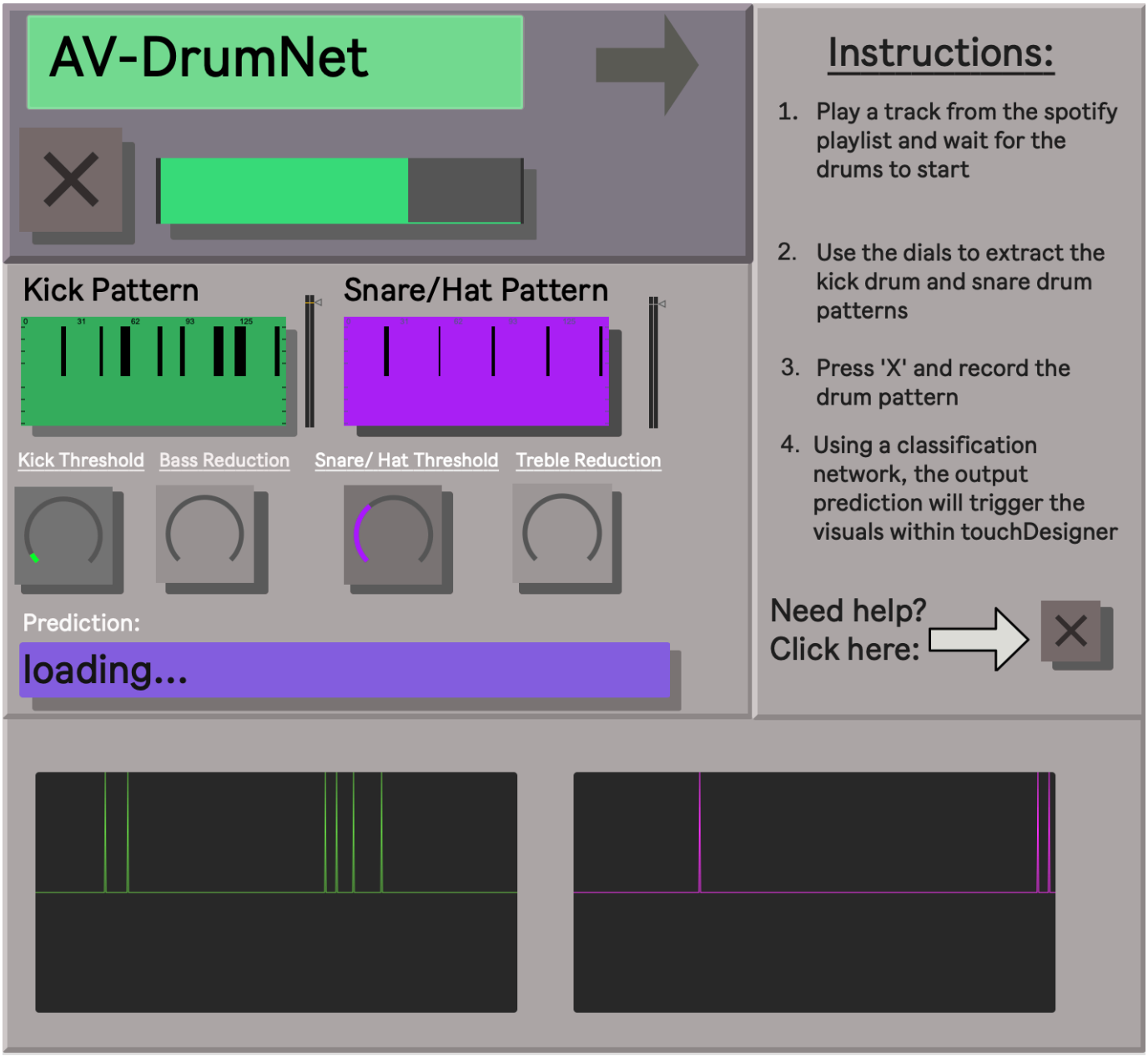
IS53043C/B/A: EMCT: COMPUTING FINAL PROJECT(2023-2024)

AV-DrumNet

By Jake Stevens

Contents


AV-DrumNet
Instructions:
1. Play a track from the spotify playlist and wait for the drums to start
2. Use the dials to extract the kick drum and snare drum patterns
3. Press 'X' and record the drum pattern
4. Using a classification network, the output prediction will trigger the visuals within touchDesigner
Need help?
Click here:
Kick Pattern
Snare/Hat Pattern
Kick Threshold
Bass Reduction
Snare/ Hat Threshold
Treble Reduction
Prediction:
loading...

Introduction

The AV-DrumNet (audio-visual drum network) is a machine learning tool that attempts to classify a selection of drum patterns in real-time, using the Fluid Corpus Manipulation Toolbox (Tremblay et al, 2022). It receives an audio signal as an input, which is streamed directly from websites such as Spotify or YouTube using the Blackhole audio driver (existential.audio, n.d.). The audio is then analysed within TouchDesigner, using a combination of audio filters and pre-built objects to detect whether a kick or snare drum is heard within the track.

