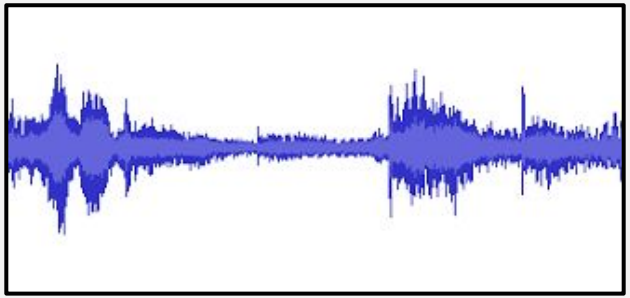
I&I - DeepLearning & Signal

Sujet : Wave-U-Net pour la séparation voix/bruit



Théo Di Piazza (theo.dipiazza@gmail.com)

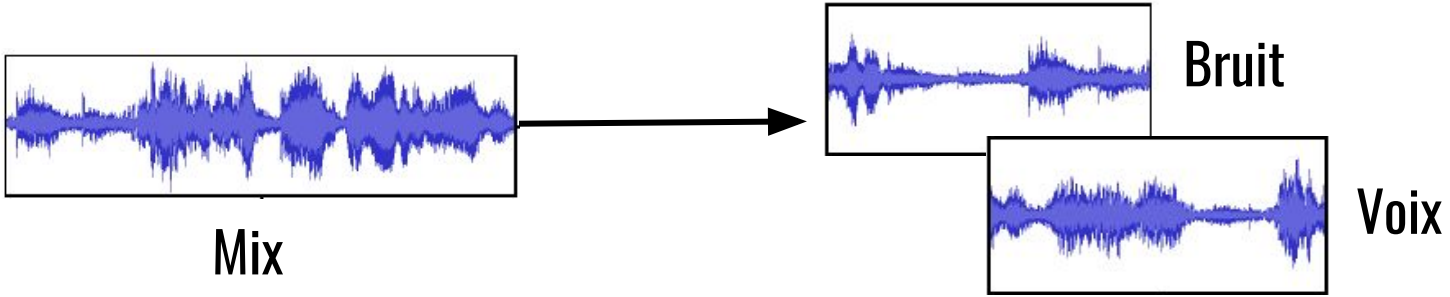
Janvier 2023

Plan

1. Introduction, contexte
2. Jeu de données
3. Théorie : Wave-U-Net
4. Entraînement : Wave-U-Net
5. Résultats Qualitatifs
6. Conclusion

1. Introduction, contexte

Comment extraire la voix d'un signal composé d'un voix et d'un bruit ?



Exemples d'utilisation d'amélioration de la voix :

- Pour la reconnaissance vocale
- Pour la reconnaissance de la parole
- Pour la qualité des vocaux

2. Jeu de données

Données d'entrée : un mix du bruit et de la voix voix.

