SPEECH SEPARATION IN THE WAVEFORM DOMAIN

Matej Hoffmann, Chuansheng Liu, Sturla Njarðarson, Dominik Stiller

Technical University of Denmark
Department of Applied Mathematics and Computer Science
2800 Kongens Lyngby, Denmark

ABSTRACT

Separating mixed signals into their respective sources is a highly underdetermined problem in machine learning. This includes the separation of music into stems and conversations into speaker utterances. For the application of hearing aids, access to individual speaker signals enables the reduction of background noise and focus on speakers. Here, we adapted the Demucs model, initially created for stereo music, for speech separation. We evaluate the performance of noisy mono two-speaker mixtures from the LibriMix dataset using quantitative metrics. We reach an SI-SDR of -1.56 dB on the validation set. The loss curves exhibit unexplainable behavior.

Index Terms— source separation, speech, deep learning

1. INTRODUCTION

Separating out a single dialogue in a room full of babble is a problem that was first described by Colin Cherry in 1953, and has since become famous as the *"Cocktail Party Problem"*. Cherry noted that the human auditory system was skilled at solving the problem, and posed the question of whether a machine with similar abilities could be created [1].

The creation of such a machine is still an open problem. A solution is of great interest in the development of hearing aids, as the hearing impaired have a hard time differentiating speech sources in noisy environments [2]. In these situations, access to individual speaker signals would allow the reduction of background noise, offering relief to the user of the hearing aid.

In recent years, deep learning models have been proposed as a solution to the source separation task. In this paper, we attempt to solve the elementary problem of two speakers talking at the same time in noise. We do this by adapting the Demucs deep neural network proposed by Défossez et al. in 2019 [3] and training it on utterances from the LibriMix dataset [4]. To the best of our knowledge, this has not been attempted before.

The objective is to be able to separate 2-speaker conversations in noise into individual speaker utterances. That is, we strive to implement a mapping

$$\mathcal{F}\{y_1 + y_2 + n\} = \begin{cases} \hat{y}_1 \\ \hat{y}_2 \end{cases}$$

where y_1 and y_2 are the ground truth speaker utterances, estimated by $\hat{y}_1$ and $\hat{y}_2$, and n is background noise. In this way, our model has two purposes: separating the superimposed speech signals and de-noising them.

2. METHODS

2.1. Dataset

From the LibriMix dataset, we use a subset called Libri2Mix, which is an automatic speech recognition corpus derived from audiobooks [4]. The dataset provides clean and mixed utterances, the latter being a mix of two speakers and noise. The dataset includes a total of 70 800 utterances which yield 292 hours of audio sampled at either 8 or 16 kHz.

We use a sampling frequency of 8 kHz and limit ourselves to 4-second utterances, removing any shorter utterances and truncating longer ones. After applying our criterion, we end up with ~64 000 mixed utterances with- superimposed noise, along with their corresponding ground truth separations. The test and validation split is listed in Table 1.

The background noise provided in the dataset is from various ambient noise recordings. This is done in an effort to imitate practical use cases.

Set	# Utterances	Hours	Split
Train	61619	68.5	96.7%
Val	2096	2.33	3.29%

Table 1: Training and validation dataset statistics. Examples from the LibriMix dataset were filtered for length > 4 s and sampled at 8 kHz.

2.2. Model

The model (Figure 1) has a U-net structure with an LSTM at the bottleneck. This structure extracts salient information by reducing the length (from 46 420 to 10 samples) and trading it for channels (from 1 to 2048). The signal is then reconstructed by the inverse operations. We implemented the model from scratch in PyTorch [1].

