Update Thesis 11-04-2022

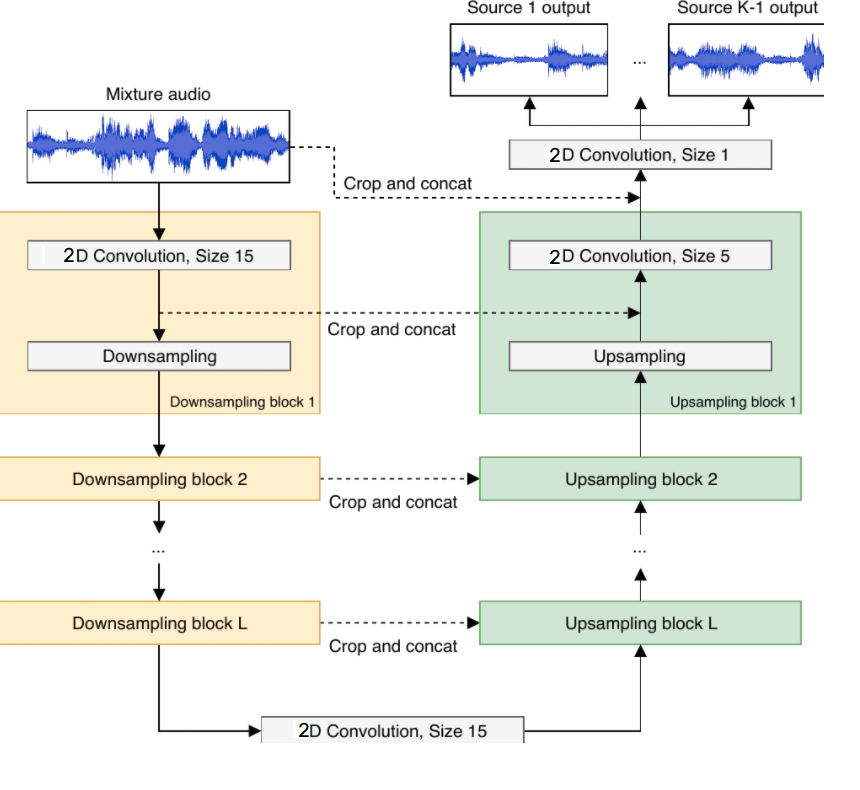
Zie code**: https://github.com/DiegoLigtenberg/Workspace-MasterThesis-MSS**

tijdje geleden:

* 4 april jaar geleden dat mijn broertje overleden was; voorbereiding / mind was even niet bij scriptie
* 2 dagen erna ziek geworden (griep), t/m gisteren dagen in bed gelegen
* Goed om weer verder te gaan!

