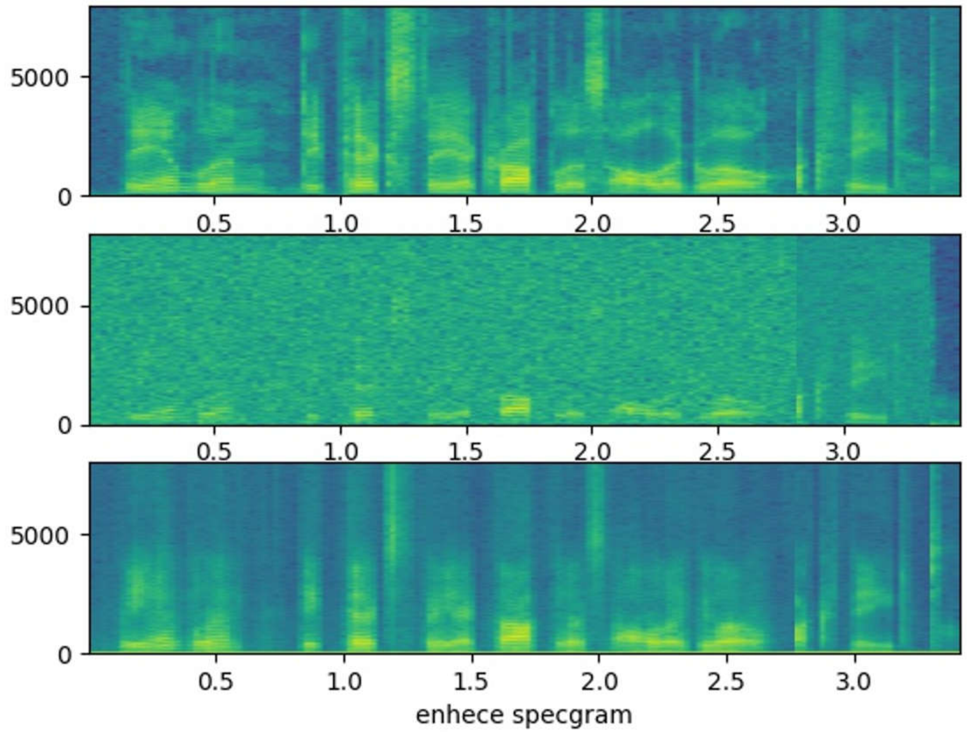
语音增强-DNN频谱映射

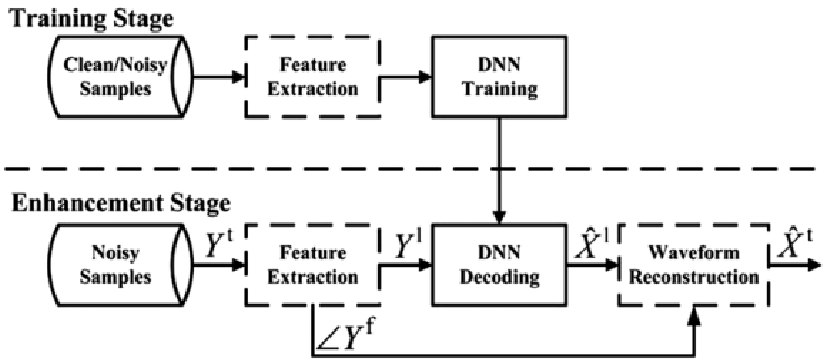
Speech Enhancement- DNN based Spectrum Mapping

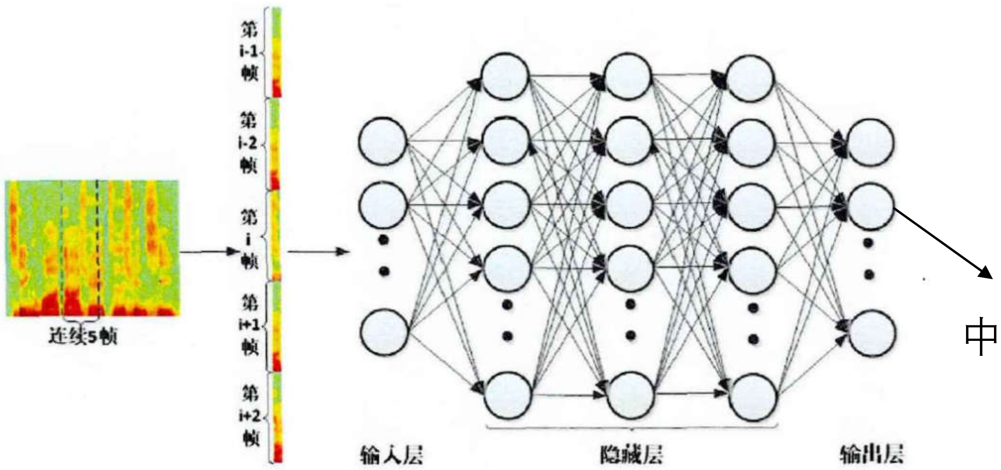


于泓
鲁东大学
信息与电气工程学院

2021.8.6

DNN 频谱映射（ DNN based Spectrum Mapping ）

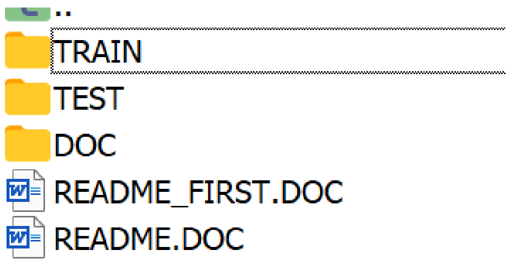




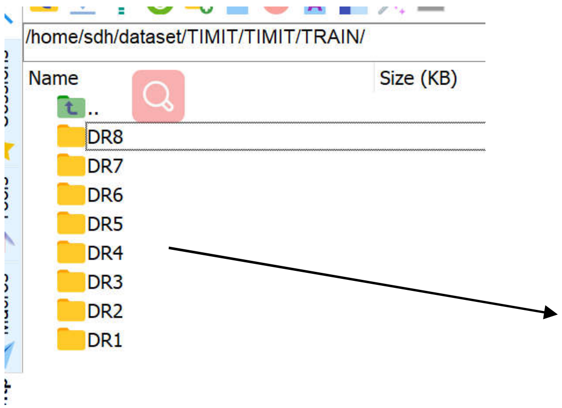
中间帧

使用MSE-loss

数据库 TIMIT



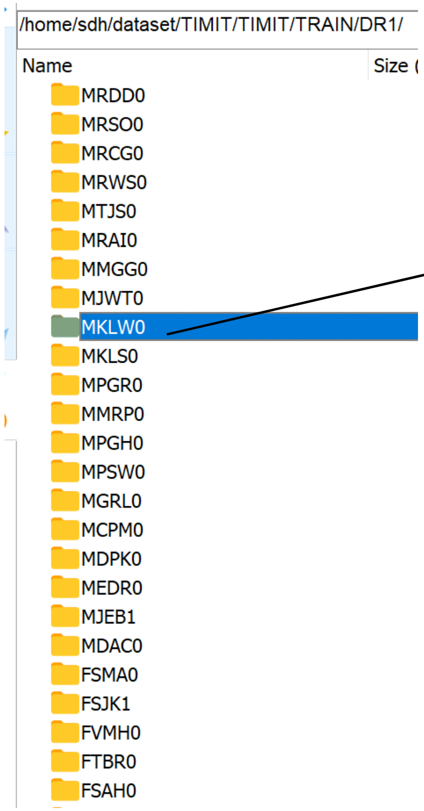
..
TRAIN
TEST
DOC
README_FIRST.DOC
README.DOC



/home/sdh/dataset/TIMIT/TIMIT/TRAIN/

Name	Size (KB)
..	
DR8	
DR7	
DR6	
DR5	
DR4	
DR3	
DR2	
DR1	

地区

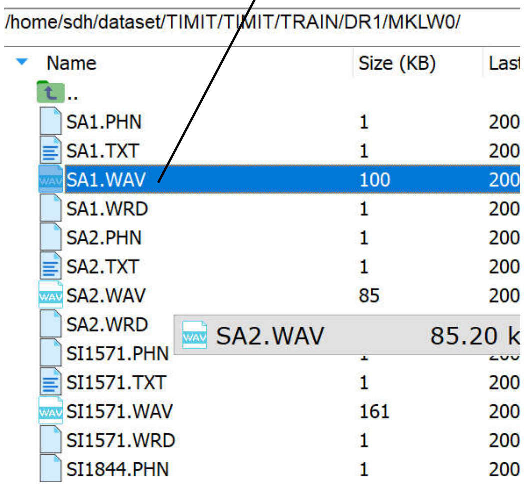


/home/sdh/dataset/TIMIT/TIMIT/TRAIN/DR1/

Name	Size (
MRDD0	
MRSO0	
MRCG0	
MRWS0	
MTJS0	
MRAI0	
MMGG0	
MJWT0	
MKLW0	
MKLS0	
MPGR0	
MMRP0	
MPGH0	
MPSW0	
MGRL0	
MCPM0	
MDPK0	
MEDR0	
MJEB1	
MDAC0	
FSMA0	
FSJK1	
FVMH0	
FTBR0	
FSAH0	

说话人

不同的语音



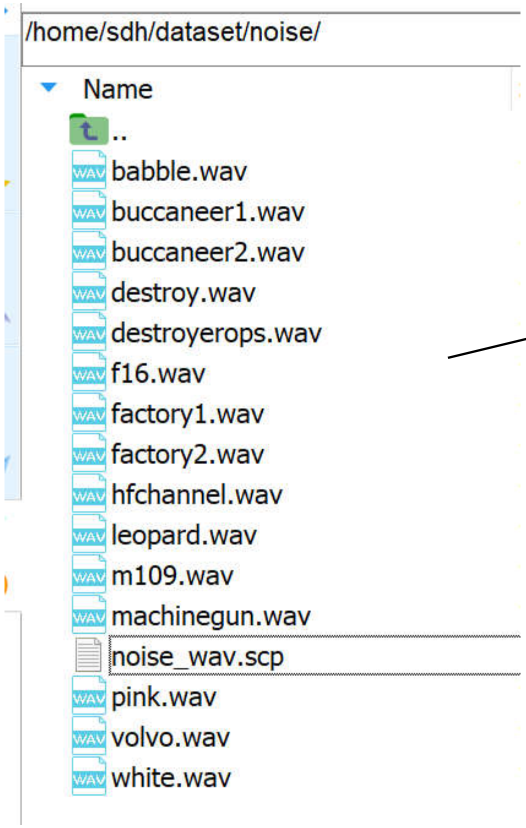
/home/sdh/dataset/TIMIT/TIMIT/TRAIN/DR1/MKLW0/

Name	Size (KB)	Last
..		
SA1.PHN	1	200
SA1.TXT	1	200
SA1.WAV	100	200
SA1.WRD	1	200
SA2.PHN	1	200
SA2.TXT	1	200
SA2.WAV	85	200
SA2.WRD		
SI1571.PHN	1	200
SI1571.TXT	1	200
SI1571.WAV	161	200
SI1571.WRD	1	200
SI1844.PHN	1	200