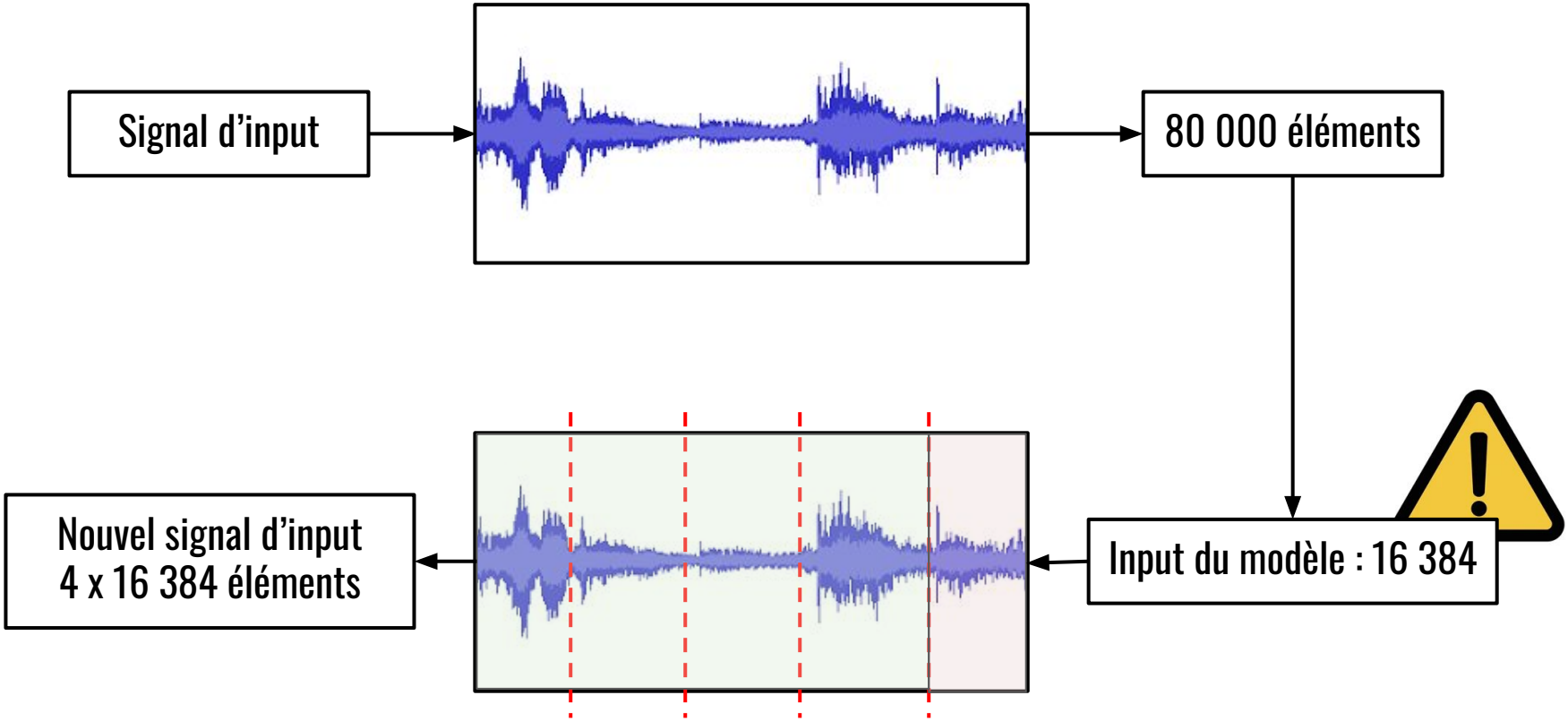
bruit + voix = mix

$$\text{Signal Normalisé} = \frac{\text{Signal}}{\| \text{Signal} \|}$$

$$\alpha \text{ bruit} + \text{voix} = \text{mix}$$

α : Coefficient compris entre 0.6 et 0.9

2. Jeu de données



2. Jeu de données

Train set *Class TrainDataset* 8224 éléments	Test set *Class TestDataset* 1248 éléments

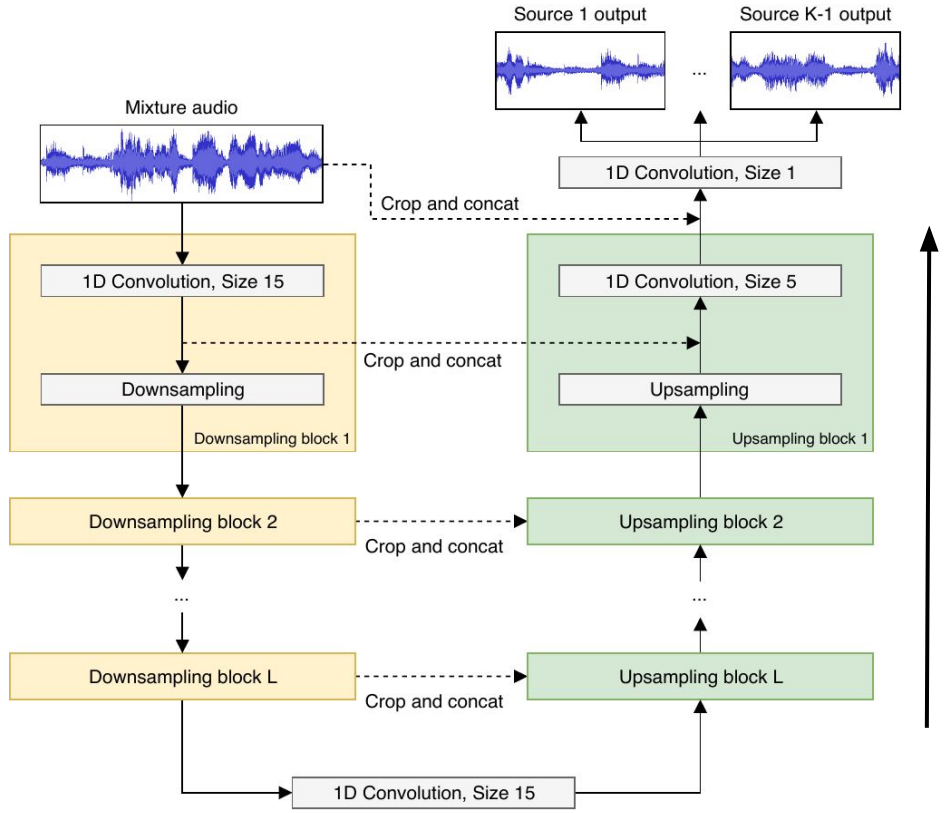
Pour chaque échantillon du jeu de données :