The result of this is then sent to Max MSP via OSC (open sound control) and formatted so it can be used as a dataset to train a classification network (mlp.classifier~). When the network is trained and in use, the user will select a song and dial in the correct parameters, before starting a 3 second timer. This will extract the drum pattern, and use it as an input for the neural network. The output will consist of five genres: Techno, Jungle, Downtempo, Afro-House and Liquid/Drum & bass. This will then be sent back into TouchDesigner via OSC which is used to choose between 5 art pieces I have created, which attempt to visually represent the tone of the genre. Within these patches, I have attempted to demonstrate a wide range of audio-visual techniques. I have used my own experience at live raves as well as a survey to assist in the creation of these.

The project began as a tool that could assist video-jockeys (VJ'S) to control live visuals in real time, by identifying characteristics from audio tracks and reacting to them. Over time the idea evolved into developing a tool that uses machine learning to classify features of audio, which can then be mapped to chosen parameters. I am very interested in the process of a neural network attempting to process abstract concepts, such as musical genre. Through these channels of thought, the AV-DrumNet seemed an appropriate exercise in these interests.

The concept of musical genre goes far deeper than drum patterns. Other elements such as vocal style, production techniques and instrument selection are all relevant. I chose drum patterns as a way to classify these genres as it is easier to process the data in real-time. I also have previous experience creating drum machines in Max MSP. Most importantly, electronic music tends to follow certain trends regarding drum pattern formation, which gives me a solid starting point to classify these genres.

I want to establish that this is not a genre detector algorithm. Its sole focus is to extract the rhythm from selected tracks and classify them, based on drum pattern conventions. Genre detection is a complex task, often achieved through the use of deep learning models and are usually not intended to be used in real-time. My research into these patterns consists of my five years of music production experience as well as a deep interest in electronic music. In combination with this, I have been using TouchDesigner for over a year, creating visualisers that attempt to emulate the tone and feel of various tracks.

Background Research

Software

- *Op.beatitude~(www.maxobjects.com, n.d.) -*

This is an external Max object that detects BPM from an audio signal. In early development I used this object to detect the tempo of the track, so that I could incorporate it when training the network. The issue with my early prototypes is that although a clear pattern can be extracted from the 'matrixctrl' object, the speed of the track cannot. This meant that 2 tracks could contain the same drum patterns yet belong to different categories due to their tempo. I used this object as an attempt to tackle this problem.

- *Derivative (2022). TouchDesigner Documentation. [online] Derivative. Available at: https://docs.derivative.ca/Palette:audioAnalysis [Accessed 16 Apr. 2024].*

The audio analysis tool from the TouchDesigner pallete has been a key component in my project. When the audio input is processed using this component, it will output a 1 if a kick or snare is detected, which is eventually converted into a list that reflects three seconds of audio. Despite being very useful, this is also the component that has restricted my project from reaching its full capability. Due to my computers inability to open the component without crashing to a frame rate of 5fps or lower, it has been difficult to understand the logic of it.

My background research into the tool suggests that it contains python script, filters and fft analysis to identify a kick or snare from the input signal. To make this work, the user has to change threshold parameters for it to be able to detect properly. The issue with this is that the timbre of the drums change every track, meaning that the thresholds have to be frequently updated. If I were to leave the thresholds as they are it would mean less hassle for the user, yet not be a pattern recognition system as it would never properly reflect the rhythm of the drums.

This is why training was difficult to automate, as every track had to be specifically dialled in to gain an accurate reading. The snare and rhythm detection is also temperamental, as the tool often recognises a hat as a snare and vice versa. This is the reason for only extracting the kick and snare signals to be used instead of the hats as well. After many failed attempts at correctly reading the drum patterns from a track, I implemented a lowpass and highpass filter before the tool which resulted in a more accurate reading.

• *Tremblay, P.A., Green, O., Roma, G., Bradbury, J., Moore, T., Hart, J., & Harker, A. (2022)* ***The Fluid Corpus Manipulation Toolbox (v.1).*** *Zenodo. doi: /10.5281/zenodo.6834643*

This toolbox contains the objects that help process the data, as well as the networks themselves (mlp.regressor - regression, mlp.classifier - classification). In order to be trained, I have used objects such as fluid.list2buf and fluid.dataset that allow me to correctly place the data into a buffer. Fluid Corpus Manipulation have also been kind enough to post tutorials on how to use these objects correctly, which I have referenced below. My research into these components consisted mainly of these tutorials and the fluid documentation.

Youtube Tutorials

• *Classifying Sounds with a Neural Network in Max by Fluid Corpus Manipulation(Manipulation, n.d.)*

This tutorial uses the fluid.mfcc~ object to convert the timbre of an instrument to a list of 13 values. It then pairs these values with a label. When the data is fitted, the result is a simple network that can differentiate between two samples. This tutorial was important in understanding how the mlp.classifier object works and how to correctly input values from a dataset.

• *Manipulation, F.C. (n.d.). Controlling a Synth using a Neural Network in Max. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=XfNZzQPdPG0 [Accessed 15 Apr. 2024].*

Similar to the sound classification tutorial, this video explains how to use the mlp.regressor object. The network receives data from an xy pad and synthesiser parameters, in order to create a controller. Alongside the classification tutorial(referenced above), this was significant in the development of the project as it clearly explained the process of adding datapoints to a regression network.