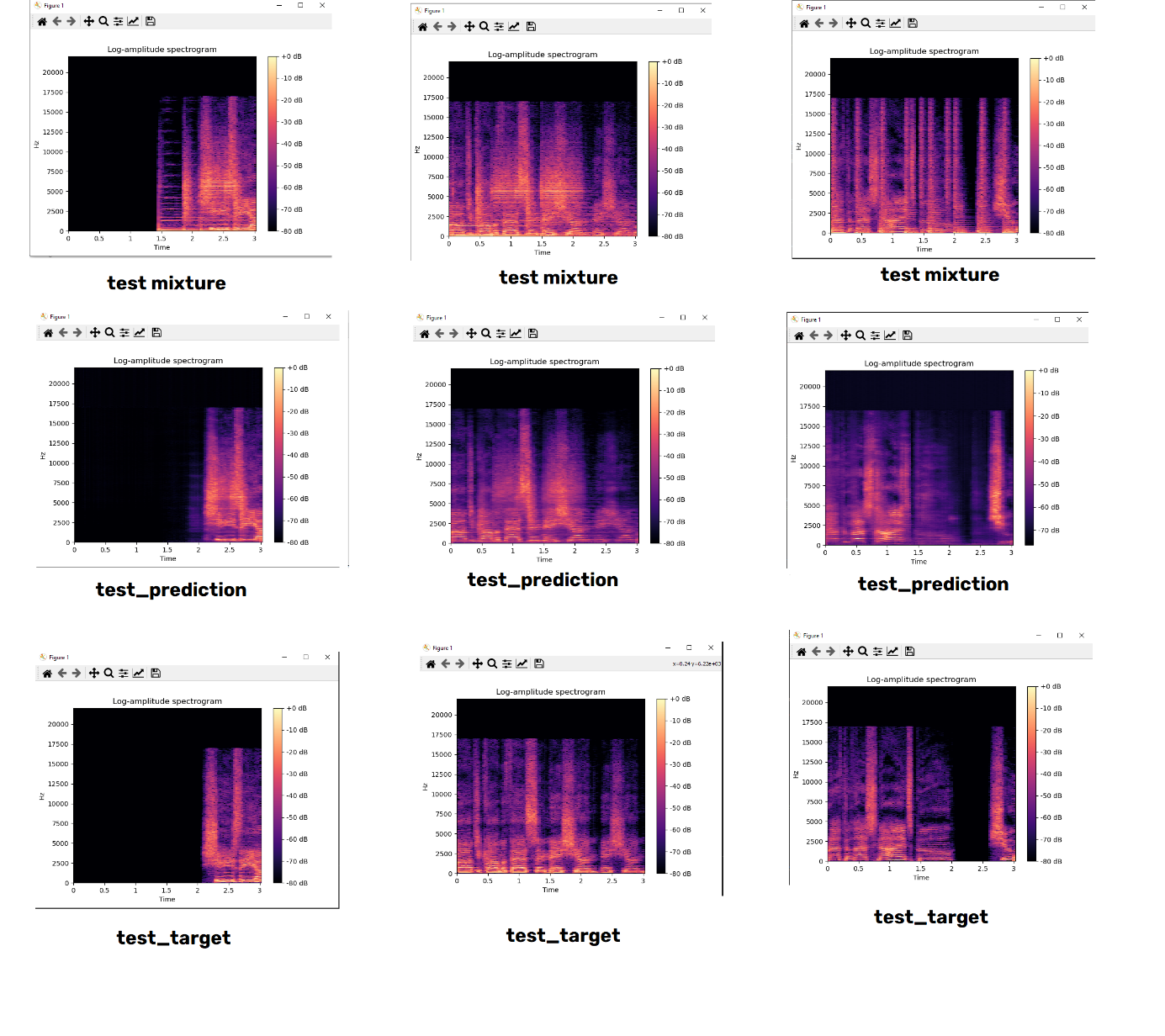
**PREPROCESSING**

* Preprocessing pipeline gemaakt voor Musdb dataset: 150 songs met gevarieerde lengte
* De 150 songs zijn verdeeld in 85 train, 15 validation, 50 test.
* De Preprocessing pipeline ziet er als volgt uit:
  1. Laad de volledige song
  2. Extract 3 second segments
  3. Augment de segments (zit in preprocessing pipeline, maar nog niet opgeslagen)
  4. Pad het signaal (laatste stuk van een song soms 2.1213 seconds -> pad naar 3 seconds door 0 toe te voegen
  5. Extract log spectrogram using stft
  6. Normalize spectrogram based on global statistics (/160)
  7. Save normalized spectrogram as .npy



**AUTO ENCODER MODEL**

* Auto encoder gemaakt met skip connections.
* Train dataset bestaat uit 6403 x 2048 x 128 npy files -> te groot om in ram te passen, dus **train on batch** implementation.
* Eerst getrained op 1 sample **mixture -> vocal**, om te kijken of het overfit met close to 0 loss.
  + Dit werkte! mse loss van 2e-5 geeft ‘perfect’ results -> architecture kan werken
* Daarna getest op grotere dataset (veel bugfixes)
  + Niet shuffelen
  + Shuffelen van X en Y waardoor labels mismatchte
  + Geen validation set
  + Verschillende learn rates getest
  + Skip connections werkte niet
* Uiteindelijk stabalized en 1 setting aan main parameters opgeslagen en mee gaan trainen
  + Results:
    - Werkt, maar overfit (voor grootste gedeelte had ik nog geen validation stats om te checken voor overfit) -> pas 2 dagen geleden implemented



* **GOALS VOOR MIXTURE -> “OTHER” (alleen instruments) om beter te maken (tot 30 april)**
  + **Add regularisation (l2 regularisation + dropout)**
  + **Loss curve voor train/test goed monitoren**
  + **Evt andere visualisations?**

Loss curves

* Validation loss decreased minder snel
* Regularisation techniques implemented -> niet super veel success

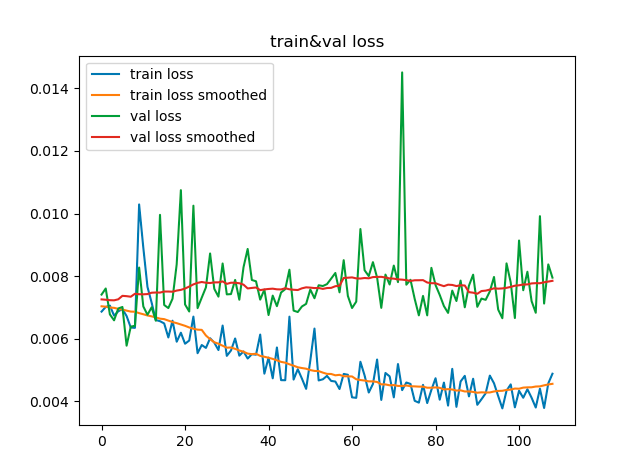
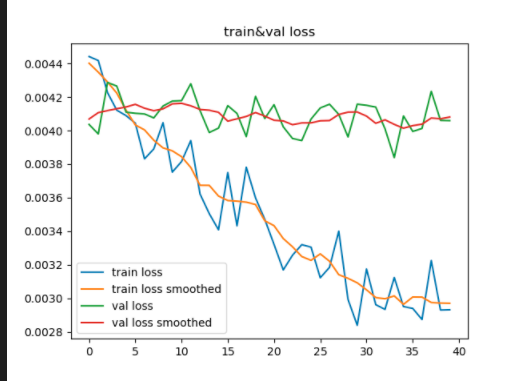
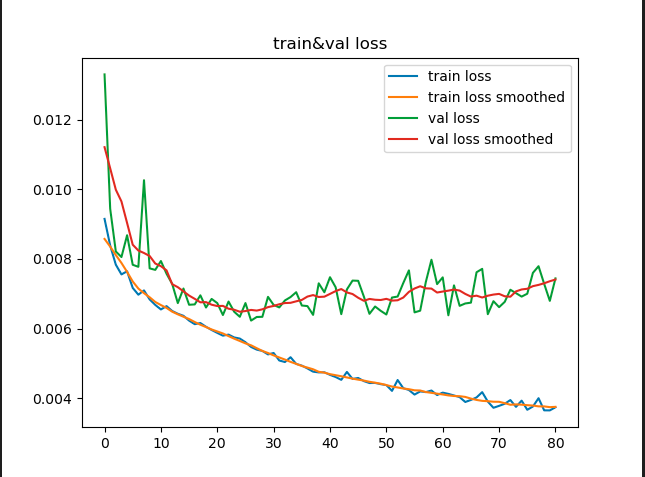


