In [1]:

```
%load_ext autoreload
%autoreload 2
#%env CUDA_VISIBLE_DEVICES=3
```

Download packages if in Google Colab

In [2]:

```
colab_requirements = [
    "pip install librosa",
    "pip install noisereduce",
    "pip install soundfile",

]

import sys, subprocess

def run_subprocess_command(cmd):
    # run the command
    process = subprocess.Popen(cmd.split(), stdout=subprocess.PIPE)
    # print the output
    for line in process.stdout:
        print(line.decode().strip())

IN_COLAB = "google.colab" in sys.modules
if IN_COLAB:
    for i in colab_requirements:
        run_subprocess_command(i)
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/
(https://us-python.pkg.dev/colab-wheels/public/simple/)
Requirement already satisfied: librosa in /usr/local/lib/python3.10/dist-packages (0.10.0.post2)
Requirement already satisfied: soxr>=0.3.2 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.3.5)
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.56.4)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.4.2)
Requirement already satisfied: joblib>=0.14 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.2.0)
Requirement already satisfied: pooch<1.7,>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.6.0)
Requirement already satisfied: lazy-loader>=0.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.2)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.10/dist-packages (from librosa) (3.0.0)
Requirement already satisfied: soundfile>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.12.1)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.5.0)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.2.2)
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in /usr/local/lib/python3.10/dist-packages (from libr
osa) (1.22.4)
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.0.5)
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.10.1)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.0->lib
rosa) (0.39.1)
```

Test noise reduction algorithm and view steps of algorithm

In [3]:

```
import IPython
from scipy.io import wavfile
import noisereduce as nr
import soundfile as sf
from noisereduce.generate_noise import band_limited_noise
import matplotlib.pyplot as plt
import urllib.request
import numpy as np
import io
%matplotlib inline
```

Load data

In [4]:

```
url = "https://raw.githubusercontent.com/timsainb/noisereduce/master/assets/fish.wav"
response = urllib.request.urlopen(url)
data, rate = sf.read(io.BytesIO(response.read()))
data = data
```

In [5]:

```
IPython.display.Audio(data=data, rate=rate)
```

Out[5]:

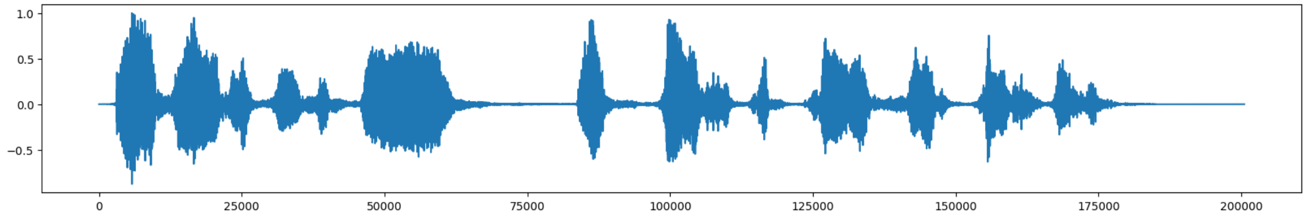
0:00 / 0:00

In [6]:

```
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(data)
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x7fb144981ab0>]
```



add noise

In [7]:

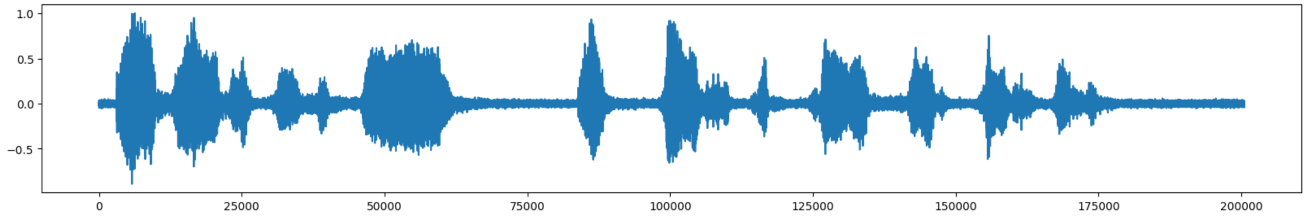
```
noise_len = 2 # seconds
noise = band_limited_noise(min_freq=2000, max_freq = 12000, samples=len(data), samplerate=rate)*10
noise_clip = noise[:rate*noise_len]
audio_clip_band_limited = data+noise
```

