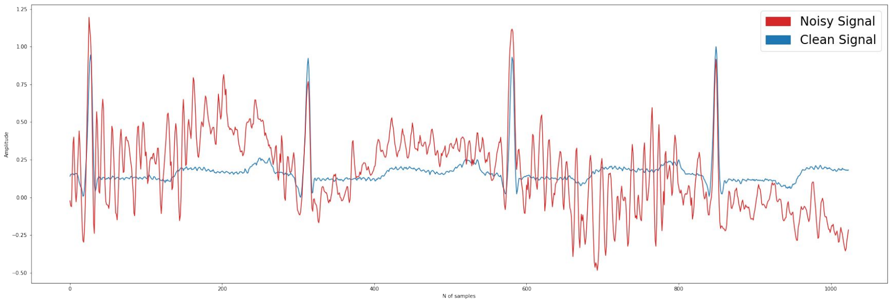
Sujet 2

Utilisation d’une méthode des auto-encodeurs entièrement convolutif pour le débruitage une base de données PCG

ANTONETTI Levis
BOUDJENIBA Oussama

Papier scientifique:

- Méthode de débruitage de signaux électriques ECG (électrocardiogramme) à l’aide d” électrodes
- détection arrhythmia, rythme cardiaque irrégulier
- approche par deep learning avec auto-encodeurs entièrement convolutionnel.



Noise Reduction in ECG Signals Using Fully Convolutional Denoising Autoencoders

HSIN-TIEN CHIANG[1], YI-YEN HSIEH[1], SZU-WEI FU[2], KUO-HSUAN HUNG[2], YU TSAO[2], AND SHAO-YI CHIEN[1], (Senior Member, IEEE)
[1]Graduate Institute of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Research Center for Information Technology Innovation, Academia Sinica, Taipei 11529, Taiwan

Corresponding author: Yu Tsao (yu.tsao@citi.sinica.edu.tw)

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 107-2633-E-002-001, Grant MOST 107-2221-E-001-012-MY2, and Grant MOST 106-2221-E-001-017-MY2, in part by the National Taiwan University under Grant NTU-107L104039, in part by the Intel Corporation, and in part by the Delta Electronics.

ABSTRACT The electrocardiogram (ECG) is an efficient and noninvasive indicator for arrhythmia detection and prevention. In real-world scenarios, ECG signals are prone to be contaminated with various noises, which may lead to wrong interpretation. Therefore, significant attention has been paid on denoising of ECG for accurate diagnosis and analysis. A denoising autoencoder (DAE) can be applied to reconstruct the clean data from its noisy version. In this paper, a DAE using the fully convolutional network (FCN) is proposed for ECG signal denoising. Meanwhile, the proposed FCN-based DAE can perform compression with regard to the DAE architecture. The proposed approach is applied to ECG signals from the MIT-BIH Arrhythmia database and the added noise signals are obtained from the MIT-BIH Noise Stress Test database. The denoising performance is evaluated using the root-mean-square error (RMSE), percentage-root-mean-square difference (PRD), and improvement in signal-to-noise ratio (SNR_{imp}). The results of the experiments conducted on noisy ECG signals of different levels of input SNR show that the FCN acquires better performance as compared to the deep fully connected neural network- and convolutional neural network-based denoising models. Moreover, the proposed FCN-based DAE reduces the size of the input ECG signals, where the compressed data is 32 times smaller than the original. The results of the study demonstrate the superiority of FCN in denoising, with lower RMSE and PRD, as well as higher SNR_{imp}. According to the results, we believe that the proposed FCN-based DAE has a good application prospect in clinical practice.

Adaptation de cette approche sur des signaux PCG

- Phonocardiogramme (PCG), enregistrements des signaux sonores des battements du cœur.
- Les enregistrements ECG semblent un peu plus bruités et bruit inégalement réparti à travers les signaux.

Noise Reduction in ECG Signals Using Fully Convolutional Denoising Autoencoders

HSIN-TIEN CHIANG[1], YI-YEN HSIEH[1], SZU-WEI FU[2], KUO-HSUAN HUNG[2], YU TSAO[2], AND SHAO-YI CHIEN[1], (Senior Member, IEEE)
[1]Graduate Institute of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Research Center for Information Technology Innovation, Academia Sinica, Taipei 11529, Taiwan
Corresponding author: Yu Tsao (yu.tsao@citi.sinica.edu.tw)

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 107-2633-E-002-001, Grant MOST 107-2221-E-001-012-MY2, and Grant MOST 106-2221-E-001-017-MY2, in part by the National Taiwan University under Grant NTU-107L104039, in part by the Intel Corporation, and in part by the Delta Electronics.

ABSTRACT The electrocardiogram (ECG) is an efficient and noninvasive indicator for arrhythmia detection and prevention. In real-world scenarios, ECG signals are prone to be contaminated with various noises, which may lead to wrong interpretation. Therefore, significant attention has been paid on denoising of ECG for accurate diagnosis and analysis. A denoising autoencoder (DAE) can be applied to reconstruct the clean data from its noisy version. In this paper, a DAE using the fully convolutional network (FCN) is proposed for ECG signal denoising. Meanwhile, the proposed FCN-based DAE can perform compression with regard to the DAE architecture. The proposed approach is applied to ECG signals from the MIT-BIH Arrhythmia database and the added noise signals are obtained from the MIT-BIH Noise Stress Test database. The denoising performance is evaluated using the root-mean-square error (RMSE), percentage-root-mean-square difference (PRD), and improvement in signal-to-noise ratio (SNR_{imp}). The results of the experiments conducted on noisy ECG signals of different levels of input SNR show that the FCN acquires better performance as compared to the deep fully connected neural network- and convolutional neural network-based denoising models. Moreover, the proposed FCN-based DAE reduces the size of the input ECG signals, where the compressed data is 32 times smaller than the original. The results of the study demonstrate the superiority of FCN in denoising, with lower RMSE and PRD, as well as higher SNR_{imp}. According to the results, we believe that the proposed FCN-based DAE has a good application prospect in clinical practice.