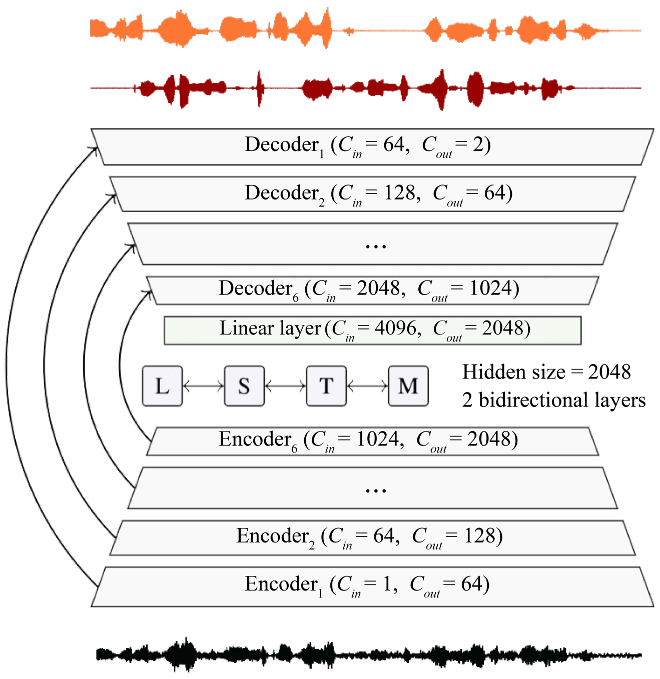
Decoder$_1$ (C_{in} = 64, C_{out} = 2)
Decoder$_2$ (C_{in} = 128, C_{out} = 64)
...
Decoder$_6$ (C_{in} = 2048, C_{out} = 1024)
Linear layer (C_{in} = 4096, C_{out} = 2048)
L
S
T
M
Hidden size = 2048
2 bidirectional layers
Encoder$_6$ (C_{in} = 1024, C_{out} = 2048)
...
Encoder$_2$ (C_{in} = 64, C_{out} = 128)
Encoder$_1$ (C_{in} = 1, C_{out} = 64)

Fig. 1: The architecture of our model: a U-net of six encoders and decoders. Figure adapted from [3].

The encoder consists of six blocks identical up to the number of channels (Figure 2, bottom). An encoder block consists of (1) a wide 1D convolution (size 8, stride 4, ReLU activation), which doubles the number of channels (only the first encoder steps from 1 to 64 channels), and (2) a 1D convolution (size 1, stride 1) with GLU activation. The second convolution doubles the number of channels but its GLU uses the second half of the channels as a mask for the first half so that the number of channels is not changed in this step. Thus, the GLU acts as attention-like masking on multi-scale representations of the input.

The LSTM has 2 bidirectional layers and 2048 hidden units per layer. The output of the LSTM is the concatenation of the forward and reverse the state for a total of 4096 features, which is reduced back to 2048 features by a linear layer.

The decoder, again, consists of six blocks identical up to the number of channels (Figure 2, top). A decoder block consists of (1) a 1D convolution (width 3, stride 1) with GLU masking, as before, and (2) a 1D transpose convolution (width 8, stride 4), which halves the number of channels, to successively construct the separated signal by trading channels for length. The width of the first convolution is called *context* because it allows the model to look to earlier and later samples to reconstruct the signal at a given point. The original authors left the context variable, but we did not find performance to change significantly when trying values between 3 and 9.