In [8]:

```
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(audio_clip_band_limited)
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x7fb1432f6ec0>]
```



In [9]:

```
IPython.display.Audio(data=audio_clip_band_limited, rate=rate)
```

Out[9]:

0:00 / 0:00

Stationary remove noise

In [10]:

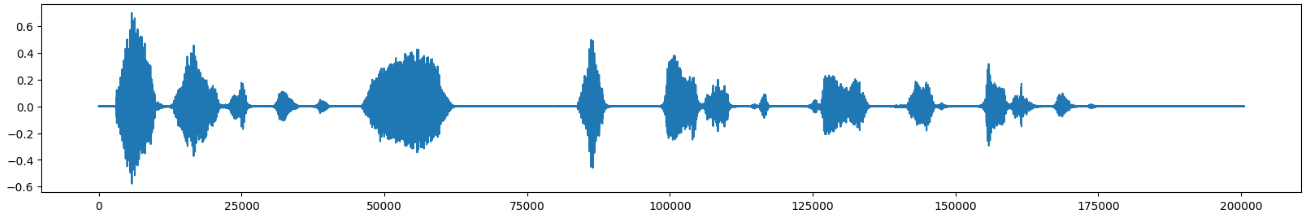
```
reduced_noise = nr.reduce_noise(y = audio_clip_band_limited, sr=rate, n_std_thresh_stationary=1.5,stationary=True)
```

In [11]:

```
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(reduced_noise)
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x7fb1391ae560>]
```



In [12]:

```
IPython.display.Audio(data=reduced_noise, rate=rate)
```

Out[12]:

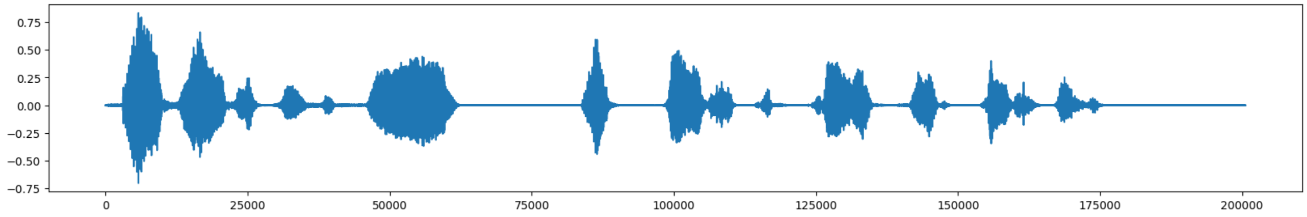
0:00 / 0:00

Non-stationary noise reduction

In [13]:
```
reduced_noise = nr.reduce_noise(y = audio_clip_band_limited, sr=rate, thresh_n_mult_nonstationary=2,stationary=False)
```

In [14]:
```
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(reduced_noise)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x7fb1394fa050>]
```

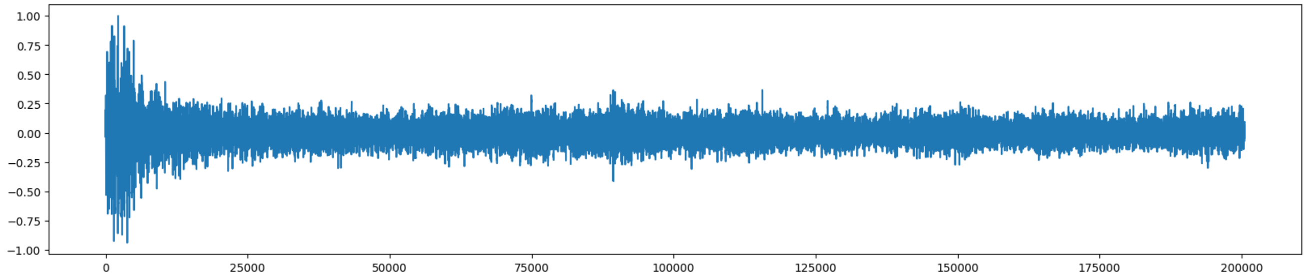


A more difficult example

In [15]:
```
url = "https://raw.githubusercontent.com/timsainb/noisereduce/master/assets/cafe_short.wav"
response = urllib.request.urlopen(url)
noise_data, noise_rate = sf.read(io.BytesIO(response.read()))
```