Figure 1 No Batch Normalisation

Figure 2: Plain model

Figure 3 Dropout + regularisation

Batch normalisation:

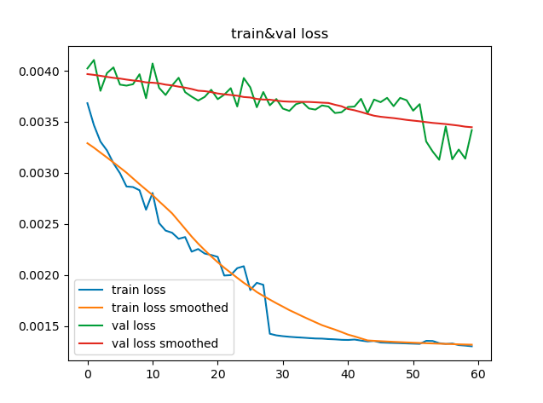
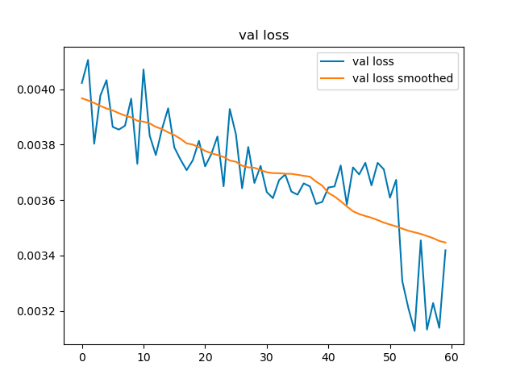
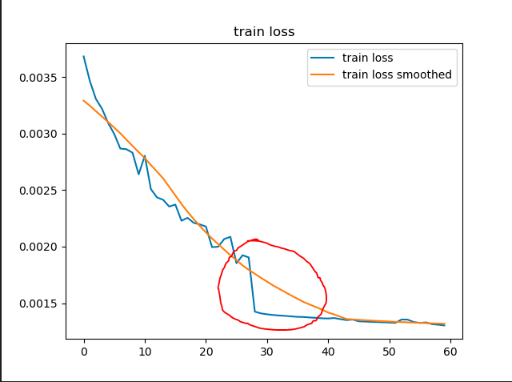
* Gekeken naar weights -> erg groot (hoort niet)

Data augmentation

* Polarity inversion
* Reverse signal
* Time stretching
* Pitch shifting

**April 26th**

**Results beginning week:**

* Trained Mixture - > Other (instrumentation without vocal) model
* Experimented with loss functions (but during training)
* Training was mostly done with grid search in mind
  + Trying different loss functions
  + Different learning rates
  + Augmented vs non augmented dataset
  + Model architecture
* Combination of everything gave final model. works ‘okay’ on test set, but does not generalize to other instrumentation. Especially saxphone + vocalinstrumentation got completely blurred.

Train/val train validation

**Mid week:**

* Started looking more into IRMAS dataset
* Cello, clarinet, flute, guitar (elec and acoustic), organ, piano, sax, trumpet, violin, **voice**
* **Voice** is also a recognition instrument -> thus creating any artefacts by removing voice does not make sense!
* **Trained new model that has other + vocal** -> thus removing the drums + bass
* **Because major change** -> looked at proposal + other paper’s doing the same thing and found 3 papers supporting my hypothesis

**3 papers supporting my hypothesis:**

**Singing Voice Separation**

<https://www.researchgate.net/profile/Bidisha-Sharma/publication/335829325_On_the_Importance_of_Audio-Source_Separation_for_Singer_Identification_in_Polyphonic_Music/links/60dd762ca6fdccb745f90fc2/On-the-Importance-of-Audio-Source-Separation-for-Singer-Identification-in-Polyphonic-Music.pdf>

* When converting songs (train + test) to only vocals using WaveUNET -> Singer classification was way easier

**Instrument Classification OLD FASST**

<https://ismir2012.ismir.net/event/papers/559_ISMIR_2012.pdf>

* When using **separated sources on both train + test ->** Detecting instruments is better

They used separate classification algorithm for each separated source -> combined the labels

* Already found slight increase when only test data was source separated with normal train data

**Instrument Classification New** (STILL FASST AND HARMONIC/Percusive from 2014 very old)

https://archives.ismir.net/ismir2018/paper/000145.pdf

* They tested whether using the separation algorithms can be used to increase performance of instrument recognition
* when the **training dataset + test dataset BOTH got separated**, they found best performance

But they used transfer learning -> because source separation is so bad that they had to use weights from previously trained model