This training portion of this tutorial is quite limited, adding few points to the dataset. By following this tutorial it meant that I understood how to transfer data between fluid objects. However, automated training and different forms of input data are not discussed in the tutorial and was something I had to experiment with, through trial and error.

• *Vanta (2020). GLSL 102 Raymarching : Beyond Copy and Paste Shadertoy - VanTa. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=h-xIuHyhQGE [Accessed 15 Apr. 2024].*

This lecture describes the processes behind ray-marching in GLSL, which I have used for the downtempo network in TouchDesigner. Through this tutorial I learnt to create 3D shapes and merge them with others. In addition to this, it provides details on how to use CHOP objects to modify values within a shader. In combination with other tools and techniques such as feedback, bloom and lens distortion, the end result of the downtempo visual is one of my personal favourites.

- *Tschepe, bileam (n.d.). Circuit Bending – TouchDesigner Tutorial 42. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=mAp_wxuuw_U [Accessed 16 Apr. 2024].*

This tutorial allowed me to create the VHS look. Using feedback techniques, it creates a flare/distortion effect that I have modified using LFO's to create a more dynamic feel.

- *Tschepe, B. (n.d.). Dynamic Texture Grids – TouchDesigner Tutorial 62. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=Eq5amV7obwg&t=1090s [Accessed 16 Apr. 2024].*

This tutorial became the starting point for the afro-house visual which was ultimately adapted in a variety of ways. It uses a reference image that is then manipulated through noise and other objects to display unique patterns.

Articles

- *Music Genre Classification Project Using Machine Learning Techniques by Raghav Agrawal* (Agrawal, 2022)-
- Velardo, V. (2020). This AI Recognises Moods in Songs and Explains How it Does it. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=doUTqWUAuDw [Accessed 15 Apr. 2024].

This article goes into the details of genre detection using machine learning. Before I focused my efforts into drum pattern recognition, I read this paper in an attempt to understand the processes behind genre and mood detection. After reading other similiar articles, I realised that deep learning models are always incorporated, as they use a spectrogram and an image classification network. This meant that real-time detection would be an extremely difficult task. I knew that deep learning would not be possible due to the complexity and the nature of the tool I was developing, so this was an important article that made me re-think how I was going to approach the project.

- *Quilez, I. (n.d.). Inigo Quilez. [online] iquilezles.org. Available at: https://iquilezles.org/articles/distfunctions/.*

This article provides a list of functions used in ray-marching to create 3d shapes. This has assisted me with the downtempo visual where I have created a vertex shader that displays an abstract shape.

Artwork

Anadol, R.(2016). Machine Hallucinations

This artwork by Refik Anadol inspired the visual for the afro-house genre. After discovering the artwork in 2021, I have been fascinated by how it translates images and adapts them into an art piece that appears as a thick liquid. The colour of the particles on screen are reflective of images sourced from the internet. In my network I have followed a similiar approach by warping instances of an image of a tribal mask. I would've loved to have taken this a step further and used a particle system, however, due to the heavy amount of processing on the CPU, I chose to stick with primarily TOP's when creating visuals patches.

Czapiga, K. (n.d.). *Cosmodernism*. [online] Cosmodernism. Available at: https://www.cosmodernism.com.

Self defined as “creative microscopy and sonic experiments”(Czapiga, 2024), Kamil Czapiga is an artist that uses liquid and microscopy to create dynamic visuals. He has been an inspiration for my “liquid” patch.

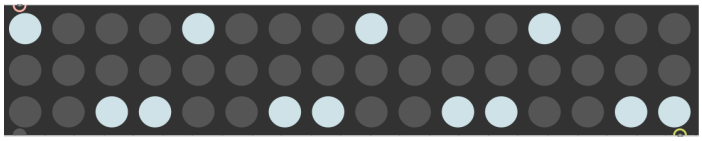
Context

Within the context of my final project, the AV-DrumNet is a drum pattern classifier that has been used as a VJ assistance tool. Despite my experience in TouchDesigner, I have not been able to demonstrate this in other assignments. I wanted to play to my strengths, hence why I used my project to trigger between multiple visual patches. However, there are many alternative uses for drum pattern classification that vary from music production to audio-visual technology.

Regression - early stages of development

During the early stages of development, my aim was to use a regression network to output values that can then be mapped onto a TouchDesigner generative system. This system would consist of a variety of TouchDesigner techniques and displayed as a projection throughout a DJ set. The values that the network was trained on consisted of results from a survey I created.

When a user records a section of the track and inputs this into the network, the result would slowly change the values from their current state to a new one (using the lag CHOP) depending on the output of the network. Therefore, the original aim for the project was more of an art piece, and became centred around visually representing a collective of opinions, using the audio input to map parameters of a generative TouchDesigner patch.



