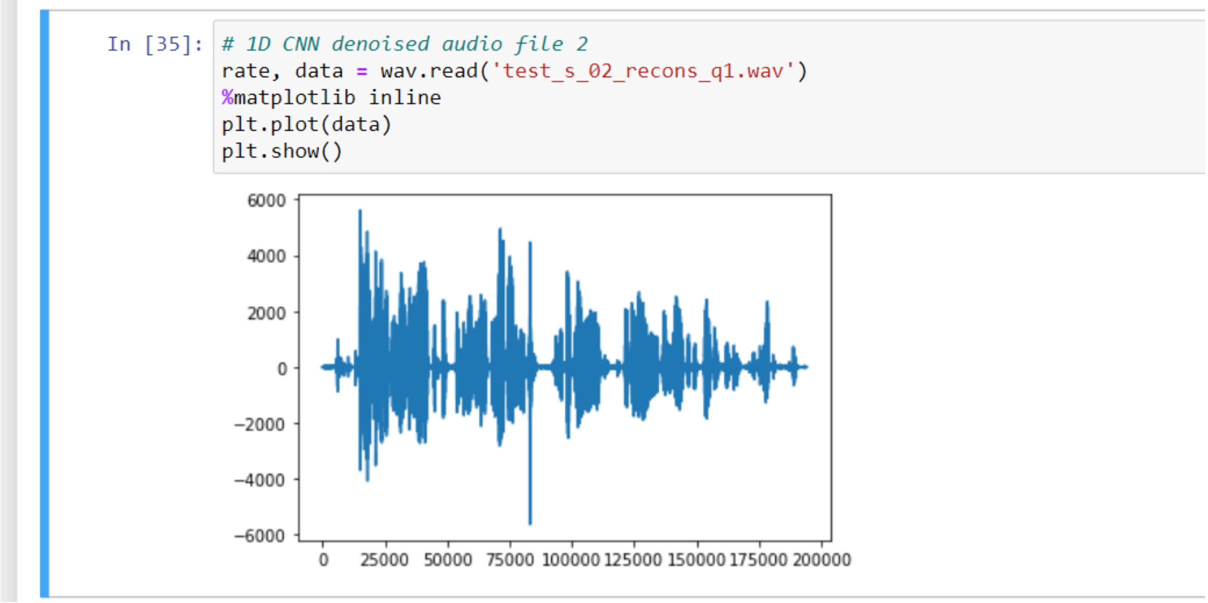
Waveforms of audio files (1D CNN)

Waveform of input noisy audio file 2



```
In [34]: # Test noisy audio file 2
         rate, data = wav.read('test_x_02.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

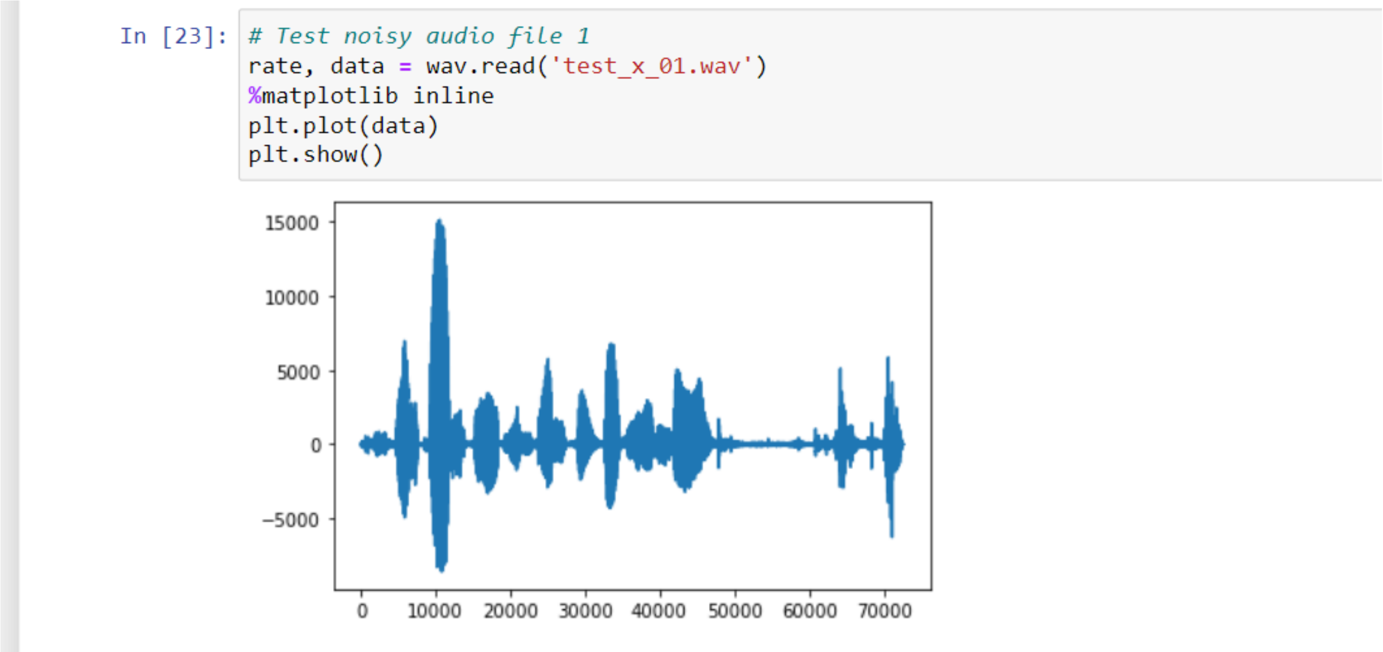
Waveform of output denoised audio file 2 by 1D CNN



