The extraction of musical instruments from bachata songs using semantic segmentation for audio source separation

Student Name: Tamas Illes

Supervisor Name: Dr Chris Willcocks

Submitted as part of the degree of MSci Natural Sciences to the

Board of Examiners in the Department of Computer Sciences, Durham University

2020.05.15.

Abstract —

Context/Background: Audio source separation is a well researched topic with various implementation designs, most of which focuses on speech and vocal separation. Little work has been done in this field involving music, with few designs using the most recent advances of convolutional neural networks.

Aims: This study investigates the possibility of using semantic segmentation and convolution layers with square-shaped kernels to separate instruments (guitar, bass, bongo, guira) from bachata music.

Method: To test the proposed idea, first a U-Net style model was implemented on the MNIST dataset to separate an integer from a mixture of numbers. Then, the same architecture was applied on the bachata audio dataset (consisting of 4-beat-long clips) to filter an instrument from a mixture of instruments using the spectrograms of the audio clips. Finally, the trained model was applied on original songs to extract instruments from them.

Results: The model achieved good outcomes on the MNIST dataset, however the edges of the predicted numbers remained slightly blurry. Training on audio clips produced similar results: an overall decreasing loss function and near perfect instrument predictions with occasional background noise. On the other hand, the quality of predictions from original songs was significantly lower. Furthermore, it was found that songs with an introduction not adopting typical 4/4 timing of the genre yielded worse predictions relative to the rest (”intro-issue”). It was also evident that the predicted audio of the guitar and bass were much clearer than those of the bongo and guira.

Conclusions The proposed architecture achieved good quality results on the MNIST and bachata audio dataset. Although the filtered instruments from songs were recognisable in a few cases, the implementation still requires further work in order to produce clear outputs with minimal background noise and to resolve the ”intro-issue”. A range of possible improvements for future research has been suggested.

Keywords — audio source separation, semantic segmentation, meta-learning, spectrogram, short-time Fourier transform, bachata

I INTRODUCTION

Audio source separation is an active research area in machine learning. Although most works focus on speech and vocal separation, successful source separation in music could lead to significant educational benefits. Many methods exist to achieve this, however, instrument extraction from songs via deep neural network became popular only in recent years. In this study we investigate the possibility of extracting musical instruments from a song using audio source separation through semantic segmentation.

We introduce a convolutional neural network which focuses on monophonic musical audio without any accompanying visual information, and uses the advanced torchaudio module by pytroch and square-shaped kernels in the convolution layers. To our best knowledge, this setup for audio source separation in music has not yet been proposed.

The fundamental idea was to extract a type of source signal (*x_groundtruth* = *x_gt*) from a mixture of source signals (*x_mix*) using knowledge about a different signal of the same type (*x_target*). Hence a combination of a particular input (*x_mix* and *x_target*) formed a task that the network attempted to solve. Since mixtures and targets were generated and selected at random, each iteration of the model yielded a new task, and therefore by definition our method was meta-learning.

A Deliverables

The basic objective of the study was to implement a semantic segmentation model on a clutter of numbers from the MNIST dataset using the popular U-Net architecture (Ronneberger et al. 2015) with skip connections. In this setup, *x_mix* was a cluttered image of random integers, and *x_target*, based on which we were trying to separate *x_gt* from *x_mix*, was a random integer whose tag was the same as *x_gt*, but the actual image differed. That is, if the mixture contained [3,8,4] and we were trying to find number 3 in this mixture, then the target and ground truth images were both of number 3 but written in two different ways, such that the ground truth image was pixel-wise present in the mixture while the target one was not. See Figure 2 for illustration.

The application of a similar U-Net scheme on audio clips instead of numbers was our intermediate deliverable. In this case, instruments represented the tags and thus *x_gt* and *x_target* corresponded to two clips of the same instrument playing different patterns, while *x_mix* was a mixture of instruments playing together (including *x_gt*), representing a song. To do this, the training clips first had to be ordered, resampled, and normalised in length, so that all of them had the same sample length, sample rate, tempo, and were not too big in size. Next, using a Short-Time Fourier Transform (STFT) we obtained the magnitude and phase matrices for each clip. The spectrograms, which are the representation of audio in the time-frequency (T-F) domain, were then calculated from the magnitude matrix and served as inputs to our convolutional neural network (CNN). The network took two spectrograms as inputs: *x_target* and *x_mix*, and produced the predicted spectrogram (*x_predicted*) of *x_gt*, using mean squared error (MSE) between *x_predicted* and *x_gt* as its loss function. This separation in the T-F domain is also called T-F masking, where the predicted spectrogram is the T-F mask. Once the model finished training, it was applied on an unseen mixture and target spectrograms for testing. The resulting T-F mask was then converted back to a magnitude matrix. Ultimately, by applying the inverse STFT on the combination of the predicted magnitude matrix and the phase matrix of the original mixture, we reconstructed the audio prediction of the chosen instrument.

Processing a full song using our trained model to extract a specific instrument was the project's advanced objective. To do so, short song clips, similar to those used for training, had to be processed. This was done by first splitting the song into clips according to their beat locations and pre-processing them to make them suitable for our neural network, which then produced the predicted audio clips. Next, these clips were post-processed to ensure their tempo matched those from the original clips before concatenating them to construct the full song length predicted audio of the chosen instrument.

B Bachata

Our project is based on bachata, which is a latin music originating from the Dominican Republic. There were several reasons for this choice. Firstly, most bachata songs consist of the same five instruments: the lead guitar (requinto), the rhythm guitar (segunda), the bass guitar (bajo) and two percussion instruments: the guira and the bongo. Moreover, the sound of these instruments has very similar pitch across bachata songs, unlike for example the electric guitar in rock music, which shows substantial variation in pitch across songs. Secondly, these instruments tend to play specific patterns in various sections of the song, regardless what song it is. These sections are: verse, chorus, and mambo (bridge). As each of the instruments have a typical pattern corresponding to section of the song, it is possible to predict what they are likely to play in a given section of a bachata song without having prior knowledge about the song. Nonetheless, another crucial motive for choosing bachata was the available training dataset. For our model to work, immense training data was required, where each instrument had its own single-track file, providing the ground truth for the network. However, due to recording studios' rights, it was impossible to gain access to multitrack versions of original bachata songs. Creating our own dataset from scratch was also futile, since we did not have enough experience in this field to compose bachata music, and our time was also limited. An alternative and perhaps even better strategy was to obtain pre-written loops from Producers Vault's Bachata DJ packs (*Dataset: bachata sample loops* n.d.). What this package contained was the required resource of individual clips containing a half-, one- or two-bar long loop of a particular instrument played in a particular section. In addition, the rich pattern variations per instrument per section allowed us to produce a very large training dataset.

In the following section, we explore the literature pertaining to this research field. Later, we propose our methods to achieve the set objectives, followed by a discussion of the results. Finally, we reflect on the limitations of the proposed design and provide suggestions for further research that could improve the outcomes.

II RELATED WORK

A A Brief History of Meta-learning

Contemporary machine learning techniques usually train models only on one specific task, which prevents them from adopting quickly when given a new, different task. Although by using deep neural networks, models proposed by Silver et al. (2016), He et al. (2015) and Devlin et al. (2018) in recent years have accomplished outstanding results, the success of these and similar designs is largely dependent on the immense available dataset and computing power. Consequently, there are clear limitations in areas where such resources are not accessible.