Réimplémentation du code sur Pytorch

- Code original de répertoire du projet ne compilait pas (pas de liste de requirements :’))
- écrit en TensorFlow v1, pb de version et de compatibilité de librairies

→ adaptation du code avec Pytorch, pour le traitement des données, création des datasets bruités, model, …

Training functions

```
def train_fn_base(model, train=None, valid=None, opt=None, loss_fr
  model.train()

  best_val_loss = 10e9

  for epoch in range(1, epochs + 1):
    loss_g = 0
    v_loss_g = 0
    v_loss_r = 0
    for _, batch in tqdm(enumerate(train, 0), unit="batch", total=
      clean = batch['clean']
      clean = clean.to(device)

      noisy = batch['noisy'].clone()
      noisy.requires_grad = True
      noisy = noisy.to(device)

      out = model(noisy)

      loss = loss_fn(out, clean)
      optimizer.zero_grad()
      loss.backward()
      optimizer.step()

      loss_g += loss.item()

    model.eval()
    for _, batch in tqdm(enumerate(valid, 0), unit="batch", total=len(valid)):
      clean = batch['clean']
      clean = clean.to(device)
```

Metrics Functions

```
# calculate SNR between a clean signal an a
def calc_snr(x,y):
  return 10 * torch.log10((x**2).sum() / ((

# calc_snr function adapted to batches to t
def calc_snr_batch(bx,by):
  return 10 * torch.log10((bx**2).sum(axis=

# calculate PRD score adapted for batches
def prd_batch(bx, by):
  return torch.sqrt(((((bx - by) ** 2).sum(
```

Dataset Classes

```
class SignalsDataset(Dataset):
    def __init__(self, d):
      self.clean = torch.tensor(d['clean'], dtype=torch.float32).unsqueeze(1)
      self.origin = torch.tensor(d['origin'], dtype=torch.float32).unsqueeze(1)
      self.recons = torch.tensor(d['recons'], dtype=torch.float32).unsqueeze(1)

    def __len__(self):
      return len(self.clean)

    def __getitem__(self, idx):
      noisy = self.origin[idx]
      clean = self.clean[idx]
      recons = self.recons[idx]
      return {
          'noisy': noisy,
          'clean': clean,
          'recons': recons,
      }

class OurSignalsDataset(Dataset):
    def __init__(self, d, snr_list=None):

      self.snr_list = snr_list

      self.clean = torch.tensor(
          d['clean'],
          dtype=torch.float32
      ).expand(
          len(snr_list),
          d['clean'].shape[0],
          d['clean'].shape[1],
      ).permute([1,0,2]).flatten(
          start_dim=0,
          end_dim=1,
```

Architecture auto-encoder (DAE) entièrement convolutionelle (FCN)

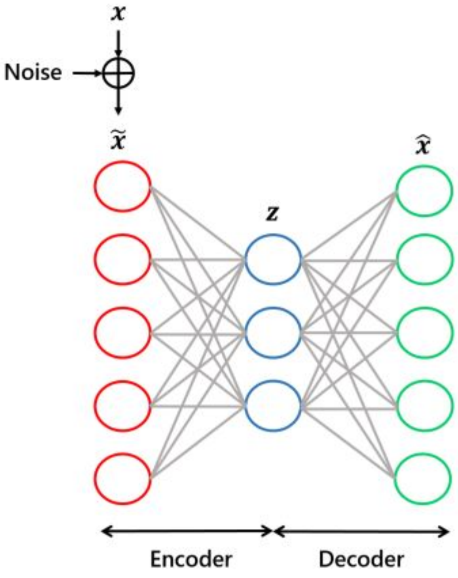


FIGURE 1. Schematic diagram of DAE.

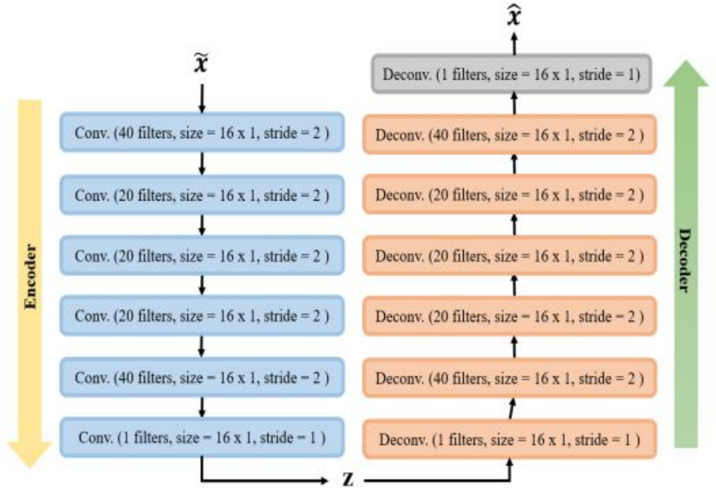


FIGURE 2. Architecture of the proposed FCN-based DAE.

Architecture auto-encoder (DAE) entièrement convolutionelle (FCN)