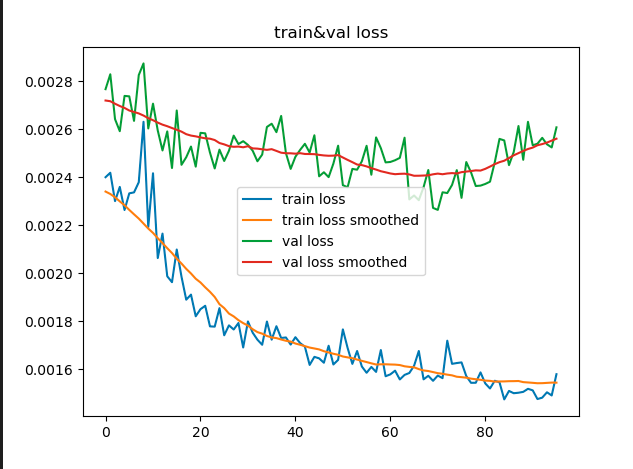
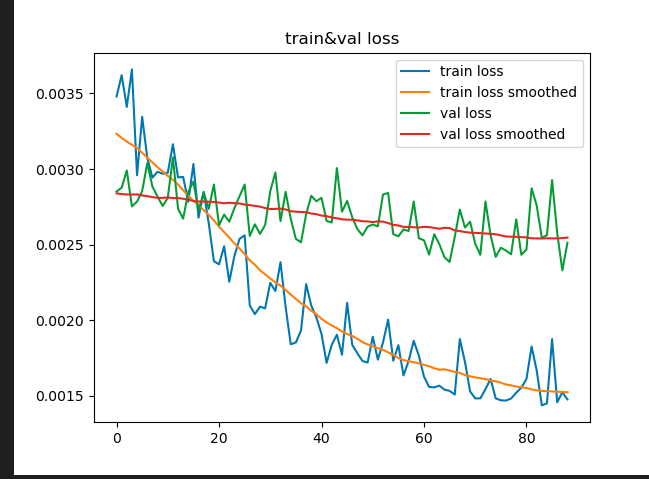
Afbeelding met tekst, hond, zoogdier, wolf

Automatisch gegenereerde beschrijvingDetection of singing foice, or instrument better if you isolate the main source -> in image you would preprocess and potentially remove the background to make network focus on the source of interest!

**End week**

**Created new Dataset -> Vocal + Other (removed drums + bass)**

* tried different architectures and attempt to systematically evaluate different hyperparameters
* Loss function mse 3e-4
* Batch normalisation -> Relu -> (Concatenate) (best order)



Other + Vocal ReLU -> BN Other + vocal BN -> ReLU

Shortly tested on IRMAS dataset -> at 50 epoch

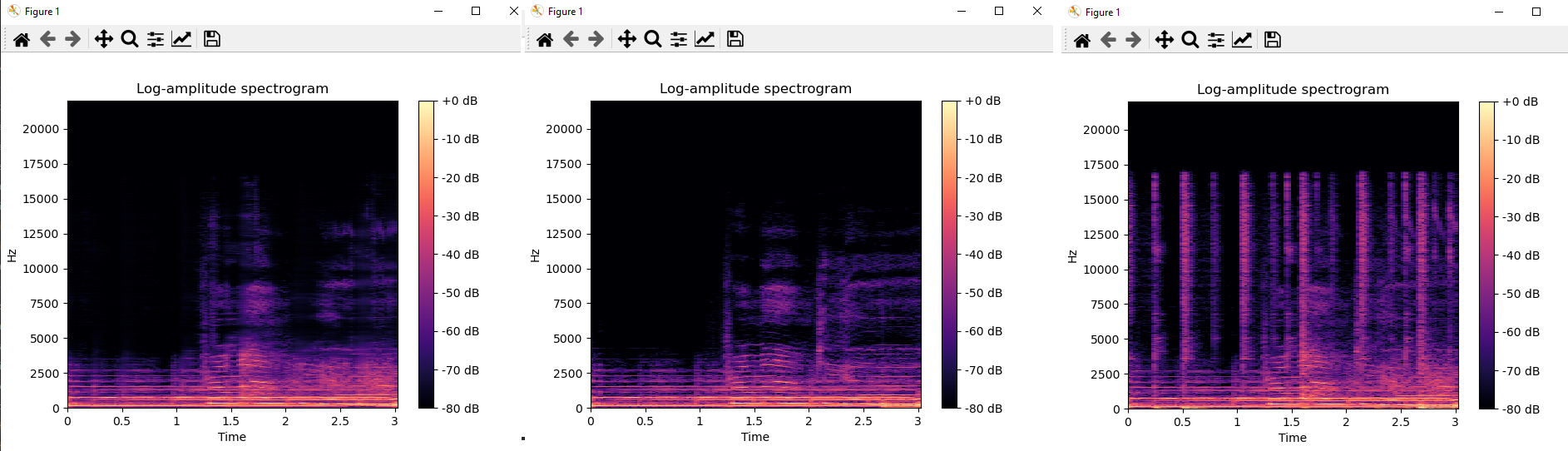
* Model is less aggressive because only removing bass and drums
* Less artefacts but also sounds bit ‘dof’
* This source separation at least feels useable! Still not perfect, but it does what it has to do
* Compared to previous papers using harmonic/percussion separation -> sounds good!