```
In [35]: # 1D CNN denoised audio file 2
         rate, data = wav.read('test_s_02_recons_q1.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

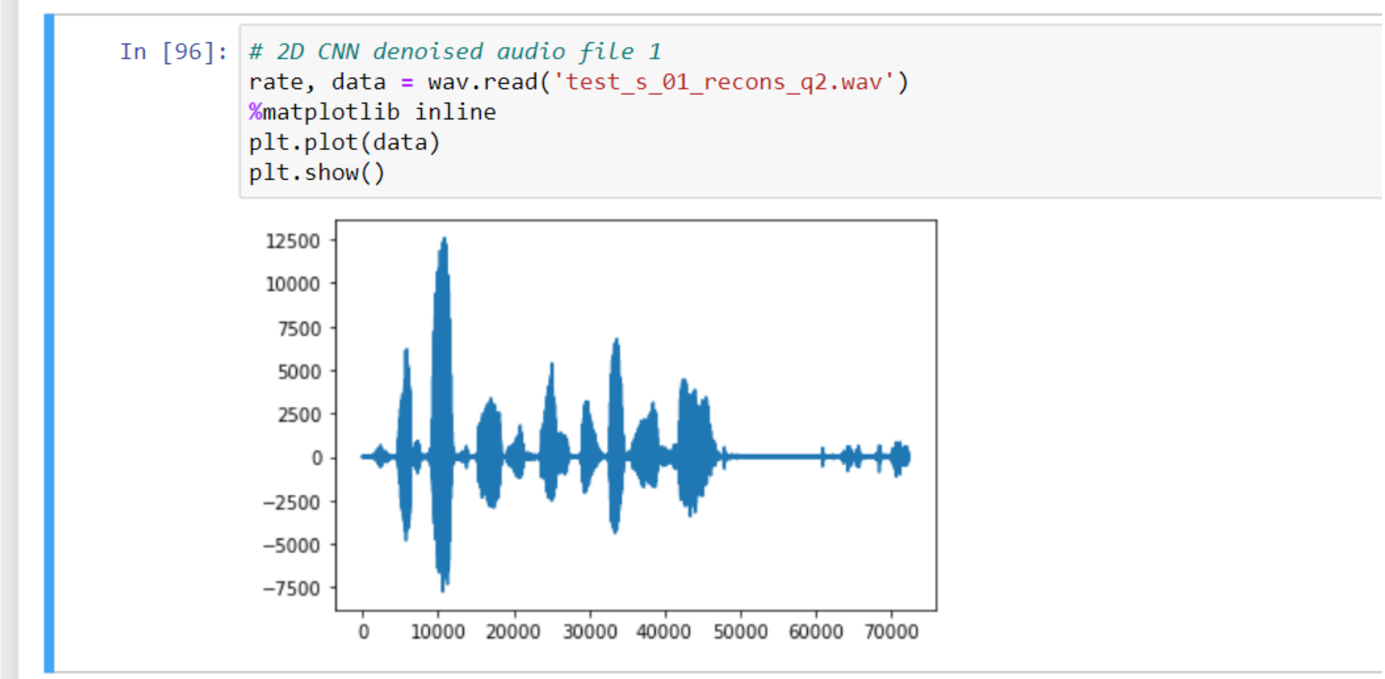
Waveforms of audio files (2D CNN)