In [16]:
```
fig, ax = plt.subplots(figsize=(20,4))
ax.plot(noise_data)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x7fb13985d9f0>]
```



In [17]:
```
IPython.display.Audio(data=noise_data, rate=noise_rate)
```

Out[17]:

0:00 / 0:00

add noise to data

In [18]:
```
snr = 2 # signal to noise ratio
noise_clip = noise_data/snr
audio_clip_cafe = data + noise_clip
```

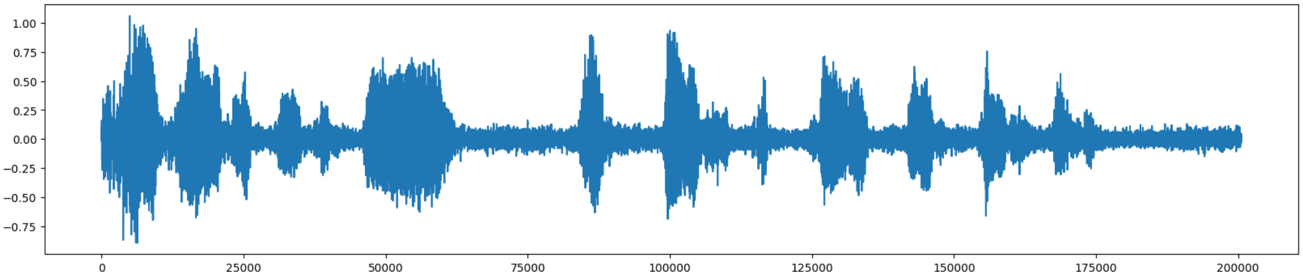
plot noisy data

In [19]:

```python
fig, ax = plt.subplots(figsize=(20,4))
ax.plot(audio_clip_cafe)
IPython.display.Audio(data=audio_clip_cafe, rate=noise_rate)
```

Out[19]:

0:00 / 0:00



Stationary remove noise

In [20]:

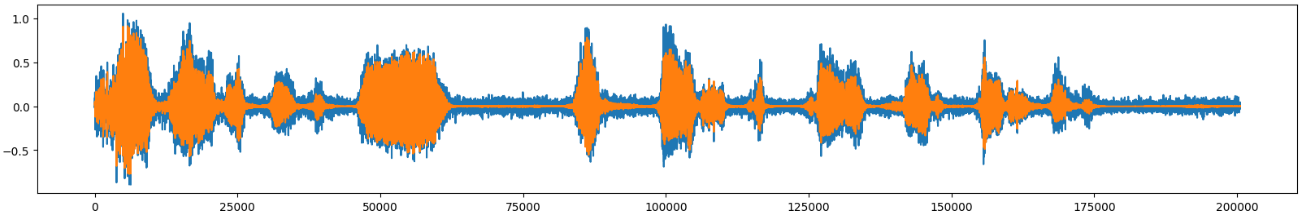
```python
reduced_noise = nr.reduce_noise(y = audio_clip_cafe, sr=rate, y_noise = noise_clip, n_std_thresh_stationary=1.5,stationary=T
```

In [21]:

```python
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(audio_clip_cafe)
ax.plot(reduced_noise)
```

Out[21]:

```
[<matplotlib.lines.Line2D at 0x7fb1394e5de0>]
```



In [22]:

```python
IPython.display.Audio(data=reduced_noise, rate=rate)
```

Out[22]:

0:00 / 0:00

Non-stationary noise reduction

In [23]:

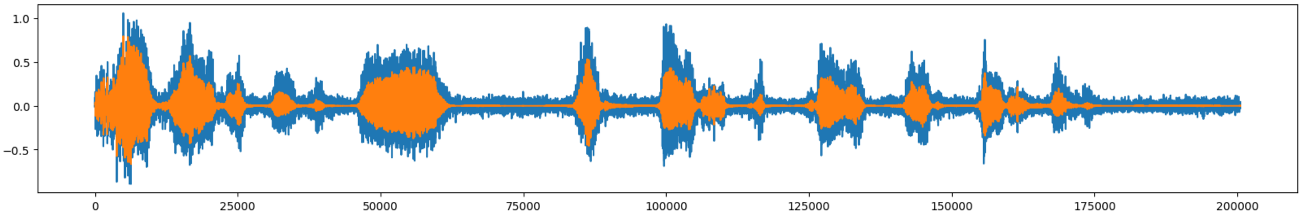
```python
reduced_noise = nr.reduce_noise(y = audio_clip_cafe, sr=rate, thresh_n_mult_nonstationary=2,stationary=False)
```