INPUT : Mix Signal : 16 384 impulsions ~ 3 secondes	OUTPUT : Voix Signal : 16 384 impulsions ~ 3 secondes

3. Résolution : Wave-U-Net

Downsampling
Réduction de dimension
Extraction de caractéristiques

```
# Block for DownSampling
for layer in range(self.nb_layers):
    # Encoder layer - DownSampling
    output = self.encoder[layer](output)
    tmp.append(output)
    # Reshape tensor
    output = output[:, :, ::2]
```



Upsampling
Augmentation de dimension
Reconstruction du signal

```
# Block for UpSampling
for layer in range(self.nb_layers):
    output = F.interpolate(output, scale
    # Skip Connection - Decoder layer
    output = torch.cat([output, tmp[self
    output = self.decoder[layer](output)
```

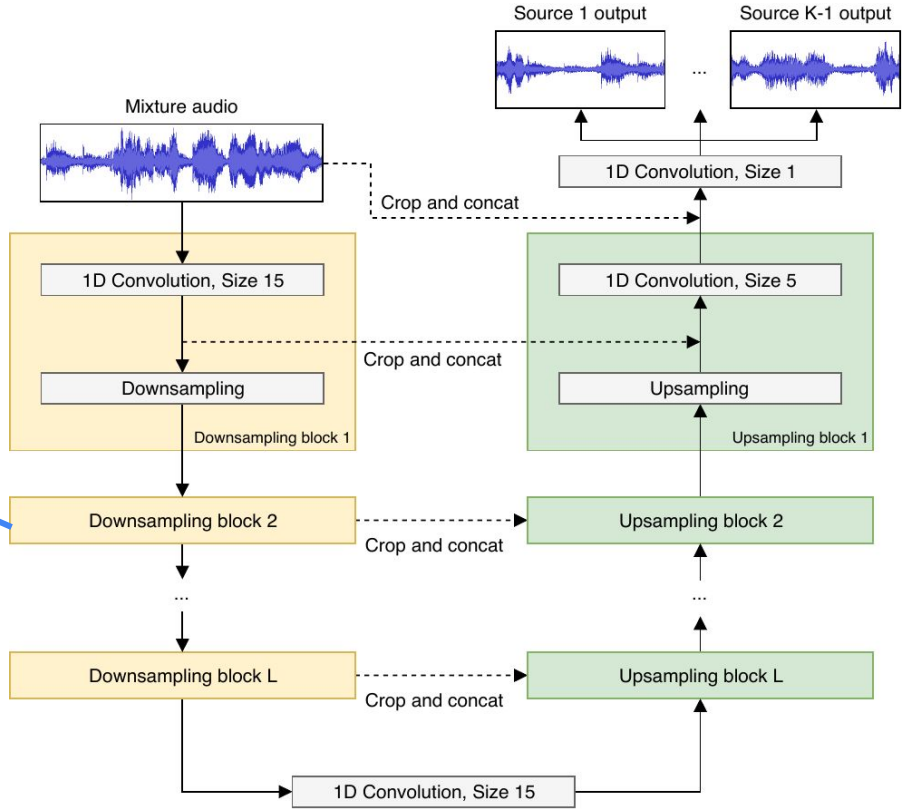
Wave-U-Net : https://arxiv.org/pdf/2002.05709.pdf

3. Résolution : Wave-U-Net

Block de Downsampling

```python
def forward(self, input):
    '''
    Forward propagation of the DownSamplingLayer.
    Input:
      input: input of the layer.
    Return:
      output: output of the layer.
    '''
    # Convolution layer
    output = self.conv(input)
    # BatchNorm after conv. layer
    output = self.batchnorm(output)
    # Activation function: LeakyReLU
    output = self.relu(output)

    return output
```