Matrixctrl object - Techno pattern

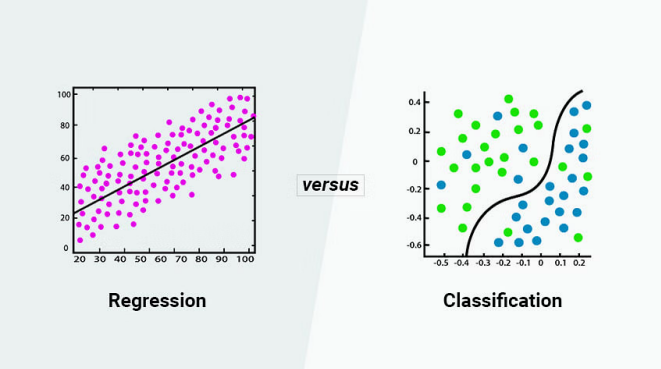
Within my “version(1)” patch (path to file included in appendix), the logic for how this would be implemented can be seen. The input for the network is a ‘matrixctrl’ object, that is used to represent the patterns in a drum sequence, in a similiar way to drum plug-ins such as the Redrum drum computer (www.reasonstudios.com, n.d.). The network is trained on a survey consisting of questions that are related to the visual representation of music genres.

These questions were answered from a range of individuals and collected within excel. I then learnt to utilise built in excel functions that calculate the mean from their answers. Following this, I assigned them to a specific line within the spreadsheet, which is then exported as a text file. Within the Max patch, a text object reads from the line containing the mean values. After this, a combination of list processing tools are used to store these values into a single list.

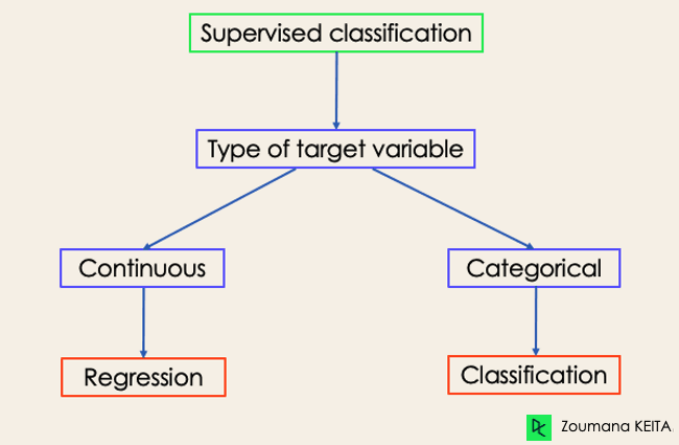
This survey list is paired with the drum patterns (another binary list). I used the op.beatitude~ object (www.maxobjects.com, n.d.) to multiply the drum pattern list by the bpm. The result of these look like this:

Drum data:(171,0,0,0,171,0,0,…….) - *drum pattern * BPM.*
Survey data:(7.1,3.2,4.8,9.2…..) - (values between 0-10)

At this point, my data collection had been been successful, but I realised the network I was using was incompatible. When training, I had 8 values from the survey paired with 6 drum patterns, for 6 genres. These 8 values remained the same for every genre, resulting in 36 drum patterns and 6 survey lists. I realised that due to an incredibly high 'fit' rate, that I was attempting to classify genres using a regression network. With this realisation, I stopped using mlp.regressor and instead began to research mlp.classifier, a classification network. By doing this, the fit rate was 150x smaller and this version of the network was more successful.



(Terra, 2022)www.simplilearn.com/regression-vs-



(Keita, 2022)

Other issues that I faced when creating this first prototype was that I knew a large part of its success would rely on the results of the survey (Results.xlsx), as well as the users ability to interpret drum patterns by ear into the matrixctrl object. When collecting the results I realised that the answers to the questions varied significantly. This meant that if the results affected a single generative network, a new genre prediction wouldn't create a dramatic enough change to be displayed visually.

Regression - sound design uses

A regression network could potentially be used within sound design, where the rhythm of the drums affect audio effect parameters. In modular synthesis, control voltage affects these parameters. In this case, the DrumNet would be used in a similiar way.

An example of this would be using the network to convert an analog signal from a piece of hardware and use it to map other parameters within a DAW. The audio signal from an analog drum

machine would run through TouchDesigner and Max MSP. The output of the network would then be used to affect other audio effects within a project. For instance, in a track, if the artist is building up to a drop using a techno-like rhythm, the tool could detect this and modulate effect parameters such as side chain compression. When the track progresses and the drums resemble a different pattern, the tool can repeat this process, manipulating other parameters. This allows the user to utilise an analog signal to modulate digital values, bridging the gap between the two technologies.

Classification - sequencing lights and visuals using a network

In live concerts, artists and performers should feel able to improvise and adapt to new musical ideas on the fly. However, in the electronic music scene, where lights and visuals have been carefully handcrafted and timestamped to specific events, improvisation is often difficult to coordinate effectively. A successful example of this would be electronic duo 'The Chemical Brothers', who's live show is widely recognised as "*a terrifying and euphoric explosion of light and sound*"(Condon, 2024). With a large team of designers and engineers, syncing visual elements to an ever-evolving set is achievable through effective communication. However, for small time performers and DJ's, the DrumNet could be used to sequence lights and visuals so that they have the freedom to play without a VJ or lighting engineer.