Meta-learning, or ”learning to learn”, is a machine learning technique which allows the model to quickly adapt to new tasks, based on experience, or meta-data, gained from a variety of (often related) training tasks. Having been trained this way, meta-learners search for the algorithm that is best suited for the given problem based on its meta-data. This learning method in neural networks, explored by Thrun & Pratt (1998) and Schmidhuber (1987), mitigates the limitations of the contemporary designs mentioned earlier. Furthermore, it provides a learning experience closer to human learning, as explained by Harlow (1949) and Lake et al. (2017). That is, humans (and animals) hardly ever start learning a new skill from scratch: instead, we use previously acquired experience and skills from similar tasks in order to solve a new, unseen problem. Meta-learning, alleviating the limitations of the standard one-task based models, led to recent successes in a variety of machine learning fields: Snell et al. (2017) in few shot image recognition, Metz et al. (2019) in unsupervised learning, and in reinforcement learning by Duan et al. (2016) and Alet et al. (2020).

B Deep Learning for Semantic Segmentation

Semantic segmentation, primary used in autonomous vehicles (Hofmarcher et al. 2019), robotics (Schnieders et al. 2019), and human-computer interaction (Panwar & Singh Mehra 2011), is the technique of labelling each pixel of an image with classes representing the region that contains the given pixel. It is essentially a pixel-level classification of the image. Since the arrival of the exceptionally successful convolutional neural networks (LeCun 1989), semantic segmentation became a well researched area in computer vision with excellent achievements in recent years.

Since semantic segmentation classifies pixels of images, it is crucial to have a coherent and universal labelling system that allows the use of multiple datasets. Papandreou et al. (2015) introduced a network which can train on a combination of weakly- and well-labelled data, using deep CNNs and fully-connected conditional random field. By reaching 73.9 % mean intersection-over-union (IOU) accuracy on PASCAL VOC 2012, this method proved that using a smart combination of weakly- and well-labelled data improves performance.

Achieving 92% mean IOU score on the ISBI cell tracking challenge in 2015, the "U-Net" architecture implemented in Ronneberger et al. (2015) became a popular state-of-the-art design in semantic segmentation. This method was built from fully convolutional layers in an autoencoder style. This novel design introduced skip connections between downsampling and upsampling paths, which preserved locality information, yielding more precise segmentation after just a few training images. Today U-Net style architectures are common designs in semantic segmentation models.

A fully convolutional DenseNet implementation was carried out in Jégou et al. (2017), in which each layer was connected to every layer in a feed-forward fashion making the training process easier and increasing the accuracy of the outputs. The architecture was a modified U-Net, using multiple dense blocks (DB) in the down- and upsampling paths: 2 DBs per path plus one in the bottleneck, with skip connections in between. Such upsampling paths using DBs outperformed other upsampling methods, achieving state-of-the-art results on the CamVid and Gatech datasets without the need of any post-processing module.

Another interesting approach was taken in a very recent study on semantic segmentation: authors of Takikawa et al. (2019) argue that processing various properties of objects (shape, color, size, etc.) separately, and thus imitating how the brain processes information, should yield to better performance in comparison to designs where all information form a single dense image representation. Therefore, a two-stream convolutional neural network was proposed, where one branch was dedicated to capture boundary related information only, using gated convolution layer (GCL) and local supervision. This architecture produced state-of-the-art results with more precise predictions around the boundaries of objects, as well as a 7% IOU improvement on thinner and smaller objects.

C Audio Source Separation Techniques

Filtering an audio source from a set of mixed signals has always been a laborious challenge in computer science. Over the past decades several attempts have been made on audio source separation, with methods spanning Gaussian Mixture Model (GMM) (Hirasawa et al. 2012), Non-negative Matrix Factorization (NMF) (Virtanen 2007), Independent Component Analysis (ICA) (Makino et al. 2004), Computational Auditory Stream Analysis (CASA) (BT et al. 2016), Pitch Estimation and Tracking (McAulay & Quatieri 1990), Sparse coding (Virtanen 2003), and Support Vector Machine (SVM) (Mozer 2002).

A classical approach used for audio source separation is the NMF. In fact, NMF has a wide range of use such as text retrieval, collaborative filtering, or spectral filtering. In Févotte et al. (2018), the authors performed spectral decomposition of sound sources using unsupervised NMF to break down a spectrogram into elementary spectral components whilst minimising the model's cost function. Although the method showed interesting and useful results on samples consisting of a single NMF component, this is rarely the case for real audio signals, since a single audio source could have multiple NMF components. Classification of individual NMF components into sources have been proposed, however additional research is still necessary in order to reduce processing time and adapt for real-time applications.

Recent studies investigate the use of deep neural networks for audio source separation. A deep CNN for estimating a binary T-F mask was proposed in Chandna et al. (2017). Vertical and horizontal convolution layers were used on spectrograms of audio mixtures in order to produce estimations for the separated sources. From these, time-frequency soft masks were computed and applied to the spectrograms of the mixtures to estimate the magnitudes of the separated sources. Output audio estimations were then obtained from these estimated magnitudes and the phases of the mixture (Liutkus & Badeau 2015). The model achieved competitive results on the Source to Distortion Ratio (SDR), Source to Interference Ratio (SIR), Source to Artifacts Ratio (SAR), and Image to Spatial distortion Ratio (ISR) tests compared to a simple multilayer perceptron, with significant improvement in processing time.

A recent study in Zhao et al. (2018) explored the visual-audio relationship of videos within the deep learning field. It combined the audio and visual information of unlabelled videos to extract audio signals from them by locating regions of the video which produce sounds and separating the sounds into components that correspond to sounds from each pixel. The model learnt to identify sounds in vision, allowing independent volume adjustments of sound sources. The authors at first separately analysed the visual and audio information of videos, which was then combined in an audio synthesiser network, computing the sound of each pixel. Similarly as in Chandna et al. (2017), the synthesiser network output a time-frequency mask which was applied to the input spectrogram using Wiener like filtering (Liutkus & Badeau 2015). Qualitative measurements showed that even with complex input spectrograms, the model successfully segmented the target instruments from the mixture. Quantitative sound separation performance tests on the Normalized Signal-to-Distortion Ratio (NSDR), SIR, and SAR metrics showed that binary masking in log frequency scale outperforms other suggested designs (within this study) as well as the baseline NMF and DeepConvSep approaches introduced in Virtanen (2007) and Chandna et al. (2017), respectively. Further subjective evaluation proved that the binary masking based architecture outperforms other designs in sound separation as well as in the visual-sound correspondence problem.

III SOLUTION AND RESULTS

A five level deep U-Net architecture was implemented in this project, going from tensor shape [B x 2 x h x w] to [B x 1024 x $h/2^4$ x $w/2^4$] during downsampling and up to [B x 1 x h x w] in the upsampling period with skip connections at every level from the downsampling path to provide relevant locality information in decoding. Here, B was the batch size and h and w were the height and width of the input images, respectively. Our model, which was fed three inputs: a mixture image, a ground truth image and a target image, attempted to separate the ground truth from the mixture based on information about the target image. Constructing the mixture by combining randomly selected distinct images allowed us to build an extensive training dataset, rather than using a premade and subsequently limited dataset. While the mixture and target images were direct inputs to our network, the ground truth image was used in the comparison with the predicted image. Since our task was to find an image in another one of the same size, it was straightforward to use MSE (mean squared error) to calculate the loss. Hence the model's objective was to minimise the loss function in Equation 1 and update the training parameters accordingly.

$$\mathcal{L}(p, \hat{p}) = \mathbb{E}\left[(p(h, w) - \hat{p}(h, w))^2\right] \tag{1}$$

Above, p and $\hat{p}$ are the ground truth and predicted image respectively, while h and w are the height and width of the images. The general outline of training and testing stages is demonstrated in Figure 1, using the described U-Net architecture shown in yellow.