In [24]:

```python
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(audio_clip_cafe)
ax.plot(reduced_noise, alpha = 1)
IPython.display.Audio(data=reduced_noise, rate=rate)
```

Out[24]:

0:00 / 0:00



In [25]:

```
IPython.display.Audio(data=reduced_noise, rate=rate)
```

Out[25]:

0:00 / 0:00

ensure that noise reduction does not cause distortion when prop_decrease == 0

In [26]:

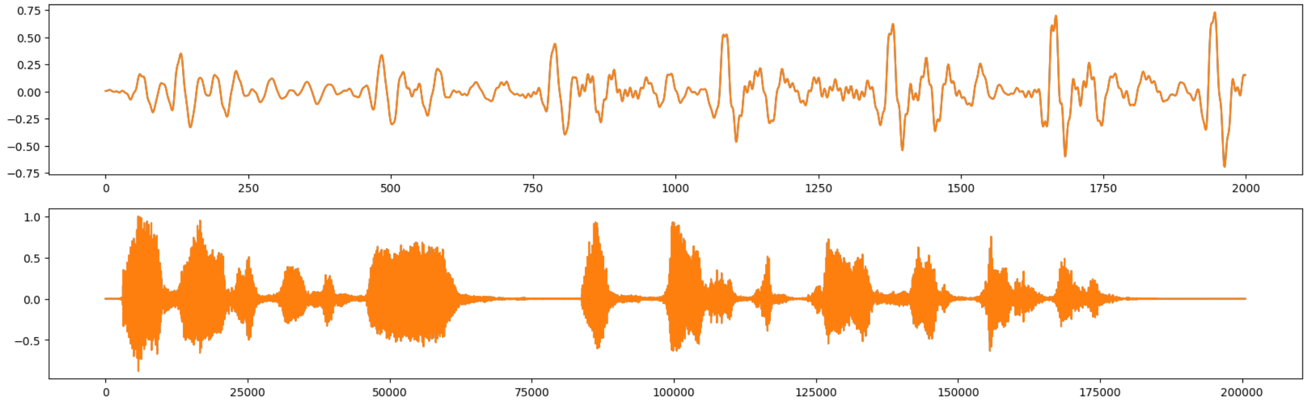
```
noise_reduced = nr.reduce_noise(y=data, sr=rate, prop_decrease=0, stationary=False)
```

In [27]:

```
fig, axs = plt.subplots(nrows=2, figsize=(20,6))
axs[0].plot(data[3000:5000])
axs[0].plot(noise_reduced[3000:5000])
axs[1].plot(data)
axs[1].plot(noise_reduced)
```

Out[27]:

```
[<matplotlib.lines.Line2D at 0x7fb1398850c0>]
```



In [28]:

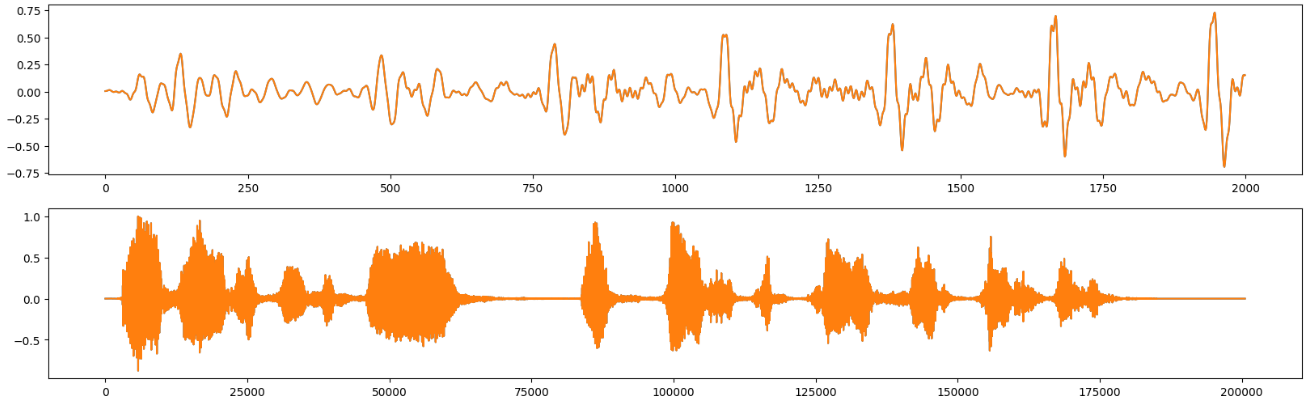
```
noise_reduced = nr.reduce_noise(y=data, sr=rate, prop_decrease=0, stationary=False)
```

