Audiosep - Separating the voice from the noise

This project report is done in the context of the Audio Separation project done at Master MVA for the lecture "Apprentissage profond et traitement du signal"
given by T. Courtat.

Yannis Kolodziej, Simon

Created on December 9 | Last edited on December 17

🔵 Introduction and contributions

Source separation is the task of isolating individual sound sources from an observed audio mixture. Its applications range from music processing, such as separating vocals and instruments in a song, to speech enhancement and military or surveillance contexts, where isolating speakers from background noise is critical.

Before deep learning, source separation methods mainly relied on signal processing and statistical modeling. Classical approaches included Independent Component Analysis (ICA), Non-negative Matrix Factorization (NMF), and probabilistic models operating in the time–frequency domain. While effective in constrained settings, their performance often degraded in real-world conditions where these assumptions were violated.

As in other signal processing domains like vision, **deep learning** marked a significant shift in audio source separation. The advances in vision, especially with the U-Net model were sucessfully adapted to the context of audio by processing the spectrograms that are also in 2D like images (**Spectro U-Net**). However, those models were still dependant of the Fourier analysis to represent the signal.

More recently, end-to-end approaches operating directly on the waveform**,** such as **Wave U-Net**, bypassed this and achieve even better performance on the new standard objective evaluation metrics like SI-SDR, PESQ and STOI.

In this work, we compare the two models evoked on our specific dataset composed of 5000 triplets of voice, noise, mix_<snr_level> and 2000 similar triplets for testing. The problem becomes a **two-source separation problem**, where one source is the voice, the other is the noise. First, we did a quick data exploration that showed scale issues in the dataset. Then we present the two models we used and how we adapted those (especially the loss function) to handle this problem. We present our experiments and conduct a quantitative and qualitative analysis. At the end, we explore **generalisation abilities** of the model on two other datasets. Regarding our contributions, both of reflect on the plan and the experiments we wanted to make, one group member focused on the first model and h-params optimization while the second did the project architecture and worked on the second model and model generalisation.

🟢 Dataset

The problem we have is to estimate the voice and the noise from the mix. Below, this is a sample of the dataset. The mix goes from -4 dB to 4 dB of SNR, which may lead to strong performance differences in the task of separating. Samples are 10s at 8 kHz. It seems normalized, amplitude values in the training_small set are between 0.2 and 0.6. The worst SNR is -8dB, whereas the best one is 1dB.

mix

Step 1

00:00/0(

Step 11

00:00/0(

voice

Step 3

00:00/0(

Step 13

00:00/0(

noise

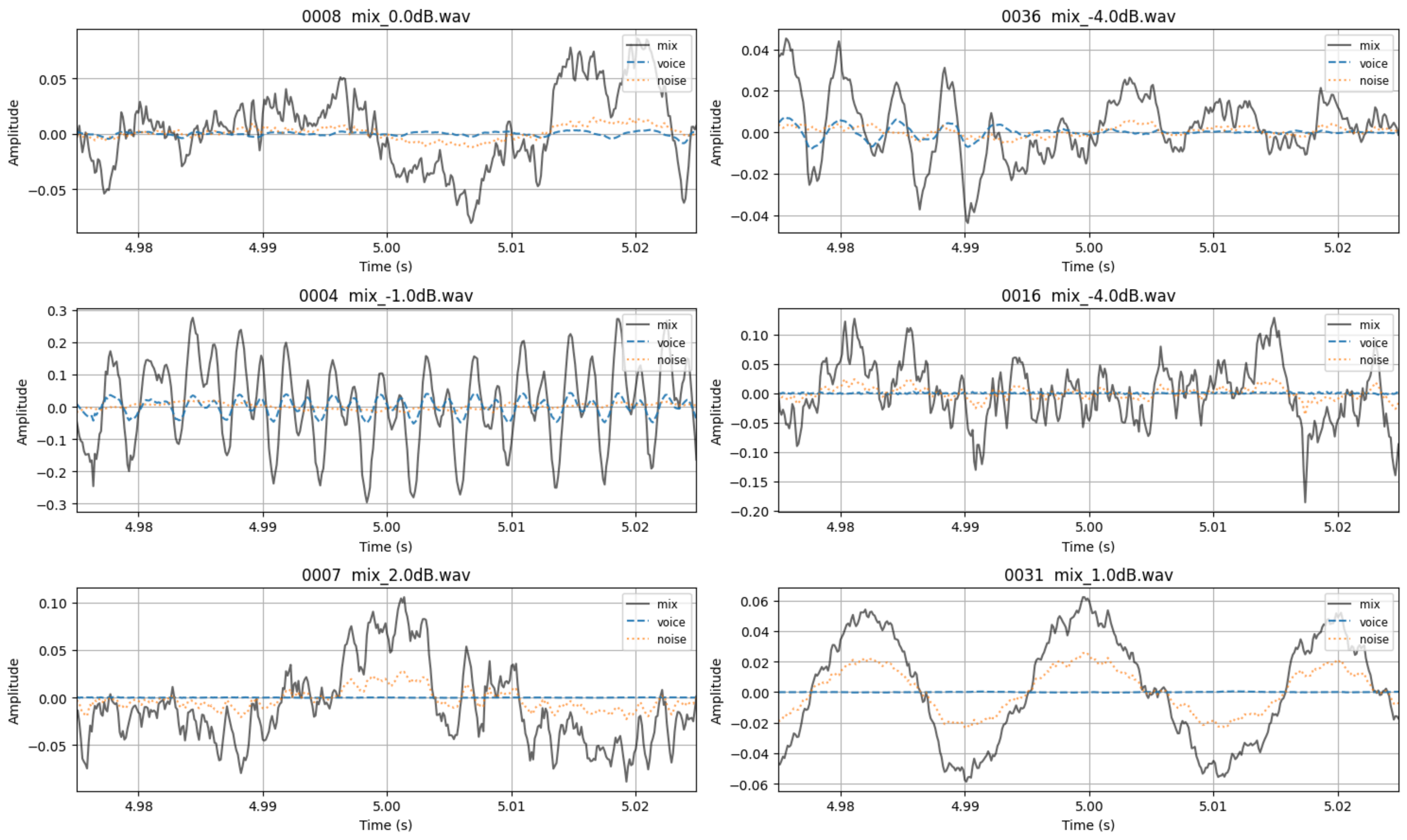
Step 0

00:00/0(

Step 10

00:00/0(

We first displayed the 3 signals, and saw that there was a **scale difference** that is particularly significant between the mix and the noise / voice.



Moreover, something to take into account is also we could call **outliers** in the signals, where voice is not present or near-silent. This could have a negative effect on metrics like **SDR**.

From those observations, we decided to keep all the samples in the signals.