[1]https://github.com/DominikStiller/dtu-speechsep

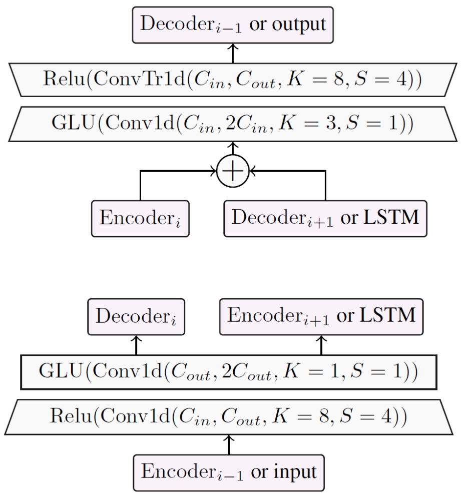
Decoder$_{i-1}$ or output
Relu(ConvTr1d($C_{in}, C_{out}, K = 8, S = 4$))
GLU(Conv1d($C_{in}, 2C_{in}, K = 3, S = 1$))
Encoder$_i$
Decoder$_{i+1}$ or LSTM
Decoder$_i$
Encoder$_{i+1}$ or LSTM
GLU(Conv1d($C_{out}, 2C_{out}, K = 1, S = 1$))
Relu(Conv1d($C_{in}, C_{out}, K = 8, S = 4$))
Encoder$_{i-1}$ or input

Fig. 2: The architecture of encoder (bottom) and decoder (top) blocks. Figure from [3].

After the final decoder layer, the reconstruction reaches the original length again but with two channels, one for each speaker.

The input needs to be padded to a valid length for the sequence of convolutions. While the input would only be 32 000 samples, it is padded with zeros to 46 420 samples, then trimmed to the original length after separation.

We experimented with dropout. In encoders and decoders, we added dropout before ReLUs and after GLUs (i.e. 2 dropout layers per block). At the bottleneck, there are dropout layers after the LSTM and the linear layer. This gives a total of 26 dropout layers.

2.3. Training and hyperparameters

We train for 360 epochs at a constant learning rate. This number of epochs worked well for music separation in the original paper [3]. We use an Adam optimizer with a learning rate of 1×10^{-4}. Learning rate schedules did not improve performance much, therefore we use a constant learning rate.

We use the sample-wise L1 loss between the separated signal and ground truth. L1 (mean absolute error) was found to be more robust than L2 (mean squared error) in the original Demucs model. While many generative audio models need to use phase-agnostic power spectrogram loss functions, the Demucs model can reconstruct the original phase through skip connections. For a longer discussion on losses, see [3].

Initially, we used a sinusoid dataset for rapid iteration. Each example is the sum of two sinusoids with random frequency, phase and amplitude. This facilitated validation of the the model code through overfitting. The faster training allowed us to find reasonable ranges for hyperparameters (e.g., context, weight decay, learning rate schedules) that allowed us to converge to optimal values on the LibriMix dataset. The final training took 25.4 h on 2 Tesla A100 PCIE 40 GB, of which the calculation of validation metrics is a significant part, however.

2.4. Evaluation metrics

We consider two metrics on our validation data to evaluate our results in addition to the L1 loss:

- *Scale-invariant source-to-distortion ratio* (SI-SDR): general measure of how good a source sounds, and
- *Speech intelligibility* (SI): measure of how well speech signals can be understood.

For speech intelligibility, we use the short-time objective intelligibility (STOI), introduced by Taal et al. in 2011 [5] . STOI is monotonically increasing with human speech intelligibility and takes values in $[0, 1]$, higher values being better.

The SI-SDR is a metric introduced by Jonathan et al. in 2018 and is a modified and more robust definition of SDR [6]. It is designed for the audio source separation task and used as an objective measure in the time domain to train deep learning models. The definition of SI-SDR is:

$$\mathrm{SI-SDR} := 10\log_{10}\left(\frac{\|e_{\text{target}}\|^2}{\|e_{\text{res}}\|^2}\right) \tag{1}$$

$$= 10\log_{10}\left(\frac{\left\|\frac{\hat{s}^T s}{\|s\|^2}s\right\|^2}{\left\|\frac{\hat{s}^T s}{\|s\|^2}s - \hat{s}\right\|^2}\right) \tag{2}$$

where s is the target source signal with allowed time-variant gain distortions, $\hat{s}$ denote an estimate of the target obtained by our model, e_{target} is obtained after scaling target signal s and e_{res} denotes component of residual which are considered as distortions. We strive for high SI-SDR values.