By training the network on drum patterns from their own songs, lights and visuals could be changed depending on the track, as it could classify a rhythm. This means the artist does not have to follow a specific setlist, resulting in a greater sense of control and freedom to experiment with new ideas. Of course it is important to mention if the DJ were to use filters or effects, the TouchDesigner audio analysis tool would need its parameters adjusted in order to correctly detect the pattern.

This is similiar to my current project, except instead of classifying genres it would be classifying songs.

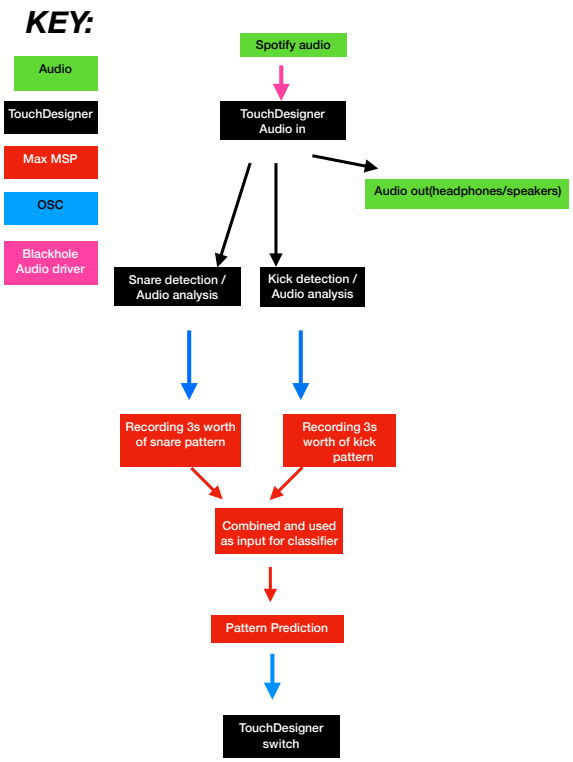
Project in depth

List of software, tools and languages:

- *Max MSP*
- *Fluid Corpus Manipulation Toolbox(Available for download in the max package manager)*
- *TouchDesigner*
- *OSC(Open sound control)*
- *Javascript (scripting.js)*
- *Blackhole audio driver*

Structure

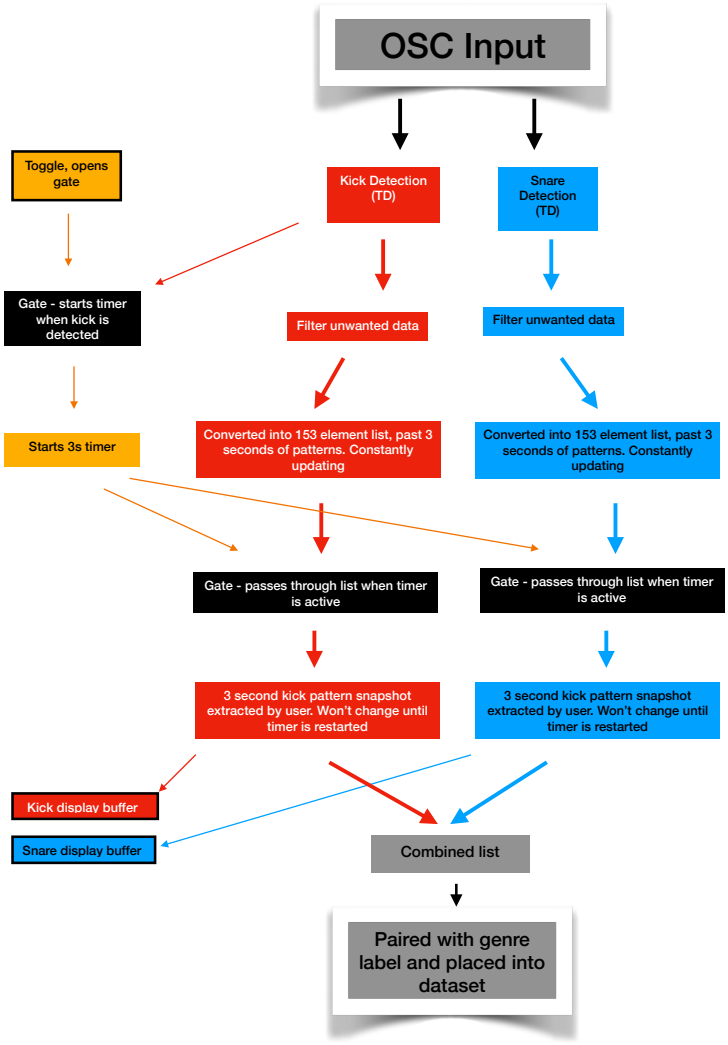
The AV-DrumNet consists of 5 major parts:



Signal flow between applications

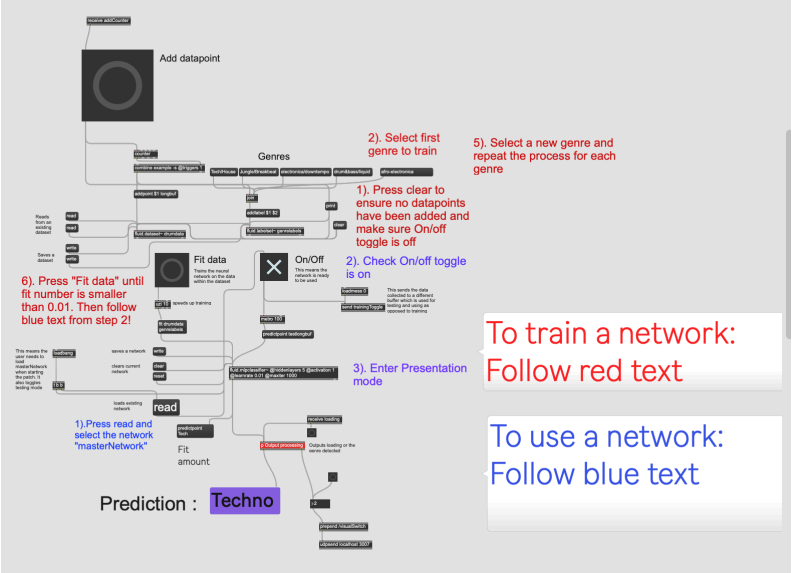
- **Extracting the kick and snare from a piece of music using the audio analysis object and blackhole audio driver**. The blackhole audio driver will split the audio signal of any apps currently running on the computer. By setting the computers' audio input and output to blackhole, the user can set up an 'audio device in' CHOP, that outputs the audio signal playing from Spotify. This signal is then sent to headphones or speaker using an 'audio device out' CHOP. After this has been set up correctly, this audio signal is sent into two audio analysis components.
- Before the kick analysis, a highpass filter is placed into the signal chain. This is to differentiate the kick drum from the bass. I have followed the same process with the snare detection as a lowpass filter isolates the snare from the hi-hats. This is not always needed but is available for the user to dial in the correct parameters until they feel that the system is correctly detecting each drum. The result is two CHOP's that output a 1 when a drum is sounded and a 0 when it is not. Each CHOP is then sent into max MSP using OSC.
- When I am training the network I am not using the audio from Spotify. Instead, I am downloading song files (path to files included in the appendix). This is so I can loop a section of the song to add patterns to the network that best reflect that genre. It also means I can avoid sections of a track that may contain silence, or drum fills. As well as this, I can test the network on the same songs by simply using a different section of the track.
- **Converting data into a three second pattern**. After the information is received from TouchDesigner into Max MSP, I've used the zl.unique object to filter out any channel names or unwanted data. The zl objects in Max are the main tools I've used to process lists in a variety of ways. After filtering out channel names, I am left with a single number that outputs 51 times a second (this due to how OSC transmits information). This number will either be 0 or 1 depending on if a hit has been detected. If there is a detection, a bang will be triggered. This activates a click~ object purely for UI purposes, as it means that I can use the live~ objects to make it easier for the users to visualise drum hits. The integer is then placed into a zl.stream object that will hold the last 153 values. 153 is the number chosen as it is 3 seconds worth of data (51x 3 = 153, multiplied by 2 as there is a list for the kick and snare pattern). These lists will be constantly updating to the last three seconds of audio.

- When the recording button is activated it will wait for the a kick drum to be detected before ‘recording the pattern’. This is a simple way of ‘quantising’ the data. It would be very difficult to train the network if the patterns were slightly skewed every recording. By waiting for the kick, the network finds it easier to identify patterns (*I will explain this in detail in the debugging section of the writeup*). Once a kick has been detected, 2 gates will open for 3 seconds. When this is over, the gates are closed and the zl.reg object will have stored a list containing values over the past 3 seconds. Each list is then sent to display buffers so the user can easily visualise the patterns. They are then joined together and placed into a single 306 element list(153 x 2) which is the input for the network.



Signal flow for pattern recording

- **Placing lists into a buffer to either be trained or used.** A switch is used to toggle training or testing mode. When training mode is activated, the list is placed into a buffer named ‘longbuf’ using the fluid.list2buf object. If I am happy for this pattern to be added as a datapoint for my network, I can select the matching label (Techno,Jungle,Liquid…..) and click the ‘add datapoint’ button. This will assign the current pattern and its matching label a unique identifier, using a counter object. Both points are then placed into data and label sets. Then, I repeat this process multiple times for each track. I train 3 tracks per genre and the result is around 400 datapoints for the final network (masterNetwork). It is important to mention that every time a new track is played I need to readjust the parameters of the audio analysis tool.



Note: training steps 3&4 are located in different parts of the patch

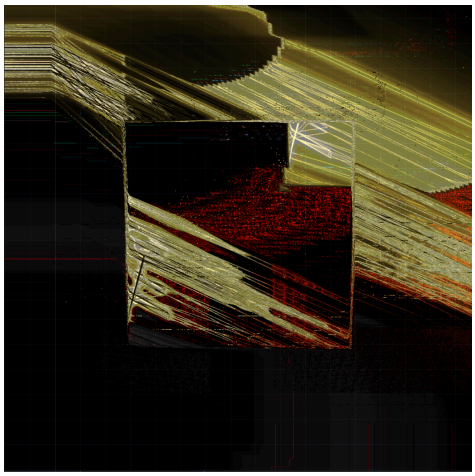
- After this, I pass the mlp.classifier the argument ‘fit data’. With assistance from the uzi object I can pass this argument hundreds of times. I usually aim for the fit value to be <0.001 before finishing training. With training is complete, I activate the user toggle and the network can now be used to detect patterns based on the training data. Inputting the pattern is the exact same process when using the network, but instead, there is no need to add points as we are no longer creating the dataset. Instead, the network will output the name of the predicted genre.
- **Sending the output of the network into touchDesigner to trigger a switch.** After I have received a prediction from the network, I then remove the word ‘predictpoint’ using the regexp object and assign each genre a number. Using OSC, this number is then sent back to TouchDesigner to control a switch TOP, which will select between 6 networks I have created(1 of which is a noise TOP when the system is loading, that represents TV static).