- Pas de couches de MaxPooling(UpSampling) entre les couches de convolution(déconvolution) pour éviter de perdre trop de détail du signal
- FCN - pas de couche entièrement connecté:
  - préserve l'info spatial
  - réduit le nombre de paramètre du réseau

```python
class DAE(nn.Module):

  def __init__(self, input_size=1024, kernel_size=16):
    super(DAE, self).__init__()

    self.activation = nn.ELU()
    self.kernel_size = kernel_size
    self.input_size = input_size
    self.padding = (kernel_size - 1) // 2

    slot = self.input_size

    self.conv1 = nn.Conv1d(1, 40, self.kernel_size, stride=2, padding=self.padding)
    self.conv_bn1 = nn.BatchNorm1d(40)
    self.conv2 = nn.Conv1d(40, 20, self.kernel_size, stride=2, padding=self.padding)
    self.conv_bn2 = nn.BatchNorm1d(20)
    self.conv3 = nn.Conv1d(20, 20, self.kernel_size, stride=2, padding=self.padding)
    self.conv_bn3 = nn.BatchNorm1d(20)
    self.conv4 = nn.Conv1d(20, 20, self.kernel_size, stride=2, padding=self.padding)
    self.conv_bn4 = nn.BatchNorm1d(20)
    self.conv5 = nn.Conv1d(20, 40, self.kernel_size, stride=2, padding=self.padding)
    self.conv_bn5 = nn.BatchNorm1d(40)
    self.conv6 = nn.Conv1d(40, 1, self.kernel_size, stride=1, padding='same')
    self.deconv0 = nn.ConvTranspose1d(1, 1, self.kernel_size - 1, stride=1, padding=self.padding)
    self.deconv_bn0 = nn.BatchNorm1d(1)
    self.deconv1 = nn.ConvTranspose1d(1, 40, self.kernel_size, stride=2, padding=self.padding)
    self.deconv_bn1 = nn.BatchNorm1d(40)
    self.deconv2 = nn.ConvTranspose1d(40, 20, self.kernel_size, stride=2, padding=self.padding)
    self.deconv_bn2 = nn.BatchNorm1d(20)
    self.deconv3 = nn.ConvTranspose1d(20, 20, self.kernel_size, stride=2, padding=self.padding)
    self.deconv_bn3 = nn.BatchNorm1d(20)
    self.deconv4 = nn.ConvTranspose1d(20, 20, self.kernel_size, stride=2, padding=self.padding)
    self.deconv_bn4 = nn.BatchNorm1d(20)
    self.deconv5 = nn.ConvTranspose1d(20, 40, self.kernel_size, stride=2, padding=self.padding)
    self.deconv_bn5 = nn.BatchNorm1d(40)
    self.deconv6 = nn.ConvTranspose1d(
        40,
        1,
        self.kernel_size - 1,
        stride=1,
        padding=(self.kernel_size - 1) // 2
    )
```

```python
def forward(self, x):
  # encoder
  x = self.conv1(x)
  x = self.conv_bn1(x)
  x = self.activation(x)

  x = self.conv2(x)
  x = self.conv_bn2(x)
  x = self.activation(x)

  x = self.conv3(x)
  x = self.conv_bn3(x)
  x = self.activation(x)

  x = self.conv4(x)
  x = self.conv_bn4(x)
  x = self.activation(x)

  x = self.conv5(x)
  x = self.conv_bn5(x)
  x = self.activation(x)

  x = self.conv6(x)
  x = self.activation(x)

  # decoder
  x = self.deconv0(x)
  x = self.deconv_bn0(x)
  x = self.activation(x)

  x = self.deconv1(x)
  x = self.deconv_bn1(x)
  x = self.activation(x)

  x = self.deconv2(x)
  x = self.deconv_bn2(x)
  x = self.activation(x)

  x = self.deconv3(x)
  x = self.deconv_bn3(x)
  x = self.activation(x)

  x = self.deconv4(x)
  x = self.deconv_bn4(x)
  x = self.activation(x)

  x = self.deconv5(x)
  x = self.deconv_bn5(x)
  x = self.activation(x)

  x = self.deconv6(x)
  x = self.activation(x)

  return x
```

Résultats du papier:

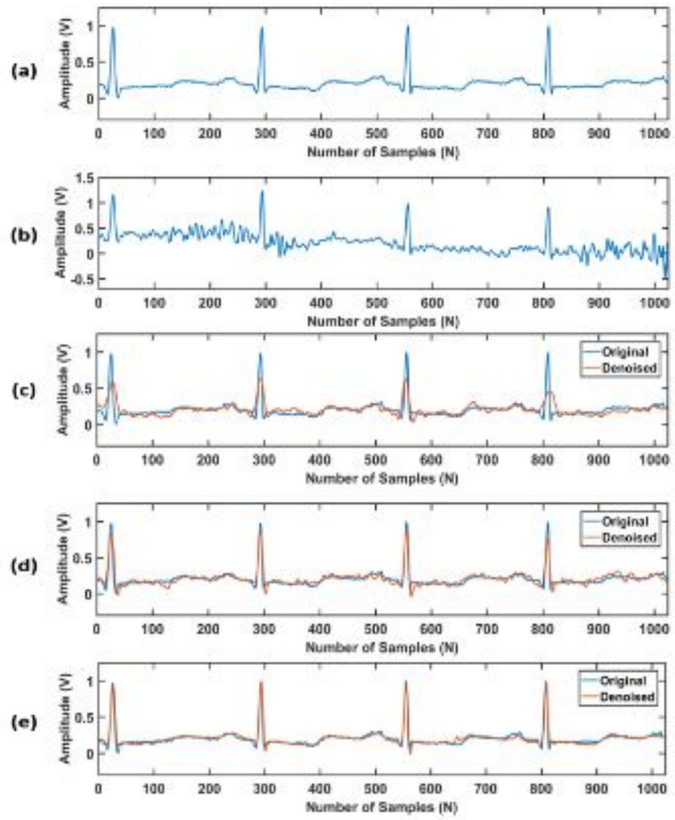


FIGURE 4. Comparison of denoised results of all evaluated methods on record 100 (a) original ECG signal (b) noisy ECG signal with an input SNR of 3dB (c) DNN (d) CNN (e) FCN.

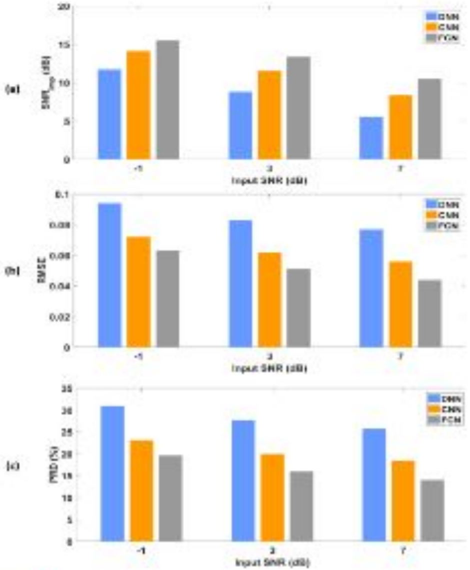
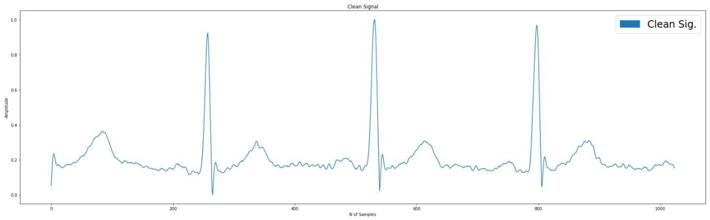
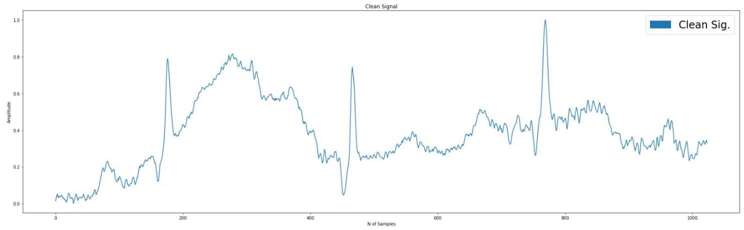