SA2.WAV 85.20 k

Fs = 16k

噪声数据库 Noise-92

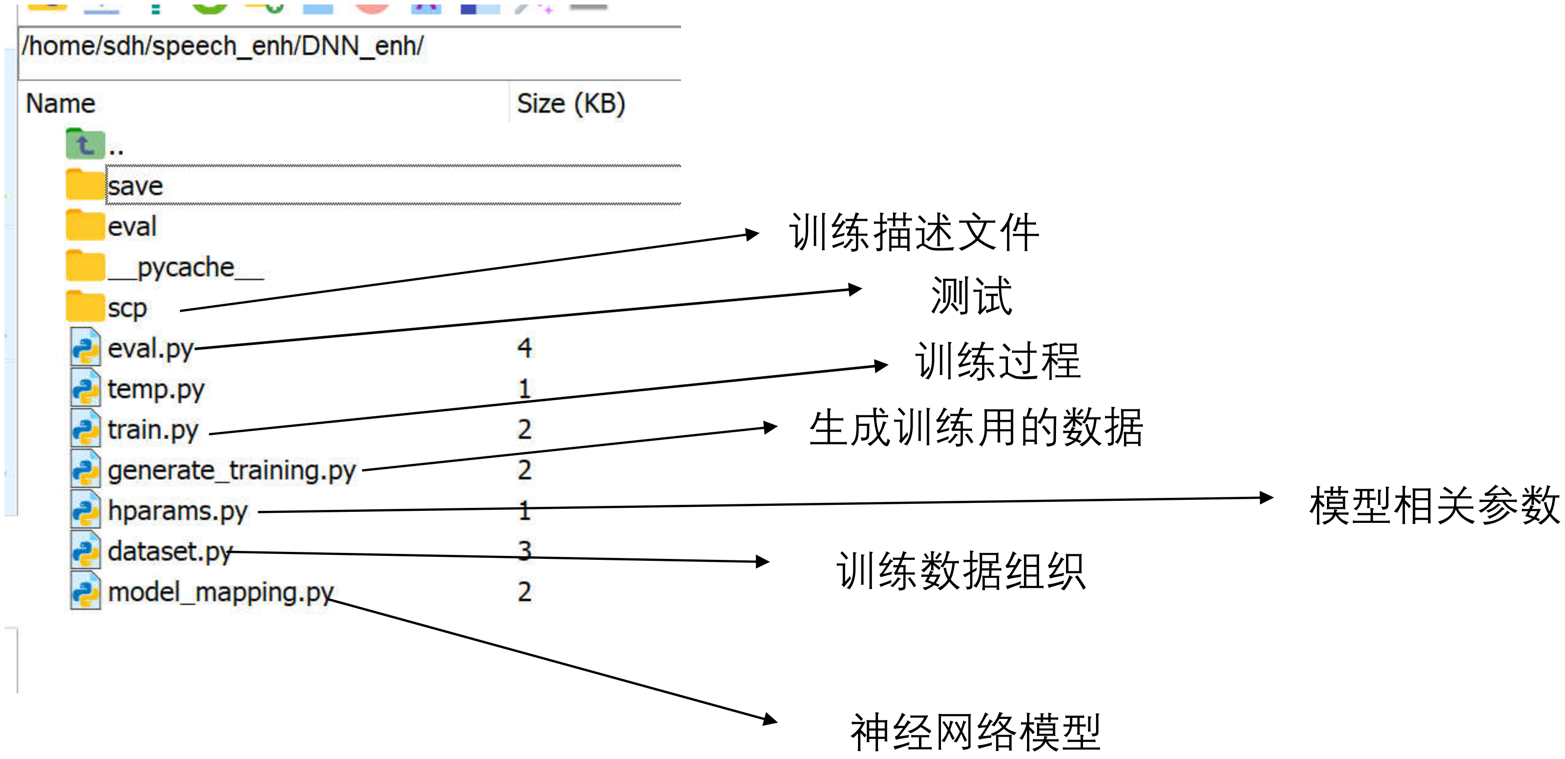


/home/sdh/dataset/noise/

```
Name
..
babble.wav
buccaneer1.wav
buccaneer2.wav
destroy.wav
destroyerops.wav
f16.wav
factory1.wav
factory2.wav
hfchannel.wav
leopard.wav
m109.wav
machinegun.wav
noise_wav.scp
pink.wav
volvo.wav
white.wav
```

15种噪声

工程目录



/home/sdh/speech_enh/DNN_enh/

Name	Size (KB)
..	
save	
eval	
__pycache__	
scp	
eval.py	4
temp.py	1
train.py	2
generate_training.py	2
hparams.py	1
dataset.py	3
model_mapping.py	2

训练描述文件

测试

训练过程

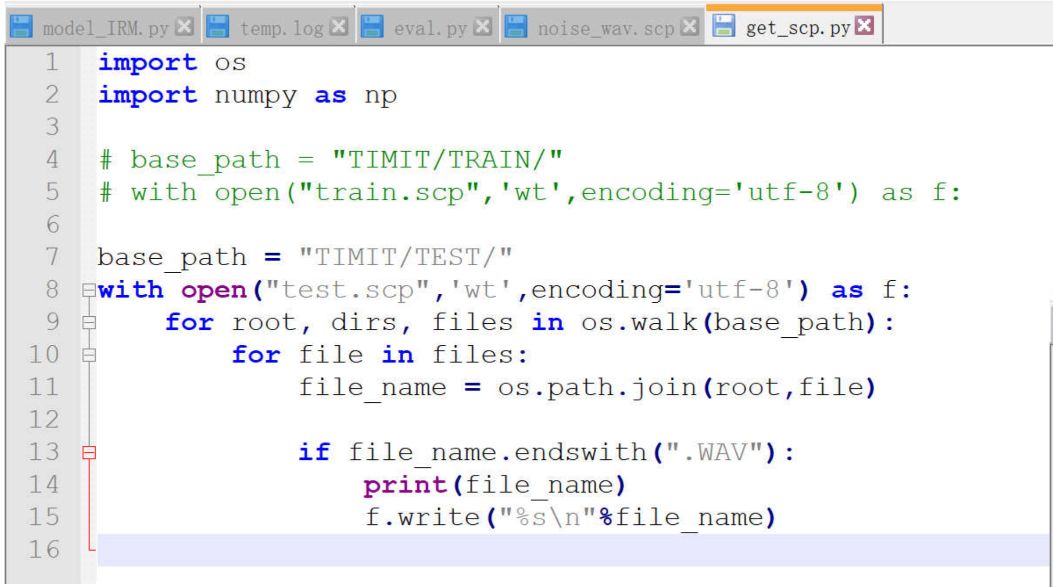
生成训练用的数据

模型相关参数

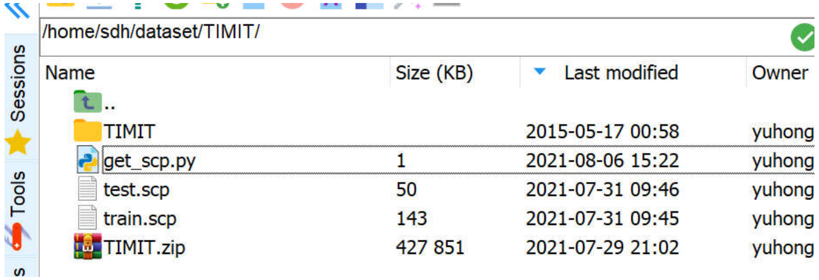
训练数据组织

神经网络模型

数据准备

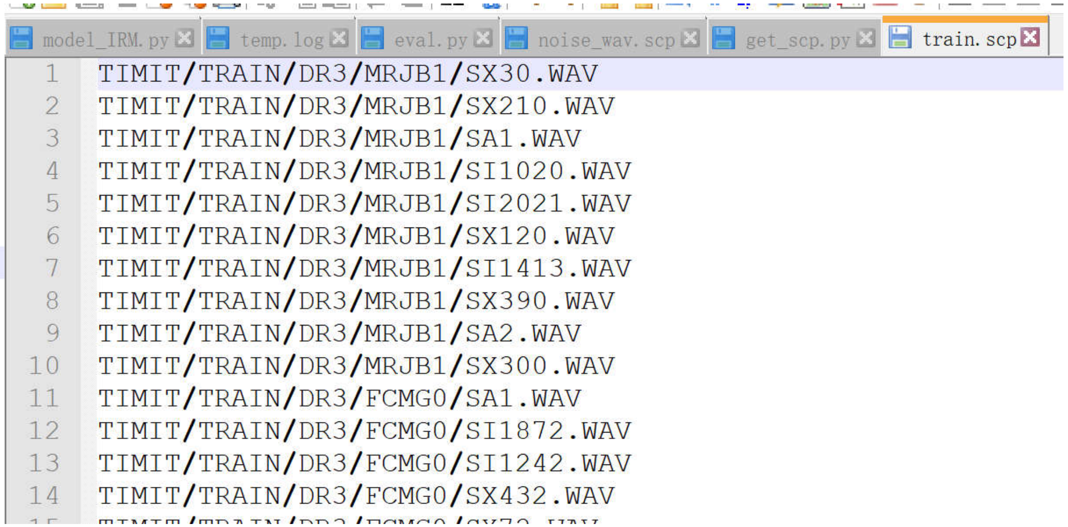


```python
import os
import numpy as np

# base_path = "TIMIT/TRAIN/"
# with open("train.scp",'wt',encoding='utf-8') as f:

base_path = "TIMIT/TEST/"
with open("test.scp",'wt',encoding='utf-8') as f:
    for root, dirs, files in os.walk(base_path):
        for file in files:
            file_name = os.path.join(root,file)

            if file_name.endswith(".WAV"):
                print(file_name)
                f.write("%s\n"%file_name)

```



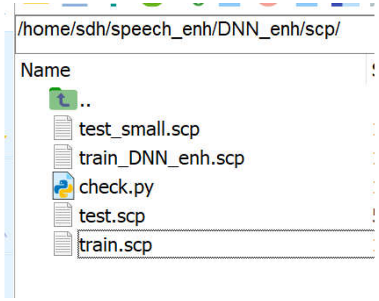
/home/sdh/dataset/TIMIT/

Name	Size (KB)	Last modified	Owner
..			
TIMIT		2015-05-17 00:58	yuhong
get_scp.py	1	2021-08-06 15:22	yuhong
test.scp	50	2021-07-31 09:46	yuhong
train.scp	143	2021-07-31 09:45	yuhong
TIMIT.zip	427 851	2021-07-29 21:02	yuhong