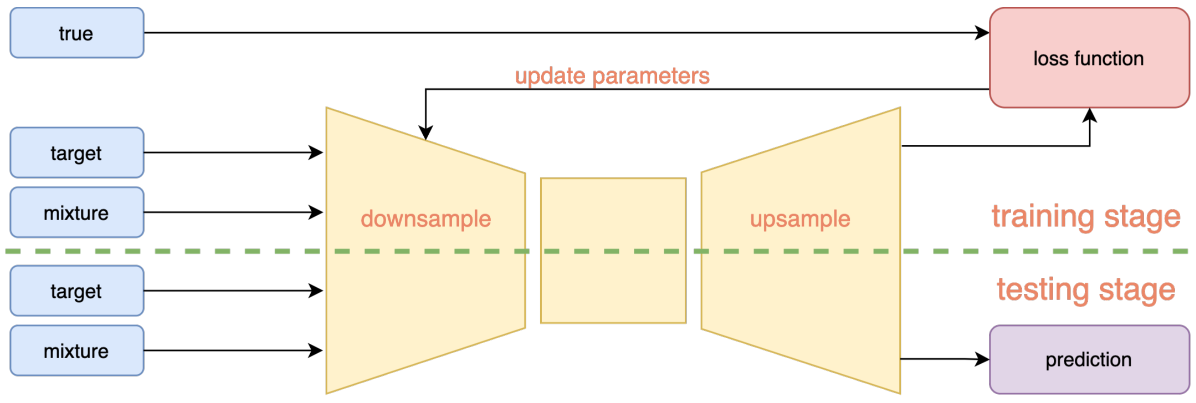

Figure 1: The general framework of the implemented models in this project.

In pursuit of to our objectives, we implemented the above-mentioned design first on the MNIST, and then on our bachata audio dataset. The application on numbers served more as a verification that our proposed design was successful, which gave the motive and reason for the implementation on the audio dataset. Therefore, in this section we discuss the model implementation and results on the numbers and audio dataset separately, following the training outline in Figure 1. Finally, we apply the trained model on original songs and analyse the outcomes.

A *MNIST Dataset*

In this section we discuss the implementation of our framework on the MNIST dataset and evaluate its performance.

A.1 Implementation

In the case of the MNIST dataset, the model was fed with three images: a mixture of numbers, and two different images (ground truth and target) of the same number. The ground truth image had to be present in the mixture, while the target image must not, as shown in Figure 2. The mixture was obtained by taking the pixel-wise maximum of multiple random images of distinct integers between 0 and 9. Due to the nature of our design it was convenient to have square-shaped inputs whose size was some power of 2. For this reason, the 28x28 pixel MNIST images were converted to 32x32 via interpolation prior to feeding them into the network. Therefore, according to our U-Net architecture, the shape of the latent space in this case was [B x 1024 x 2 x 2].

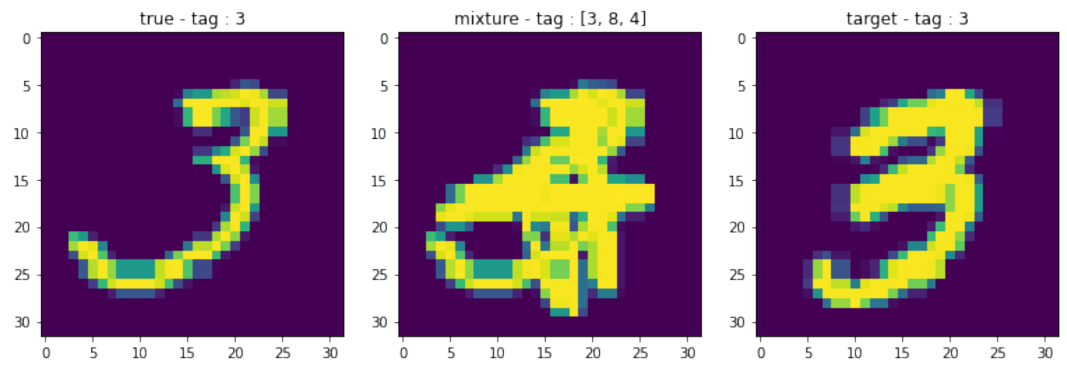

Figure 2: An example of an image that we are trying to find (*x_gt*, left) in a mixture (*x_mix*, middle) based on a different image of the same number (*x_target*, right).

A.2 Results and Evaluation

To evaluate its performance, we trained our model using a batch size of 64 and a mixture of 4 numbers (as would be the case for audio). One can see in Figure 3 the evolution of the predictions: 8 samples of 5 random instances were taken every 1000 iterations starting from 0. The observed prediction accuracy increased over time, most noticeably in the first few thousand iterations: by the 4^{th} and 5^{th} thousand iteration most predicted numbers were clearly recognisable, and further training appeared to produce slightly more defined predictions. Therefore, our visual perception was that the model in early stages outlines the sought number from the mixture, followed by further fine tuning in later stages. This notion is also visible in Figure 4, where we plotted the loss behaviour over 100 epochs using 100 iterations per epoch (left). For better visualisation we also calculated the mean of every 5 and 10 epochs with corresponding error bars in the middle and on the left of the figure, respectively. One can see the monotonic decrease in loss (or increase in accuracy) on the middle graph till the 40^{th} epoch (4000^{th} iteration, as expected) followed by a rather jagged period with random spikes, but overall still decreasing, as shown on the right. These findings further support our earlier observation on the prediction evolution in early stages.

We also noticed that despite a very long training period, the predictions still remained slightly blurry and sometimes messy in comparison to their corresponding ground truth. By expanding

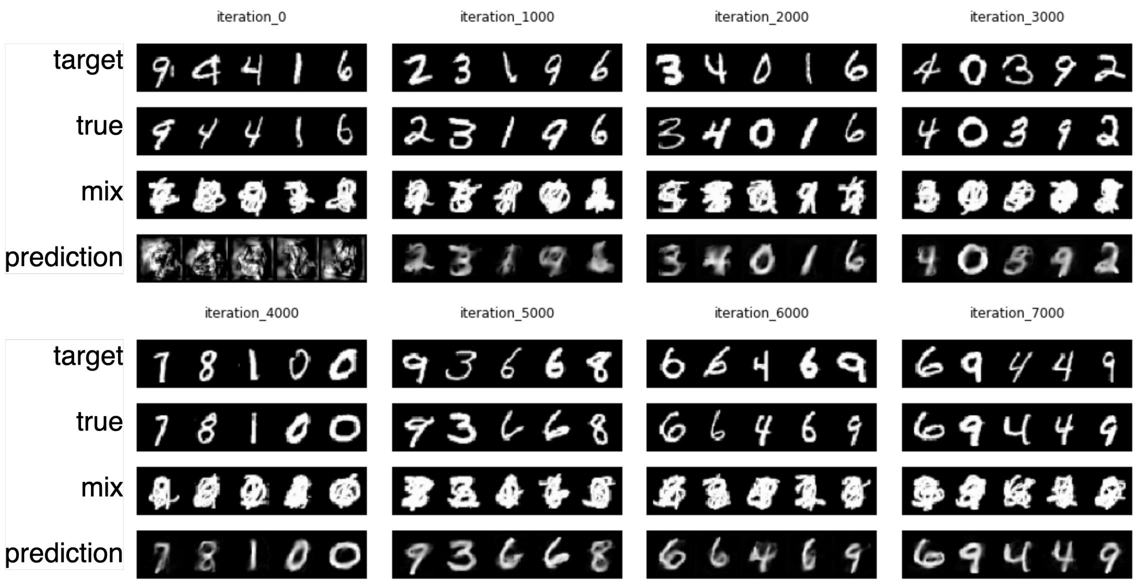

Figure 3: The evolution of number predictions: 8 samples of 5 random instances at every 1000 iterations.