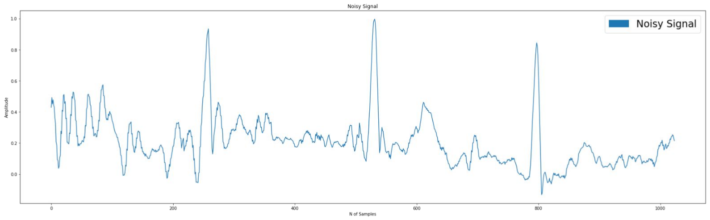
FIGURE 3. Comparison of the denoising performance of all evaluated methods at different input SNR levels (a) SNR_{imp} of DNN, CNN and FCN with varying input SNR levels (b) RMSE of DNN, CNN and FCN with varying input SNR levels (c) PRD of DNN, CNN and FCN with varying input SNR levels.

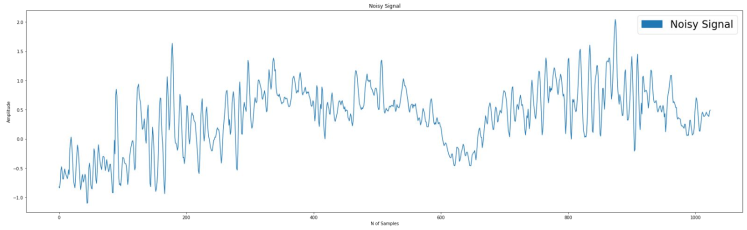
Comparaison des résultats avec architectures DNN et CNN

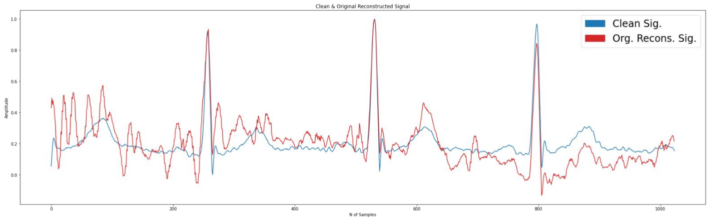
1 - Entrainement du modèle avec les données disponible dans le répertoire Noise-Reduction-in-ECG-Signals