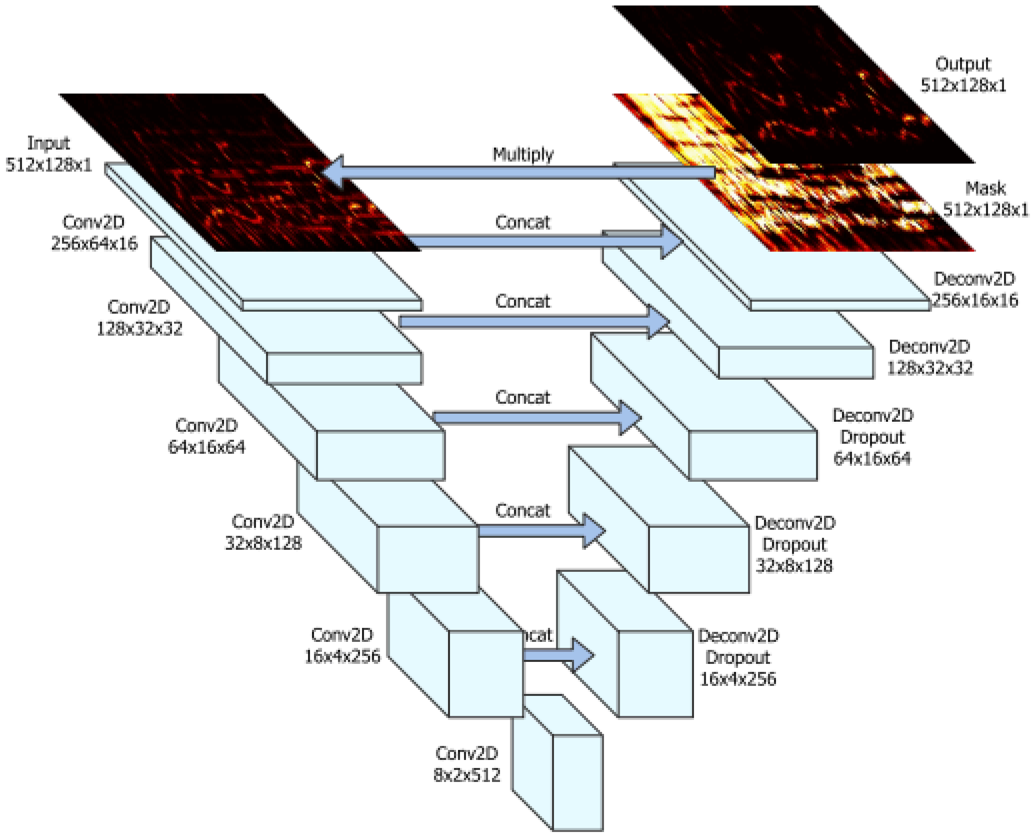
Since the mixture is the only signal that is normally available at inference time (in real conditions), we just normalise mix, voice, noise by the max of mix.

Models

We followed a classical approach of comparing two U-Net models for this task, respectively in frequency domain for the Spectrogram U-Net and time domain for the Wave U-Net.

🟢 Baseline Spectrogram U-Net

The first model we used is a spectrogram U-Net, from the original paper from Jansson et al., 2017. This is convolutional autoencoder that compresses the spectrograms and reconstructs target sources by learning masks, with skip connections preserving fine spectral details. The paper originally implemented two U-Nets to have the possibility to have different strategies in the future for the source network and the noise network, which is not our case, so we change the number of **output_channels to 2** to predict both noise and voice. **We also kept a baseline implementation in our experiments (only predict voice**).



Spectrogram U-Net architecture

For both architecture, we kept a **L1-loss** as mentioned in the paper. We didn't have time to experiment the impact of choosing another loss, but for instance we expect L2-loss to penalize more the differences, that could lead to **over-smoothing**.

$$L(X, Y; \Theta) = ||f(X, \Theta) \odot X - Y||_{1,1}$$

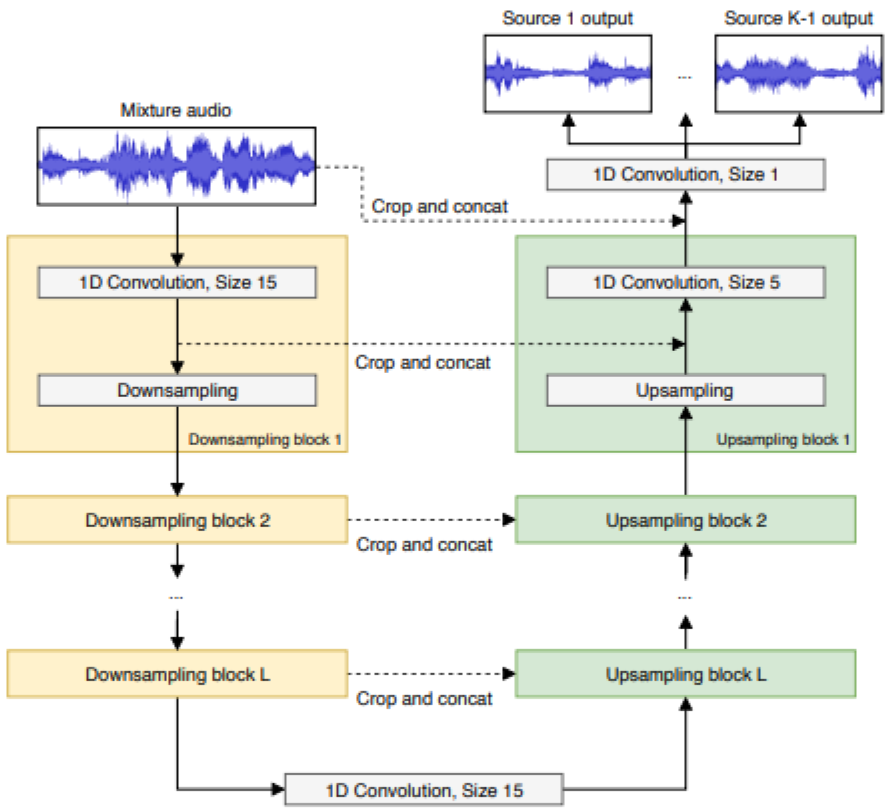
where:

- X: The magnitude of the spectrogram of the original, mixed signal

- Y : The magnitude of the spectrogram of the target audio (either the vocal or instrumental component)
- $f(X, \Theta)$: The mask generated by the network model with parameters Θ
- $\odot$: Denotes element-wise multiplication applied to the mask and the input
- $||\cdot||$1,1: The L1,1 norm, defined as the sum of the absolute values of the matrix elements

🔵 Wave U-Net

In contrast to Spectro U-Net, Wave U-Net is an autoencoder operating directly on raw waveforms.



Wave U-Net architecture

Architecture

We re-implemented the architecture from scratch based on the original paper. We didn't add the Resample layer though, due to lack of time. We also removed the TanH layer at the end since the output was not constrained between -1 and 1. The code has been made generic for further HP optimization.

Context aware prediction

Something that authors add in their implementation that we reused is context aware prediction. The model takes a larger input window than the region it predicts, using surrounding temporal context to **avoid border artifacts** caused by convolution and downsampling. **Only the central part of the output is supervised and kept**, ensuring reliable predictions while discarding boundary-affected samples.

Speech aware chunking

We adopted for Wave-U-Net a **speech-aware chunk selection strategy** where we only sample chunks from the signal that have a sufficient voice energy. This avoids to get random crops of the signal that would not bear significant information.

