Wave-U-Net

https://github.com/f90/Wave-U-Net-Pytorch

http://ismir2018.ircam.fr/doc/pdfs/205_Paper.pdf

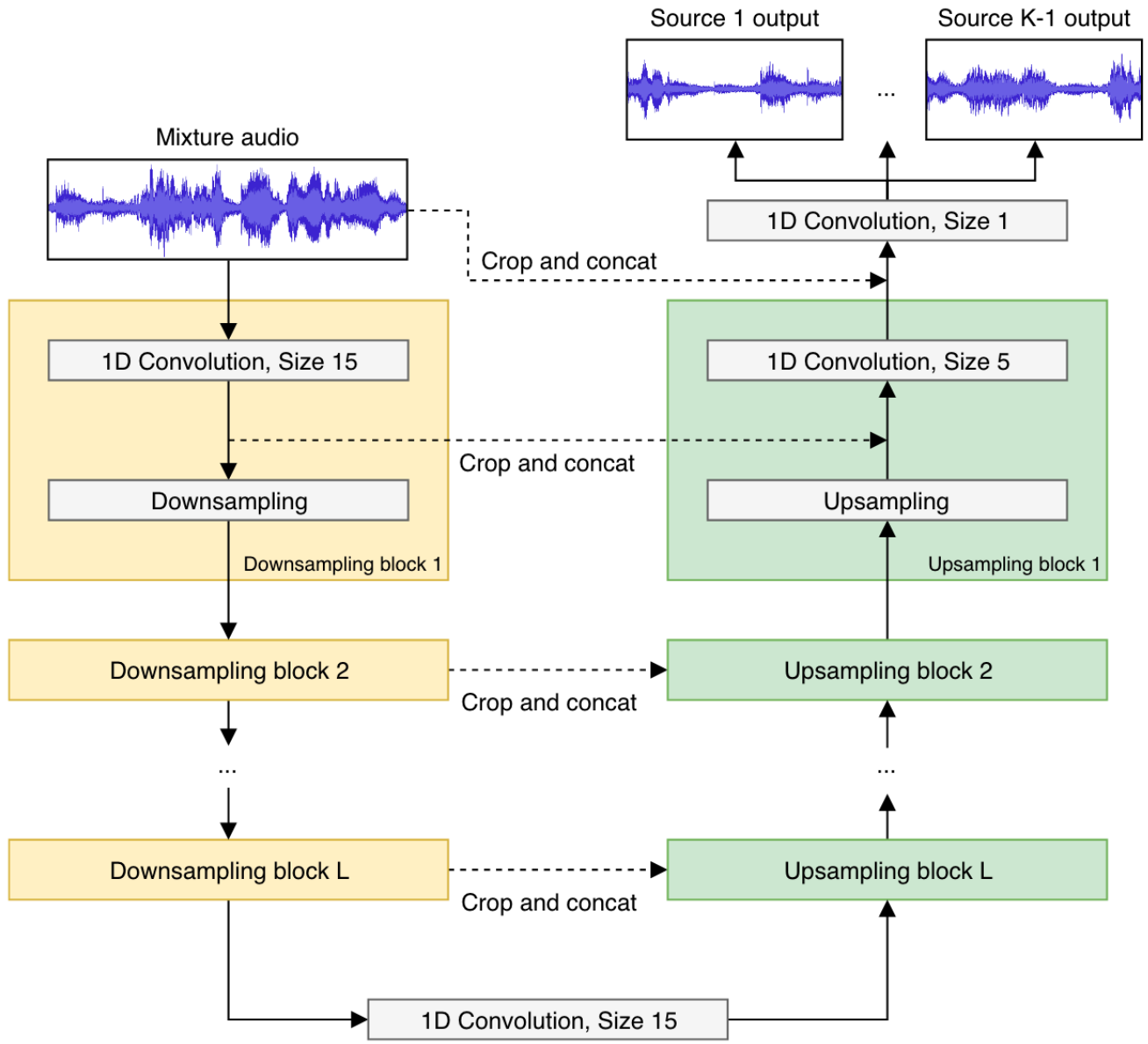

Figure 1. Our proposed Wave-U-Net with K sources and L layers. With our difference output layer, the K-th source prediction is the difference between the mixture and the sum of the other sources.

- The network has L levels in total, with each successive level operating at half the time resolution as the previous one.

- For K sources to be estimated, the model returns predictions in the interval (−1, 1), one for each source audio sample.