Wave-U-Net : https://arxiv.org/pdf/2002.05709.pdf

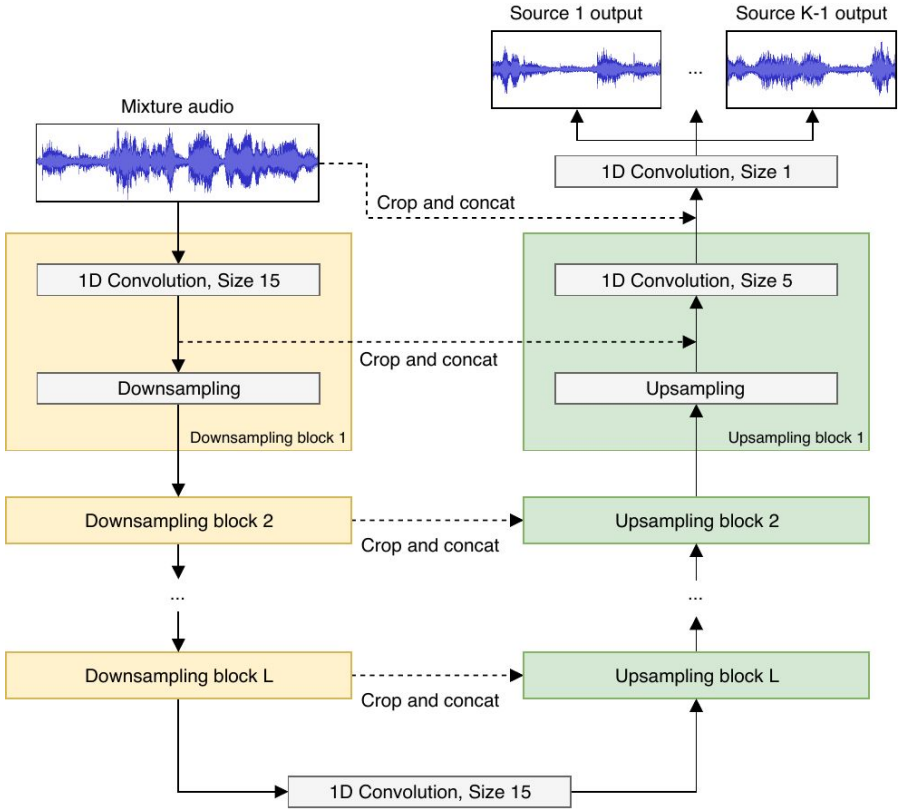
3. Résolution : Wave-U-Net

Interval Channels

Filtre appliqué à des intervalles de temps fixes

Exemple

- Pour un signal défini par :
  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- Convolution classique :
  [(0 + 1 + 2), (1 + 2 + 3), ..., (7 + 8 + 9)]
- Convolution par interval channels :
  [(0 + 3 + 6), (1 + 4 + 7), ..., (3 + 6 + 9)]



Wave-U-Net : https://arxiv.org/pdf/2002.05709.pdf

4. Entraînement : Wave-U-Net

5 architectures seront testées

	Layers	Interval Channels
Modèle 1	6	24
Modèle 2	10	24
Modèle 3	12	24
Modèle 4	6	12
Modèle 5	6	18

Détails

100 epochs

Adam Optimizer

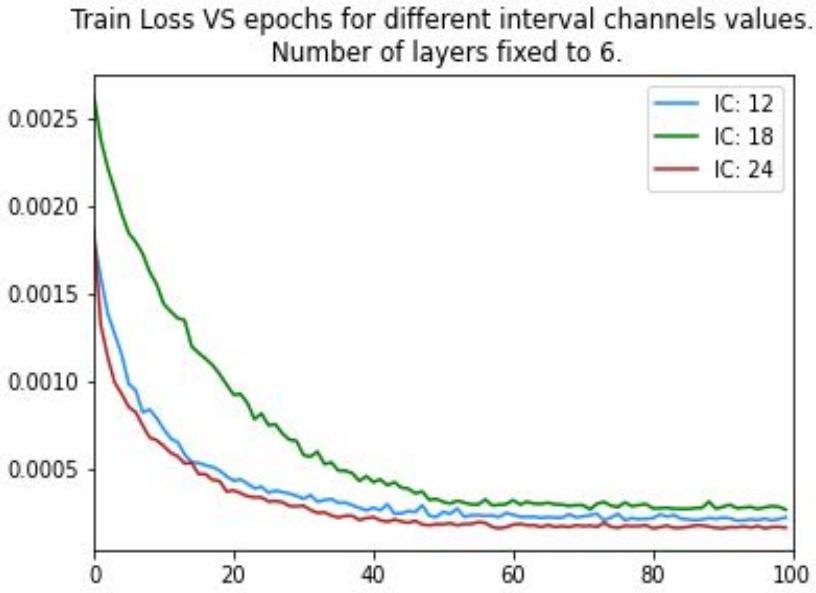
- lr pour 1-50 epochs: 0.0001
- lr pour 51-100 epochs : 0.00001

betas = (0.9, 0.999)