Waveform of input noisy audio file 1



```
In [23]: # Test noisy audio file 1
         rate, data = wav.read('test_x_01.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

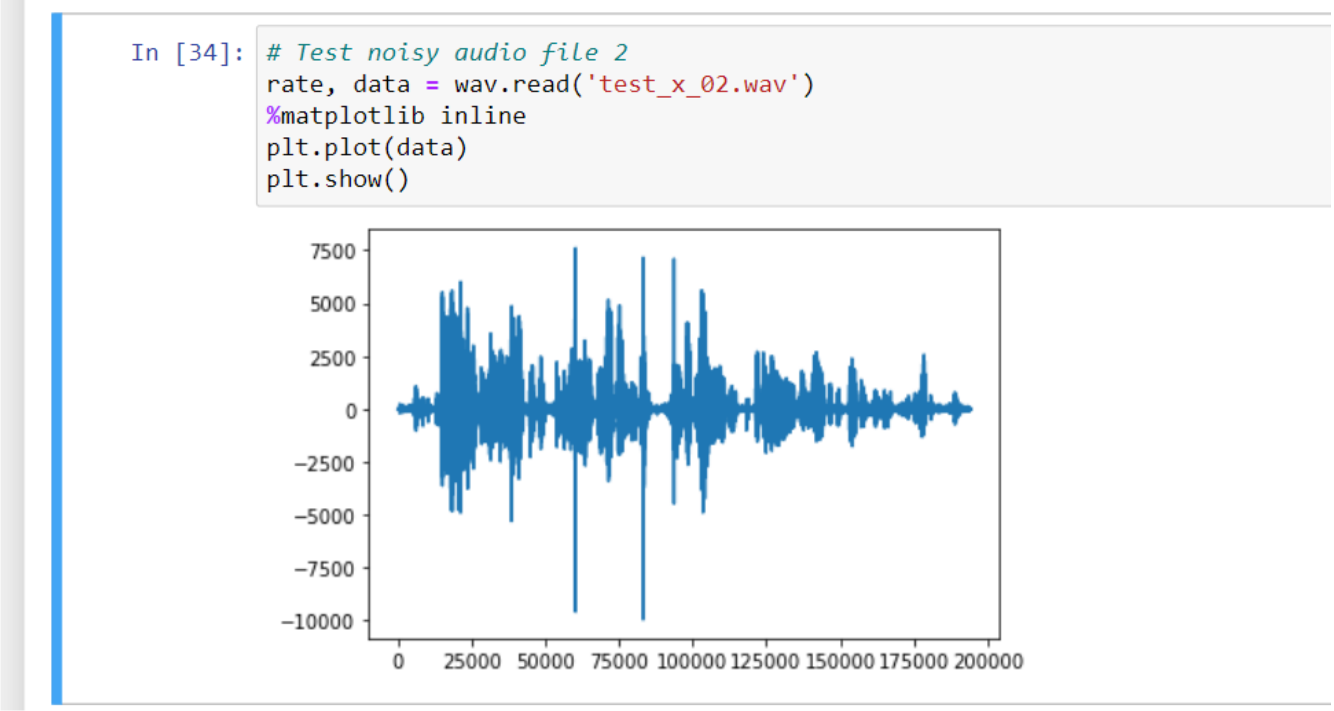
2D CNN Denoised audio file 1



```
In [96]: # 2D CNN denoised audio file 1
         rate, data = wav.read('test_s_01_recons_q2.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

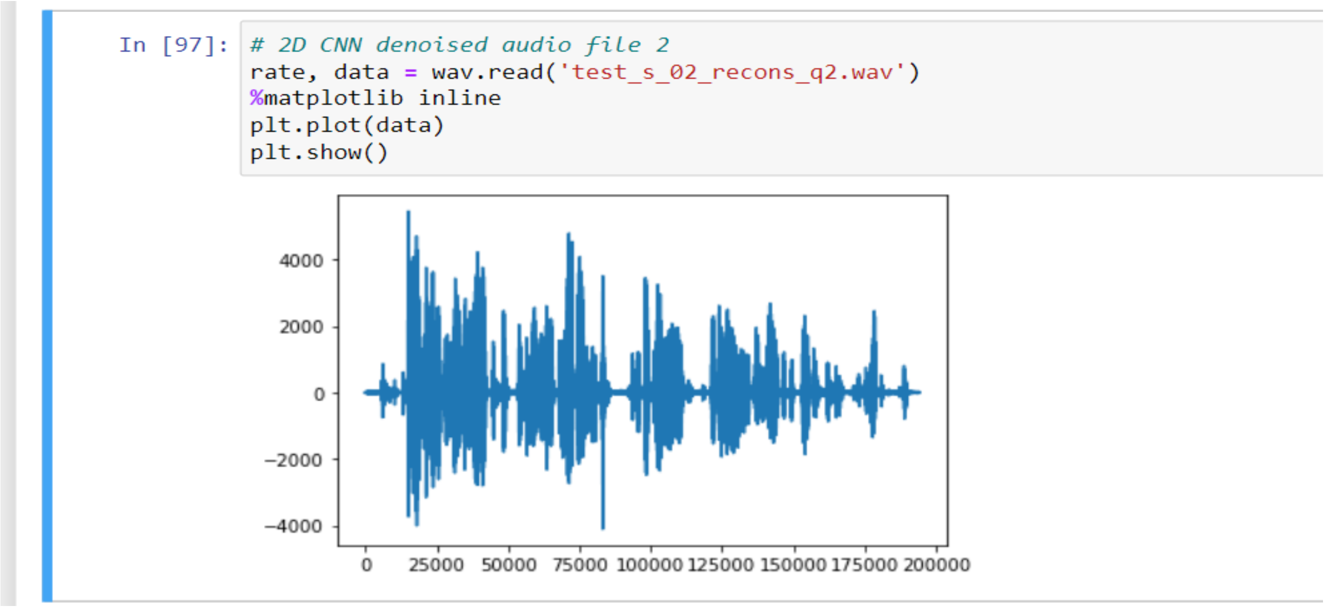
Waveforms of audio files (2D CNN)