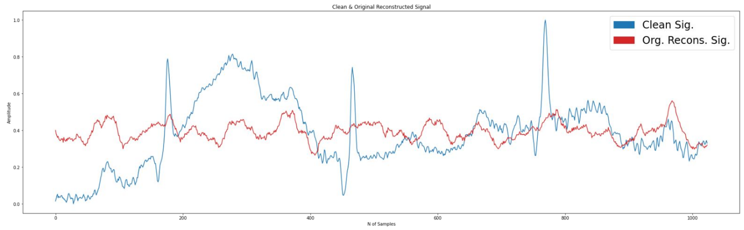


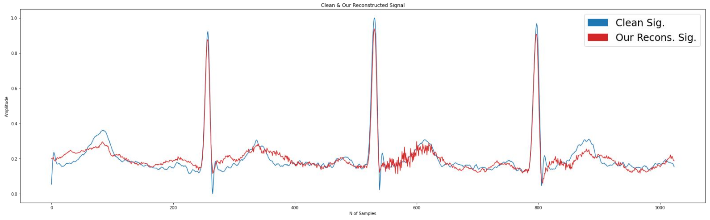


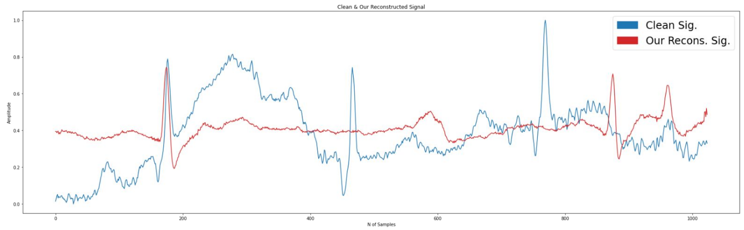




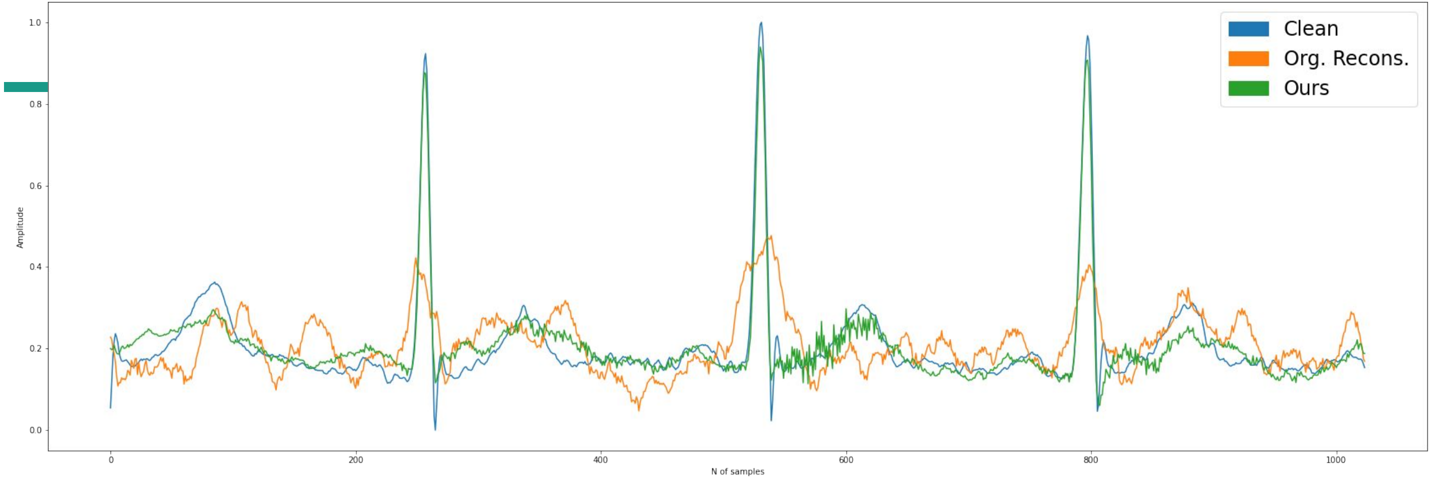








1 - Entrainement du modèle avec les données disponible dans le répertoire [Noise-Reduction-in-ECG-Signals]



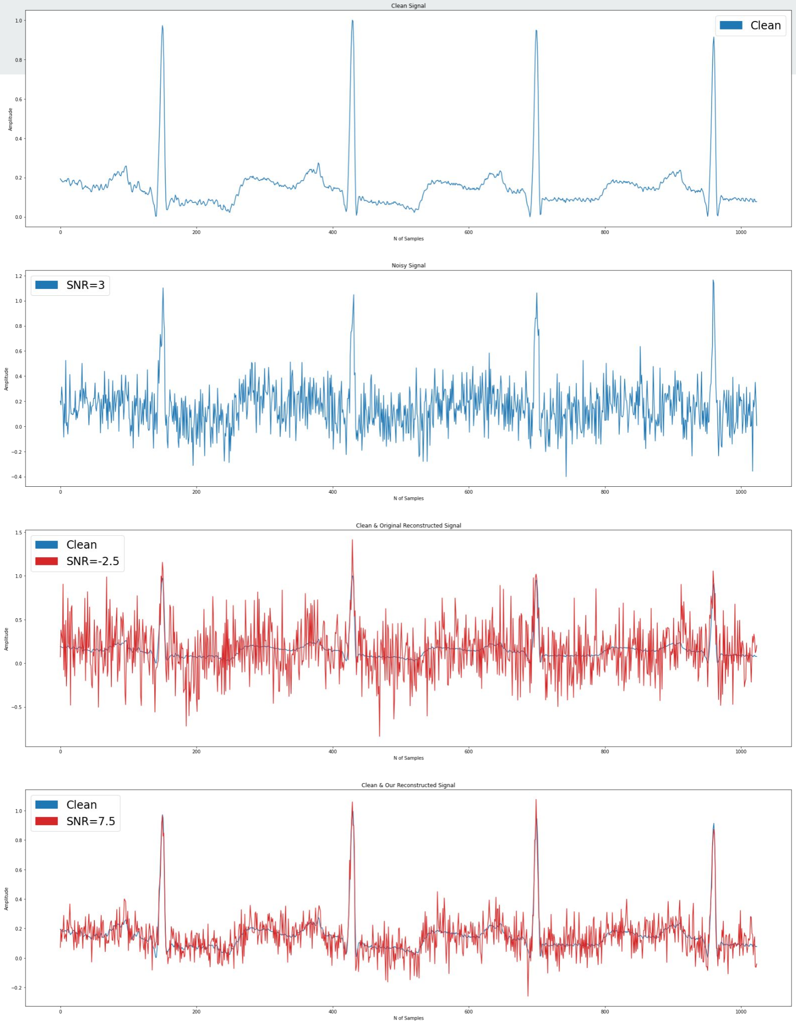

```
100%  72/72 [00:01<00:00, 61.72batch/s]
100%  10/10 [00:00<00:00, 106.10batch/s]
epoch 96:
MSE training loss: 0.00776607948096676
MSE validation loss: 0.00554668391123414
MSE reconstruc loss: 0.010975828813388943
found better valid loss... saving model...
```

Métriques utilisées

root mean square error: $$RMSE = \sqrt{\frac{1}{N} \times \sum_{n=1}^{N} (x_i - \hat{x}_i)^2}$$	*percentage root mean square difference:* $$PRD = \sqrt{\frac{\sum_{n=1}^{N} (x_i - \hat{x}_i)^2}{\sum_{n=1}^{N} x_i^2} \times 100}$$	*SNR difference between orginal et denoised input:* $$SNR_{in} = 10 \times \log_{10}\left(\frac{\sum_{n=1}^{N} x_i^2}{\sum_{n=1}^{N} (\tilde{x}_i - x_i)^2}\right)$$ $$SNR_{out} = 10 \times \log_{10}\left(\frac{\sum_{n=1}^{N} x_i^2}{\sum_{n=1}^{N} (\hat{x}_i - x_i)^2}\right)$$ $$SNR_{imp} = SNR_{out} - SNR_{in}$$

2 - Entrainement du modèle sur le nouveau dataset créé à partir des données cleans.

- Bruitées en fonctions de différents SNR=[-2.5, -1, 0, 1, 2.5, 3, 5, 7.5].



2 - Entrainement du modèle sur le nouveau dataset créé à partir des données cleans.