STOI and the SI-SDR are functions of clean and processed (degraded) signals, which in our case are the ground truth utterances and the utterances separated by our network. We note that these quantities are correlated since they are all affected by the SNR [7]. However - we believe that together, they provide a useful understanding of the performance of our system.

We furthermore employ human evaluations for the audio quality and speech intelligibility measures, along with a visual comparison between predictions and ground truths.

3. RESULTS

In this section, we compare experimental results on the Demucs with different batch sizes B and dropout probabilities p. The loss/metric curves during training are shown in Figures 4 to 7, the final loss/metric values are shown in Table 2.

Figure 4 shows how the two hyperparameters affect the L1 loss for both training and validation. It is worth noting that for the hyperparameters we tested, the loss does not change much beyond 200 epochs. Changing the batch size has little influence on the loss: the two respective curves (blue and orange) converge to approximately the same value, but the loss curve with 64 batches behaves more smoothly than the one with 8 batches. With regards to dropout probability, the loss curves diverge and adding dropout layers indeed reduces the validation loss. After 200 epochs, the training loss with 0.2 dropout probability (green) is more than three times higher than that without dropout, while the validation loss with dropout is 10% lower. Moreover, there are drastic changes in training and validation losses at around 162 epochs if the model has dropout layers. Unfortunately, we could not find a compelling reason for this behavior.

The loss curves in Figure 4 are rich in information. The training loss decreases steadily without dropout, but remains high if dropout layers are added. In general, a low training loss corresponds with a low validation SI-SDR, which is interesting but this may just be a spurious correlation. The validation loss seems correlated to the training loss: it decreases sharply at first, but once the training loss decreases (i.e. the model learns), the validation loss increases, then plateaus at a high level.

The curves of metrics have a similar trend. From Figures 5 and 6, we notice that the validation SI-SDR and STOI curves have similar shapes. However, the SI-SDR for batch size 8 converges to -6.84 and slightly improves to -5.14 after increasing the batch size to 64, while the STOI still converges to about 0.49.

Figures 6 and 7 show that both validation SI-SDR and STOI benefit from using dropout. Compared to curves without dropout, after 200 epochs, SI-SDR and STOI with 0.2 dropout probability are 3.58 and 0.155 higher respectively. Likewise, metric curves incur unknown sudden jumps at the same position as loss curves. All three validation SI-SDR curves in Figures 5 to 7 behave similarly with corresponding training losses. It is strange because they should have opposite trends in theory.

In general, a smaller batch size leads to more stochastic behaviors of loss curves but does not change the final result significantly. After introducing the dropout, training loss increases as expected, while validation performance gets big improvements.

Param.		Train.	Val.		
B	p	L1	L1	SI-SDR	STOI
8	0	5.44	18.1	-6.84	0.489
64	0	4.94	18.2	-5.14	0.491
64	0.2	16.6	16.4	-1.56	0.616

Table 2: Effect of different batch sizes B and dropout probabilities p on the training and validation set performance, averaged over the last ten epochs. L1 loss is given in $\times 10^{-3}$.

4. DISCUSSION

Overall, we had some success with our objective of separating two speech signals. The performance is best when both speakers have similar volume levels. Our model showed skill on the training data but did not generalize well, which manifested in confusion between speakers in validation predictions (Figure 3). We could not resolve the confusion with wider contexts and more LSTM layers, which should have given the model more sequential information to work with. Regularization with dropout, weight decay and smaller batch sizes did not improve validation performance. Resampling and normalization helped the original authors with music separation but deteriorated performance for speech.

In the future, we would like to explore more approaches to improve the generalization and also try to separate three or more speaker mixtures. Trying to train the model with different or extended dataset could also improve the results.