Choice of the loss

In original Wave-U-Net, authors used a **MSE-loss** to train the model, but in our case this will be a problem. Indeed, the MSE-loss is expressed like this:

$$\mathrm{MSE}(\hat{x}, x) = \frac{1}{T}\sum_t (\hat{x}_t - x_t)^2$$

But if we add a scaling factor α to both signal and mix, the MSE loss changes (whereas the shape remains unchanged) $\mathrm{MSE}(\alpha\hat{x}, \alpha x) = \frac{1}{T}\sum_t (\alpha\hat{x}_t - \alpha x_t)^2 = \alpha^2\,\mathrm{MSE}(\hat{x}, x)$.

Consequently, the MSE-loss might indeed learn waveform basics statistics like means instead of the shape of the signal.

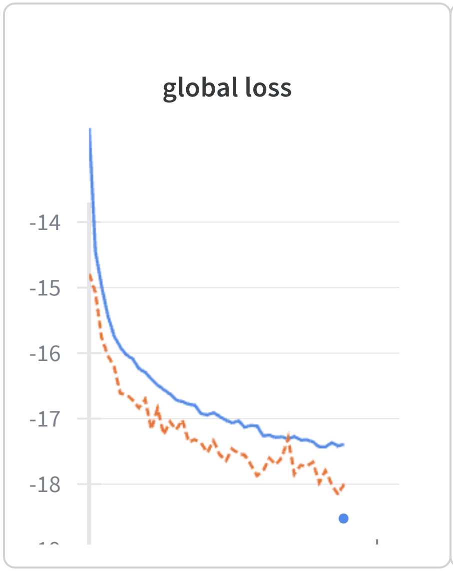
To address this issue, we consider the **Scale-Invariant Signal-to-Noise Ratio (SI-SNR)**, which is explicitly invariant to global amplitude scaling and therefore better suited to source separation tasks. However, when used alone as a loss function, SI-SNR does not constrain the absolute amplitude of the reconstructed signals. In practice, we observed that this can lead to amplification effects in the model outputs. While this may be acceptable depending on the application, we opted to introduce an additional constraint by adding a **mixture-consistency penalty based on the MSE** between the reconstructed mixture ($\hat{v} + \hat{n}$) and the input mixture x.

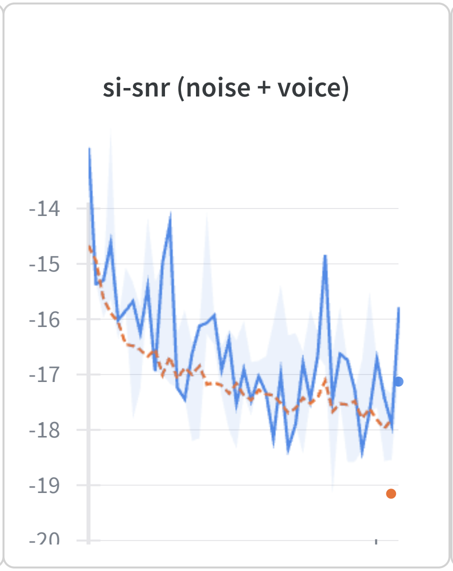
Consistent with the dataset specifications, we supervise both the voice and the noise components. An alternative approach would be to estimate only the voice signal and treat the noise as a residual. However, supervising only the voice introduces additional degrees of freedom: the noise estimate $\hat{n}$ can become arbitrary as long as $\hat{v} + \hat{n} \approx x$, and the model may also arbitrarily **swap the output channels.** This ambiguity is clearly reflected in unstable training behaviour.

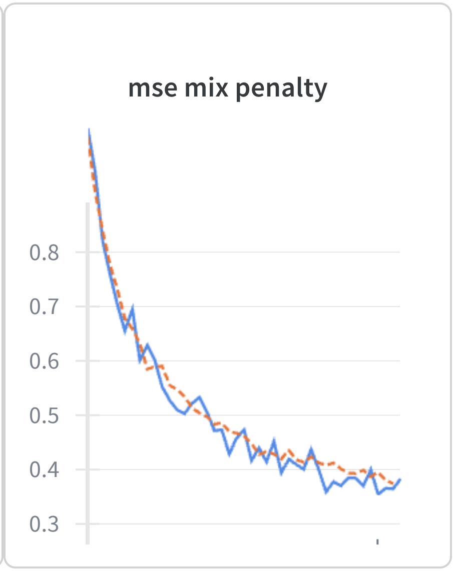
To mitigate this issue, we tried (once, removed for code claritity) **Permutation Invariant Training (PIT)**, which resolves the source–output assignment ambiguity by selecting the permutation that minimizes the loss. Additionally, we introduce a **gating mechanism on the SI-SNR** computation to ignore silent or near-silent segments, as SI-SNR becomes ill-defined in such cases due to the $\log(0)$behaviour.

$$\mathcal{L} = -\text{SI-SNR}(\hat{v}, v) - \lambda_n \text{SI-SNR}(\hat{n}, n) + \lambda \left\| (\hat{v} + \hat{n}) - x \right\|^2$$

with $\hat{v}$ the reconstructed voice and $\hat{n}$ the reconstructed noise, λ the coefficient of the penalisation. We fix λ_n, λ as hyperparameters controlling the relative weight of each term. As illustrated in our experiments, omitting PIT leads to highly unstable SI-SNR behaviour, confirming the necessity of permutation-invariant supervision.







🟢 Implementation

Concerning the software stack, we relied on two software utility :

- **Pytorch Lightning**, we can be seen as a Pytorch wrapper that helps to organize and design reproducible ML projects. It implements for instance a Trainer class that does all the training logic based on methods you can implement in the module class. It has also native logging, callbacks for batch_end or test_end stages that avoid to mix everything together. Additionally, it is made to be cross-platform so we didn't have to adapt the code to run on GPU or CPU.
- **WandB,** which is a standard of ML experiment tracking.

For **training**, at the moment we felt confident in our implementation, we used **wandb sweep** for h param optimization and configured our models so that number of layers, filters, and some other parameters related to our specific implementations (such as λ in wave unet) were arguments.

The search of h-params was run locally, so we add to make choices.