Résultats métriques

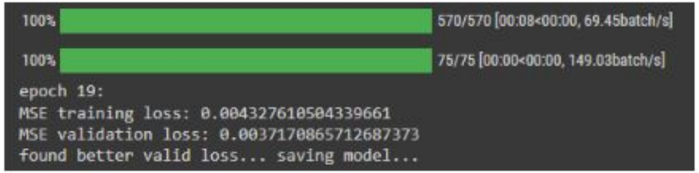


fig8. best performances model 2

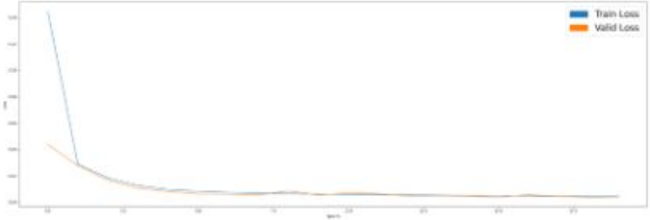


fig9. train and validation curves

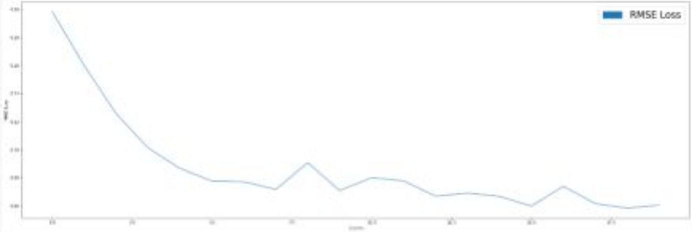


fig10. loss curve RMSE

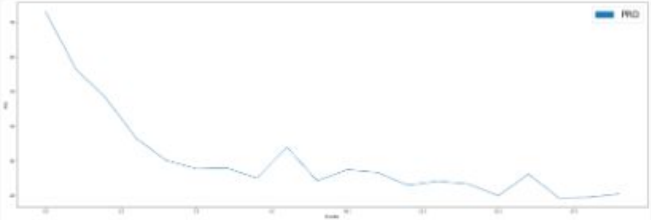


fig11. loss curve PRD

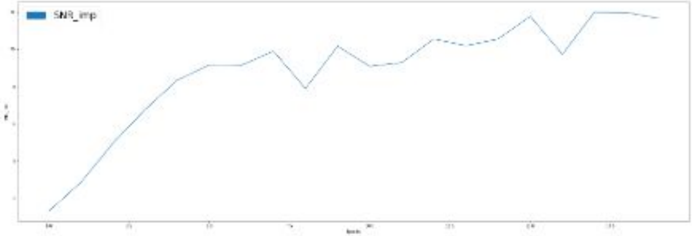


fig12. loss curve SNR_imp

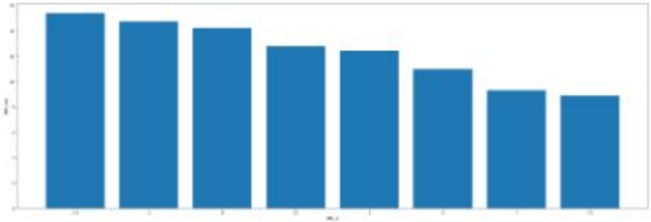
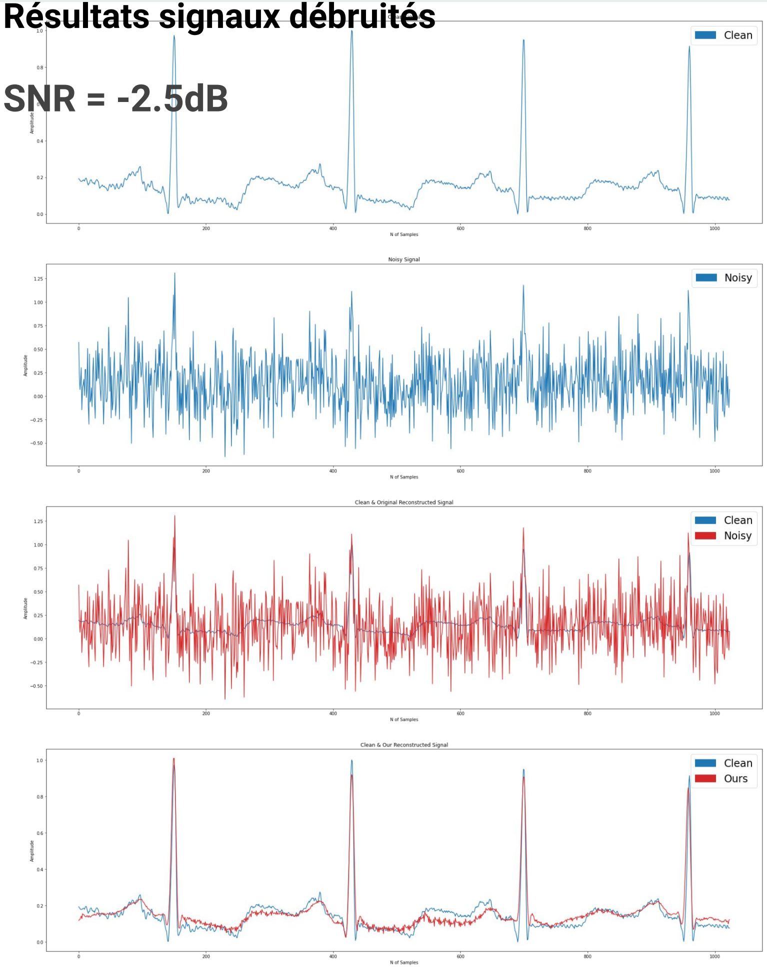