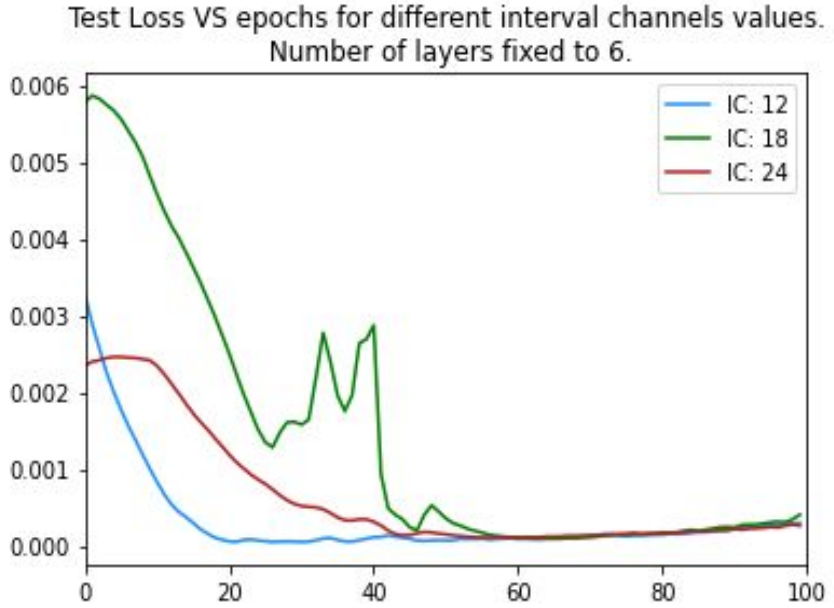
Loss : L1-Norm

4. Entraînement : Wave-U-Net

Comparaison ≠ Interval channels - Loss VS Epochs



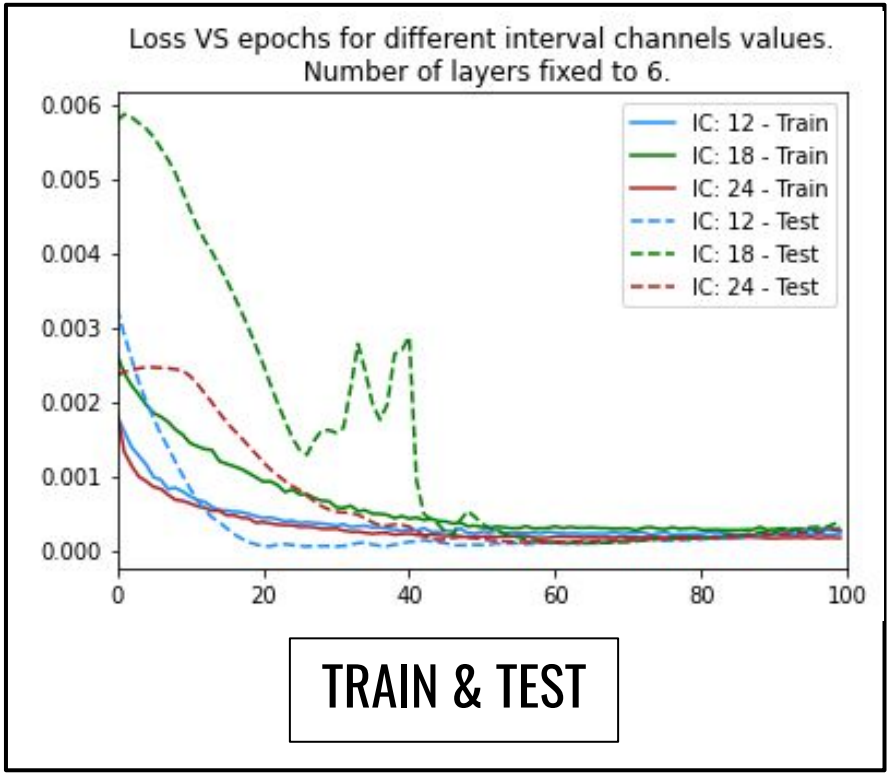
TRAIN



TEST

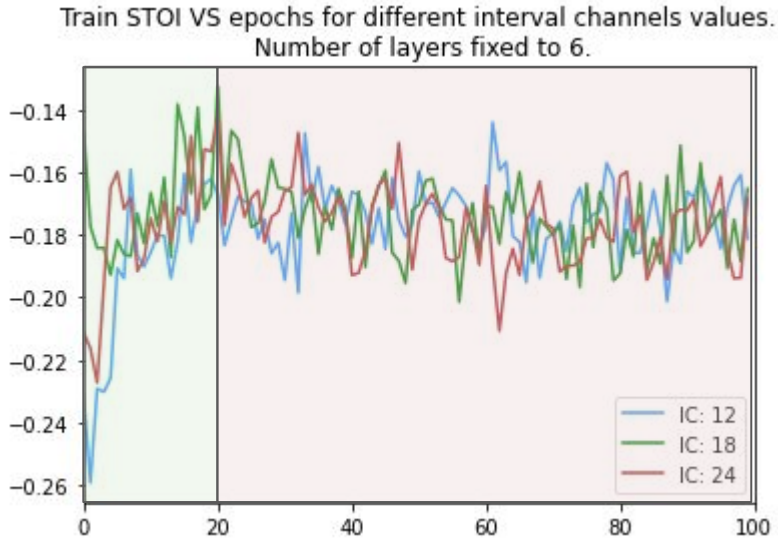
4. Entraînement : Wave-U-Net

Comparaison ≠ Interval channels - Loss VS Epochs

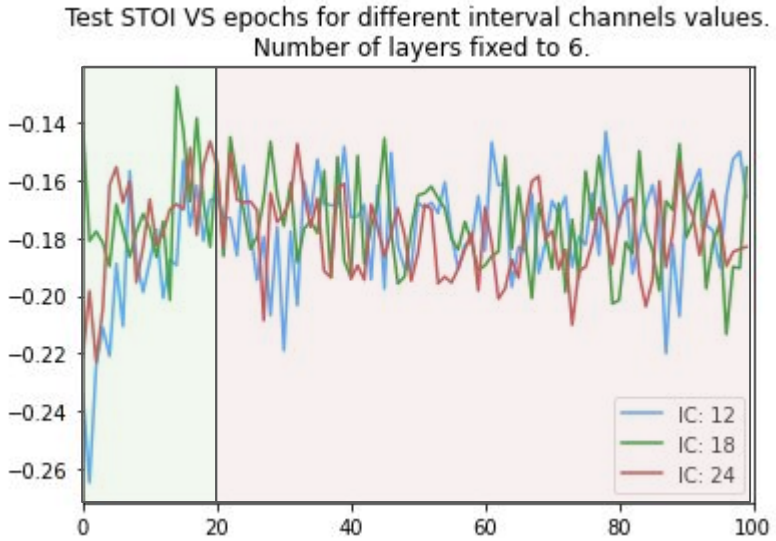


4. Entraînement : Wave-U-Net