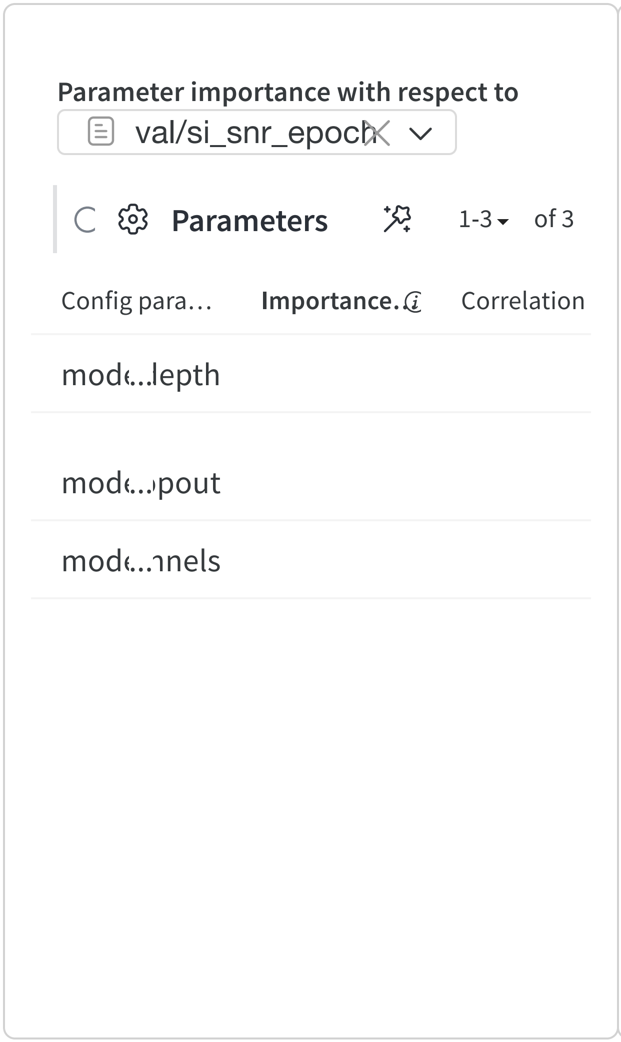
For **evaluation,** we chose to compute all the metrics on the test set and log it to wandb. Our implementation enable to configure a testing configuration (.yaml) file and run the test. We have one config for each model where we specified arguments such as the checkpoint, datamodule used etc. We logged also the Audio to WandB so that we can associate one run with the model performance.

Test table example

Results

🟢 Hyper parameter tuning

Spectogram unet tuning (random search), maximizing si_snr epoch

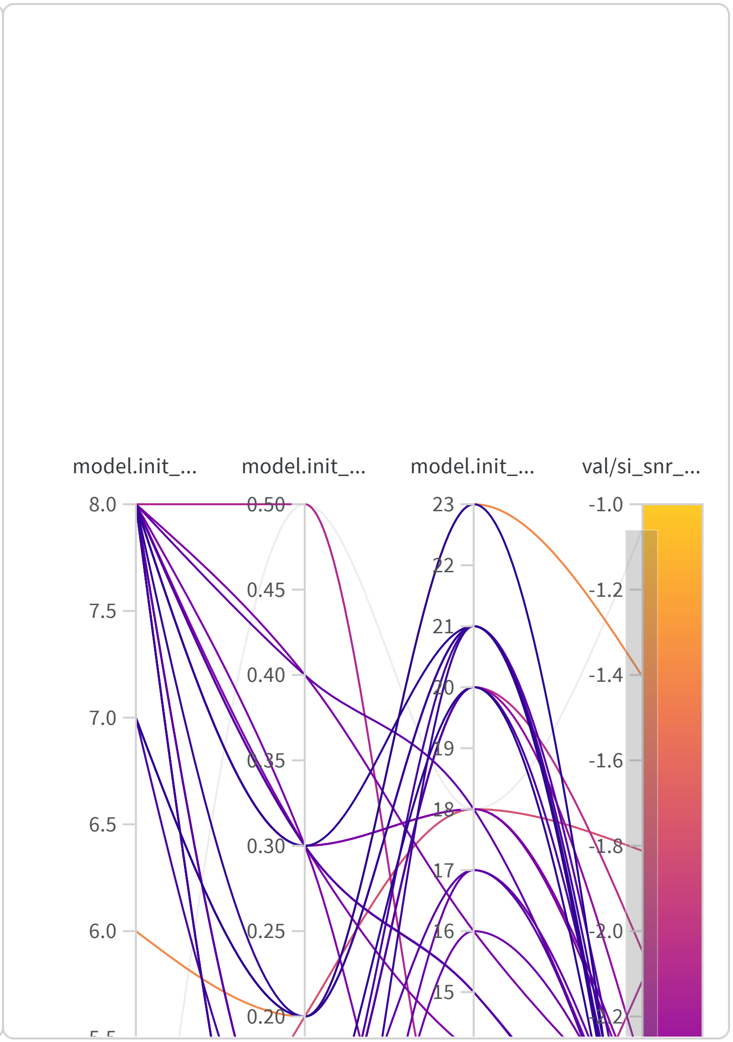


Parameter importance with respect to

val/si_snr_epoch

Parameters 1-3 of 3

Config para…	Importance.	Correlation
mode…lepth		
mode…pout		
mode…nnels		



🔵 Model performance comparison on multiple runs

We recall the definition of the metrics we used:

SNR: Measures how much signal power dominates error power, sensitive to amplitude scaling. Penalize silent segments. Baseline

$$\mathrm{SNR}(\hat{x}, x) = 10\log_{10}\frac{\|x\|^2}{\|\hat{x}-x\|^2}$$

SDR: Measures overall distortion relative to the true source. Accounts for interferences, noise and artifacts. Sensitive to scale, undefined for silent targets. Standard

$$\mathrm{SDR} = 10\log_{10}\frac{\|x_{\text{target}}\|^2}{\|e_{\text{interf}}+e_{\text{noise}}+e_{\text{artif}}\|^2}$$

with $\hat{x} = x_{\text{target}} + e_{\text{interf}} + e_{\text{noise}} + e_{\text{artif}}$

SI-SNR: measures the quality of the separated speech while ignoring differences in the scale (amplitude) of the signal.

It's also differentiable and can be used as a **training loss**.

Given an estimated clean source $\hat{\mathrm{s}} \in \mathbb{R}^{1\times \mathrm{t}}$ and a target clean source $\mathrm{s} \in \mathbb{R}^{1\times \mathrm{t}}$ (where t is the signal length), the calculation follows these steps:

Both $\hat{\mathrm{s}}$ and s are normalized to have zero-mean to ensure scale-invariance, in the same way that SI-SDR does.

Calculate the target signal component ($\mathrm{s}_{\text{target}}$) by projecting the estimated signal onto the target signal:

$$\mathrm{s}_{\mathrm{target}} = \frac{\langle \mathrm{x}_{\mathrm{proj}}, \mathrm{x}_{\mathrm{target}} \rangle \mathrm{x}_{\mathrm{target}}}{\|\mathrm{x}_{\mathrm{target}}\|^2}$$

Calculate the noise error component ($\mathrm{e}_{\mathrm{noise}}$) by subtracting the target projection from the estimated signal:

$$\mathrm{e}_{\mathrm{noise}} = \mathrm{x}_{\mathrm{proj}} - \mathrm{s}_{\mathrm{target}}$$

Compute the ratio of the target signal power to the noise power in decibels:

$$\text{SI-SNR} := 10 \log_{10} \frac{\|\mathrm{s}_{\mathrm{target}}\|^2}{\|\mathrm{e}_{\mathrm{noise}}\|^2}$$

SI-SDR: measures the quality of a source separation estimate ($\hat{\mathrm{s}}$) by comparing it to the ground truth target (s). Unlike standard SNR or SDR, it first aligns the scale of the target to the estimate via orthogonal projection to ensure the metric is invariant to the gain (volume) of the output.