fig13. SNR_imp ratio

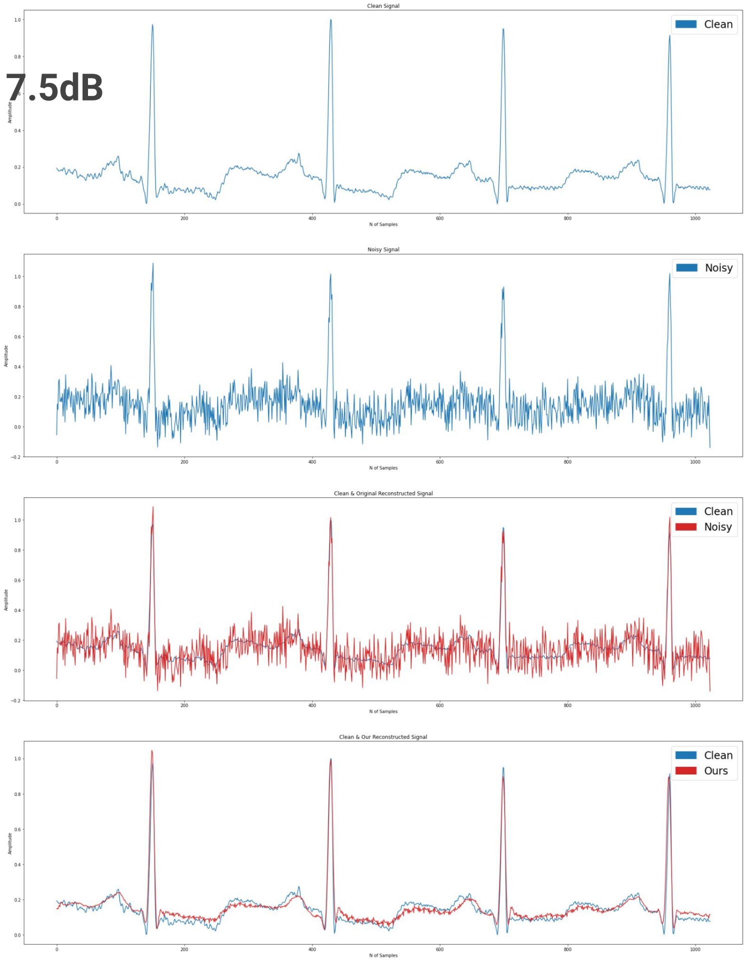
2 - Entrainement du modèle sur le nouveau dataset créé à partir des données cleans.

Résultats signaux débruités

SNR = -2.5dB



SNR = 7.5dB



3 - Application du modèle au signaux PCG

- Standardisation des signaux PCG

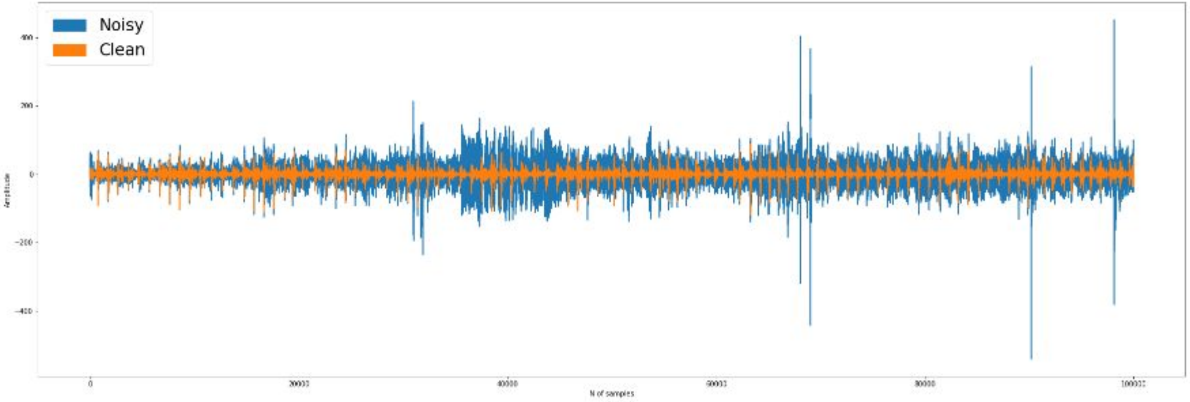


fig16. échantillon signal PCG brut

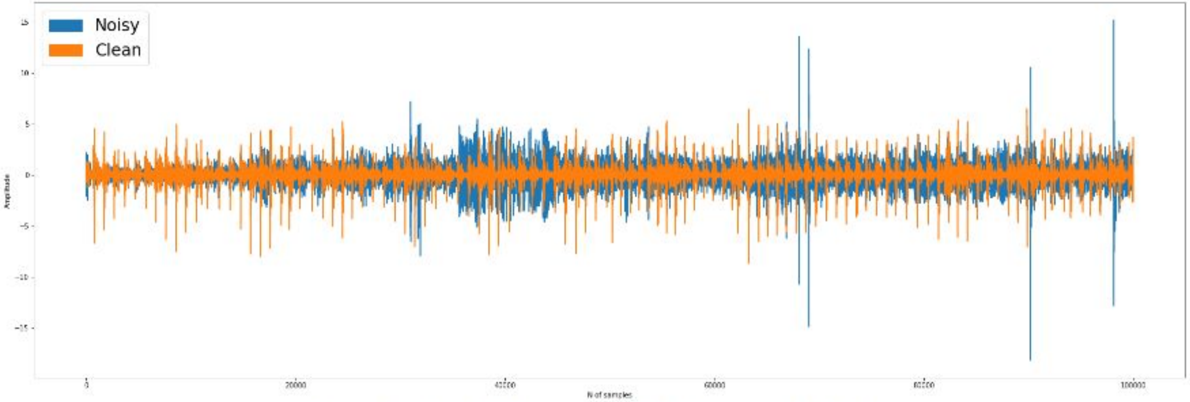
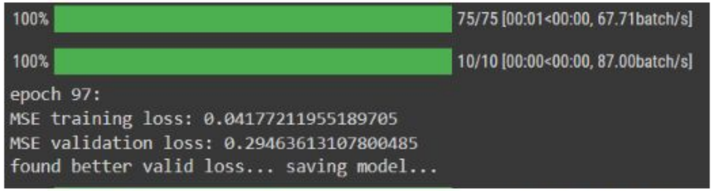


fig17. échantillon signal PCG standardisé

3 - Entrainement du modèle sur le dataset des signaux PCG

Résultats métriques



```
100% 75/75 [00:01<00:00, 67.71batch/s]
100% 10/10 [00:00<00:00, 87.00batch/s]
epoch 97:
MSE training loss: 0.04177211955189705
MSE validation loss: 0.29463613107800485
found better valid loss... saving model...
```

fig18 best performances model 3 (PCG)