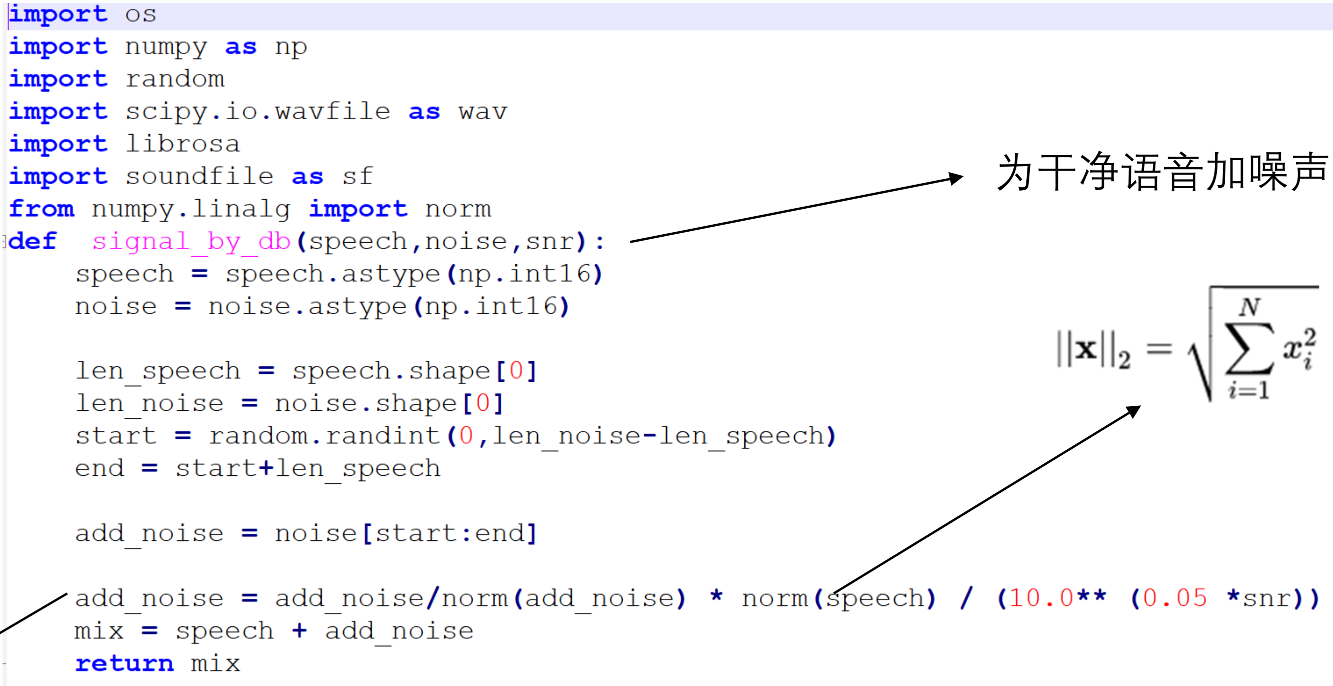
```
TIMIT/TRAIN/DR3/MRJB1/SX30.WAV
TIMIT/TRAIN/DR3/MRJB1/SX210.WAV
TIMIT/TRAIN/DR3/MRJB1/SA1.WAV
TIMIT/TRAIN/DR3/MRJB1/SI1020.WAV
TIMIT/TRAIN/DR3/MRJB1/SI2021.WAV
TIMIT/TRAIN/DR3/MRJB1/SX120.WAV
TIMIT/TRAIN/DR3/MRJB1/SI1413.WAV
TIMIT/TRAIN/DR3/MRJB1/SX390.WAV
TIMIT/TRAIN/DR3/MRJB1/SA2.WAV
TIMIT/TRAIN/DR3/MRJB1/SX300.WAV
TIMIT/TRAIN/DR3/FCMG0/SA1.WAV
TIMIT/TRAIN/DR3/FCMG0/SI1872.WAV
TIMIT/TRAIN/DR3/FCMG0/SI1242.WAV
TIMIT/TRAIN/DR3/FCMG0/SX432.WAV
```

生成noisy数据



/home/sdh/speech_enh/DNN_enh/scp/

```
..
test_small.scp
train_DNN_enh.scp
check.py
test.scp
train.scp
```

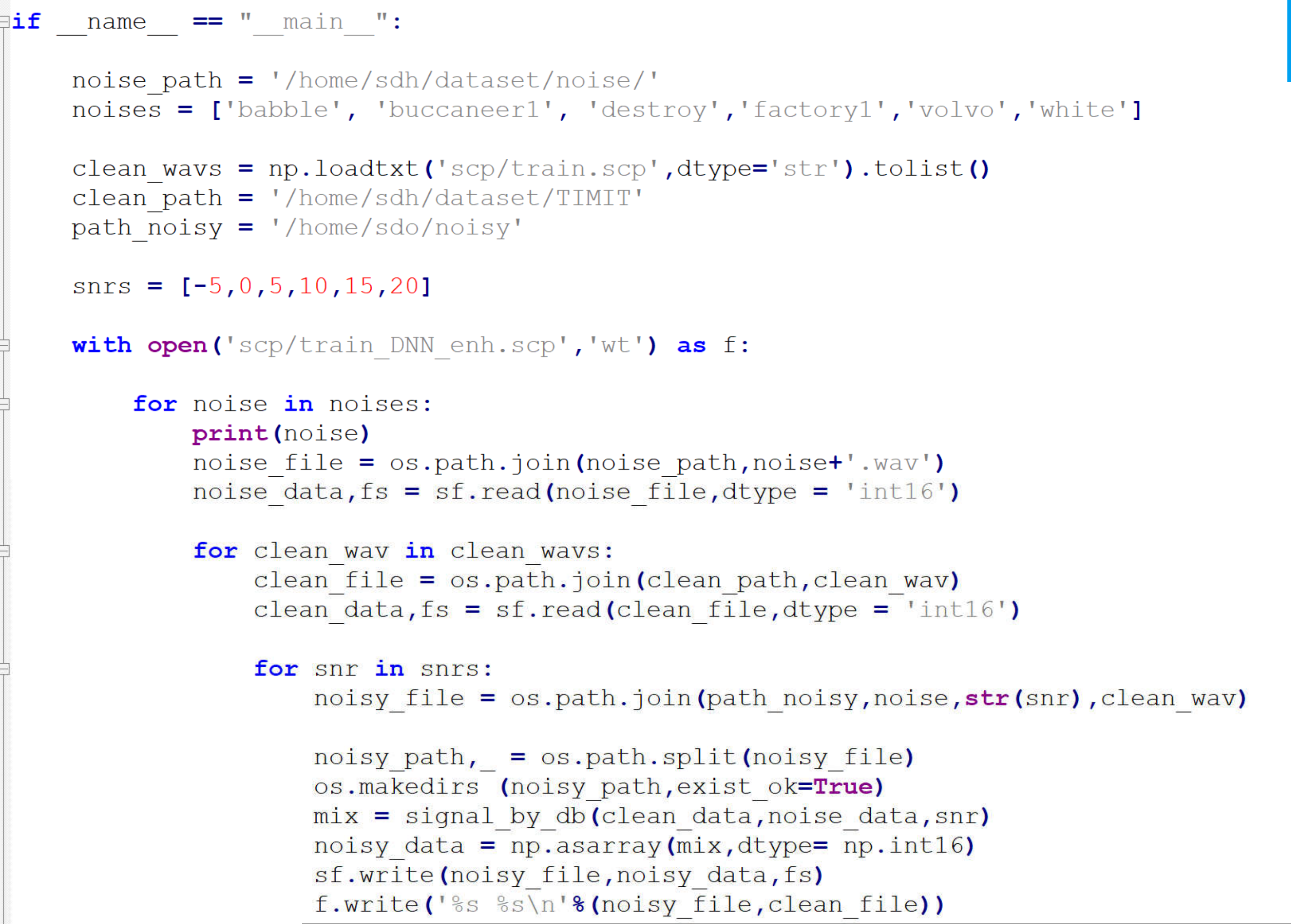


```
import os
import numpy as np
import random
import scipy.io.wavfile as wav
import librosa
import soundfile as sf
from numpy.linalg import norm
def  signal_by_db(speech,noise,snr):
    speech = speech.astype(np.int16)
    noise = noise.astype(np.int16)

    len_speech = speech.shape[0]
    len_noise = noise.shape[0]
    start = random.randint(0,len_noise-len_speech)
    end = start+len_speech

    add_noise = noise[start:end]

    add_noise = add_noise/norm(add_noise) * norm(speech) / (10.0** (0.05 *snr))
    mix = speech + add_noise
    return mix
```

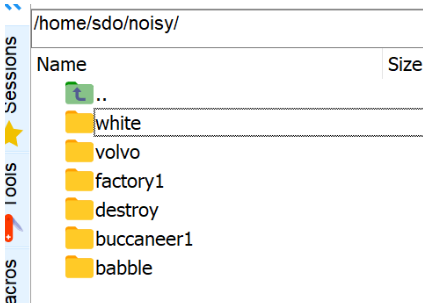
为干净语音加噪声

$$||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$$

$$\mathrm{SNR(dB)} = 10\log_{10}\left(\frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}}\right) = 20\log_{10}\left(\frac{A_{\mathrm{signal}}}{A_{\mathrm{noise}}}\right)$$

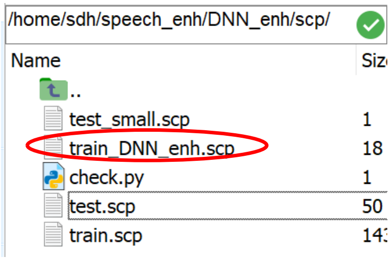


```
if __name__ == "__main__":

    noise_path = '/home/sdh/dataset/noise/'
    noises = ['babble', 'buccaneer1', 'destroy','factory1','volvo','white']

    clean_wavs = np.loadtxt('scp/train.scp',dtype='str').tolist()
    clean_path = '/home/sdh/dataset/TIMIT'
    path_noisy = '/home/sdo/noisy'

    snrs = [-5,0,5,10,15,20]

    with open('scp/train_DNN_enh.scp','wt') as f:

        for noise in noises:
            print(noise)
            noise_file = os.path.join(noise_path,noise+'.wav')
            noise_data,fs = sf.read(noise_file,dtype = 'int16')

            for clean_wav in clean_wavs:
                clean_file = os.path.join(clean_path,clean_wav)
                clean_data,fs = sf.read(clean_file,dtype = 'int16')

                for snr in snrs:
                    noisy_file = os.path.join(path_noisy,noise,str(snr),clean_wav)

                    noisy_path,_ = os.path.split(noisy_file)
                    os.makedirs  (noisy_path,exist_ok=True)
                    mix = signal_by_db(clean_data,noise_data,snr)
                    noisy_data = np.asarray(mix,dtype= np.int16)
                    sf.write(noisy_file,noisy_data,fs)
                    f.write('%s %s\n'%(noisy_file,clean_file))
```