Waveform of input noisy audio file 2



```
In [34]: # Test noisy audio file 2
         rate, data = wav.read('test_x_02.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

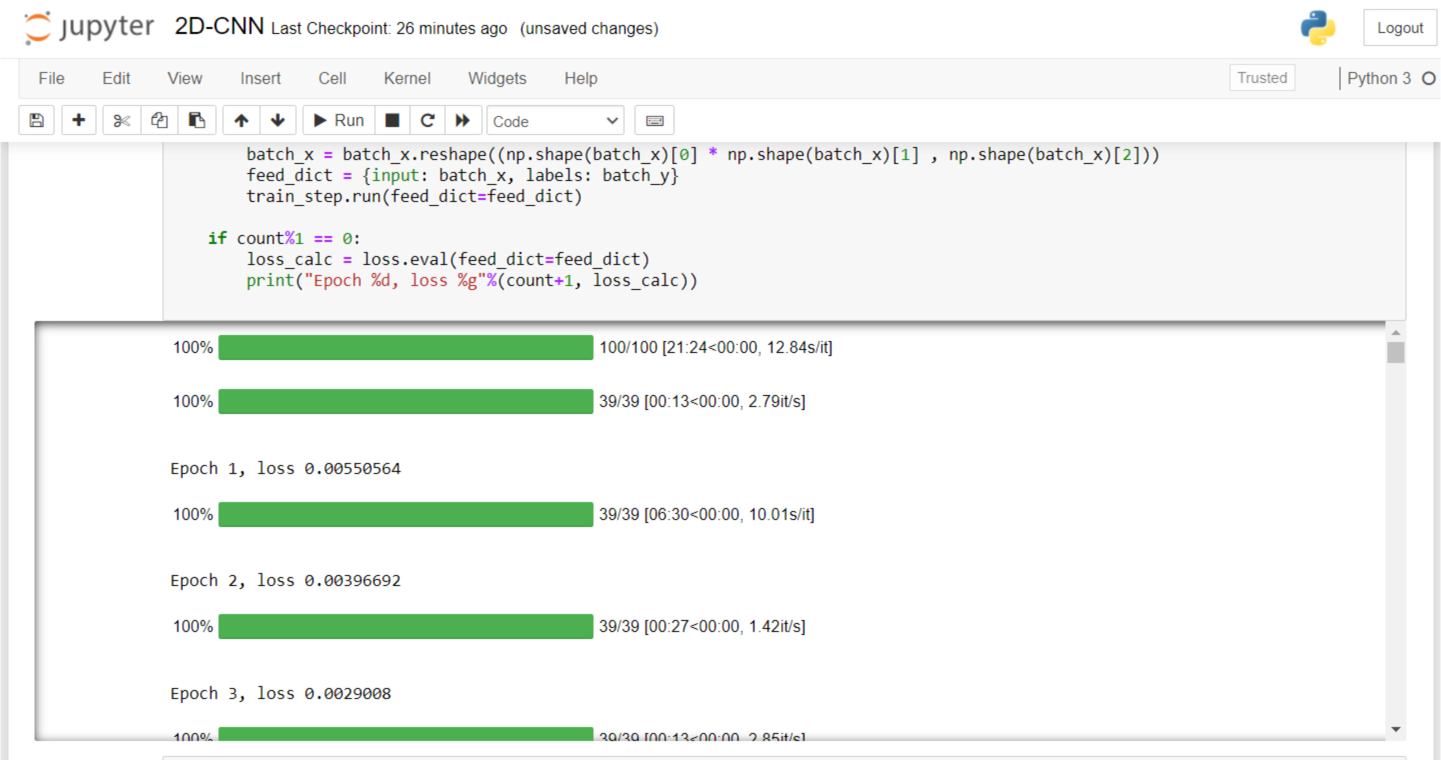
Waveform of denoised audio file 2 by 2D CNN



```
In [97]: # 2D CNN denoised audio file 2
         rate, data = wav.read('test_s_02_recons_q2.wav')
         %matplotlib inline
         plt.plot(data)
         plt.show()
```

2D CNN training

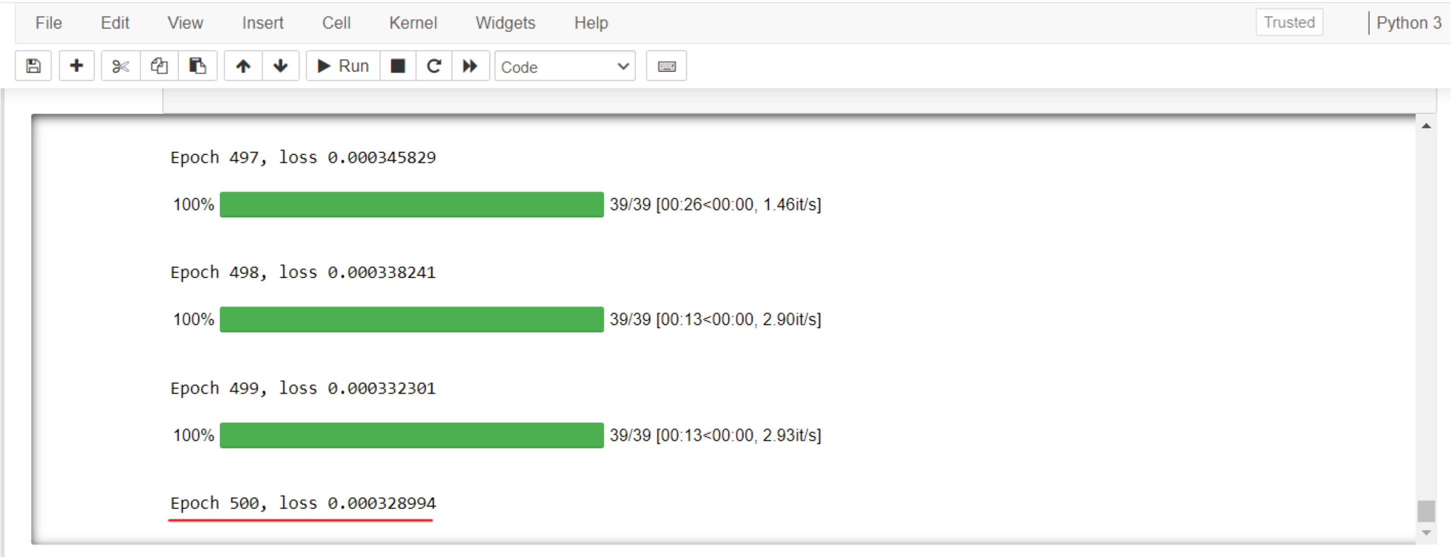
Training started