TouchDesigner

This section assumes basic knowledge of TouchDesigner. I have included definitions for some of the abbreviations in the glossary.

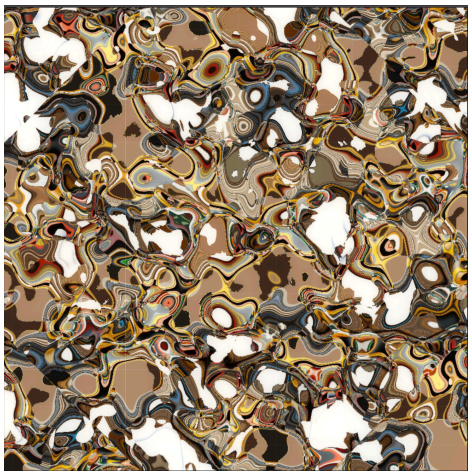
Techno:

This TouchDesigner patch begins with a SOP(3D shape) that has been distorted using the twist operator. Its’ dimensions are manipulated using noise CHOPs. When this is rendered, I use an edge TOP along with noise displacement and a limit, to create generative patterns of paint splatters and geometry. Multiple feedback loops are placed within the patch so that these patterns fade instead flash. Then, I displace the entire image with a square that creates a window-like effect. The symbolism of the square is a nod to the ‘4 on the floor’ kick pattern that occurs in techno music. After this, using multiple blur TOP’s (where the brightness is synced with the kick drum), I combine each component and am left with a glowing golden structure, displaced by a square with a generative coloured overlay.



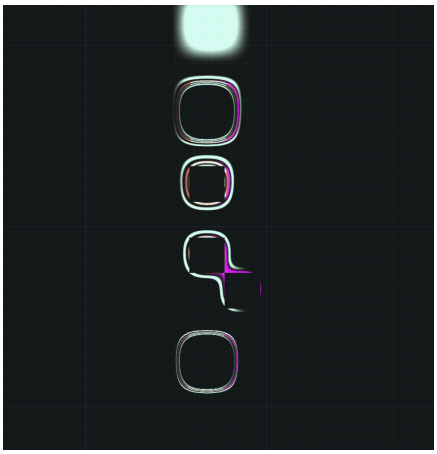
Afro-House:

With assistance from a Bileam Tschepe tutorial(tschepe, n.d.),I am using an input image combined with noise to remap an image of a tribal mask that creates a grid of textures. This is then mapped to an object which is a combination of noise and the outline of the mask, creating an image that resembles molten gold.I have taken inspiration from Refik Anadol’s artwork that uses images as particles(Anadol, n.d.).



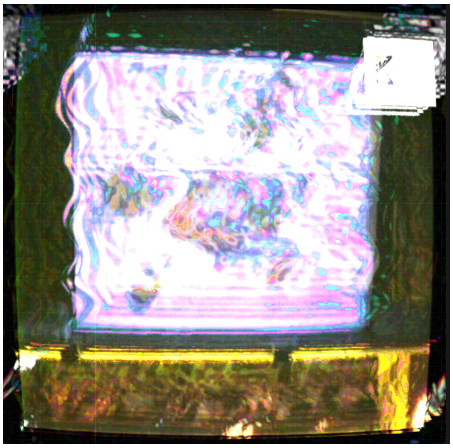
Liquid:

For this visual, I have taken a displaced image, and converted it to a 4 channel CHOP. A twisted box has then rendered using the GEO comp. The values from the displaced image(r,g,b,a) are then mapped to the translate and scale parameters within the instancing tab of the GEO comp. The result is a dark, glitchy piece of geometry that is pixelised and blurred to create a slight glow. This final image is then displaced slightly to create the effect of a moving liquid, inspired by the art of Kamil Czapiga(Czapiga, n.d.). This is also a nod to the name of the genre itself.



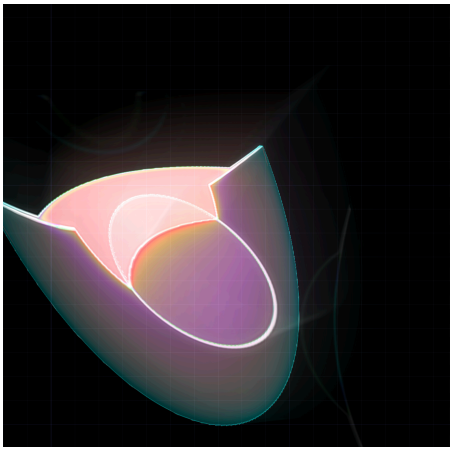
Jungle

This patch is heavily influenced by Bileam Tschepe's Circuit bending tutorial(tschepe, n.d.). When asked to create visuals for a jungle-based rave, I was instructed to keep things simplistic. After learning about the culture, I realised a lot of emphasis is placed on maintaining a rugged, urban aesthetic, as jungle music originated in underground rave scenes across the UK in the 1990's. I wanted to create something simple that clearly communicates this. I found old ministry of sound footage from YouTube and encapsulated different elements of the tutorial to create a VHS, distorted look. Finishing off with a lens distortion that emulates the look of an old TV set, with the soft round corners and a slight convex.