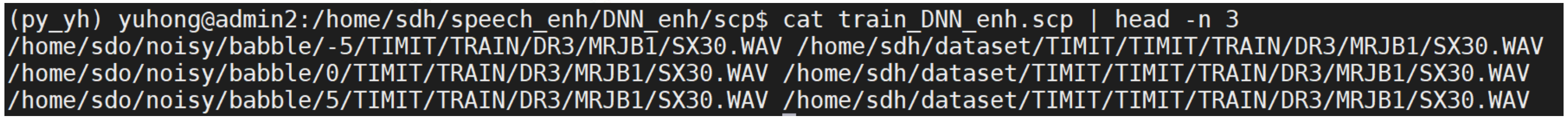
/home/sdo/noisy/

Name	Size
..	
white	
volvo	
factory1	
destroy	
buccaneer1	
babble	



/home/sdh/speech_enh/DNN_enh/scp/

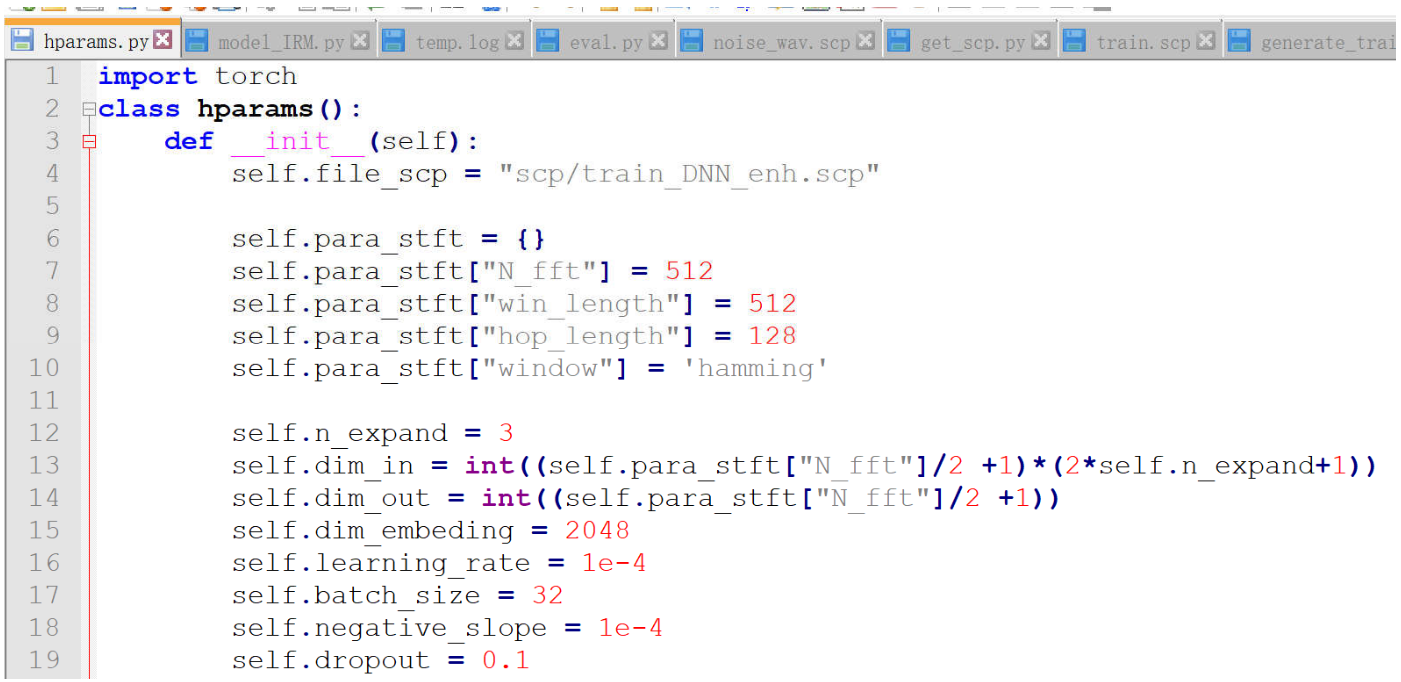
Name	Siz
..	
test_small.scp	1
train_DNN_enh.scp	18
check.py	1
test.scp	50
train.scp	14



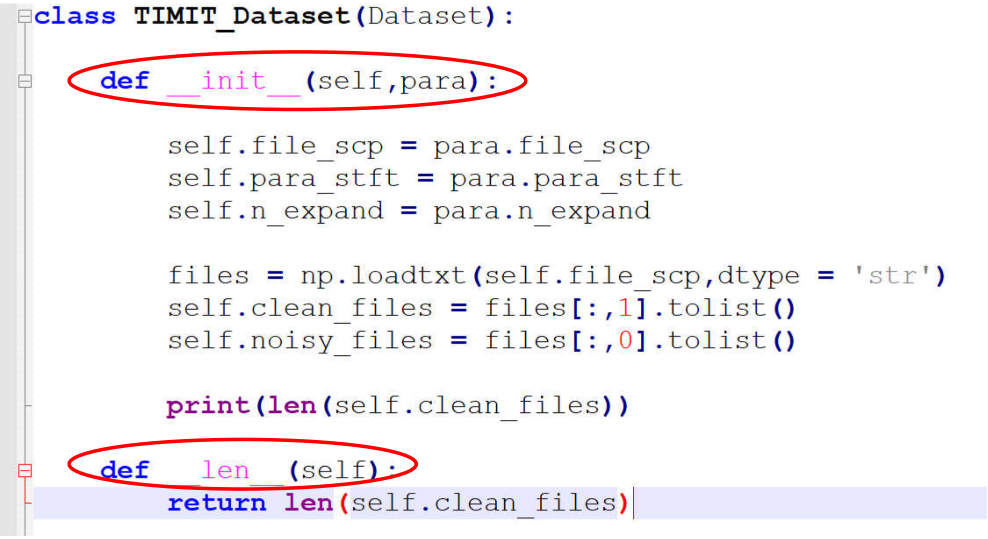
```
(py_yh) yuhong@admin2:/home/sdh/speech_enh/DNN_enh/scp$ cat train_DNN_enh.scp | head -n 3
/home/sdo/noisy/babble/-5/TIMIT/TRAIN/DR3/MRJB1/SX30.WAV /home/sdh/dataset/TIMIT/TIMIT/TRAIN/DR3/MRJB1/SX30.WAV
/home/sdo/noisy/babble/0/TIMIT/TRAIN/DR3/MRJB1/SX30.WAV /home/sdh/dataset/TIMIT/TIMIT/TRAIN/DR3/MRJB1/SX30.WAV
/home/sdo/noisy/babble/5/TIMIT/TRAIN/DR3/MRJB1/SX30.WAV /home/sdh/dataset/TIMIT/TIMIT/TRAIN/DR3/MRJB1/SX30.WAV
```

数据加载

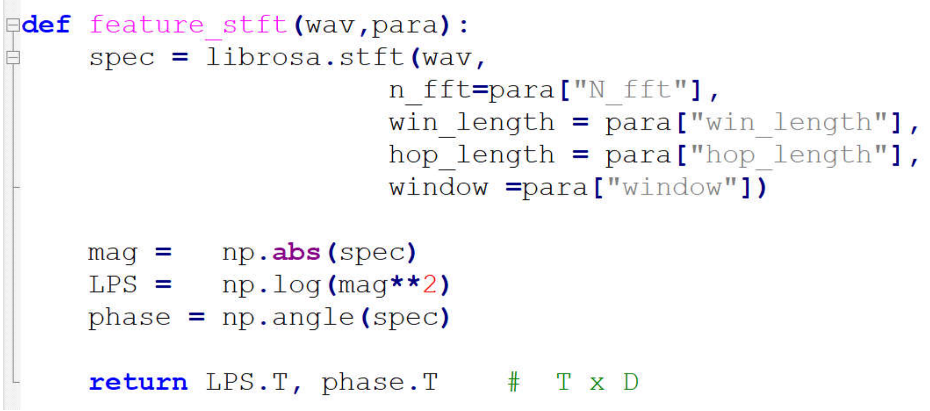
```python
import os
import torch
import numpy as np
from torch.utils.data import Dataset,DataLoader
from hparams import hparams
import librosa
import random
import soundfile as sf
```



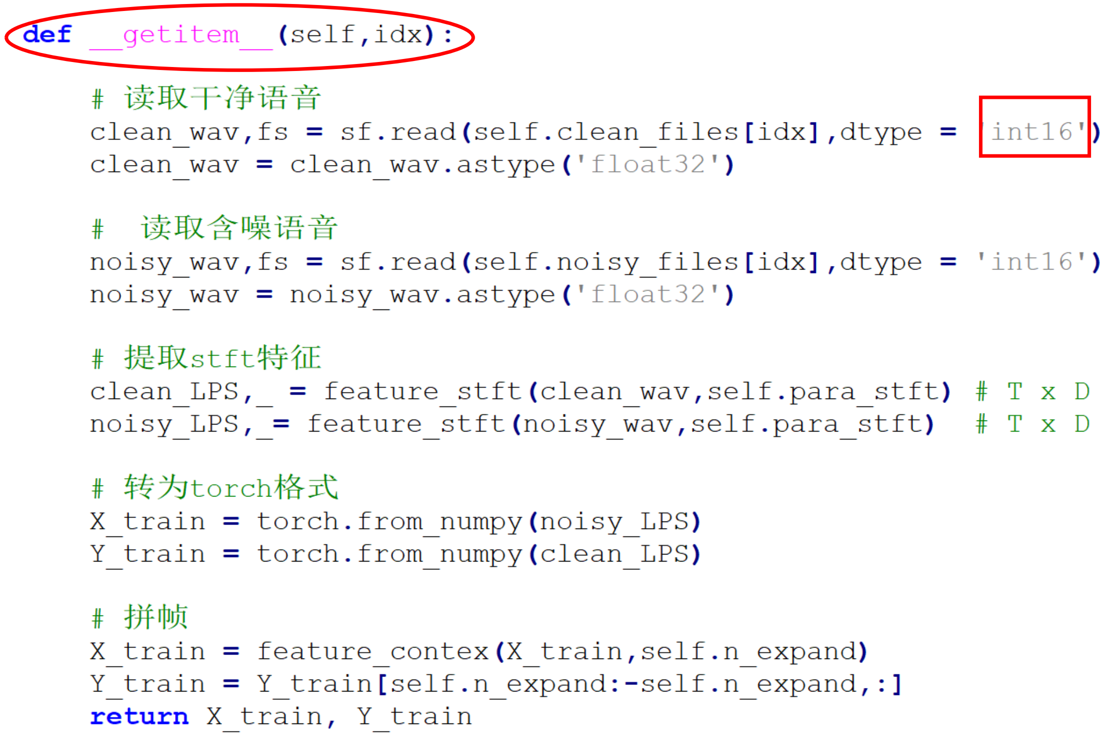
```python
import torch
class hparams():
    def __init__(self):
        self.file_scp = "scp/train_DNN_enh.scp"

        self.para_stft = {}
        self.para_stft["N_fft"] = 512
        self.para_stft["win_length"] = 512
        self.para_stft["hop_length"] = 128
        self.para_stft["window"] = 'hamming'

        self.n_expand = 3
        self.dim_in = int((self.para_stft["N_fft"]/2 +1)*(2*self.n_expand+1))
        self.dim_out = int((self.para_stft["N_fft"]/2 +1))
        self.dim_embeding = 2048
        self.learning_rate = 1e-4
        self.batch_size = 32
        self.negative_slope = 1e-4
        self.dropout = 0.1
```



```
class TIMIT_Dataset(Dataset):

    def __init__(self,para):

        self.file_scp = para.file_scp
        self.para_stft = para.para_stft
        self.n_expand = para.n_expand

        files = np.loadtxt(self.file_scp,dtype = 'str')
        self.clean_files = files[:,1].tolist()
        self.noisy_files = files[:,0].tolist()

        print(len(self.clean_files))

    def __len__(self):
        return len(self.clean_files)
```