```
        batch_x = batch_x.reshape((np.shape(batch_x)[0] * np.shape(batch_x)[1] , np.shape(batch_x)[2]))
        feed_dict = {input: batch_x, labels: batch_y}
        train_step.run(feed_dict=feed_dict)

    if count%1 == 0:
        loss_calc = loss.eval(feed_dict=feed_dict)
        print("Epoch %d, loss %g"%(count+1, loss_calc))
```

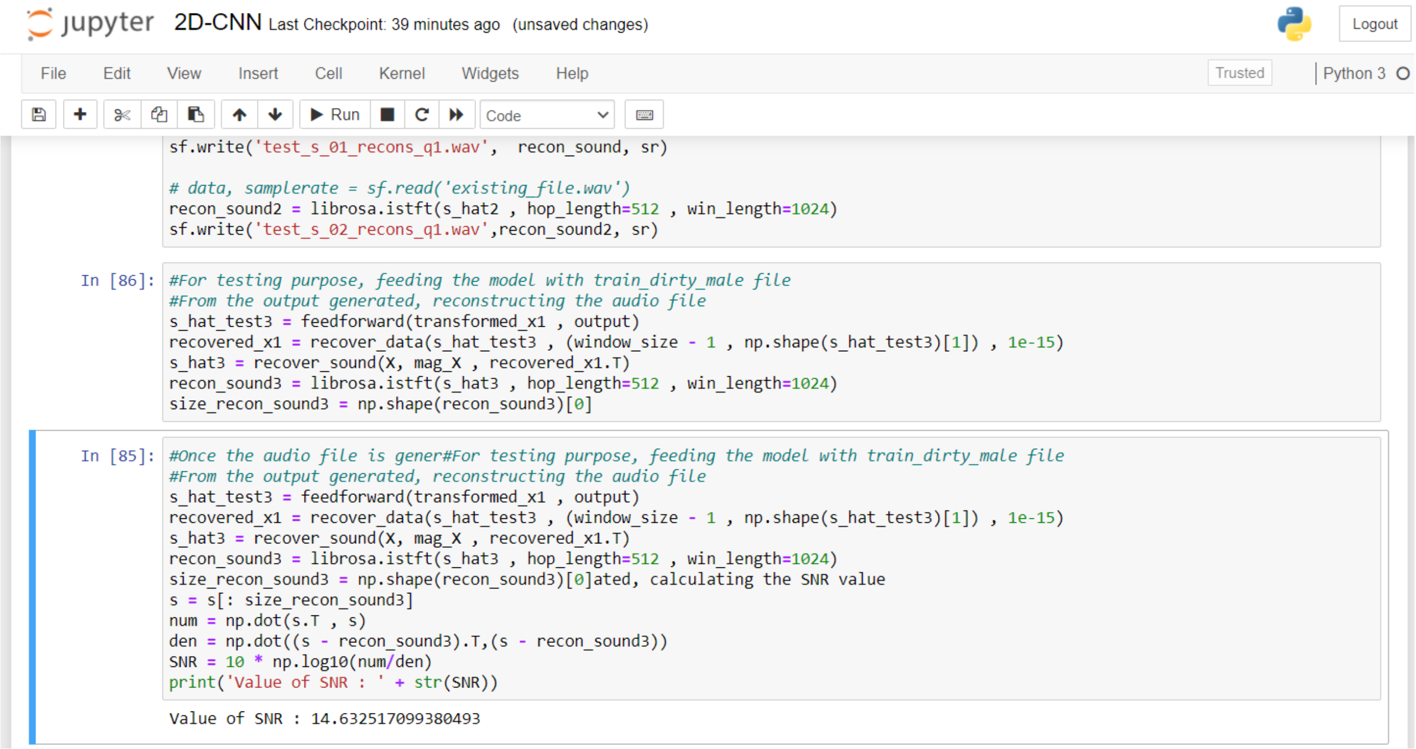
```
100% 100/100 [21:24<00:00, 12.84s/it]

100% 39/39 [00:13<00:00, 2.79it/s]