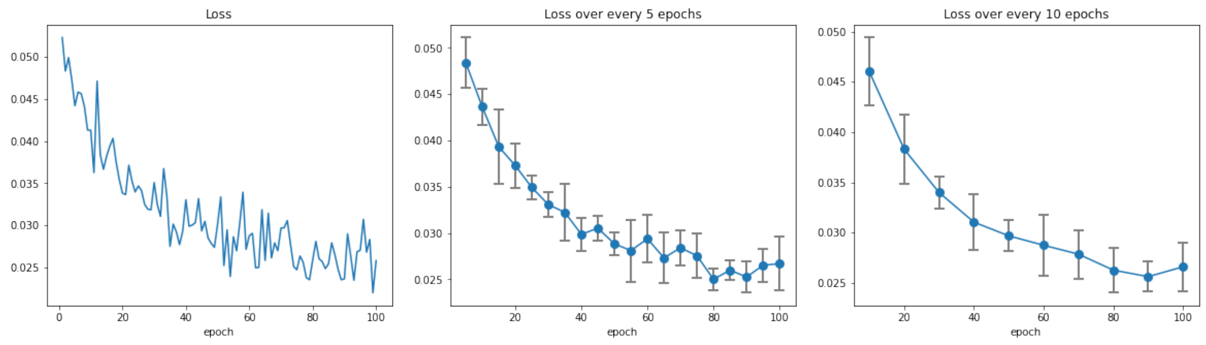

Figure 4: The loss behaviour of our predictions on the MNIST dataset per epoch (left). In addition, the mean of this loss over every 5 epochs (middle) and every 10 epochs (right) was measured, with corresponding error bars.

the model with a generative adversarial network (GAN), we could possibly achieve well defined and non-blurry predictions, however testing this hypothesis was beyond the scope of this study.

Still, these results proved the potential of the proposed architecture and therefore gave further support for the implementation and investigation of the same model structure on our audio dataset.

B Audio Dataset

Motivated by the performance of the model on the MNIST dataset, in the following section we discuss the application and results of the same neural network architecture on the monophonic bachata audio dataset.

B.1 Implementation

Before building our model, the bachata loops had to be carefully ordered and normalised, as follows.

Pre-processing. Our bachata audio dataset consisted of a large number of half-, one- and two-bar (1 bar = 8 beats) long loops of various individual and mixed bachata instruments, although we were only interested in the basic single-instrument loops: bongo, guira, bass, and guitar (Hernandez 1995). Therefore in the first stage of pre-processing we chose a bar length to work with and ordered the relevant loops into instrument categories. We proceeded with the 1-bar long loops, since their number of samples was the highest. Next, to speed up computation we resampled the clips from 44100 Hz to 11025 Hz, although it should be noted that this reduced the quality of the clips. At this point, although we selected only 1 bar long loops, there was still slight variations in their sample lengths and tempo, which would not be suitable for further computation and comparison. Because of this, we normalised the resampled clips to have a consistent sample length of 36000 samples, which corresponded to 147 BPM (Beats Per Minute), by stretching or shrinking them in the time domain using the *stretch* function provided by gaganbahga (2018). The stretching factor was defined as the sample length of the clip divided by 36000. Besides an audio array and stretching factor, the window size of the Fast Fourier Transform (FFT) used in the stretching process (*nfft*) also had to be defined. We found that using a common *nfft* value for all audio clips would result in serious quality loss for some instruments, which is likely to be due to their varying pitch. For instance, the guira has a much higher pitch than the bass, meaning that the rate at which the sound of the bass vibrates is much lower than that of the guira. Consequently, we found that the following instrument-specific optimal *nfft* values led to acceptable stretched audio qualities: bongo – 80; guira – 75; guitar – 160; bass – 500. These numbers also reflect the pitch variation across instruments: higher *nfft* corresponds to lower pitch. Although bachata songs commonly consist of full bars, they occasionally contain a half-bar at the end of the *verse* or *chorus* sections, which would cause a shift in the pattern of the next section by half a bar. For this reason, we further split our training loops into 2 half-bar long clips, which resulted in twice as many training data with sample length of 18000.

The model. Figure 5 demonstrates the flowchart of the model we built in this section.

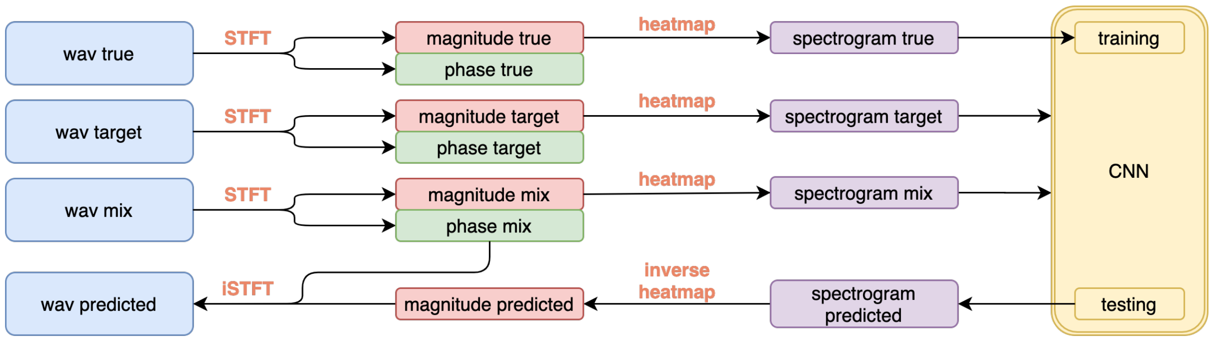

Figure 5: The flowchart of our design on the audio dataset, implemented within the framework described earlier.

Once we achieved an ordered and downsampled audio dataset with consistent sample length, we had to construct the input mixture for the model. This was done iteratively during training by summing up the waveforms of the randomly chosen audio clips with corresponding weights. These weights were introduced to account for discrepancies in volume: the guitar samples were much louder, while the guira ones were much weaker in the mixture relative to the other instruments, which reflected an inaccurate balance with respect to actual bachata songs. Thus, we multiplied the waveform of the guira samples by 3, and divided those of the guitar by 3, which resulted in a more authentic mixture. The bongo and bass audio samples remained intact.

These audio clips then had to be converted into images due to the nature of convolutional neural networks. A useful visual representation of an audio clip is its spectrogram, which captures the frequency against time information of the audio. To do this, first a Short-Time Fourier Transform (STFT) had to be applied to the clips with optimised input parameters to obtain complex matrices ($\mathbf{Z}_{STFT}$) of size 256x256. From this we derived the magnitude ($\mathbf{M}$) and phase ($\mathbf{P}$) matrices of the clips (Zhao et al. 2018):

$$\mathbf{M} = \mid \mathbf{Z}_{STFT} \mid = \sqrt{\mathcal{R}(\mathbf{Z}_{STFT}) \odot \mathcal{R}(\mathbf{Z}_{STFT}) + \mathcal{I}(\mathbf{Z}_{STFT}) \odot \mathcal{I}(\mathbf{Z}_{STFT})}$$
$$\mathbf{P} = \arctan\left(\frac{\mathcal{I}(\mathbf{Z}_{STFT})}{\mathcal{R}(\mathbf{Z}_{STFT})}\right) \tag{2}$$

Note that the piecewise operations in Equation 2 maintained the size of the matrices and that the columns and rows represented time and frequency, respectively, in both $\mathbf{M}$ and $\mathbf{P}$. Furthermore, while the values of $\mathbf{M}$ implied the intensity of a particular frequency at a given time, those of $\mathbf{P}$ indicated the piecewise angle of the complex matrix $\mathbf{Z}_{STFT}$. Although the phase matrices were not used in the formulation of spectrograms, they were essential in the reconstruction of the predicted audio. Using the phase information of the mixture in audio retrieval after T-F masking is called Wiener-like filtering (Liutkus & Badeau 2015), which is a common practice in the audio source separation framework. Phase prediction via deep learning is an active research area and was beyond the scope of this study.