Downsample (DS), Upsample(US)

Block	Operation	Shape
	Input	$(16384, 1)$
DS, repeated for $i = 1, \ldots, L$	`Conv1D`$(F_c \cdot i, f_d)$	
	`Decimate`	$(4, 288)$
	`Conv1D`$(F_c \cdot (L+1), f_d)$	$(4, 312)$
US, repeated for $i = L, \ldots, 1$	`Upsample`	
	`Concat`(DS block i)	
	`Conv1D`$(F_c \cdot i, f_u)$	$(16834, 24)$
	`Concat`(Input)	$(16834, 25)$
	`Conv1D`$(K, 1)$	$(16834, 2)$

Table 1. Block diagram of the base architecture. Shapes describe the final output after potential repeated application of blocks, for the example of model M1, and denote the number of time steps and feature channels, in that order. DS block i refers to the output before decimation. Note that the US blocks are applied in reverse order, from level L to 1.

Downsample (DS)

- Conv1D(x,y) denotes a 1D convolution with x filters of size y. It includes zero-padding for the base architecture, and is followed by a LeakyReLU activation (except for the final one, which uses tanh).

- Decimate discards features for every other time step to halve the time resolution.

Upsample(US)

- Upsample performs upsampling in the time direction by a factor of two, for which we use linear interpolation.

- ??? Concat(x) concatenates the current, high-level features with more local features x.

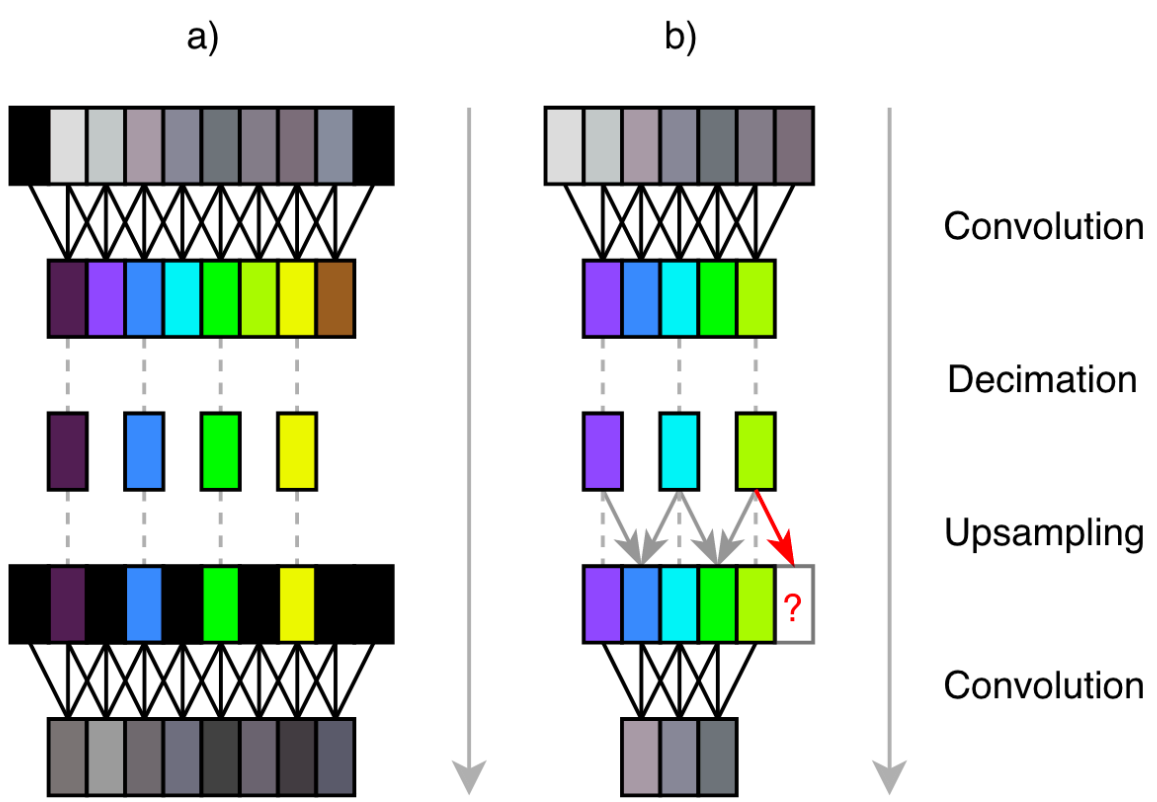

Figure 2. a) Common model (e.g. [7]) with an even number of inputs (grey) which are zero-padded (black) before convolving, creating artifacts at the borders (dark colours). After decimation, a transposed convolution with stride 2 is shown here as upsampling by zero-padding intermediate and border values followed by normal convolution, which likely creates high-frequency artifacts in the output. b) Our model with proper input context and linear interpolation for upsampling from Section 3.2.2 does not use zero-padding. The number of features is kept uneven, so that upsampling does not require extrapolating values (red arrow). Although the output is smaller, artifacts are avoided.