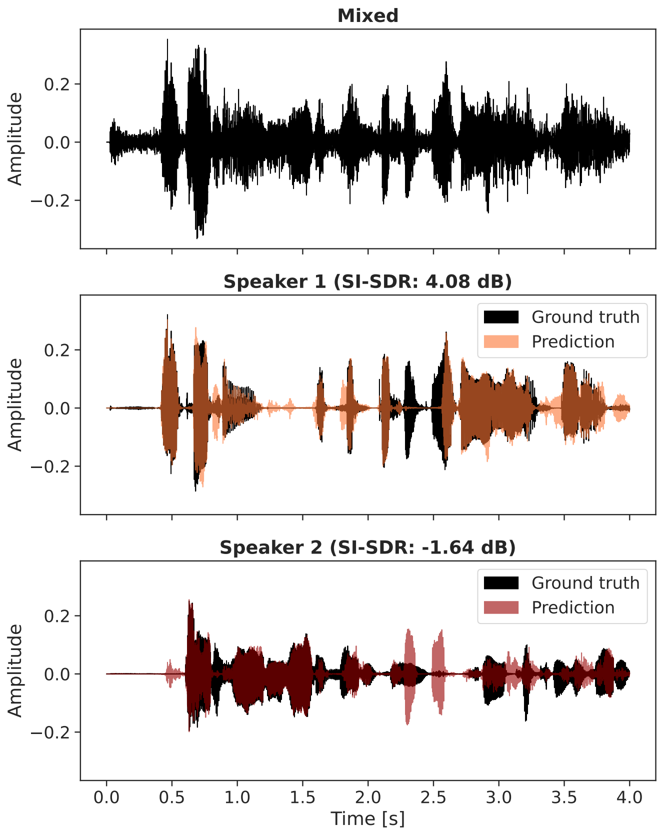

Fig. 3: Prediction of a validation example. The top shows the input, followed by a comparison between ground truth and predicted signals. Confusion is present around 2.5 s.

5. REFERENCES

[1] E. Colin Cherry, "Some Experiments on the Recognition of Speech, with One and with Two Ears," *The Journal of the Acoustical Society of America*, vol. 25, no. 5, pp. 975–979, Sept. 1953.

[2] Simon Haykin and Zhe Chen, "The Cocktail Party Problem," *Neural Computation*, vol. 17, no. 9, pp. 1875–1902, Sept. 2005.

[3] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach, "Music source separation in the waveform domain," 2019.

[4] Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent, "Librimix: An open-source dataset for generalizable speech separation," 2020.

[5] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, sep 2011.

[6] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R. Hershey, "Sdr - half-baked or well done?," 2018.

[7] Xiaodong Xu, Ronan Flynn, and Michael Russell, "Speech intelligibility and quality: A comparative study of speech enhancement algorithms," in *2017 28th Irish Signals and Systems Conference (ISSC)*, Killarney, Co Kerry, Ireland, June 2017, pp. 1–6, IEEE.