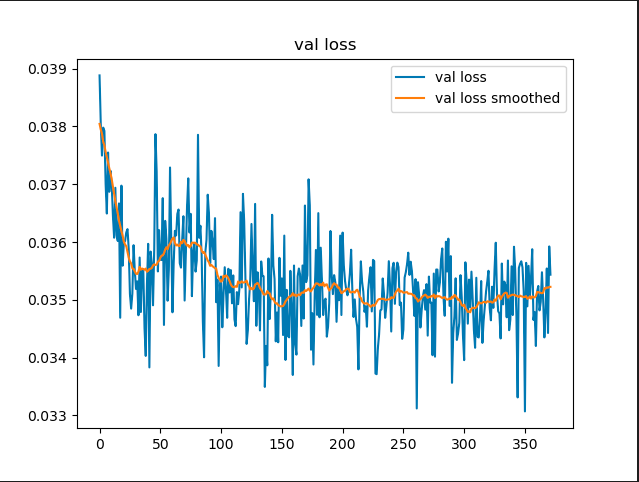
**May 3th**

Final model trained

* Model able to extract “other + vocals” from input mixture (aka removing drum + bass)

Result spectrogram



Prediction target mixture

**May 10th**

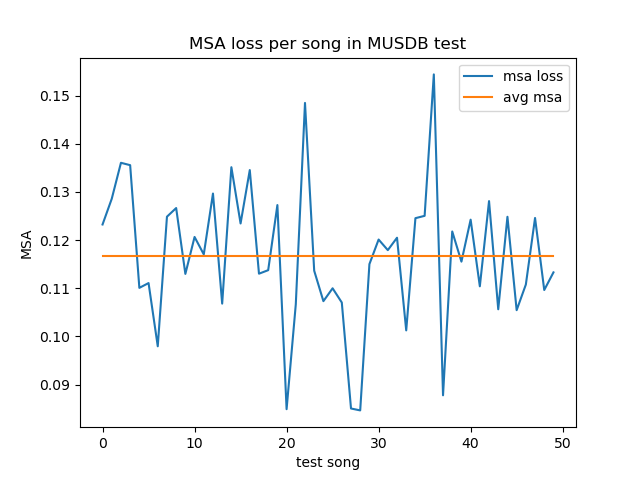
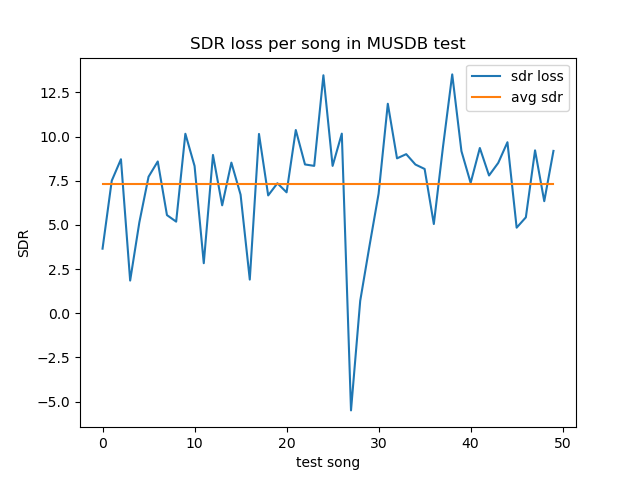
Generalized code + Evaluation MusDB + Post Processing

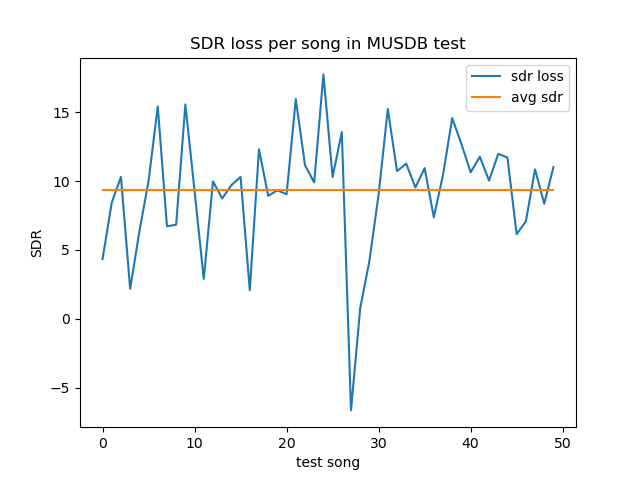
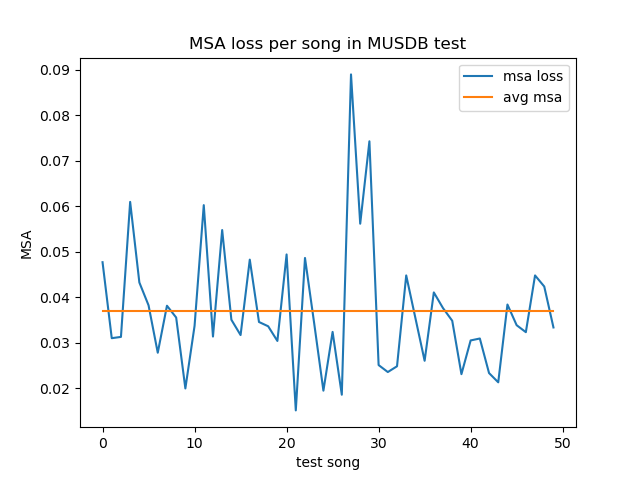
Generalized code

* 1 function allows us to put music files in ‘track input’
  + Separator.input\_to\_waveform -> converts all waveforms to prediction of model with or without post processing -> very convenient!
  + Also Separator.evaluate\_MUSDB evaluates whole musdb test set!