- In extensions of the base architecture, where Conv1D does not involve zero-padding, x is centre-cropped first so it has the same number of time steps as the current layer.

- We employ convolutions without implicit padding and instead provide a mixture input larger than the size of the output prediction, so that the convolutions are computed on the correct audio context (see Figure 2b). Since this reduces the feature map sizes, we constrain the possible output sizes of the network so that feature maps are always large enough for the following convolution.

Learned upsampling

$$f_{t+0.5} = \sigma(w) \odot f_t + (1 - \sigma(w)) \odot f_{t+1} \qquad (1)$$

Training

- 75 tracks from the training partition of the MUSDB [17] multi-track database are randomly assigned to our training set, and the remaining 25 tracks form the validation set, which is used for early stopping. Final performance is evaluated on the MUSDB test partition comprised of 50 songs. For singing voice separation, we also add the whole CCMixter database [9] to the training set.
- As data augmentation for both tasks, we multiply source signals with a factor chosen uniformly from the interval [0.7, 1.0] and set the input mixture as the sum of source signals. No further data preprocessing is performed, only a conversion to mono (except for stereo models) and downsampling to 22050 Hz.

Training

- During training, audio excerpts are sampled randomly and inputs padded accordingly for models with input context. As loss, we use the mean squared error (MSE) over all source output samples in a batch. We use the ADAM optimizer with learning rate 0.0001, decay rates $\beta 1 = 0.9$ and $\beta 2 = 0.999$ and a batch size of 16. We define 2000 iterations as one epoch, and perform early stopping after 20 epochs of no improvement on the validation set, measured by the MSE loss. Afterwards, the last model is fine-tuned further, with the batch size doubled and the learning rate lowered to 0.00001, again until 20 epochs without improvement in validation loss. Finally, the model with the best validation loss is selected.

Training

- 7 models with variation are trained. Details in the paper. Unorganized.

Metric

Source-to-Distortion Ratio (SDR)

$$\mathrm{SDR} := 10 \log_{10} \left(\frac{\| s_{\text{target}} \|^2}{\| e_{\text{interf}} + e_{\text{noise}} + e_{\text{artif}} \|^2} \right)$$

SDR is usually considered to be an overall measure of how good a source sounds. If a paper only reports one number for estimated quality, it is usually SDR.

Results

		M1	M2	M3	M4	M5	M7	U7	U7a
Voc.	Med.	3.90	3.92	3.96	4.46	**4.58**	**3.49**	2.76	2.74
	MAD	3.04	3.01	3.00	3.21	3.28	2.71	2.46	2.54
	Mean	-0.12	0.05	0.31	**0.65**	0.55	-0.23	-0.66	**0.51**
	SD	14.00	13.63	13.25	13.67	13.84	13.00	12.38	10.82
Acc.	Med.	7.45	7.46	7.53	**10.69**	10.66	**7.12**	6.76	6.68
	MAD	2.08	2.10	2.11	3.15	3.10	2.04	2.00	2.04
	Mean	7.62	7.68	7.66	**11.85**	11.74	**7.15**	6.90	6.85
	SD	3.93	3.84	3.90	7.03	7.05	4.10	3.67	3.60

Table 2. Test set performance metrics (SDR statistics, in dB) for each singing voice separation model. Best performances overall and among comparison models are shown in bold.

	Vocals				Other			
	Med.	MAD	Mean	SD	Med.	MAD	Mean	SD
M6	3.0	2.76	-2.10	15.41	2.03	1.64	1.68	6.14
	Bass				**Drums**			
	Med.	MAD	Mean	SD	Med.	MAD	Mean	SD
M6	2.91	2.47	-0.30	13.50	4.15	1.99	2.88	7.68

Table 3. Test performance metrics (SDR statistics, in dB) for our multi-instrument model

Train

- python3.6 train.py --dataset_dir /PATH/TO/MUSDB18HQ