Comparaison ≠ Interval channels - STOI VS Epochs



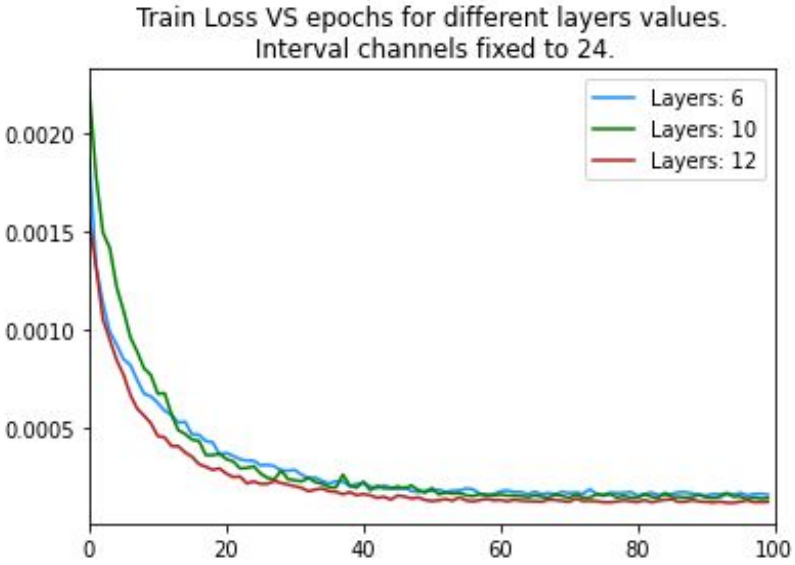
TRAIN



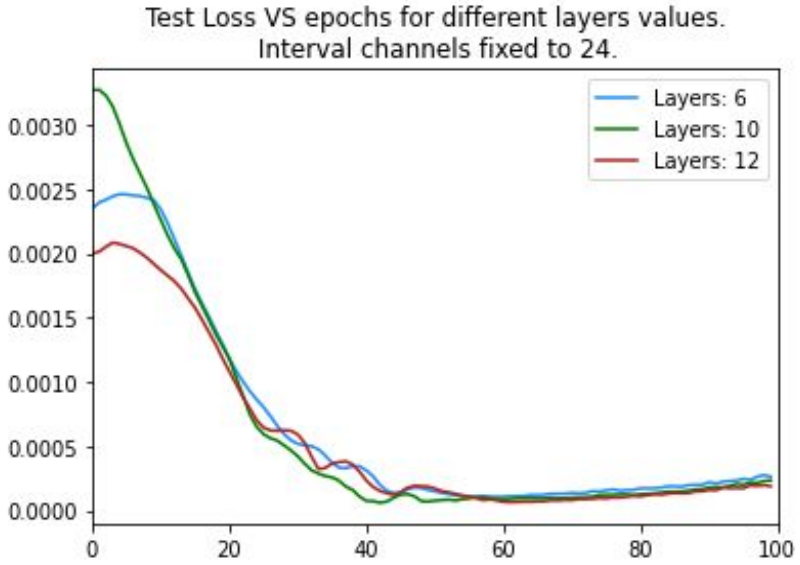
TEST

4. Entraînement : Wave-U-Net

Comparaison ≠ Layers - Loss VS Epochs



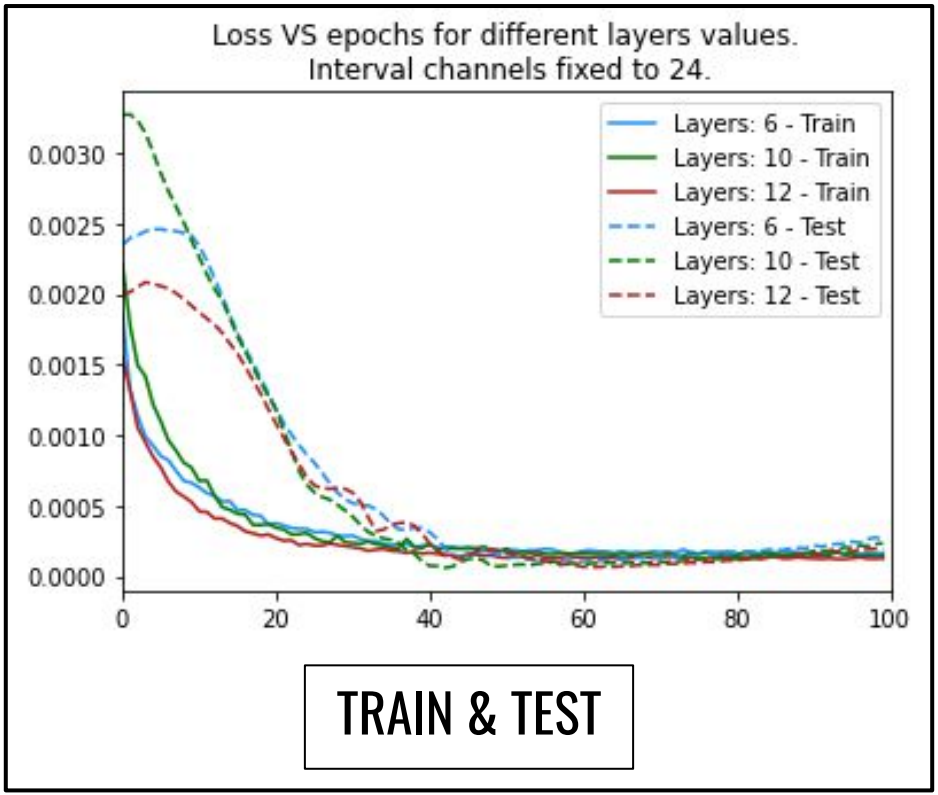
TRAIN



TEST

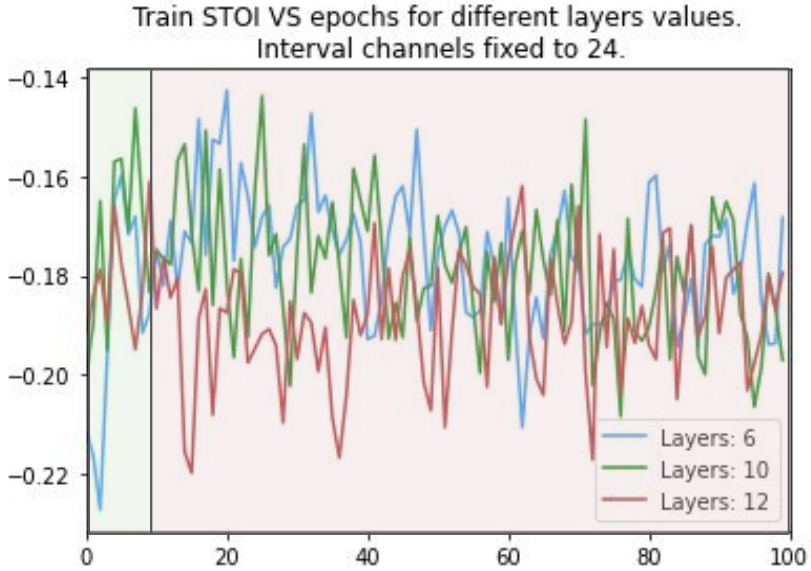
4. Entraînement : Wave-U-Net

Comparaison ≠ Layers - Loss VS Epochs

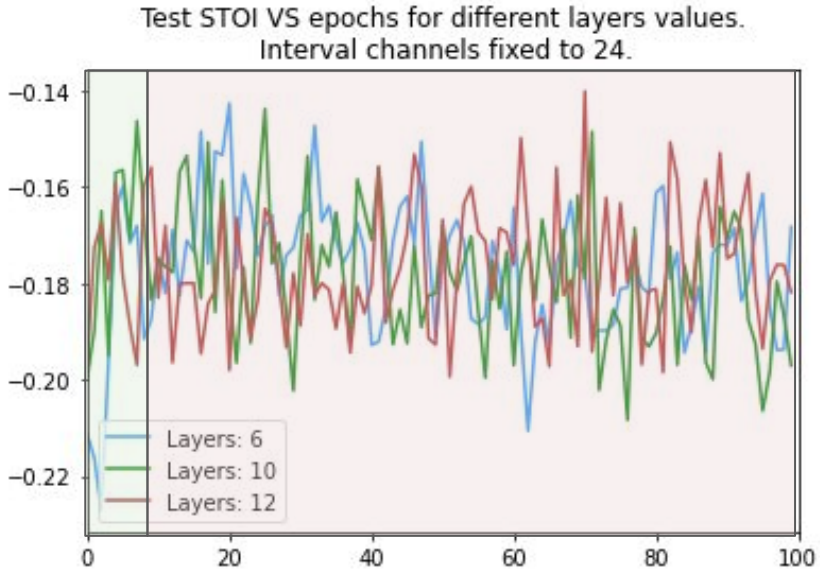


4. Entraînement : Wave-U-Net

Comparaison ≠ Layers - STOI VS Epochs



TRAIN



TEST

5. Résultats qualitatifs

	Résultat 1 : Bruit : Helsinki	Résultat 2 : Bruit : Lisbonne + Voix	Résultat 3 : Bruit : Barcelone + Rires
mix			
voix			
prediction			

6. Conclusion

Prise en main de la méthode Wave-U-Net
Importance des méthodes de Data Augmentation
Evaluation des performances via STOI
Comparaison pour plusieurs architectures

Code disponible :
https://colab.research.google.com/drive/1CQ_PhvaVYoV2VIP7YWnzxhNAO6FFmchl?usp=sharing

Amélioration des performances du modèles
⇒ ≠ Hyper-parameters
⇒ ≠ Data augmentation

Obtenir un signal reconstruit intact

Main references

Wave-U-Net : https://arxiv.org/abs/1806.03185

U-Net : https://arxiv.org/abs/1505.04597

STOI : https://arxiv.org/pdf/1806.03185.pdf