Epoch 1, loss 0.00550564

100% 39/39 [06:30<00:00, 10.01s/it]

Epoch 2, loss 0.00396692

100% 39/39 [00:27<00:00, 1.42it/s]

Epoch 3, loss 0.0029008

100% 39/39 [00:13<00:00, 2.85it/s]
```

Training ended



```
Epoch 497, loss 0.000345829

100% 39/39 [00:26<00:00, 1.46it/s]

Epoch 498, loss 0.000338241

100% 39/39 [00:13<00:00, 2.90it/s]

Epoch 499, loss 0.000332301

100% 39/39 [00:13<00:00, 2.93it/s]

Epoch 500, loss 0.000328994
```

2D CNN SNR: 14.63251


jupyter 2D-CNN Last Checkpoint: 39 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
sf.write('test_s_01_recons_q1.wav',  recon_sound, sr)

# data, samplerate = sf.read('existing_file.wav')
recon_sound2 = librosa.istft(s_hat2 , hop_length=512 , win_length=1024)
sf.write('test_s_02_recons_q1.wav',recon_sound2, sr)
```

In [86]:

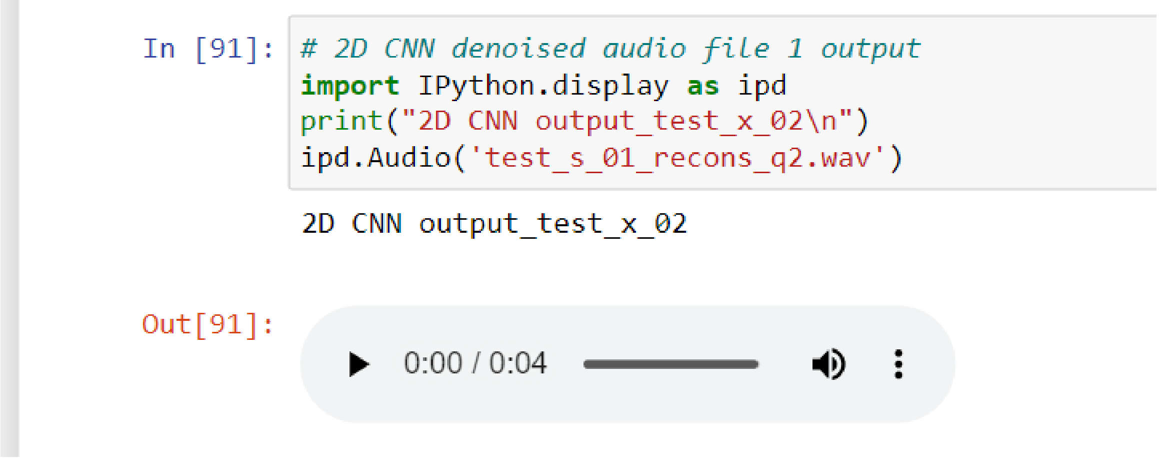
```
#For testing purpose, feeding the model with train_dirty_male file
#From the output generated, reconstructing the audio file
s_hat_test3 = feedforward(transformed_x1 , output)
recovered_x1 = recover_data(s_hat_test3 , (window_size - 1 , np.shape(s_hat_test3)[1]) , 1e-15)
s_hat3 = recover_sound(X, mag_X , recovered_x1.T)
recon_sound3 = librosa.istft(s_hat3 , hop_length=512 , win_length=1024)
size_recon_sound3 = np.shape(recon_sound3)[0]
```

In [85]:

```
#Once the audio file is gener#For testing purpose, feeding the model with train_dirty_male file
#From the output generated, reconstructing the audio file
s_hat_test3 = feedforward(transformed_x1 , output)
recovered_x1 = recover_data(s_hat_test3 , (window_size - 1 , np.shape(s_hat_test3)[1]) , 1e-15)
s_hat3 = recover_sound(X, mag_X , recovered_x1.T)
recon_sound3 = librosa.istft(s_hat3 , hop_length=512 , win_length=1024)
size_recon_sound3 = np.shape(recon_sound3)[0]ated, calculating the SNR value
s = s[: size_recon_sound3]
num = np.dot(s.T , s)
den = np.dot((s - recon_sound3).T,(s - recon_sound3))
SNR = 10 * np.log10(num/den)
print('Value of SNR : ' + str(SNR))
```