A. TRAINING CURVES

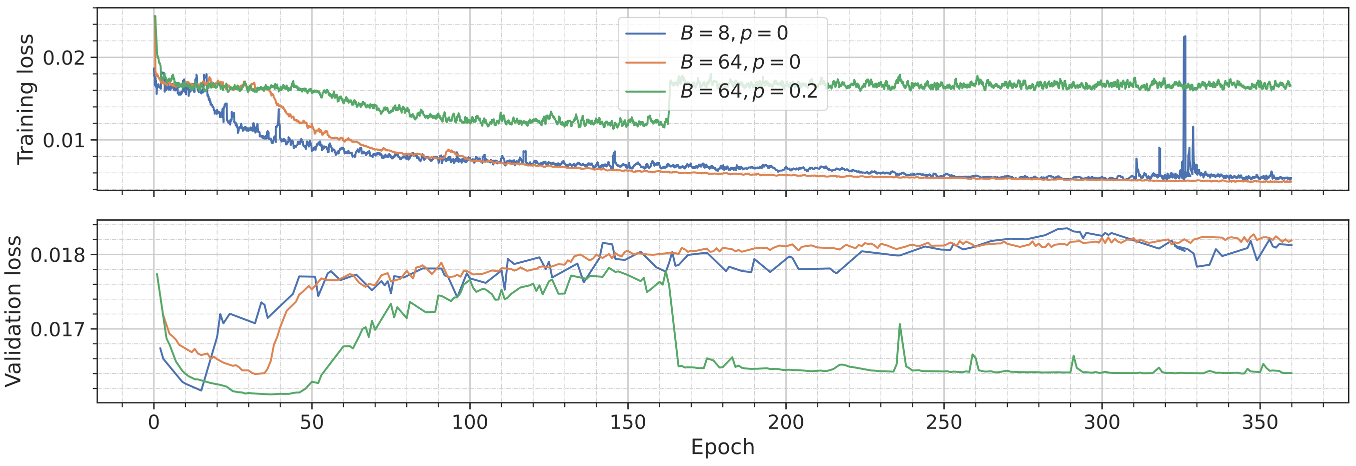

Fig. 4: Training and validation L1 loss of the three experiments. Curves have been smoothed.

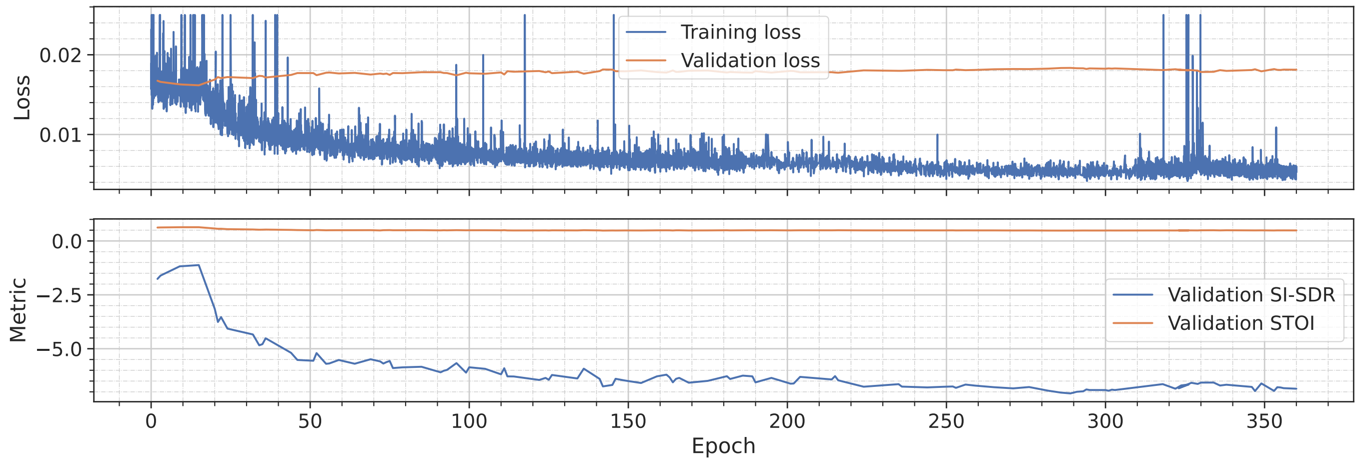

Fig. 5: Losses (lower is better) and metrics (higher is better) for experiment with $B = 8$ and $p = 0$.