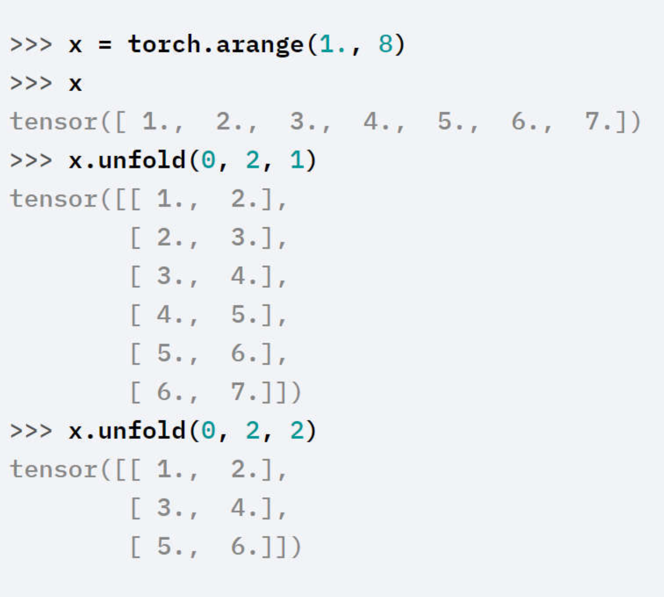


```
def feature_stft(wav,para):
    spec = librosa.stft(wav,
                        n_fft=para["N_fft"],
                        win_length = para["win_length"],
                        hop_length = para["hop_length"],
                        window =para["window"])

    mag =   np.abs(spec)
    LPS =   np.log(mag**2)
    phase = np.angle(spec)

    return LPS.T, phase.T    #  T x D
```

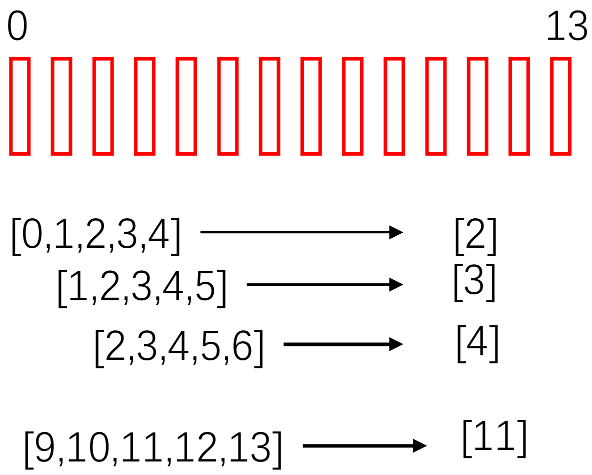


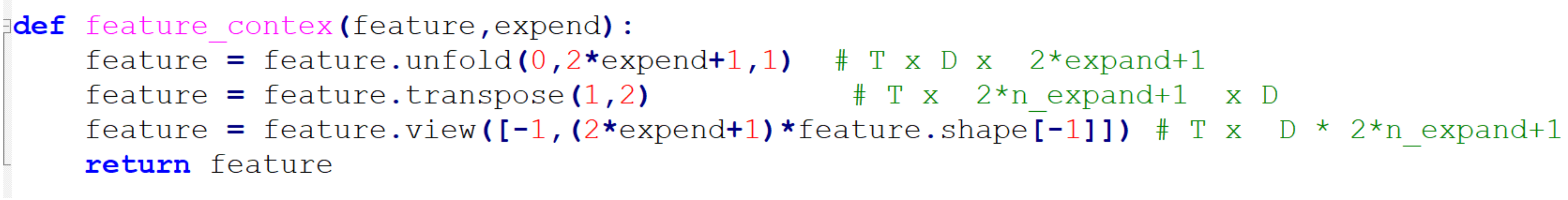
```
def __getitem__(self,idx):

    # 读取干净语音
    clean_wav,fs = sf.read(self.clean_files[idx],dtype = 'int16')
    clean_wav = clean_wav.astype('float32')

    #  读取含噪语音
    noisy_wav,fs = sf.read(self.noisy_files[idx],dtype = 'int16')
    noisy_wav = noisy_wav.astype('float32')

    # 提取stft特征
    clean_LPS,_ = feature_stft(clean_wav,self.para_stft) # T x D
    noisy_LPS,_= feature_stft(noisy_wav,self.para_stft)  # T x D

    # 转为torch格式
    X_train = torch.from_numpy(noisy_LPS)
    Y_train = torch.from_numpy(clean_LPS)

    # 拼帧
    X_train = feature_contex(X_train,self.n_expand)
    Y_train = Y_train[self.n_expand:-self.n_expand,:]
    return X_train, Y_train
```

Tensor.unfold(*dimension*, *size*, *step*) → Tensor



```
>>> x = torch.arange(1., 8)
>>> x
tensor([ 1.,  2.,  3.,  4.,  5.,  6.,  7.])
>>> x.unfold(0, 2, 1)
tensor([[ 1.,  2.],
        [ 2.,  3.],
        [ 3.,  4.],
        [ 4.,  5.],
        [ 5.,  6.],
        [ 6.,  7.]])
>>> x.unfold(0, 2, 2)
tensor([[ 1.,  2.],
        [ 3.,  4.],
        [ 5.,  6.]])
```

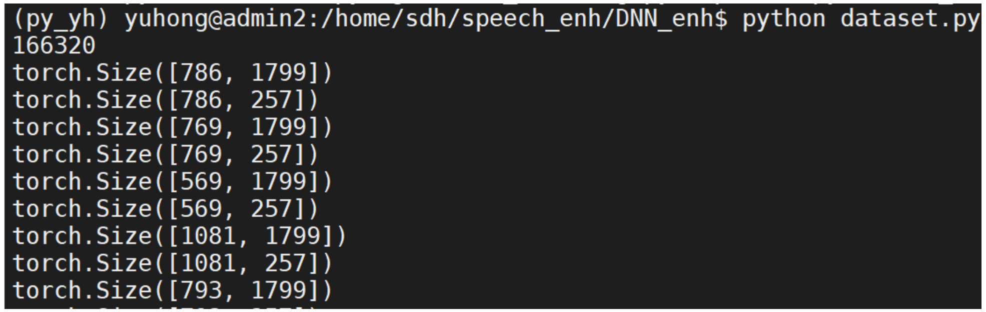




```
def feature_contex(feature,expend):
    feature = feature.unfold(0,2*expend+1,1)  # T x D x  2*expand+1
    feature = feature.transpose(1,2)            # T x  2*n_expand+1  x D
    feature = feature.view([-1,(2*expend+1)*feature.shape[-1]]) # T x  D * 2*n_expand+1
    return feature
```

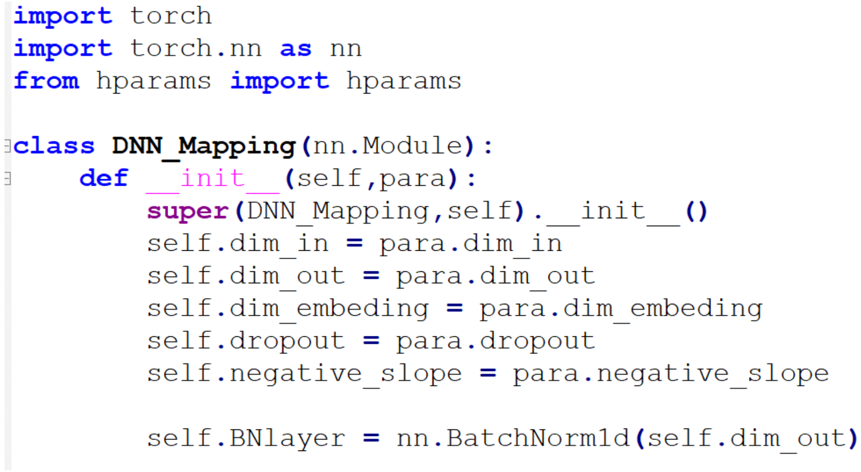
```
def my_collect(batch):
    batch_X = [item[0] for item in batch]
    batch_Y = [item[1] for item in batch]
    batch_X = torch.cat(batch_X,0)
    batch_Y = torch.cat(batch_Y,0)
    return[batch_X.float(),batch_Y.float()]
```

```
if __name__ == '__main__':

    # 数据加载测试
    para = hparams()

    m_Dataset= TIMIT_Dataset(para)

    m_DataLoader = DataLoader(m_Dataset,batch_size = 2,shuffle = True, num_workers = 4, collate_fn = my_collect)

    for i_batch, sample_batch in enumerate(m_DataLoader):
        train_X = sample_batch[0]
        train_Y = sample_batch[1]
        print(train_X.shape)
        print(train_Y.shape)
```

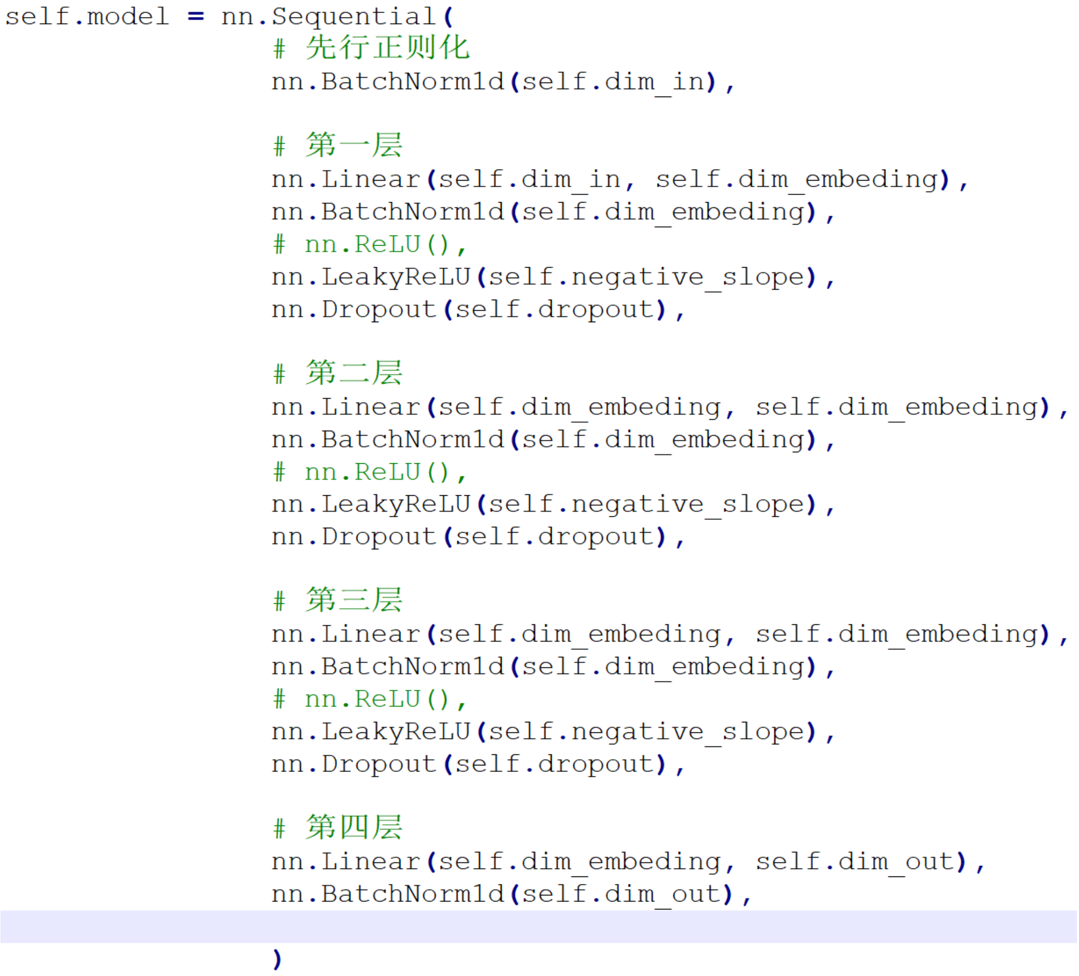


```
(py_yh) yuhong@admin2:/home/sdh/speech_enh/DNN_enh$ python dataset.py
166320
torch.Size([786, 1799])
torch.Size([786, 257])
torch.Size([769, 1799])
torch.Size([769, 257])
torch.Size([569, 1799])
torch.Size([569, 257])
torch.Size([1081, 1799])
torch.Size([1081, 257])
torch.Size([793, 1799])
```