Evaluation MUSDB TEST DATA

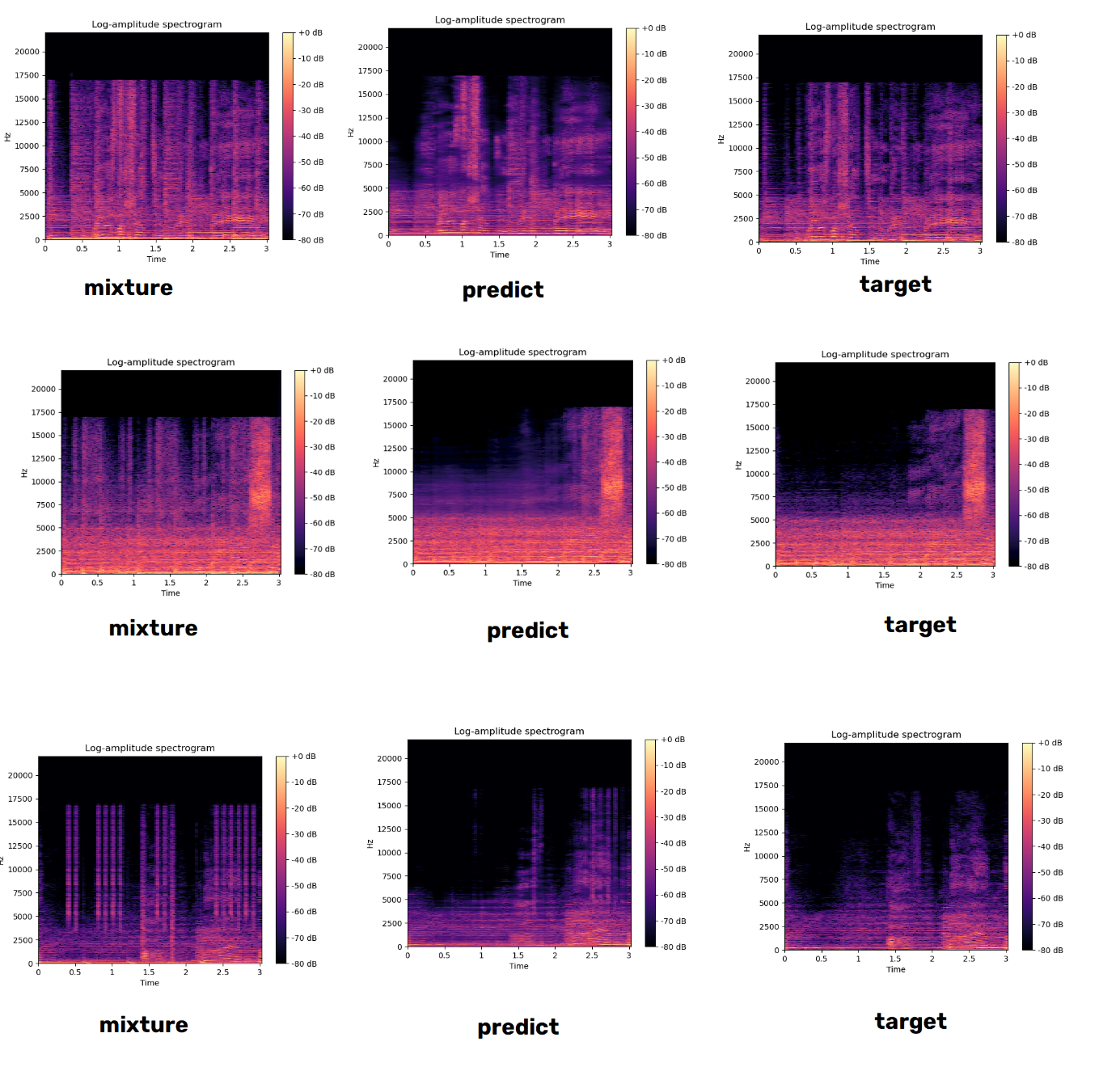
* Computed both SDR and Mean Absolute error on whole MUSDB test set (50 songs)
* generally good except for 1 outlier song (27)
  + this song is very very sparse in leftover sound
  + lots of what is in target song is delay of the main leftover sound (difficult), model goes silent



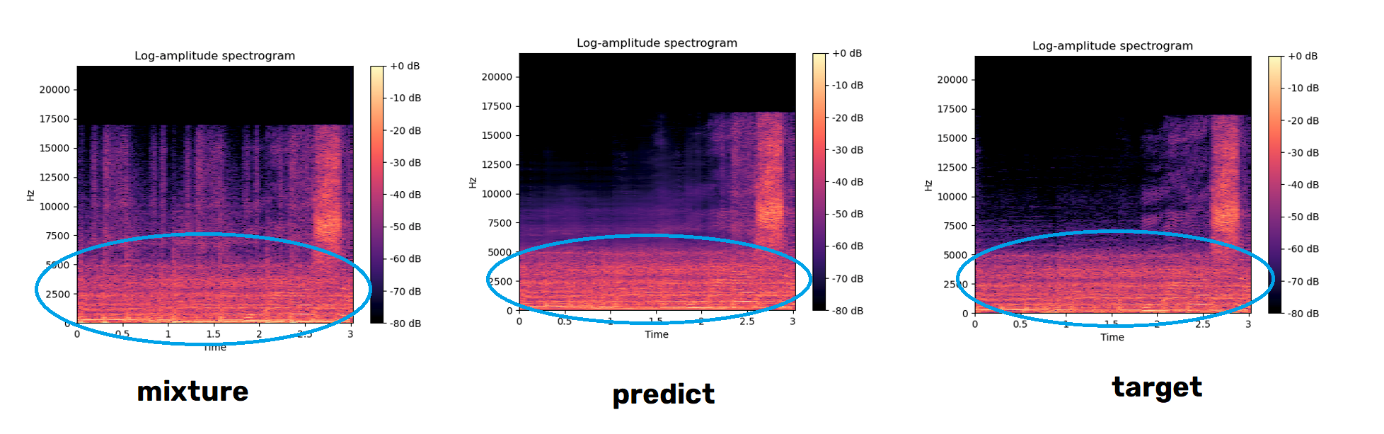


Example mixture predict target in MUSDB

* often does it job, but NOT without ERROR!