The obtained magnitude matrices were spectrograms of the original clips, however further computation was required to reveal their important features for the neural network. By applying the following *heatmap* computations to them, we obtained images like the examples shown in Figure 6.

$$\mathbf{S}^* = log_{10}(\mathbf{M} + 1) \tag{3}$$

$$\mathbf{S} = \frac{\mathbf{S}^*}{max(\mathbf{S}^*)} \tag{4}$$

Equation 3 depicts the spectrograms scaled up to a visible spectrum which clearly indicated the beat locations and the corresponding frequency magnitudes (Zhao et al. 2018). This was followed by a normalisation in Equation 4, which standardised all input spectrograms within a given batch to have values between [0,1], yielding consistency across tasks.

Ultimately, the obtained spectrograms of size 256x256 could be fed into our neural network. A detailed picture of the proposed U-Net architecture for this audio dataset is presented in Figure 7. Batch size of 32 was used in our experiments, and thus the network encoded the [32 x 2 x

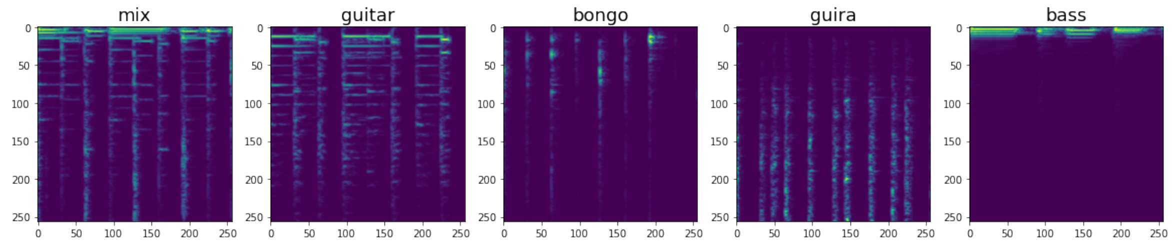

Figure 6: An example of a spectrogram of a mixture (left) and of the underlying instruments within: guitar, bongo, guira, and bass.

256 x 256] shaped inputs to [32 x 1024 x 16 x 16] shaped images in the latent space, which was then decoded to attain the predicted spectrograms, $\mathbf{S}_p$, of shape [32 x 1 x 256 x 256]. The output spectrogram values were then clamped between 0 and 1 before computing the loss.

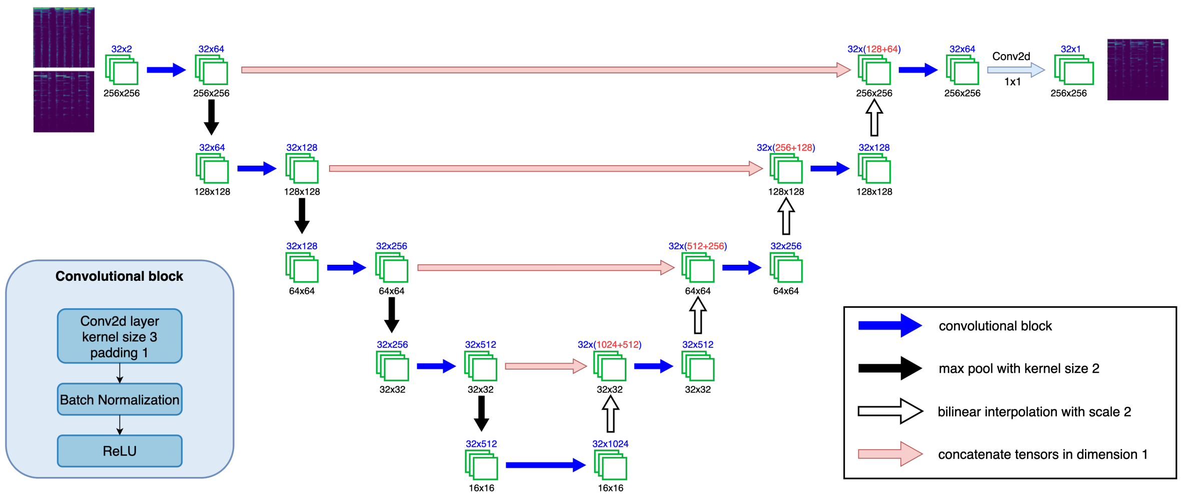

Figure 7: A detailed diagram of the implemented U-Net architecture on the audio dataset.

Following the flowchart in Figure 5, to retrieve audio data from the predicted spectrogram one had to reverse the computations in Equation 3 and Equation 2 (Zhao et al. 2018):

$$\mathbf{M}_p = 10^{\mathbf{S}_p} - 1, \tag{5}$$

$$\mathcal{R}(\mathbf{Z}_{STFT_p}) = \mathbf{M}_p \cdot cos(\mathbf{P}_{mix}) \qquad \mathcal{I}(\mathbf{Z}_{STFT_p}) = \mathbf{M}_p \cdot sin(\mathbf{P}_{mix}), \tag{6}$$

where $\mathbf{M}_p$ and $\mathbf{Z}_{STFT_p}$ were the predicted magnitude and predicted complex STFT matrices respectively, and $\mathbf{P}_{mix}$ was the phase matrix of the input mixture. By applying the inverse STFT on $\mathbf{Z}_{STFT_p}$ we acquired the predicted audio clip of the extracted instrument.

B.2 Results and Evaluation

The decisions to normalise the input spectrograms via Equation 4 prior to training and to clip the predicted output values between 0 and 1 was based on performance results after experimentation with various model settings. In Figure 8 we demonstrate the loss results of the model that was trained on 100 iterations for 100 epochs in three different ways: without clipping or normalisation (left), with clipping but without normalisation (middle), and using both clamp and normalisation (right).

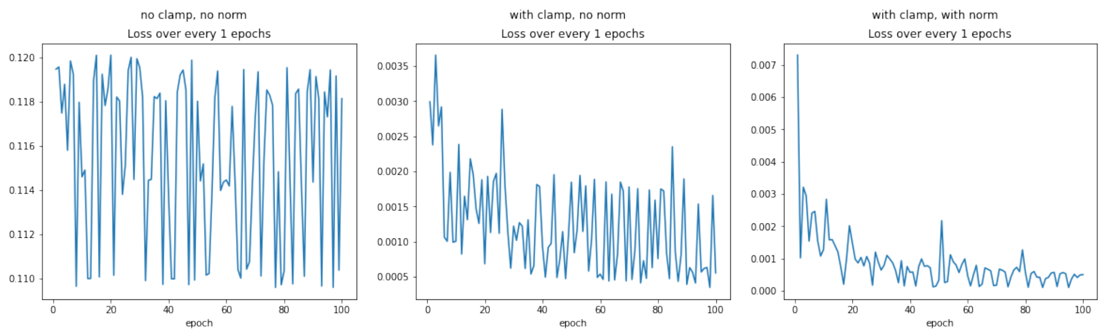

Figure 8: Accuracy behaviour of the model trained (on 100 iterations for 100 epochs) without clipping or normalisation (left), with clipping but without normalisation (middle), and with both clipping and normalisation (right).

Although the graphs may seem random, we can see a decreasing behaviour as we add clamping and normalisation to our design. Therefore throughout the remainder of this paper we will only consider models built with these functionalities.

In Figure 9 we further examined the loss behaviour of the network over 100 epochs, with 100 iterations per epoch. Plotting the loss at every 5, 10 and 20 epochs with corresponding error bars showed an overall decreasing tendency, as desired. The irregularity and jaggedness of the loss per epoch graph (top left) was possibly due to the random choice of instruments during training. To examine this, we trained the network with 20 iterations for 50 epochs per instrument, meaning that instead of randomly choosing a source to extract, we trained 5 models: one for each instrument, and one with random choice. The resulting losses are shown in Figure 10.