It involves decomposing the estimated signal $\hat{\mathrm{s}}$ into two orthogonal components: the target signal scaled by α ($\mathrm{e}_{\mathrm{target}}$) and the residual distortion/noise ($\mathrm{e}_{\mathrm{res}}$).

Ensure both signals have zero-mean to guarantee scale invariance:

$$\begin{gathered}\mathrm{s} \leftarrow \mathrm{x}_{\mathrm{target}} - \mathrm{mean}(\mathrm{s}) \\ \hat{\mathrm{s}} \leftarrow \mathrm{x}_{\mathrm{proj}} - \mathrm{mean}(\hat{\mathrm{s}})\end{gathered}$$

Find the scalar α that minimizes the residual energy (orthogonal projection of $\hat{\mathrm{s}}$ onto s):

$$\alpha = \frac{\hat{\mathrm{s}}^{\mathrm{T}} \mathrm{s}}{||\mathrm{s}||^2} = \frac{< \hat{\mathrm{s}}, \mathrm{s} >}{||\mathrm{s}||^2}$$

Calculate the scaled target component and the noise residual component:

$$e_{\text{target}} = \alpha s$$
$$e_{\text{res}} = \hat{s} - e_{\text{target}}$$

The SI-SDR in decibels (dB) is the logarithmic ratio of the energy of the scaled target to the energy of the residual error:

$$\text{SI-SDR} = 10 \log_{10} \left(\frac{||e_{\text{target}}||^2}{||e_{\text{res}}||^2} \right)$$

STOI: computes the average linear correlation between short-time temporal envelopes of the reference and estimate.
https://lightning.ai/docs/torchmetrics/stable/audio/short_time_objective_intelligibility.html

$$\text{STOI} = \frac{1}{N} \sum_{k,n} \text{corr}\left(E_x(k,n), E_{\hat{x}}(k,n)\right)$$

PESQ: takes into considerations characteristics such as: audio sharpness, call volume, background noise, clipping, audio interference etc. PESQ returns a score between -0.5 and 4.5.

$$\text{PESQ} = f\left(P(x), P(\hat{x})\right)$$

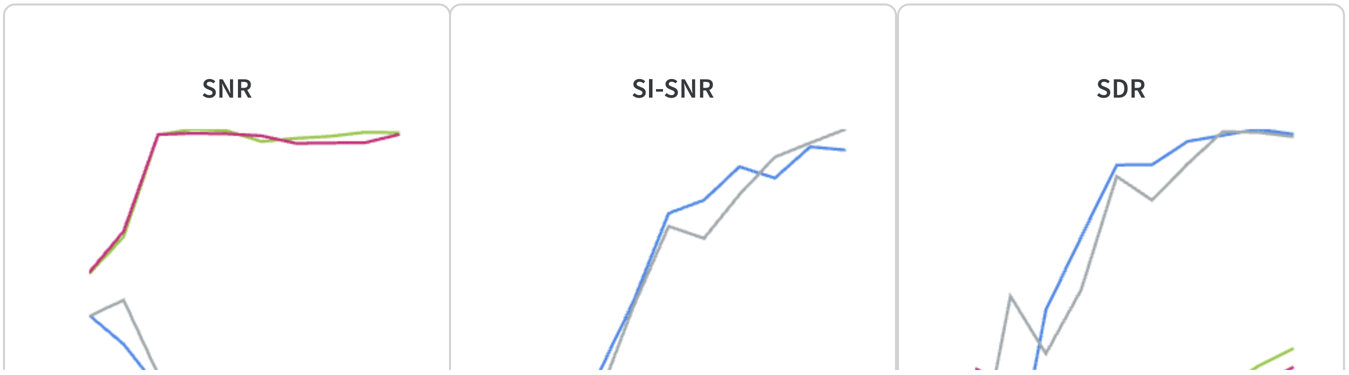
where $P(\cdot)$ is a psychoacoustic model of human hearing

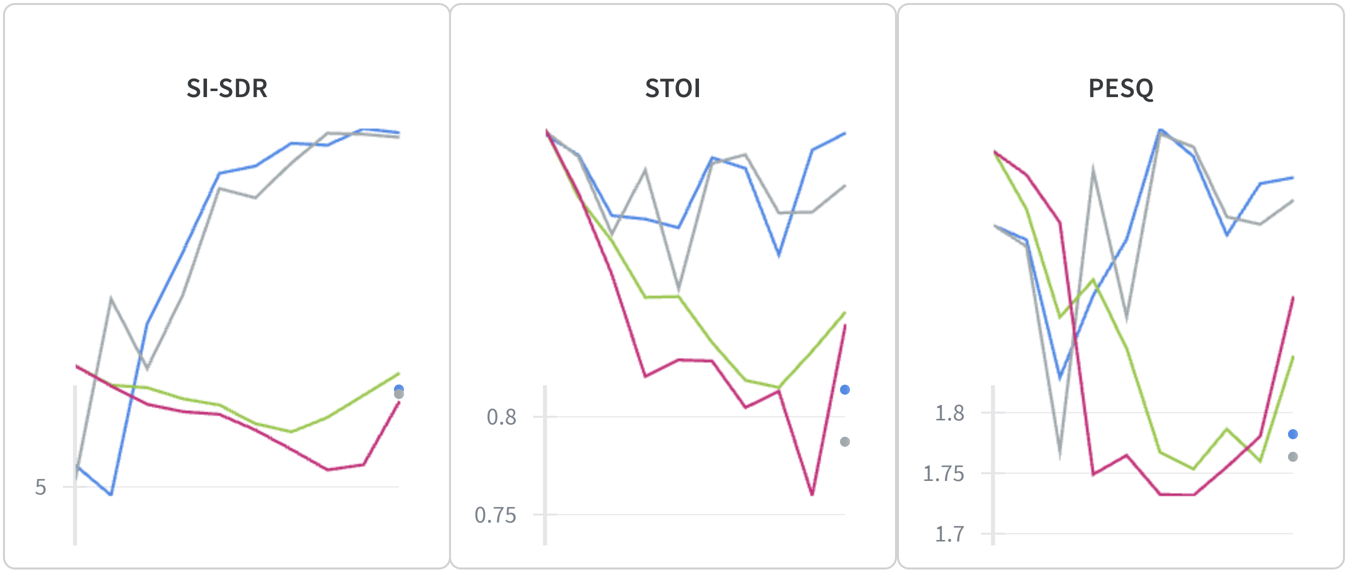
Here is a quick recap:

Metric	Scale-Invariant	Perceptual	Differentiable	Typical use	Equation

SNR	🔴	🔴	🟢	Baseline	$\mathrm{SNR}(\hat{x}, x) = 10 \log_{10} \frac{\|x\|^2}{\|\hat{x}-x\|^2}$
SDR	🔴	🔴	🔴	Baseline	$\mathrm{SDR} = 10 \log_{10} \frac{\|x_{\mathrm{target}}\|^2}{\|e_{\mathrm{interf}}+e_{\mathrm{noise}}+e_{\mathrm{artif}}\|^2}$
SI-SNR	🟢	🔴	🟢	Loss	$\mathrm{SI\text{-}SNR} := 10 \log_{10} \frac{\|s_{\mathrm{target}}\|^2}{\|e_{\mathrm{noise}}\|^2}$
SI-SDR	🟢	🔴	🔴	Evaluation	$\mathrm{SI\text{-}SDR} = 10 \log_{10} \left(\frac{\|e_{target}\|^2}{\|e_{res}\|^2} \right)$
STOI	🟢	🟢	🔴	Intelligibility	$\mathrm{STOI} = \frac{1}{N} \sum_{k,n} \mathrm{corr}\left(E_x(k,n), E_{\hat{x}}(k,n)\right)$
PESQ	🟢	🟢	🔴	Speech quality	$\mathrm{PESQ} = f\left(P(x), P(\hat{x})\right)$

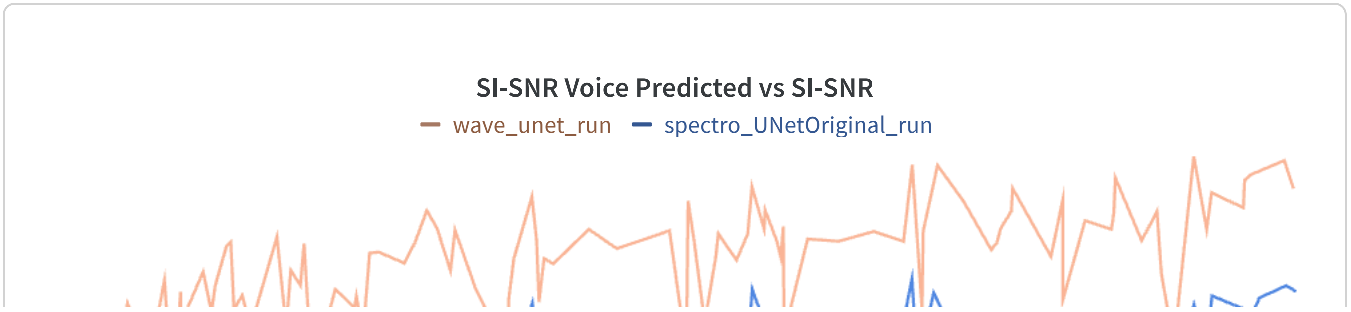
Here is the performance of our models evaluated on the test set.

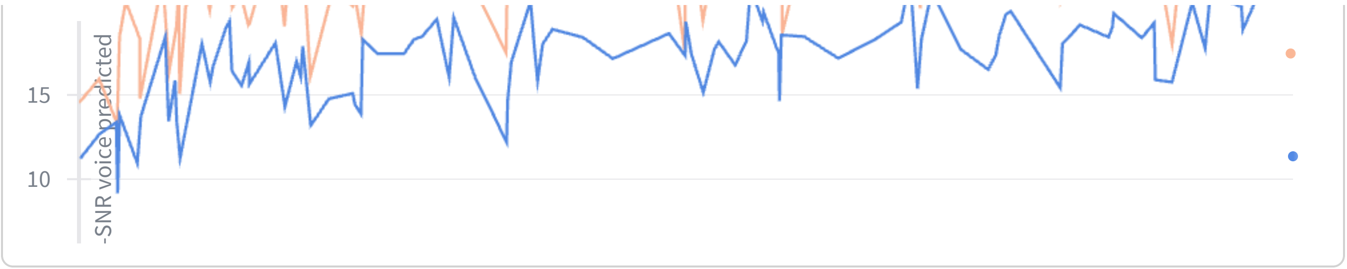


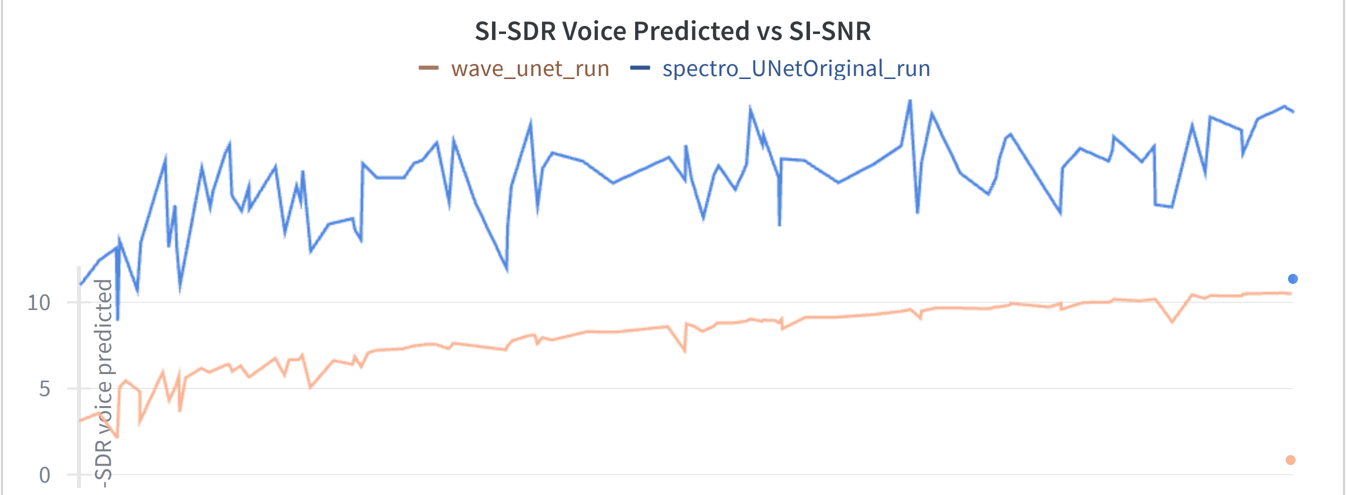


🟢 Impact of SNR levels

For quantitative evaluation, we focus on **SI-SDR** and **SI-SNR** metrics since they are scale-invariant.







Plot of SI-SDR and SI-SNR over 100 sample of the test dataset

```
[runs.summary["test_waveunet_table"], runs.summary["test_table"]].concat
```

Filter

	predicted_voice	target_voice	mix	predicted_noise
1	00:	00:		

First sample of each model in the test runs (hyperparameters test and retry runs are excluded)

Observations and analysis

- We see that both metrics increase with SNR-level as expected.
- Wave U-Net is better for SI-SNR and Spectro Skip better for SI-SDR. That's because Wave U-Net optimize for this metric
- Amplitude out of Wave U-Net is stronger
- Spectro Original cuts the amplitude of the signal

Qualitative analysis

Here we can find the results of one sample for the 3 architectures:

🟢 Spectrogram U-Net

🔵 Wave U-Net

🔵 Generalisation abilities

To evaluate the generalisation capabilities of our model, **we constructed a synthetic test** dataset by mixing clean voice and noise from the original datasets at controlled signal-to-noise ratios (SNRs) (the same as test_set in order). We record a voice reading a text from Pline l'Ancien, Histoire Naturelle (random book found on the library), sliced and resampled to 10s chunks of 8Khz.