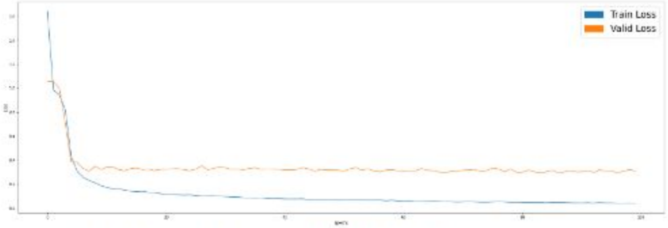


fig19. train and validation curves

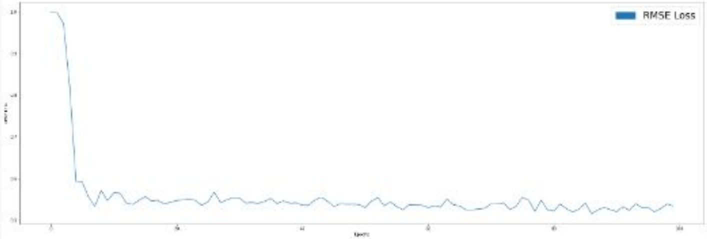


fig20. loss curve RMSE

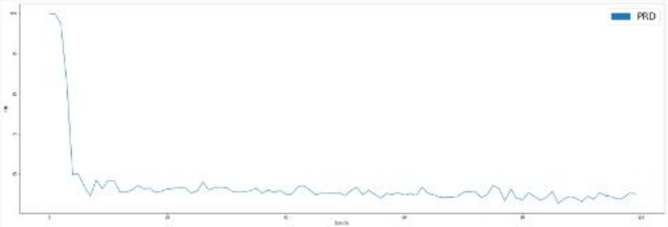


fig21. loss curve PRD

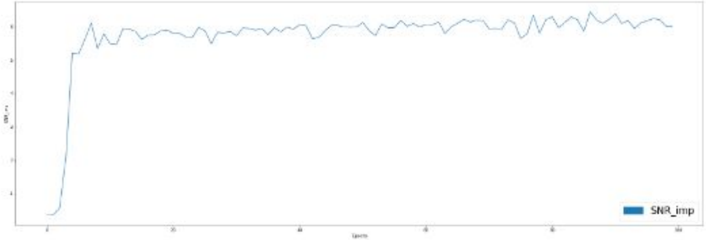
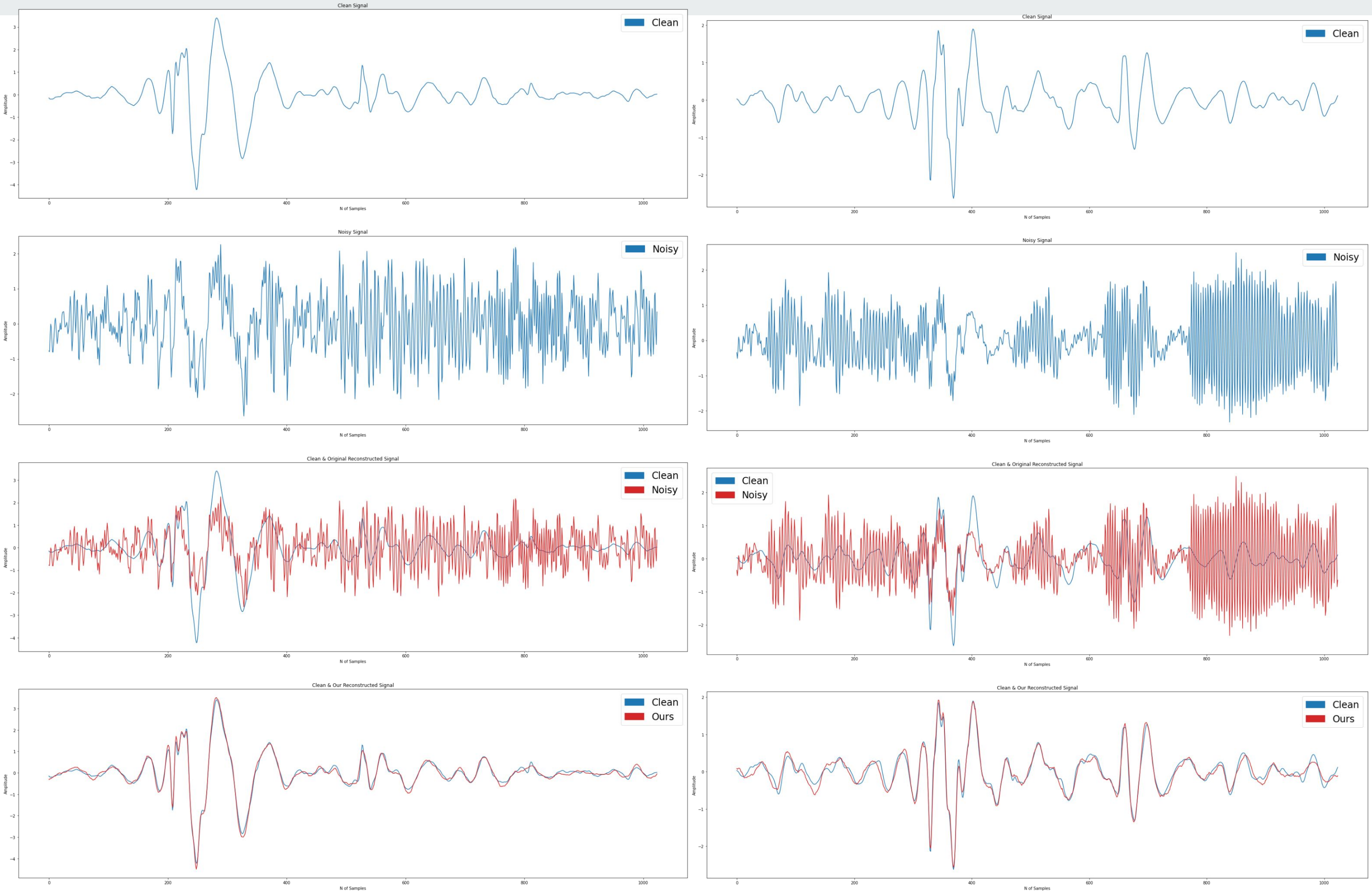


fig22. loss curve SNR_imp

3 - Entrainement du modèle sur le dataset des signaux PCG

Résultats signaux débruités:

- **meilleur cas 1**
- **meilleur cas 2**



3 - Entrainement du modèle sur le dataset des signaux PCG

Résultats signaux débruités:

- **pire cas**
- **échantillon du test set**