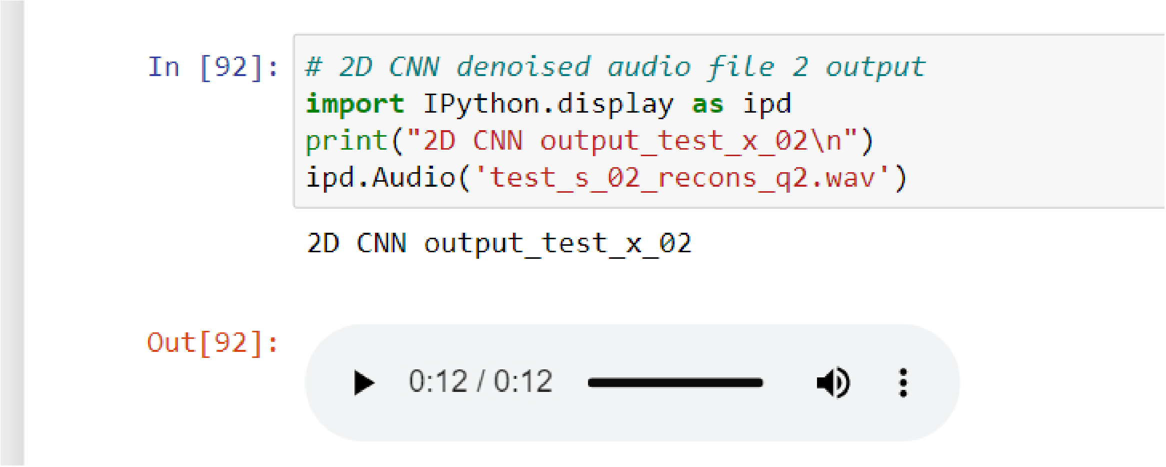
```
Value of SNR : 14.632517099380493
```

Waveform of 2D CNN Denoised audio file 1



```
In [91]: # 2D CNN denoised audio file 1 output
         import IPython.display as ipd
         print("2D CNN output_test_x_02\n")
         ipd.Audio('test_s_01_recons_q2.wav')

         2D CNN output_test_x_02

Out[91]:
```

Waveform of 2D CNN Denoised audio file 2



```
In [92]: # 2D CNN denoised audio file 2 output
         import IPython.display as ipd
         print("2D CNN output_test_x_02\n")
         ipd.Audio('test_s_02_recons_q2.wav')

         2D CNN output_test_x_02

Out[92]:
```