In dataset it was quite consistently occurring that 0-5500 frequency range had biggest amplitudes (-40DB to 0 DB) and everything above was around -60DB To -80DB.



Post Processing

After careful listening to multiple songs from MUSDB + Other inference songs:

* Frequencies > 5500 were often times interpreted as just **noise** of high percussion/**overtones**

Therefore implemented post processing that only focussed on these frequencies < 5500

* Overall SDR decreased (for some songs it even increased due to almost unhearable artefacts in high frequency range BUT human judgement was **WAY** better!
* In target these high frequencies >5500 mostly contain
  + breathing from voices, or highest frequencies (**overtones**) of certain piano/guitar
  + I have an example of piano -> **that’s not what makes a piano a piano**

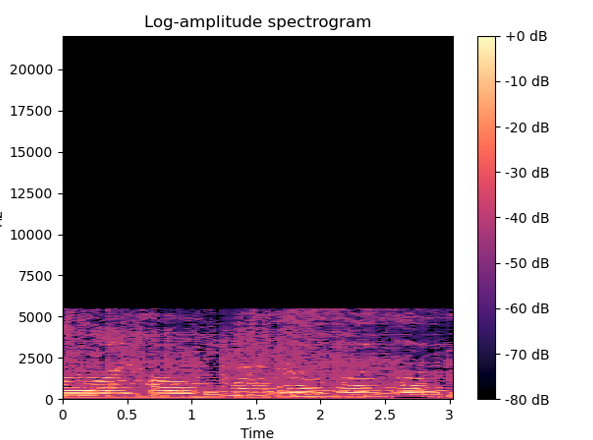
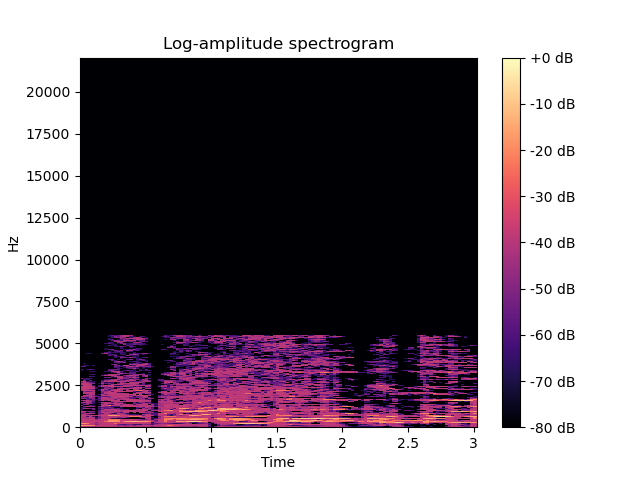
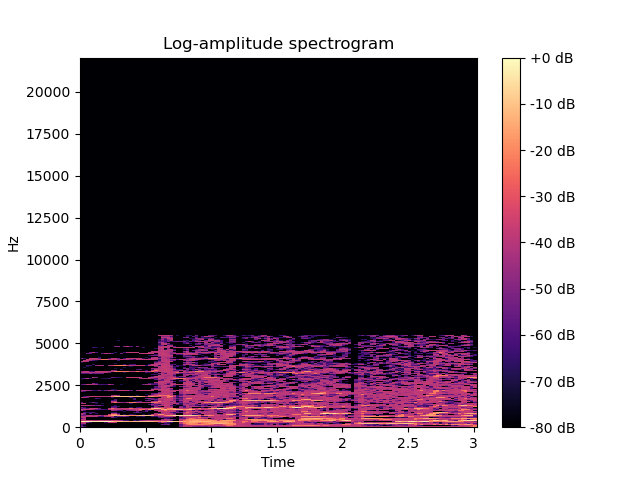
The algorithm

* Break wall limiter on all frequencies > 5500
* All frequencies (also below 5500) softer than -40DB are removed in 0-5500 frequency range
  + This removed most leftover artefacts from very soft drums!
* Also experimented with converting to mp3 -> this sort of worked but still noise -> this was better because neural network actually predicted the drums/bass as **low(er)** **amplitudes** but simple not -80DB -> this completely removes all noise!



Goal is to use at least 2, but probably a total of 3 models for instrument classification testing

1. Base model
2. Raw Neural network output model
3. Post processed network output model (based on what humans find nice (only ~200-5500 hz) rest too many artefacts (best human judge score) (see image for sdr score)

Post Processing on certain tracks (otherwise drum artefact frequencies are removed!)