In [29]:

```
fig, axs = plt.subplots(nrows=2, figsize=(20,6))
axs[0].plot(data[3000:5000])
axs[0].plot(noise_reduced[3000:5000])
axs[1].plot(data)
axs[1].plot(noise_reduced)
```

Out[29]:

```
[<matplotlib.lines.Line2D at 0x7fb139cda7a0>]
```



Reduce noise over batches in parallel on long signal

In [30]:

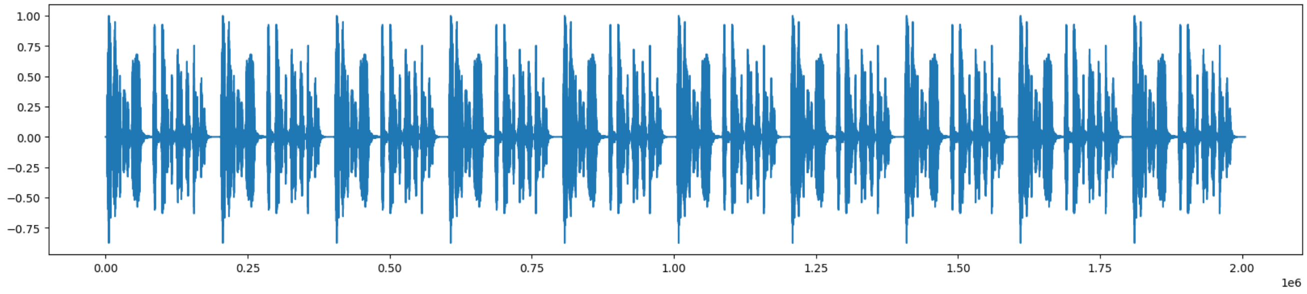
```
long_data = np.tile(data, 10)
len(long_data)/rate
```

Out[30]:

```
45.47437641723356
```

In [31]:
```
fig, ax = plt.subplots(figsize=(20,4))
ax.plot(long_data)
```

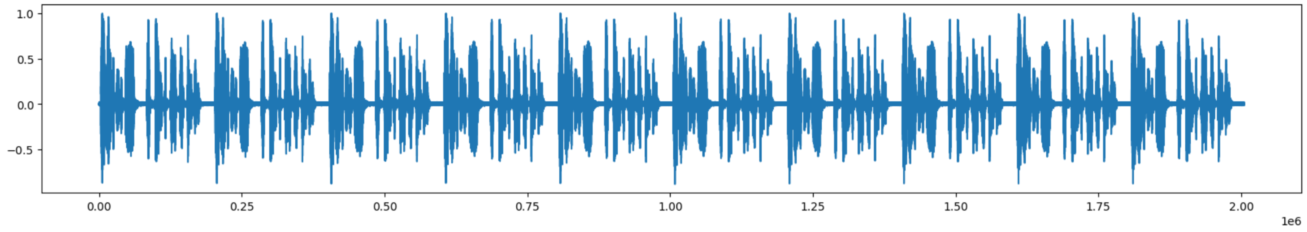
Out[31]:
```
[<matplotlib.lines.Line2D at 0x7fb139a9d3f0>]
```



In [32]:
```
noise = band_limited_noise(min_freq=2000, max_freq = 12000, samples=len(long_data), samplerate=rate)*10
audio_clip_band_limited = long_data+noise
```

In [33]:
```
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(audio_clip_band_limited)
```