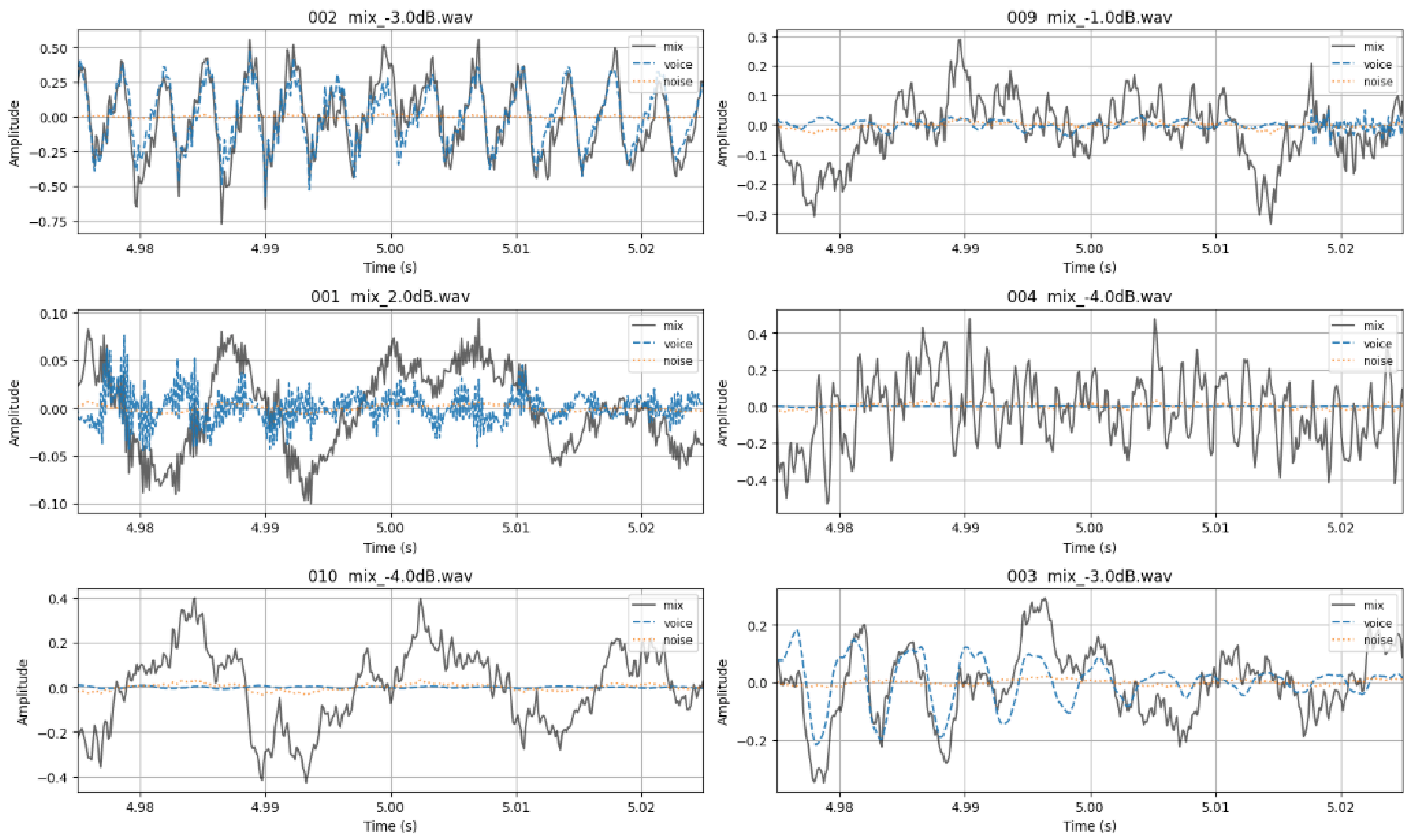
For each mixture, the noise signal is scaled such that the ratio between the voice power and the noise power matches a desired target SNR, before summation. Finally, the resulting mixture is peak-normalized to avoid clipping. We took the same snr levels (sorted) than the test set of the original dataset, so that we could compare the performance.

The scaling factor applied to the noise is computed as:

$$\alpha = 10^{-\frac{\mathrm{SNR}}{20}} \cdot \frac{\|v\|_2}{\|n\|_2},$$

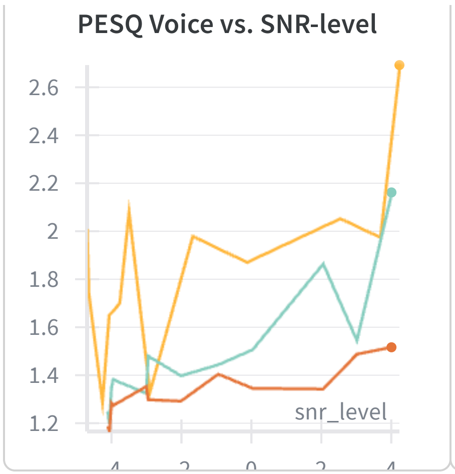
where v and n denote the clean voice and noise signals, respectively, and the mixture is obtained as $x = v + \alpha n$.

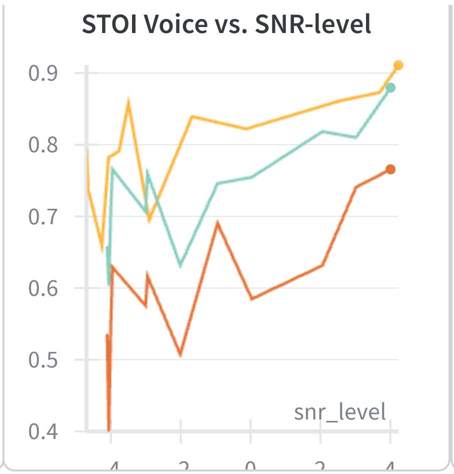
By employing this technique, we obtain similar waveforms:

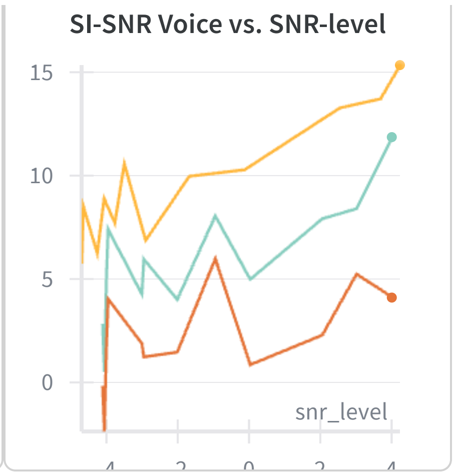


Here are the results:

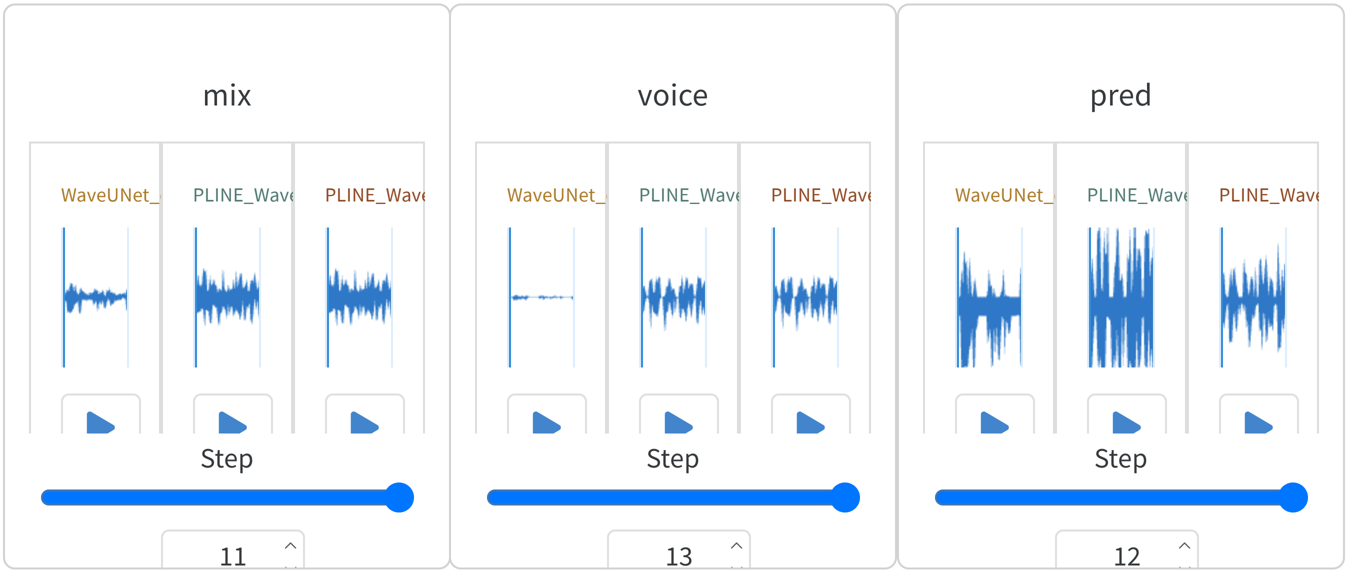
- the yellow curve represent a baseline on the original dataset,
- the green (blue) is the same model architecture on the new dataset,
- and the orange is a smaller architecture.


PESQ Voice vs. SNR-level
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
snr_level
-4
-2
0
2
4

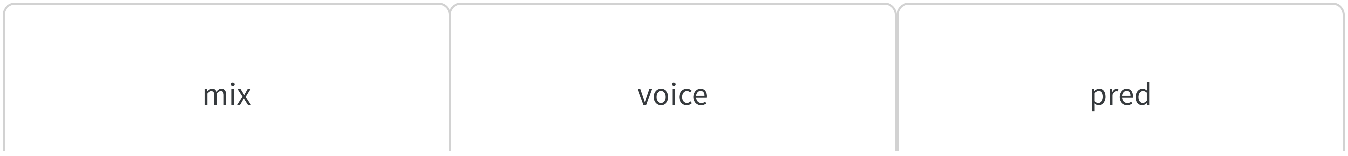

STOI Voice vs. SNR-level
0.9
0.8
0.7
0.6
0.5
0.4
snr_level
-4
-2
0
2
4

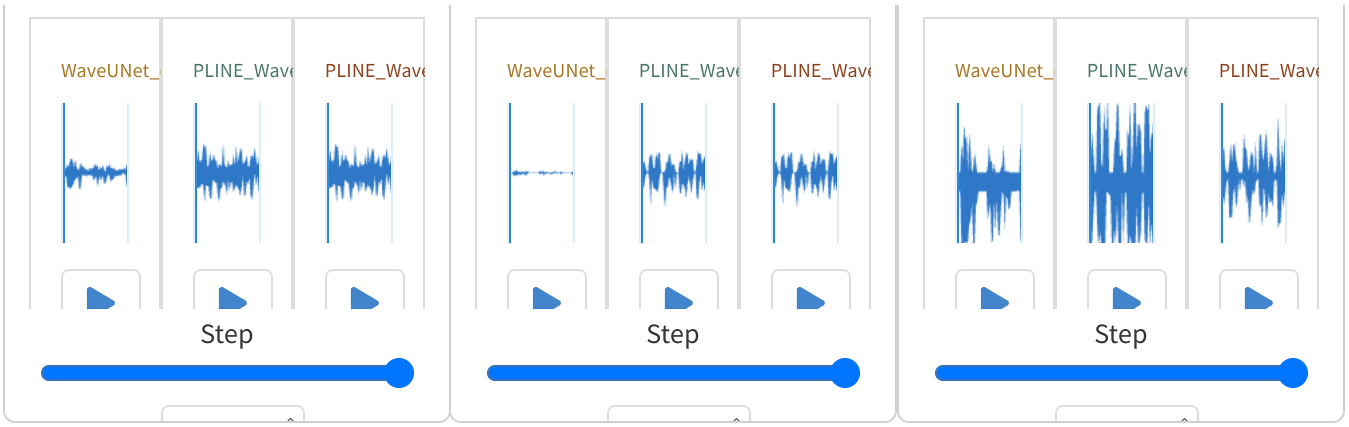

SI-SNR Voice vs. SNR-level
15
10
5
0
snr_level
-4
-2
0
2
4

At **-4 dB:**


mix
WaveUNet_
PLINE_Wave
PLINE_Wave
Step
11
voice
WaveUNet_
PLINE_Wave
PLINE_Wave
Step
13
pred
WaveUNet_
PLINE_Wave
PLINE_Wave
Step
12

At **4 dB:**


mix
voice
pred



Compared to the baseline (yellow), the same model architecture (PLINE green) is performing quite well on metrics, especially for STOI. However, the listening give other results : the orange model is better at reconstructing the voice (less noisy residuals). This might be due to model overfitting, and remains the **importance of qualitative analysis** in addition to metric-base judgment.

🟢 Conclusion

Future work

- Data augmentation (e.g, mixing with a different SNR)
- Leveraging hyper parameters founds using WandB Sweep
- An online version of the algorithm for real time noise suppression
- Extracting noise as well (the model is already trained for it, but we used voice as the basic metric for comparing results)
- Normalizing the volume between each model (U-net with spectrogram is lower)
- Clean up all those runs

WEEKLY MOST ACTIVE	RUNS
yannis-kolodziej	442
simon84	439

References

Jansson et al., 2017, Singing voice separation with deep U-Net convolutional networks.

Ju et al., 2016, Permutation Invariant Training of Deep Models for Speaker-Independent Multi-talker Speech Separation

Li et al., 2018, A Novel Bandit-Based Approach to Hyperparameter Optimization

C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, “A Short-Time Objective Intelligibility Measure for Time-Frequency Weighted Noisy Speech”, IEEE Int. Conf. Acoust., Speech, Signal Processing, Dallas, United States, pp. 4214-4217, 2010.

https://en.wikipedia.org/wiki/Perceptual_Evaluation_of_Speech_Quality