5) Block Diagram

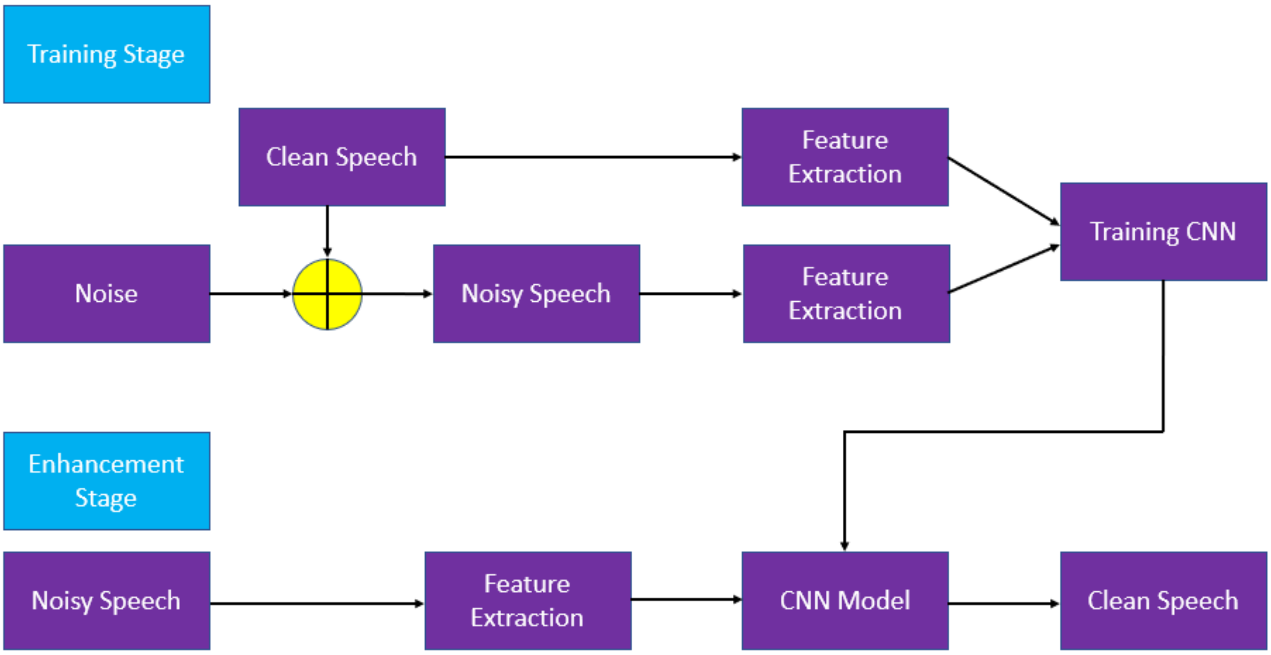


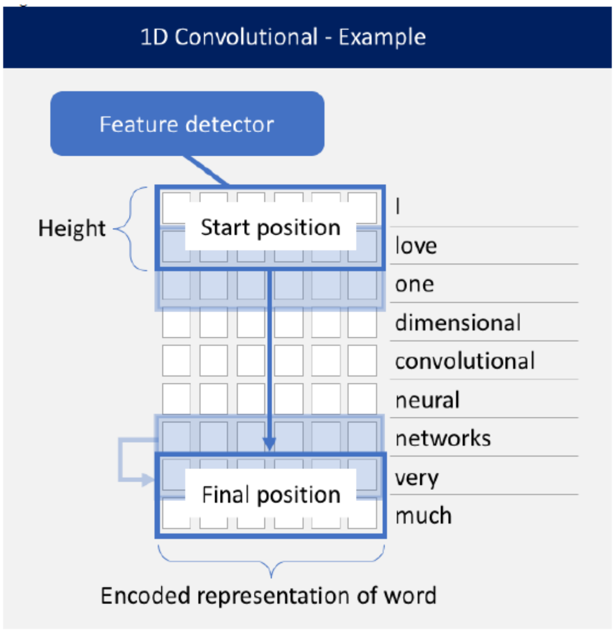
Figure 1: Block Diagram

6) Algorithm Used

1) Loading training and testing input audio files using load() function of librosa by which audio files are converted into NumPy array.
2) Computing short time fourier transform on all those input audio files array, using stft() function of librosa which transforms this array (consisting amplitude in time domain) to frequency domain. This helps in removing high frequency noise.
3) Calculate magnitude of all input files using NumPy.
4) Defining CNN model specifications like learning rate, epochs, no. of layers, filters, kernels. After every convolutional layer we are adding pooling layer. Non linearity ReLu has been applied to feature maps output by convolutional layer.
5) We defined mean square loss function with Adam optimizer to minimize the loss between expected and model output to train deep learning model
6) We use neural networks to find a transformation between the already pre-processed noisy clip and the clean clip.
7) Then we performed maxpooling and added a fully connected layer at the end to reduce the dimensionality and get the desired dimensions.
8) After training the model, we tested the performance of the model by calculating the Signal-to-Noise Ratio (SNR) value and performed ISTFT to check how the audio sounded.

1D CNN design implemented:

- Two convolution layers with filters 16 and 32 respectively.
- Also kernel sizes of 16, 8 respectively.
- Same padding is used.
- ReLU activation function is used in all the convolution layers.
- Max pooling layers are implemented one each after the convolution layer.
- Flattening is implemented for the last max pooling layer.
- A dense layer of 513 units with a ReLU activation.
- Adam optimizer, mean squared error loss function are used.
- 1000 epochs are used for training.

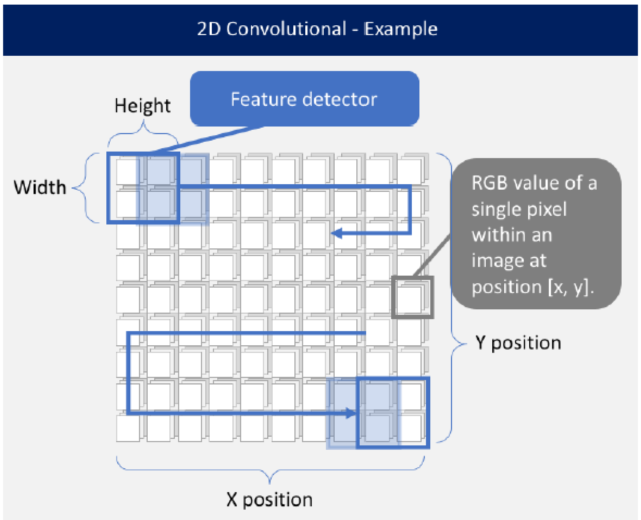


"In this example for natural language processing, a sentence is made up of 9 words. Each word is a vector that represents a word as a low dimensional representation. The feature detector will always cover the whole word. The height determines how many words are considered when training the feature detector. In our example, the height is two. In this example the feature detector will iterate through the data 8 times."

Figure 2: 1D CNN by Blog/Good Audience

2D CNN design implemented:

- Input layer with Tensor Shape as (-1,20,513,1).
- Two convolution layers used with filters of 16,32 are used respectively.
- Also kernel size of (4,4) for all of the above layers.
- ReLU activation is used in all the convolution layers.
- Max pooling layers are used after each convolution layer, with pool_size of (2,2).
- Final max pooling layer is flattened.
- A dense layer is used with 513 hidden units with ReLU activation function.
- Adam optimizer (0.0002 learning rate) and Mean squared error Loss function are used.
- Batch size of 64 and 500 epochs for training.


2D Convolutional - Example
Height
Feature detector
Width
RGB value of a single pixel within an image at position [x, y].
Y position
X position

"In this example for computer vision, each pixel within the Image is represented by its x· and y position as well as three values (RGB). The feature detector has a dimension of 2 x 2 in our example. The feature detector will now slide both horizontally and vertically across the image.

Figure 3: 2D CNN by Blog/Good Audience

7) Observations and Investigations

Comparison

Here, we compare the performances of speech denoising first computing stft then applying deep learning model by 1D CNN & 2D CNN.

Method	Accuracy (SNR)
1D CNN	17. 641 DB
2D CNN	14. 632 DB

Table 1: Performance of stft + deep learning-based method

8) Conclusion

Hence it is evident from the above results that using (1D CNN + stft) gives better result compared to (2D CNN + stft) for speech denoising by removal of noise from noisy speech and speech enhancement.

By having SNR of 17.64 DB by 1D CNN compared to 14.632 DB by 2D CNN.

9) Code Link: https://github.com/mohdshahbaz123/Speech-Denoising-using-Deep-Learning