Our findings indicated that the bass extraction had the least loss throughout the training period, and the guitar had higher loss than any other instrument. This was because, as is visible in Figure 6, the bass spectrograms had the clearest distinction between high and low regions, while those of the guitar were more complex. The bongo and guira spectrograms were somewhere in between the two, with more ambiguous active regions than those of the bass, but less sophisticated than those of the guitar. For this reason their loss fell between the bass and guitar losses, as one can see in Figure 10. Therefore, we deduced that the irregularity in the loss function with random instrument choices (in purple) was due to its randomness.

It is important to note that we did not have a testing dataset, since at every iteration the network was fed a new, unseen task, and therefore testing our model simply meant to execute one more iteration with a mixture and target inputs. For the same reason, the loss calculation in the testing period was unnecessary, since the training loss already indicated the expected loss in testing. Whilst we could not produce any further quantitative measures to test our model, since the project was audio based, we were able to listen to the predicted outcomes and compare them with their ground truth audio. To augment our quantitative results with qualitative findings, a number of example audio samples are accessible via Table 1 (click on either the example number or speaker icon). Each example consists of a random mixture and the predicted (*pred*) and ground

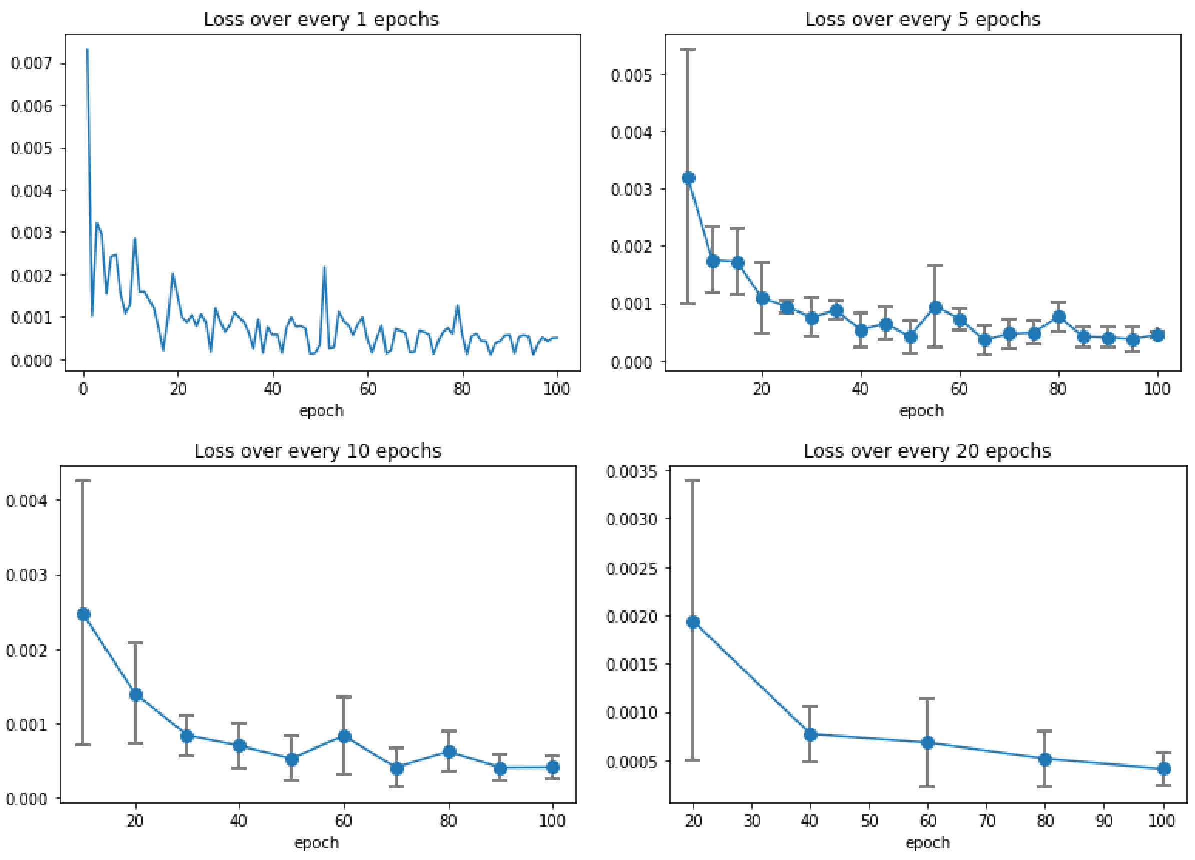

Figure 9: Training loss of the model on the audio dataset over 100 epochs with 100 iterations in each epoch. Measurements were taken at every epoch (top left) and the mean loss is shown over every 5 (top right), 10 (bottom left), and 20 (bottom right) epochs with corresponding error bars.

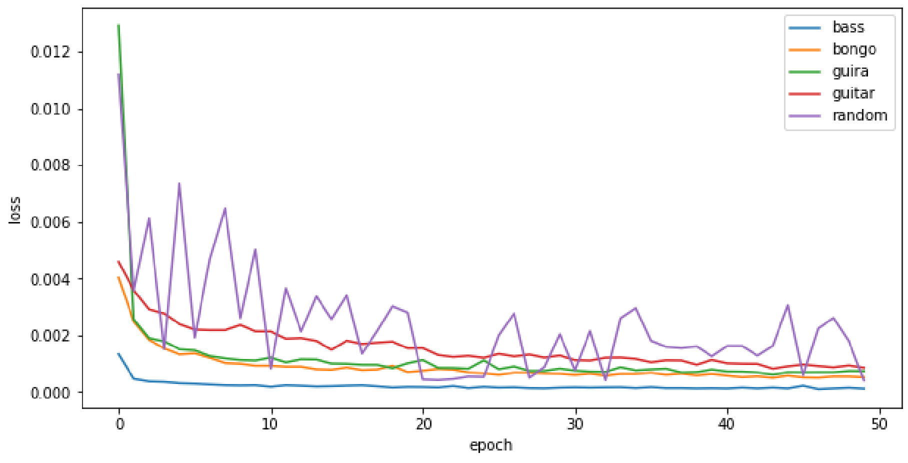

Figure 10: The accuracy of the model trained using single instruments on 20 iterations for 50 epochs. The training loss of a model with random instrument choice under the same experimental setup was also measured (purple).

truth (*true*) audio of each instrument.

example	**mixture**	**bongo**		**guira**		**bass**		**guitar**	
		true	*pred*	*true*	*pred*	*true*	*pred*	*true*	*pred*
1	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊
2	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊
3	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊
4	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊
5	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊	🔊

Table 1: Some of the audio samples used for qualitative evaluation: a set of mixtures and the underlying ground truth and predicted audio clips of its instruments (click on the speaker icon to listen). Also available in *Results: audio samples* (n.d.).

It is undeniable that the model achieved positive results, however, in some cases the predicted instruments did not sound as clear as their corresponding ground truth. This was likely due to lacking accurate phase for estimation: as mentioned earlier, the phase of the mixture was used in combination with the predicted magnitude in order to construct the audio prediction (Liutkus & Badeau 2015).

C Application on Songs

After successfully filtering instruments from half-bar long clips, our final deliverable was to carry out instrument extraction on full songs using our trained model. The implementation and results of this process is detailed in this section.