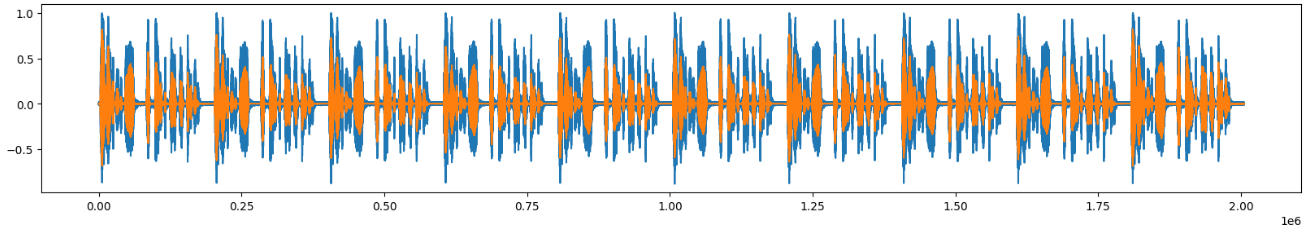
Out[33]:
```
[<matplotlib.lines.Line2D at 0x7fb139937b50>]
```



In [34]:
```
reduced_noise = nr.reduce_noise(
    y=audio_clip_band_limited,
    sr=rate,
    thresh_n_mult_nonstationary=2,
    stationary=False,
    n_jobs=2,
)
```

In [35]:
```
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(audio_clip_band_limited)
ax.plot(reduced_noise)
```

Out[35]:
```
[<matplotlib.lines.Line2D at 0x7fb1390a48e0>]
```



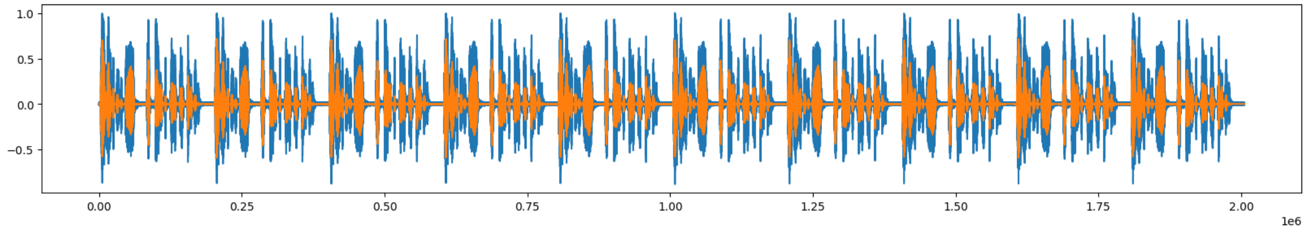
In [36]:
```
reduced_noise = nr.reduce_noise(
    y=audio_clip_band_limited,
    sr=rate,
    thresh_n_mult_nonstationary=2,
    stationary=True,
    n_jobs=2,
)
```

In [37]:

```python
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(audio_clip_band_limited)
ax.plot(reduced_noise)
```

Out[37]:

```
[<matplotlib.lines.Line2D at 0x7fb13951fca0>]
```



Reduce noise on only a subset of a long clip

In [38]:

```python
from noisereduce.noisereduce import SpectralGateStationary
```

In [39]:

```python
sg = SpectralGateStationary(
    y = data,
    sr = rate,
    y_noise=None,
    prop_decrease=1.0,
    time_constant_s=2.0,
    freq_mask_smooth_hz=500,
    time_mask_smooth_ms=50,
    n_std_thresh_stationary=1.5,
    tmp_folder=None,
    chunk_size=600000,
    padding=30000,
    n_fft=1024,
    win_length=None,
    hop_length=None,
    clip_noise_stationary=True,
    use_tqdm=False,
    n_jobs=1,
)
```

In [40]:

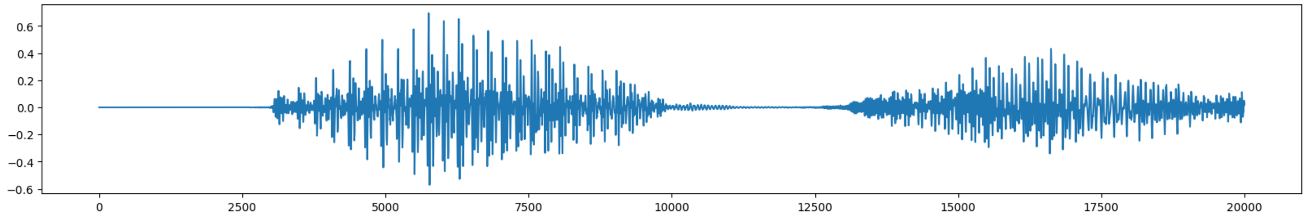
```python
subset_noise_reduce = sg.get_traces(start_frame = 10000, end_frame = 20000)
```