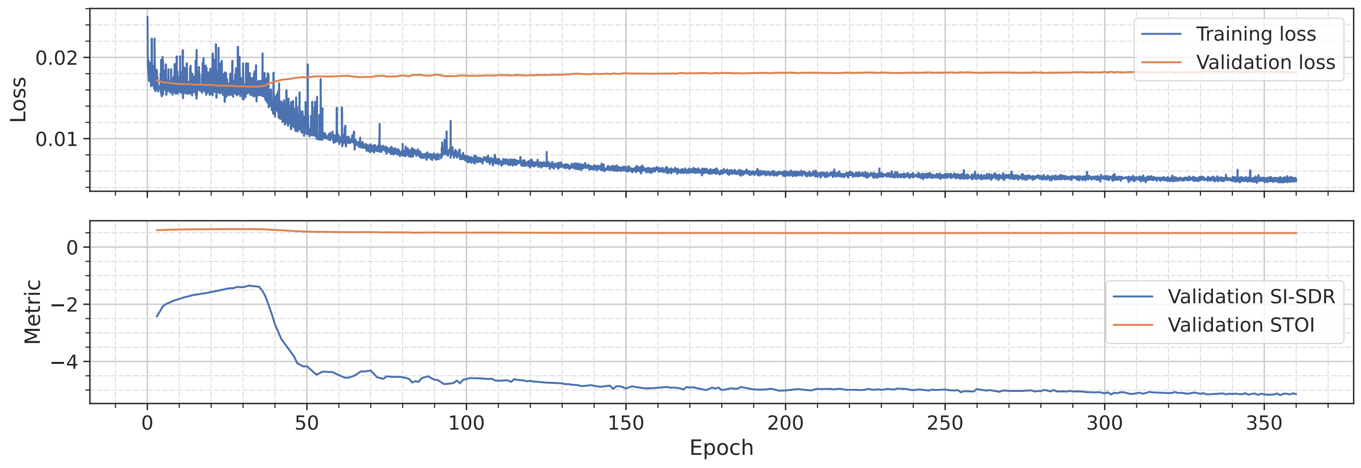

Fig. 6: Losses (lower is better) and metrics (higher is better) for experiment with $B = 64$ and $p = 0$.

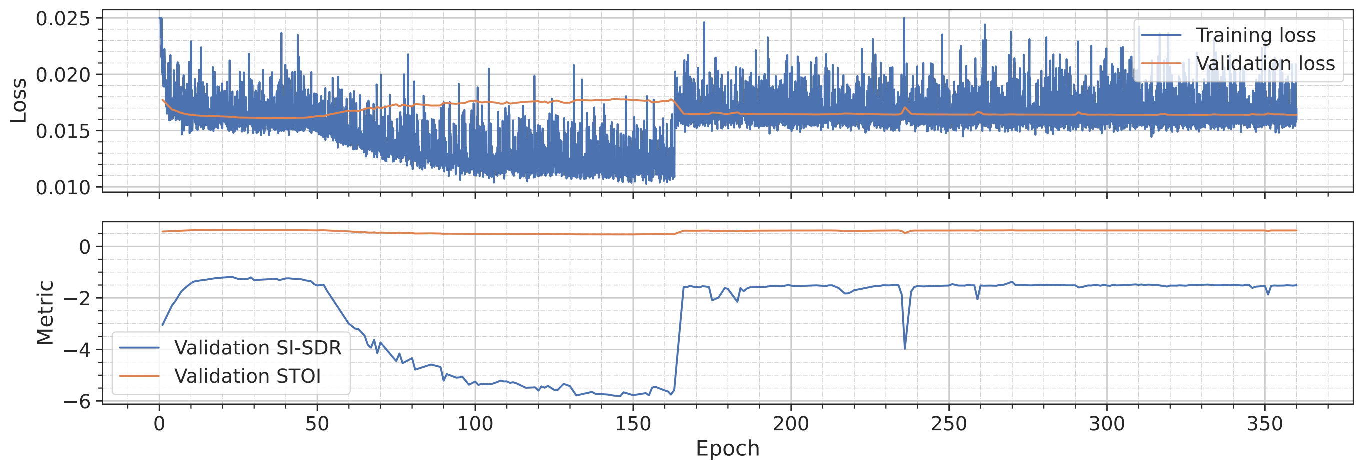

Fig. 7: Losses (lower is better) and metrics (higher is better) for experiment with $B = 64$ and $p = 0.2$.