**Goal for next time is:**

* Download whole IRMAS dataset (train + test) -> only got parts right now
* Generate 2 extra versions of IRMAS dataset (prediction and prediction + post processing)

So that we have BASE dataset and compare it with two generated datasets

* Figure out how to do Multiclass classification
* F1 Accuracy Precision Recall will be metrics -> how does it work with more than 1 predictions
* Train set is 3 seconds, test set is 5-20 sec -> how to make logic for what predictions are
* Highly likely that I will train new model
* If Leftover time -> Attempt Transfer learning (freezing autoencoder) -> **probably will not work due to skip connections**, model does not have to learn much for pieces it can just copy, only learned how to adjust bass / drums -> not necessarily instruments

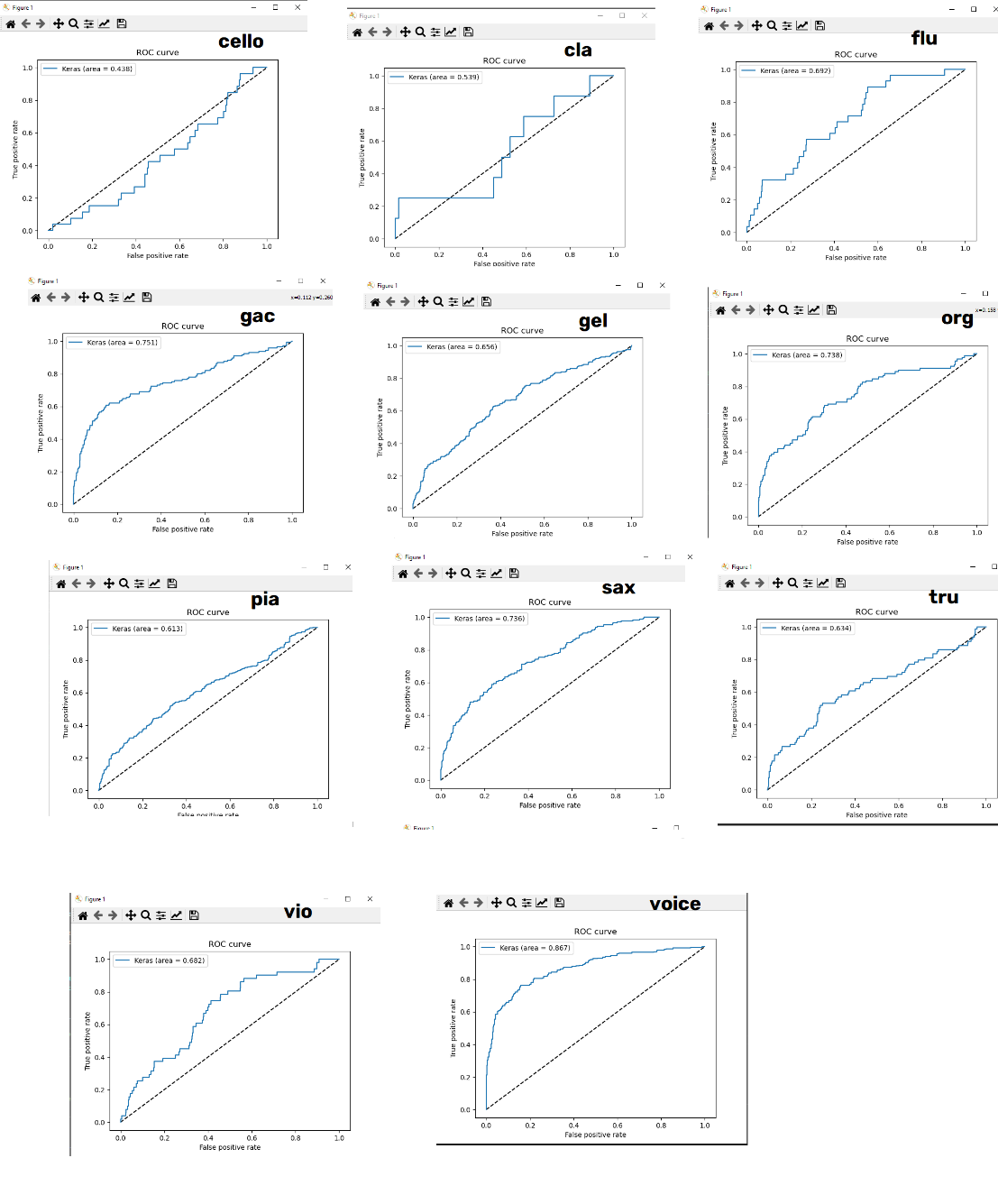
**June 8th**

* Trained model on train data (only one hot encoded)
  + 6705 train samples ( 3 seconds) single-label
  + 805 test samples ( 3 seconds) multi-label
  + It actually does do correct inference on test data!
* Trained without dropout, and without proper metrics (only bce loss, and binary accuracy)
* Model grossly overfits on train data! Why is this happening?
  + No dropout (regularisation)

Afbeelding met tafel

Automatisch gegenereerde beschrijving

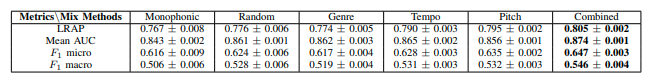
Results



Base No Post Processing With Post Processing

mean auc .66 .67 .68

Other paper results (with heavy data addition)



How To generate CI,

Friend of mine told me to sample auc’s from data met terugleggen -> 1000x en dan krijg je CI

Next Goals

* Figure out how to get AUC during training, kind of annoying right now/manual
* Ranking score (LRAP), probably scores that should be sufficient)

Compare these scores with base, post process, no postprocess