C.1 Implementation

Since the model was trained on 4-beat long clips, our task was to split the bachata songs into clips of appropriate length and size, feed them to our network, and connect the resulting clips. The flowchart in Figure 11 details these steps. Initially we identified the beat locations of the

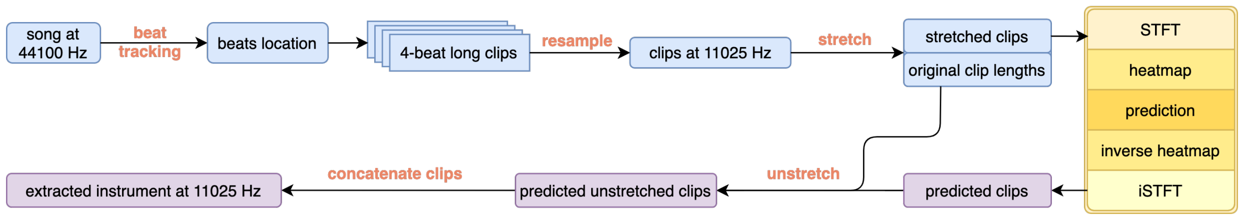

Figure 11: The outline of the application of the trained model on bachata songs.

song using beat tracking, which allowed us to split the song into consecutive clips of 4 beats, starting from the first detected beat. Next, we resampled and stretched them so that their sample rate and tempo was 11025 Hz and 147 BPM, respectively, which was consistent to that of the training data. Note that the stretching factor in this case was the sample length of the clip divided by 18000. The example in Figure 12 compares a spectrogram of a mixture from the training dataset (left) to one of a clip from a song (right). The difference is obvious: the left spectrogram, which we trained our model on, is much less noisy, while the right one is much more cluttered. Seeing this we suspected that the predictions from songs would be less clear than those from subsection B.

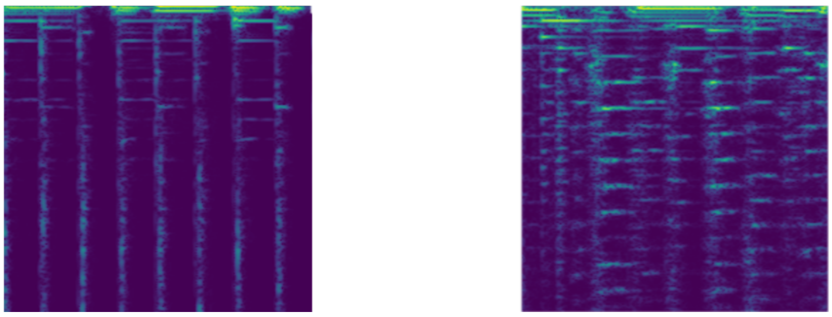

Figure 12: Comparing spectrograms of a training mixture (left) and a song clip (right).

The pre-processed clips were then fed into the trained model, however due to the architecture of our network, it was necessary to provide a sample spectrogram of the wanted instrument with appropriate shape. Therefore, once we chose an instrument to filter, a random clip of this instrument was selected from the training dataset to serve as the target image. The already trained network then output a predicted spectrogram, from which the audio prediction was retrieved using the same methods as in subsection B. Next, to recover the original tempo of the song, the predicted audio was un-stretched (or un-shrunk) using the original sample length of the input clip. The resulting clips were then concatenated to one another to form the full-length extracted instrument audio at 11kHz sample rate.

C.2 Results and Evaluation

Since ground truth audio was not available for original songs, we could not present quantitative measurements for these experiments. We could however, provide qualitative audio samples of the outcomes: the original audio and the extracted instrument predictions of 6 example songs are accessible through Table 2 (click on either the song name or speaker icon). In general, the predic-

Song name	original	bongo	guira	bass	guitar	assessment
Vanidosa	🔊	🔊	🔊	🔊	🔊	good
El guardaespaldas	🔊	🔊	🔊	🔊	🔊	good
Te robare	🔊	🔊	🔊	🔊	🔊	fair
Vagabundo, borracho & loco	🔊	🔊	🔊	🔊	🔊	fair
A donde va el amor	🔊	🔊	🔊	🔊	🔊	poor
Por favor	🔊	🔊	🔊	🔊	🔊	poor

Table 2: Examples of songs and their extracted instruments alongside with a qualitative assessment on them (click on the speaker icon to listen). Also available in *Results: song samples* (n.d.).

tions were much more ambiguous than those from subsection B, which was expected since the songs may have used unseen instruments, multiple guitars, vocals, possibly different volume balance between instruments, various unseen patterns or rhythms, and perhaps they were recorded in a way that produced unclear spectrograms in our model unlike the ones we used during training. Nevertheless, we managed to carry out a relative comparison. In total we analysed 14 songs, of which 2 we found to have relatively good results, 3 fair results, and the remainder poor quality results. Two examples of each are available through Table 2. Our assessment indicated a success rate of 1 out of 7, although it is important to note that successful extractions were still far from studio quality. We also noticed that the bass and guitar extractions were much more successful than those of the bongo or guira. The bass predictions were mostly clear, as expected, and despite having relatively complicated spectrograms, the guitar extractions produced surprisingly good results. The bongo and guira predictions suffered the most from the lack of an accurate phase matrix: often they sounded like background noise rather than clearly defined instruments. In addition, in most cases one could still hear some of the underlying music in the background of the predictions. This effect was most evident in the bongo and guira extractions.

Furthermore, after attaching the predictions together, we noticed that the beats on the edges of any two adjacent clips were more powerful than the rest, which was undetectable when listening to only a single predicted audio clip. Since this effect only happened on the edge of neighbouring clips, these dominant beats coincided with the starting beat of each predicted clip. We suspected that the unsynchronised edges of the predicted spectrograms could be the reason for this, however further investigation was outside the scope of this project.

Whilst initially considered problematic, the strong starting beats revealed the reason behind another issue. Namely, since they indicated the locations of the predicted clips within the full-length audio prediction, we could track which clips were processed by the model. This revealed that some songs were not sliced into clips as they should have, meaning the original songs were cut into clips without respecting the rhythmical structure of the song. For instance, a 4-beat-long clip could correspond to beats 1-2-3-4 or beats 3-4-1-2 in the original song. The former is what our model was trained on, while the latter is essentially an overlap between two half-bar clips in the song. We noticed that this effect was present in all songs with poor quality results. What these songs also had in common was a short introduction section of varying length at the beginning of the song, which was not always comprised of multiples of 4 beats. Our beat tracking method however detected beat locations from the very beginning of the song, meaning that an introduction, whose beat count was not a multiple of 4 could shift the rest of the clips relative to the original song, disregarding the rhythmical structure of it. An example of this effect is illustrated in Figure 13, where an introduction of 3 beats caused the model input clips to be shifted by 1 beat.

While all poor quality predictions suffered from this ”intro-issue”, having an introduction with suitable beat count did not have any negative effect in further predictions. An excellent example to demonstrate this is the song ”Vanidosa” in Table 2, whose introduction comprised a half-bar guitar clip, yet the predictions from this song were one of the best in our list. Conversely the song ”Por favor”, having only a 2 beat long introduction, resulted in shifted clips by 2 beats. Therefore we could deduce that the poor quality predictions were highly likely due to this ”intro-issue”. The assessment ”fair” meant that the intro-issue was not present, but the predictions were not very clear. Resolving this problem with the introductions would allow for more meaningful comparison between songs.

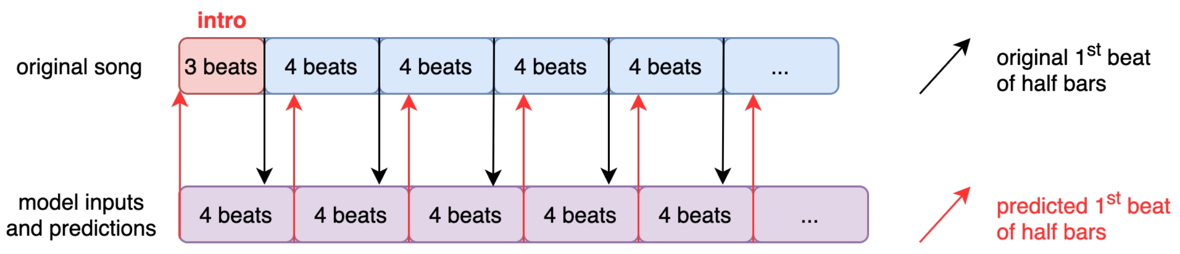

Figure 13: Demonstration of the ”intro-issue”. The introduction of a song comprised of 3 beats caused a shift of 1 beat in the model input clips and their corresponding predictions.