神经网络模型

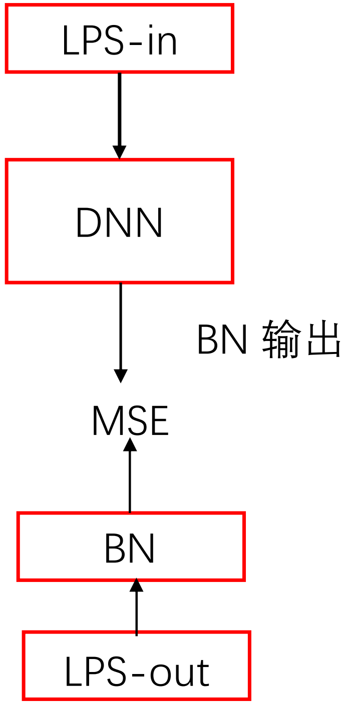


```python
import torch
import torch.nn as nn
from hparams import hparams

class DNN_Mapping(nn.Module):
    def __init__(self,para):
        super(DNN_Mapping,self).__init__()
        self.dim_in = para.dim_in
        self.dim_out = para.dim_out
        self.dim_embeding = para.dim_embeding
        self.dropout = para.dropout
        self.negative_slope = para.negative_slope

        self.BNlayer = nn.BatchNorm1d(self.dim_out)
```



```python
self.model = nn.Sequential(
                # 先行正则化
                nn.BatchNorm1d(self.dim_in),

                # 第一层
                nn.Linear(self.dim_in, self.dim_embeding),
                nn.BatchNorm1d(self.dim_embeding),
                # nn.ReLU(),
                nn.LeakyReLU(self.negative_slope),
                nn.Dropout(self.dropout),

                # 第二层
                nn.Linear(self.dim_embeding, self.dim_embeding),
                nn.BatchNorm1d(self.dim_embeding),
                # nn.ReLU(),
                nn.LeakyReLU(self.negative_slope),
                nn.Dropout(self.dropout),

                # 第三层
                nn.Linear(self.dim_embeding, self.dim_embeding),
                nn.BatchNorm1d(self.dim_embeding),
                # nn.ReLU(),
                nn.LeakyReLU(self.negative_slope),
                nn.Dropout(self.dropout),

                # 第四层
                nn.Linear(self.dim_embeding, self.dim_out),
                nn.BatchNorm1d(self.dim_out),

                )
```

```
    for m in self.modules():
        if isinstance(m, nn.Linear):
            nn.init.xavier_normal_(m.weight.data)


def forward(self,x,y=None, istraining = True):
    out_enh = self.model(x)
    if istraining:
        out_targrt = self.BNlayer(y)
        return out_enh,out_targrt
    else:
        return out_enh
```



LPS-in

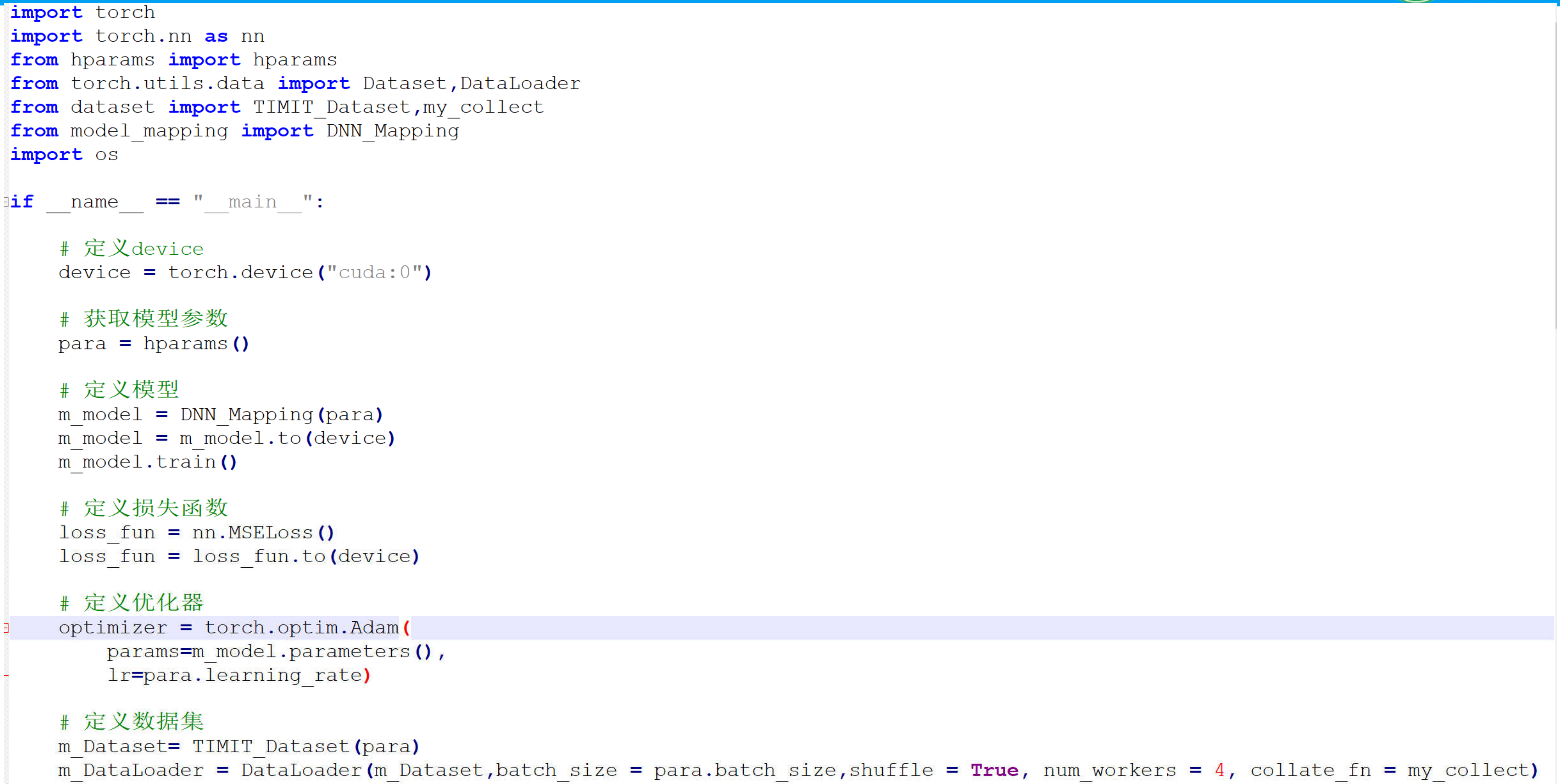
DNN

BN 输出

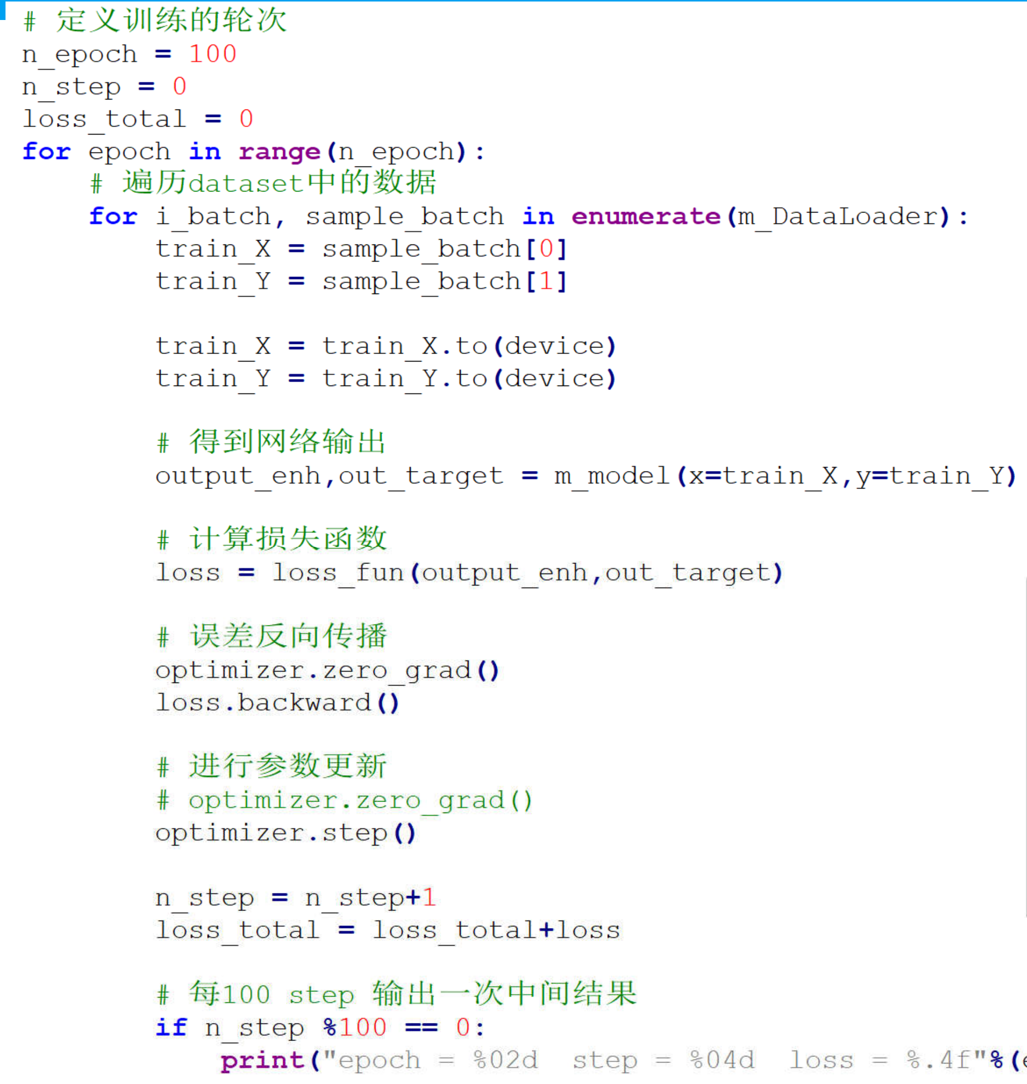
MSE

BN

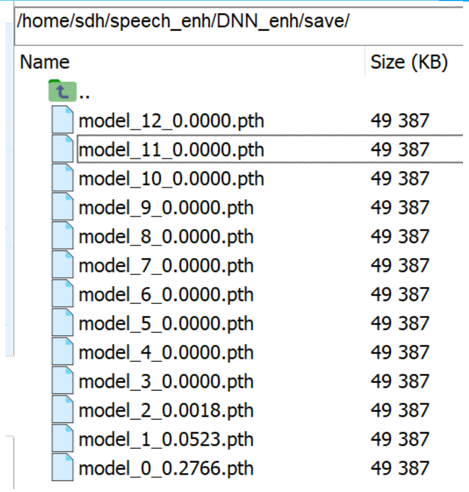
LPS-out



```
import torch
import torch.nn as nn
from hparams import hparams
from torch.utils.data import Dataset,DataLoader
from dataset import TIMIT_Dataset,my_collect
from model_mapping import DNN_Mapping
import os

if __name__ == "__main__":

    # 定义device
    device = torch.device("cuda:0")

    # 获取模型参数
    para = hparams()

    # 定义模型
    m_model = DNN_Mapping(para)
    m_model = m_model.to(device)
    m_model.train()

    # 定义损失函数
    loss_fun = nn.MSELoss()
    loss_fun = loss_fun.to(device)

    # 定义优化器
    optimizer = torch.optim.Adam(
        params=m_model.parameters(),
        lr=para.learning_rate)

    # 定义数据集
    m_Dataset= TIMIT_Dataset(para)
    m_DataLoader = DataLoader(m_Dataset,batch_size = para.batch_size,shuffle = True, num_workers = 4, collate_fn = my_collect)
```