In [41]:

```python
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(subset_noise_reduce)
```

Out[41]:

```
[<matplotlib.lines.Line2D at 0x7fb139a273d0>]
```



Multichannel noise

In [42]:

```python
audio_clip_cafe_2_channel = np.vstack([audio_clip_cafe, audio_clip_cafe])
audio_clip_cafe_2_channel.shape
```

Out[42]:

```
(2, 200542)
```

In [43]:

```python
reduced_noise = nr.reduce_noise(y = audio_clip_cafe_2_channel, sr=rate, n_std_thresh_stationary=1.5,stationary=True)
```

In [44]:

```python
reduced_noise.shape
```

Out[44]:

```
(2, 200542)
```

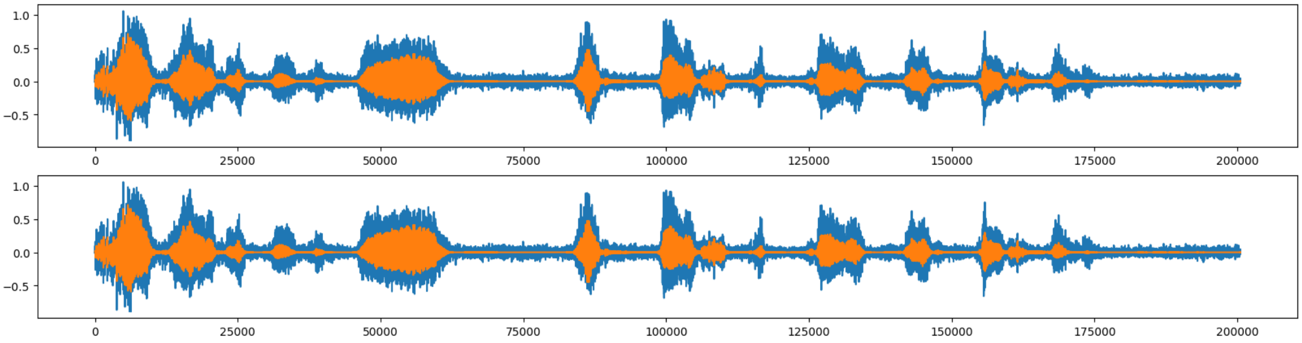
In [45]:

```
fig, axs = plt.subplots(nrows= 2, figsize=(20,5))
axs[0].plot(audio_clip_cafe_2_channel[0])
axs[1].plot(audio_clip_cafe_2_channel[1])

axs[0].plot(reduced_noise[0])
axs[1].plot(reduced_noise[1])
```

Out[45]:

```
[<matplotlib.lines.Line2D at 0x7fb13991ff40>]
```



In [46]:

```
IPython.display.Audio(data=reduced_noise, rate=rate)
```

Out[46]:

0:00 / 0:00

In [47]:

```
reduced_noise = nr.reduce_noise(y = audio_clip_cafe, sr=rate, thresh_n_mult_nonstationary=2,stationary=False)
```

In [48]:

```
reduced_noise.shape
```

Out[48]:

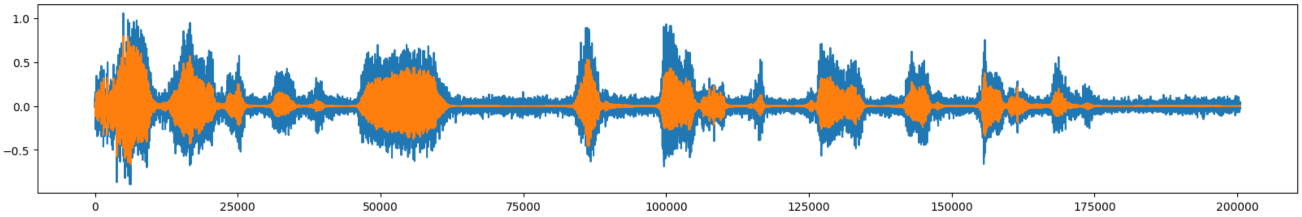
```
(200542,)
```

In [50]:

```
fig, ax = plt.subplots(figsize=(20,3))
ax.plot(audio_clip_cafe)
ax.plot(reduced_noise, alpha = 1)
IPython.display.Audio(data=reduced_noise, rate=rate)
```

Out[50]:

0:00 / 0:00



In [49]:

```

```