IV LIMITATIONS AND FUTURE WORK

Although we proved that our basic design idea worked, the results from subsection C showed that there is still a lot of scope for improvements. Below we collected ideas that could be investigated or areas that should be improved:

- Expanding our design with a Generative Adversarial Network (GAN) for more defined and less noisy predictions.
- Experimenting with a modified U-Net design, for instance, by expanding it with residual blocks to achieve better predictions.
- Using a better phase retrieval method to obtain accurate phase estimation. Perhaps predicting the phase matrix of instruments through a deep neural network.
- Training and testing on audio with 44kHz instead of 11kHz to improve audio quality.
- Examining whether the choice of the target audio matters when testing on full songs.
- Resolving the louder starting beat problem in the predicted clips from full songs.
- Training song-section-specific (verse, chorus, mambo) models, so that after classifying which section of the song is played in the clip, the appropriate model could be applied.
- Resolving the ”intro-issue” by estimating the downbeat locations instead of standard beat locations, which would indicate the correct start of every 4-beat clip.
- Generalising the model for unseen instruments.

V CONCLUSION

We saw in subsection A the evolution of number predictions and their decreasing loss function over time, as expected. Although the final predictions were clearly recognisable, they remained slightly blurry with undefined edges. This could be mitigated by adding a GAN to our design. Next, in subsection B we converted the pre-processed audio clips into spectrograms, from which the model produced a T-F mask of the target instrument. These predicted spectrograms were then converted back to audio. The loss function was once again clearly decreasing over time, and the final audio predictions were also very close to the ground truth with some occasional background noise. Finally, while the application on original bachata songs in subsection C was relatively successful in a few cases, the majority of the songs turned out to have poor quality predictions. Our findings suggest that the reason behind this was likely the ”intro-issue”, which could be resolved by identifying and using the downbeat locations in the song instead of the standard beat locations. Despite this, the quality of predictions that did not suffer from this ”intro-issue” was still undoubtedly lower than those from the training period, and therefore a range of possible mitigations and further improvements was proposed to increase the overall quality of the predictions. Finally, a rather surprising and unexpected result we encountered was the high quality results of the guitar predictions in almost all songs.

In conclusion, our idea to use semantic segmentation with square-shaped kernels in the convolution layers for monophonic audio source separation proved to be successful. We reached our basic and intermediate objectives with remarkable outcomes, however further work is required to achieve our advanced deliverable with good quality predictions.

References

Alet, F., Schneider, M. F., Lozano-Perez, T. & Kaelbling, L. P. (2020), ‘Meta-learning curiosity algorithms’, *ICLR* .

BT, S., SL, D. & I, W. (2016), ‘Computational models of auditory scene analysis: A review’, *Front Neurosci* .

Chandna, P., Miron, M., Janer, J. & Gomez, E. (2017), ‘Monoaural audio source separation using deep convolutional neural networks’.

Dataset: bachata sample loops (n.d.). `https://www.producersvault.com`.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), ‘Bert: Pre-training of deep bidirectional transformers for language understanding’.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I. & Abbeel, P. (2016), ‘RL2: Fast reinforcement learning via slow rein- forcement learning’.

Févotte, C., Vincent, E. & Ozerov, A. (2018), Single-channel audio source separation with NMF: divergences, constraints and algorithms, *in* ‘Audio Source Separation’, Springer.

gaganbahga (2018), *time-stretch*. `https://github.com/gaganbahga/time_stretch`.

Harlow, H. F. (1949), ‘The formation of learning sets’, *Psychological Review* .

He, K., Zhang, X., Ren, S. & Sun, J. (2015), ‘Deep residual learning for image recognition’.

Hernandez, D. P. (1995), *Bachata: A Social History of a Dominican Popular Music*, Temple University Press.

Hirasawa, Y., Yasuraoka, N., Takahashi, T., Ogata, T. & Okuno, H. G. (2012), A gmm sound source model for blind speech separation in under-determined conditions, *in* ‘Latent Variable Analysis and Signal Separation’, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 446–453.

Hofmarcher, M., Unterthiner, T., Arjona-Medina, J., Klambauer, G., Hochreiter, S. & Nessler, B. (2019), 'Visual scene understanding for autonomous driving using semantic segmentation'.

Jégou, S., Drozdzal, M., Vázquez, D., Romero, A. & Bengio, Y. (2017), 'The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation', *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* pp. 1175–1183.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B. & Gershman, S. J. (2017), 'Building machines that learn and think like people', *Beh. and Brain Sc.* .

LeCun, Y. (1989), 'Generalization and network design strategies'.

Liutkus, A. & Badeau, R. (2015), 'Generalized wiener filtering with fractional power spectrograms'.

Makino, S., Araki, S., Mukai, R. & Sawada, H. (2004), 'Audio source separation based on independent component analysis'.

McAulay, R. J. & Quatieri, T. F. (1990), 'Pitch estimation and voicing detection based on a sinusoidal speech model', *International Conference on Acoustics, Speech, and Signal Processing* .

Metz, L., Maheswaranathan, N., Cheung, B. & Sohl-Dickstein, J. (2019), 'Meta-learning update rules for unsupervised representation learning', *ICLR* .

Mozer, M. (2002), 'Monaural separation and classification of mixed signals: A support-vector regression perspective'.

Panwar, M. & Singh Mehra, P. (2011), 'Hand gesture recognition for human computer interaction', *2011 International Conference on Image Information Processing* .

Papandreou, G., Chen, L.-C., Murphy, K. P. & Yuille, A. L. (2015), 'Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation', *ICCV* pp. 1742–1750.

Results: audio samples (n.d.). `https://www.dropbox.com/s/3qy3gjo6r4wixbn/sample_clips.zip?dl=0`.

Results: song samples (n.d.). `https://www.dropbox.com/s/u479sggdw25nljo/song_examples.zip?dl=0`.

Ronneberger, O., Fischer, P. & Brox, T. (2015), 'U-net: Convolutional networks for biomedical image segmentation', *ArXiv* .

Schmidhuber, J. (1987), 'Evolutionary principles in self-referential learning', *On learning how to learn: The meta-meta-... hook* .

Schnieders, B., Luo, S., Palmer, G. & Tuyls, K. (2019), 'Fully convolutional one-shot object segmentation for industrial robotics', *AAMAS* .

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L. & et al (2016), 'Mastering the game of go with deep neural networks and tree search'.

Snell, J., Swersky, K. & Zemel, R. S. (2017), 'Prototypical networks for few shot learning', *NeurIPS* .

Takikawa, T., Acuna, D., Jampani, V. & Fidler, S. (2019), 'Gated-scnn: Gated shape cnns for semantic segmentation', *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* pp. 5228–5237.

Thrun, S. & Pratt, L. (1998), 'Learning to learn: Introduction and overview'.

Virtanen, T. (2003), 'Sound source separation using sparse coding with temporal continuity objective'.

Virtanen, T. (2007), 'Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria', *IEEE Transactions on Audio, Speech, and Language Processing* **15**(3), 1066–1074.

Zhao, H., Gan, C., Rouditchenko, A., Vondrick, C., McDermott, J. & Torralba, A. (2018), 'The sound of pixels', *The European Conference on Computer Vision (ECCV)* .