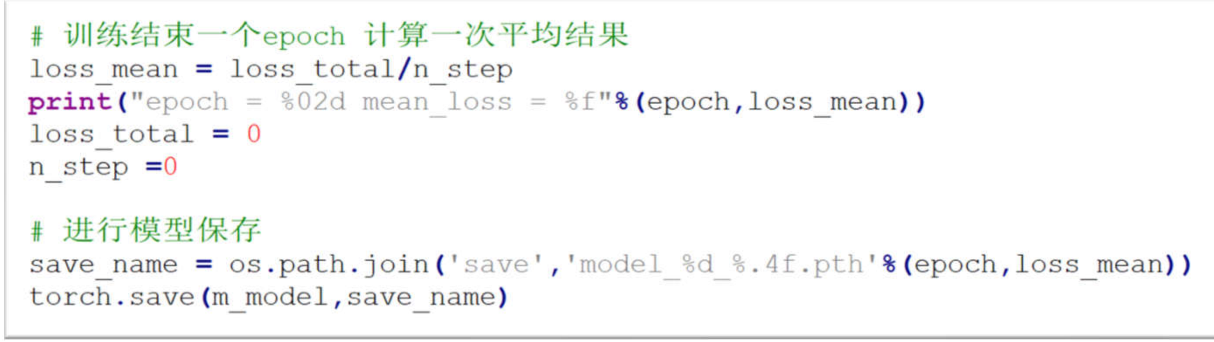


```
# 定义训练的轮次
n_epoch = 100
n_step = 0
loss_total = 0
for epoch in range(n_epoch):
    # 遍历dataset中的数据
    for i_batch, sample_batch in enumerate(m_DataLoader):
        train_X = sample_batch[0]
        train_Y = sample_batch[1]

        train_X = train_X.to(device)
        train_Y = train_Y.to(device)

        # 得到网络输出
        output_enh,out_target = m_model(x=train_X,y=train_Y)

        # 计算损失函数
        loss = loss_fun(output_enh,out_target)

        # 误差反向传播
        optimizer.zero_grad()
        loss.backward()

        # 进行参数更新
        # optimizer.zero_grad()
        optimizer.step()

        n_step = n_step+1
        loss_total = loss_total+loss

        # 每100 step 输出一次中间结果
        if n_step %100 == 0:
            print("epoch = %02d  step = %04d  loss = %.4f"%(epoch,n_step,loss))
```



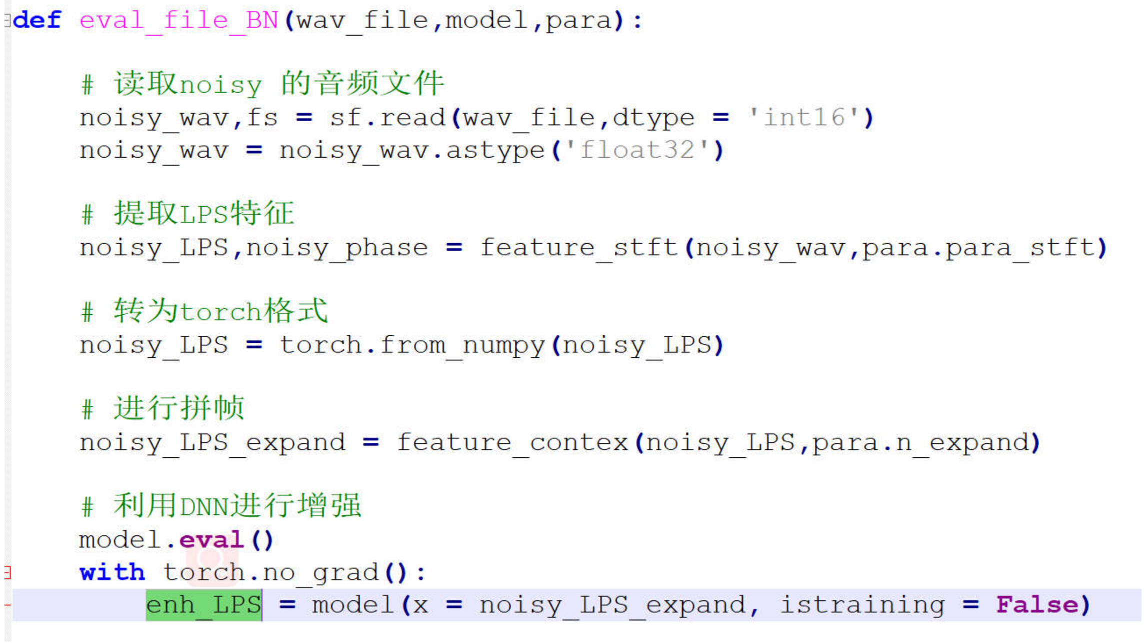
/home/sdh/speech_enh/DNN_enh/save/

Name	Size (KB)
..	
model_12_0.0000.pth	49 387
model_11_0.0000.pth	49 387
model_10_0.0000.pth	49 387
model_9_0.0000.pth	49 387
model_8_0.0000.pth	49 387
model_7_0.0000.pth	49 387
model_6_0.0000.pth	49 387
model_5_0.0000.pth	49 387
model_4_0.0000.pth	49 387
model_3_0.0000.pth	49 387
model_2_0.0018.pth	49 387
model_1_0.0523.pth	49 387
model_0_0.2766.pth	49 387

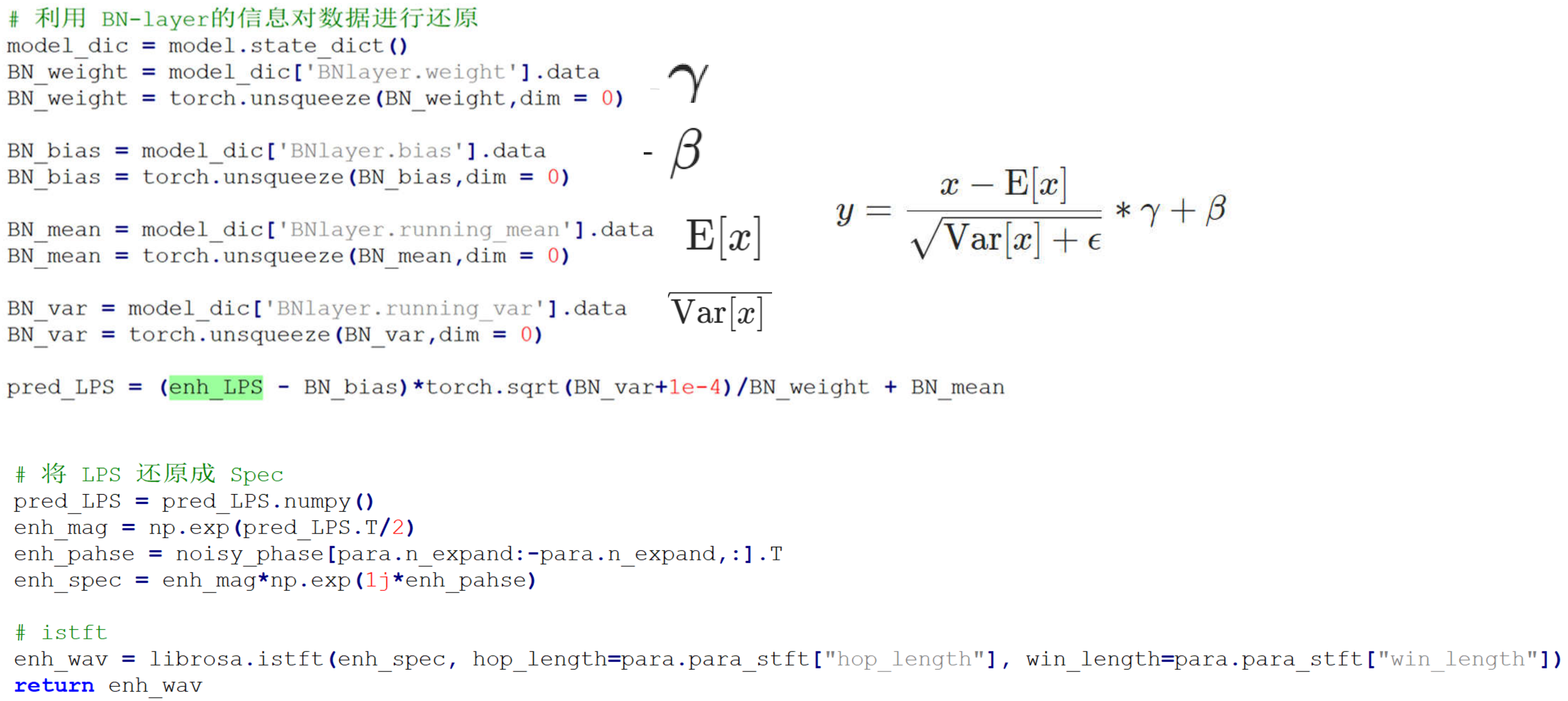


```
# 训练结束一个epoch 计算一次平均结果
loss_mean = loss_total/n_step
print("epoch = %02d mean_loss = %f"%(epoch,loss_mean))
loss_total = 0
n_step =0

# 进行模型保存
save_name = os.path.join('save','model_%d_%.4f.pth'%(epoch,loss_mean))
torch.save(m_model,save_name)
```

测试



```
def eval_file_BN(wav_file,model,para):

    # 读取noisy 的音频文件
    noisy_wav,fs = sf.read(wav_file,dtype = 'int16')
    noisy_wav = noisy_wav.astype('float32')

    # 提取LPS特征
    noisy_LPS,noisy_phase = feature_stft(noisy_wav,para.para_stft)

    # 转为torch格式
    noisy_LPS = torch.from_numpy(noisy_LPS)

    # 进行拼帧
    noisy_LPS_expand = feature_contex(noisy_LPS,para.n_expand)

    # 利用DNN进行增强
    model.eval()
    with torch.no_grad():
        enh_LPS = model(x = noisy_LPS_expand, istraining = False)
```



```
# 利用 BN-layer的信息对数据进行还原
model_dic = model.state_dict()
BN_weight = model_dic['BNlayer.weight'].data
BN_weight = torch.unsqueeze(BN_weight,dim = 0)

BN_bias = model_dic['BNlayer.bias'].data
BN_bias = torch.unsqueeze(BN_bias,dim = 0)

BN_mean = model_dic['BNlayer.running_mean'].data
BN_mean = torch.unsqueeze(BN_mean,dim = 0)

BN_var = model_dic['BNlayer.running_var'].data
BN_var = torch.unsqueeze(BN_var,dim = 0)

pred_LPS = (enh_LPS - BN_bias)*torch.sqrt(BN_var+1e-4)/BN_weight + BN_mean


# 将 LPS 还原成 Spec
pred_LPS = pred_LPS.numpy()
enh_mag = np.exp(pred_LPS.T/2)
enh_pahse = noisy_phase[para.n_expand:-para.n_expand,:].T
enh_spec = enh_mag*np.exp(1j*enh_pahse)

# istft
enh_wav = librosa.istft(enh_spec, hop_length=para.para_stft["hop_length"], win_length=para.para_stft["win_length"])
return enh_wav
```

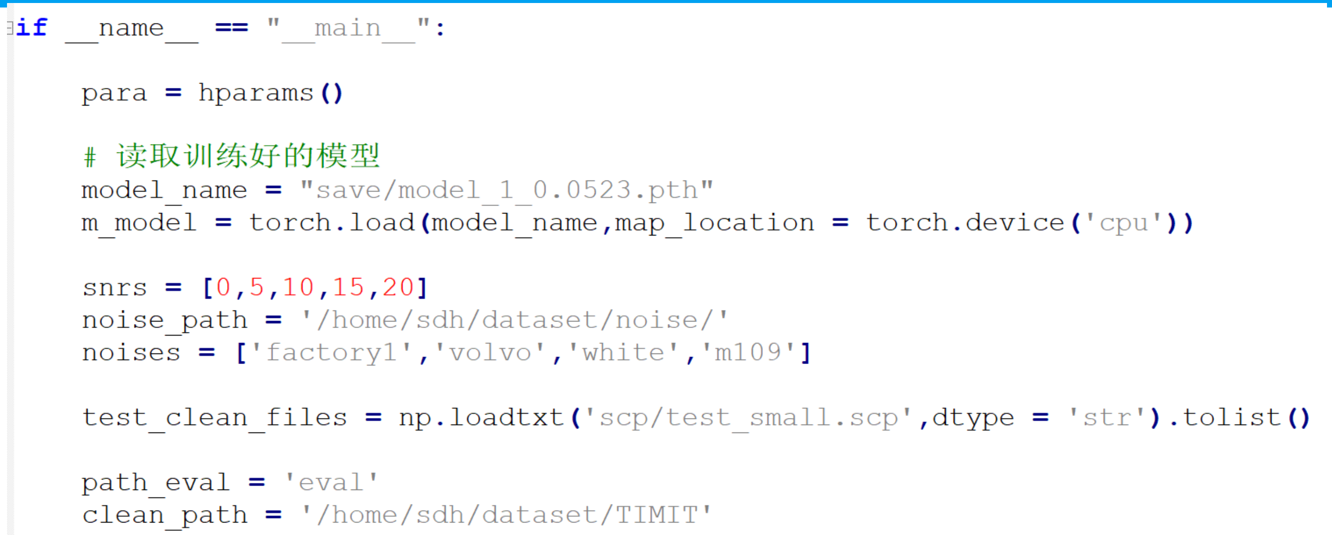
γ

β

$\mathrm{E}[x]$

$\mathrm{Var}[x]$

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$



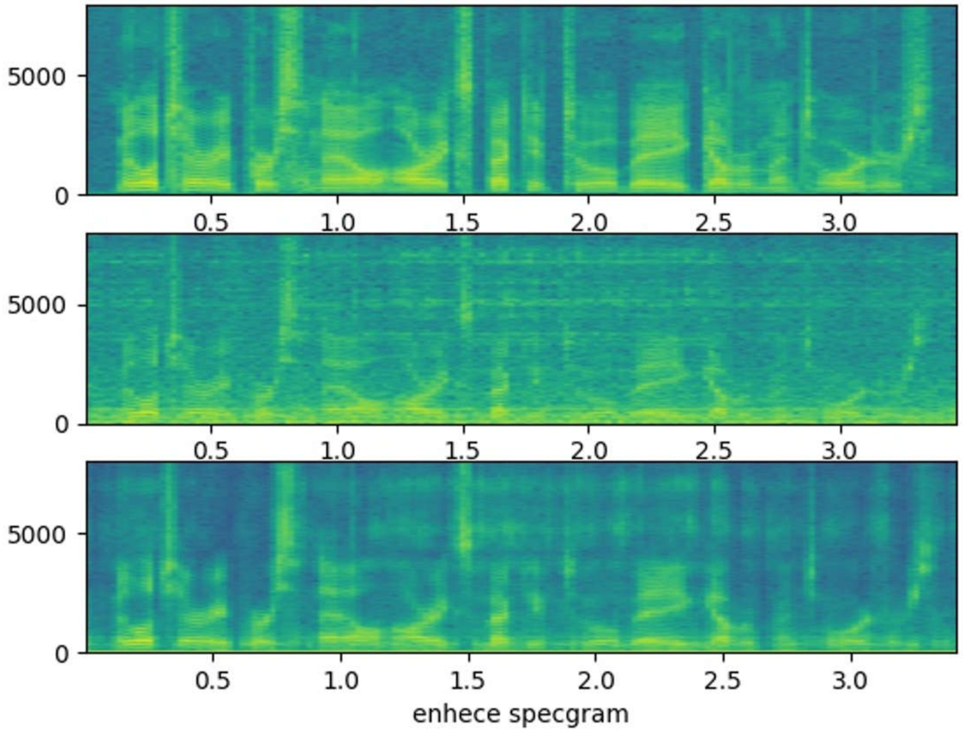
```
if __name__ == "__main__":

    para = hparams()

    # 读取训练好的模型
    model_name = "save/model_1_0.0523.pth"
    m_model = torch.load(model_name,map_location = torch.device('cpu'))

    snrs = [0,5,10,15,20]
    noise_path = '/home/sdh/dataset/noise/'
    noises = ['factory1','volvo','white','m109']

    test_clean_files = np.loadtxt('scp/test_small.scp',dtype = 'str').tolist()

    path_eval = 'eval'
    clean_path = '/home/sdh/dataset/TIMIT'
```

```
for noise in noises:
    print(noise)
    noise_file = os.path.join(noise_path,noise+'.wav')
    noise_data,fs = sf.read(noise_file,dtype = 'int16')

    for clean_wav in test_clean_files:

        # 读取干净语音并保存
        clean_file = os.path.join(clean_path,clean_wav)
        clean_data,fs = sf.read(clean_file,dtype = 'int16')
        id = os.path.split(clean_file)[-1]
        sf.write(os.path.join(path_eval,id),clean_data,fs)

        for snr in snrs:
            # 生成noisy文件
            noisy_file = os.path.join(path_eval,noise+'-'+str(snr)+'-'+id)
            mix = signal_by_db(clean_data,noise_data,snr)
            noisy_data = np.asarray(mix,dtype= np.int16)
            sf.write(noisy_file,noisy_data,fs)

            # 进行增强
            print("enhancement file %s"%(noisy_file))
            enh_data = eval_file_BN(noisy_file,m_model,para)

            # 信号正则
            max_ = np.max(enh_data)
            min_ = np.min(enh_data)
            enh_data = enh_data*(2/(max_ - min_)) - (max_+min_)/(max_-min_)
            enh_file = os.path.join(path_eval,noise+'-'+str(snr)+'-'+'enh'+'-'+id)
            sf.write(enh_file,enh_data,fs)
```

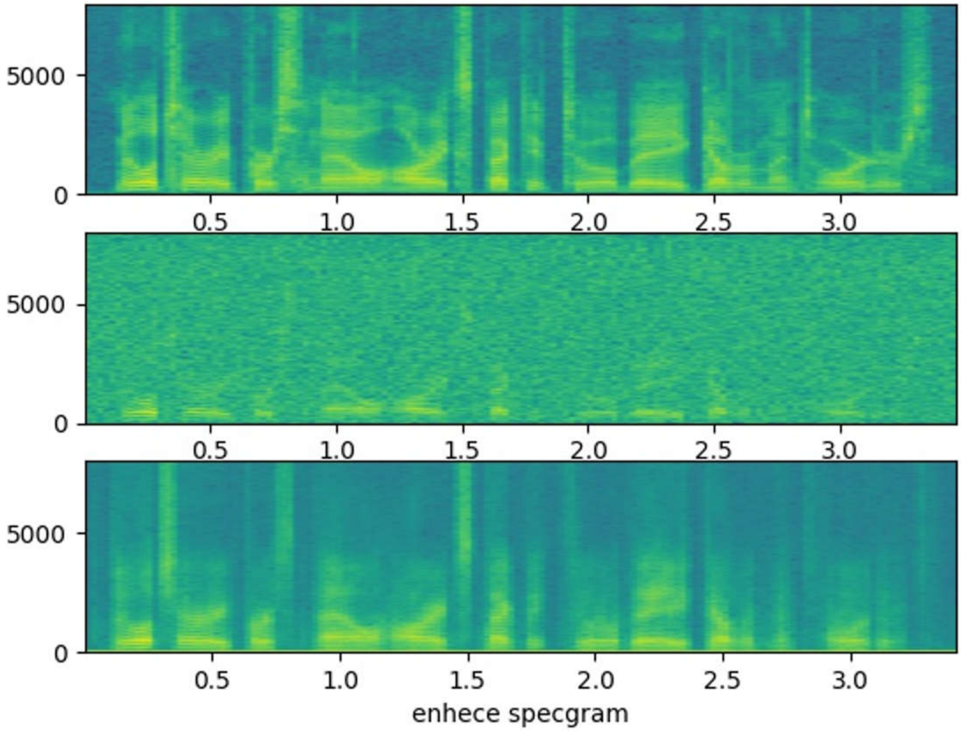
将数值转为-1~+1

```
# 绘图
fig_name = os.path.join(path_eval,noise+'-'+str(snr)+'-'+id[:-3]+'jpg')

plt.subplot(3,1,1)
plt.specgram(clean_data,NFFT=512,Fs=fs)
plt.xlabel("clean specgram")
plt.subplot(3,1,2)
plt.specgram(noisy_data,NFFT=512,Fs=fs)
plt.xlabel("noisy specgram")
plt.subplot(3,1,3)
plt.specgram(enh_data,NFFT=512,Fs=fs)
plt.xlabel("enhece specgram")
plt.savefig(fig_name)
```

部分结果



M109-0db



White 0DB