Downtempo

Using a GLSL TOP, I have created a vertex shader that displays a 3D object, which has been created from spheres and octahedrons. The diffuse lighting function is used with a variety of LFO's to create ambient lighting that randomly displays different faces of the shape. I have combined this with 'bloom', multiple blur TOP's (to create a glow), and the edge TOP. At the end of the signal chain I finish off the network with a radial blur, and lens distortion. The result of this is a white and red glowing shape, that morphs from its glowing edges to a solid white. For this network, I did not want there to be any audio-reactivity from the drums as I feel that the genre tends to revolve around synthesis



or vocals. In addition to this, the minimalist features in downtempo are best accompanied by a slow moving visual which I feel has been successfully achieved in this example.

Creative Process

How the project aims have changed over the process of development

There are many elements of my project that may be difficult to conceptualise for someone who is not familiar with machine learning or music production. Because of this, I spent a lot of time on the development of a user interface. I wanted to create a design that was not overwhelming and could be accessible for users of all backgrounds. I opted for a retro design, emulating the feel of an Akai MPC from the mid 2000's. As it is used in a musical context, I wanted to steer away from words such as "buffer", "classification" or "training". When I created my first prototype, containing a matrix controller, I wanted the user to input the data themselves. This was an attempt to make machine learning more accessible, demonstrating the process of data preparation and training. However, due to feedback received from users of varied understanding, I opted to hide as many of the processes as possible. This is so that the users had less questions, and could focus on using the tool to the best of their ability.

Input Processing

"The matrix controller will be confusing for users that have no experience in music production"
Patrick Hartono - personal tutor

"Using a matrix controller means the network will struggle to process tempo" - Patrick Hartono

Discovery

After receiving feedback from my personal tutor, I realised that a matrix controller may not be suitable for the end user. My original idea of adapting the matrix controller input was by creating a step sequencer like UI, which assisted the user when inputting the pattern. As a prototype, I used the op.beatitude~ object(an external max patch)to control a metro object that moved a blinking light. However, with my experience in creating sequencers in max, I was well aware that often the timing can be skewed and this would be extremely temperamental. Not to mention that the bpm from op.beatitude~ can be 1-2 values skewed, resulting in further inaccuracy.

Define

The main problem I faced was to create a way that I could incorporate tempo and drum pattern into a single input for the network. Assisted by my personal tutor, we came up with a way to use the audio analysis tool in TouchDesigner, which would output bangs when a drum is detected. This would then be sent into Max MSP using OSC and I would have to experiment with thresholds to find a suitable way to input the data.

Develop

I began by formatting the data in a variety of different ways. This included converting the bangs into a signal before recording them into a buffer. I also attempted to create multi-channel buffers, which contained separate lists for the kick and snare(this will be explained in further detail in the debugging portion of the writeup).

Deliver

After creating a suitable way to input the data to train the network, I began training with a variety of different genres, all with their own drum conventions. These early networks can be found in the past logs folder in my submission. Other ways that I experimented with training included the amount of datapoints, the sections of tracks, as well as a lot of music listening, in order to find songs that best represented drum patterns within that genre. Ultimately, I settled on 5 different labels(genres) that have been trained on 400-500 datapoints(masterNetwork).

After developing this network, I attempted to create another large network(WithArtifacts), adding my own 'artifacts'. This meant I added my own mistakes in threshold parameters, or recorded patterns from parts of the song that do not reflect the genre as much. I believed that this would be more accurate as it would take into account the slight mistakes made by the user. However, the result was less accurate and was not used. Past networks can be found in the past logs folder, that vary from genre classification to song classification, all of which occurred during this phase of testing.

User Interface

"I am unsure what the dials do"(User1,2024)

"It is difficult to see when a hit has been detected as sometimes it is detected for a long amount of time"(User2, 2024)

"If I have not heard a genre before, how do I know what the pattern should be like?"(User3, 2024)

Discovery

After finishing the final prototype I began to ask for feedback from users that would not have seen similiar technology before. To achieve my aims mentioned previously, I wanted this to be a tool accessible for people unfamiliar with machine learning or music production. Most of the feedback I received when conducting this research was that they were unsure how to dial in the correct parameters.

Define

I knew that I could not simplify the controls any further as the current controls were vital. Instead, I had to find a way to better communicate how the system is used. A user suggested the idea of displaying examples of existing patterns, to demonstrate drum conventions within a genre. By understanding these, it meant that users could look for patterns within the music. Potentially resulting in better results when adjusting threshold parameters.

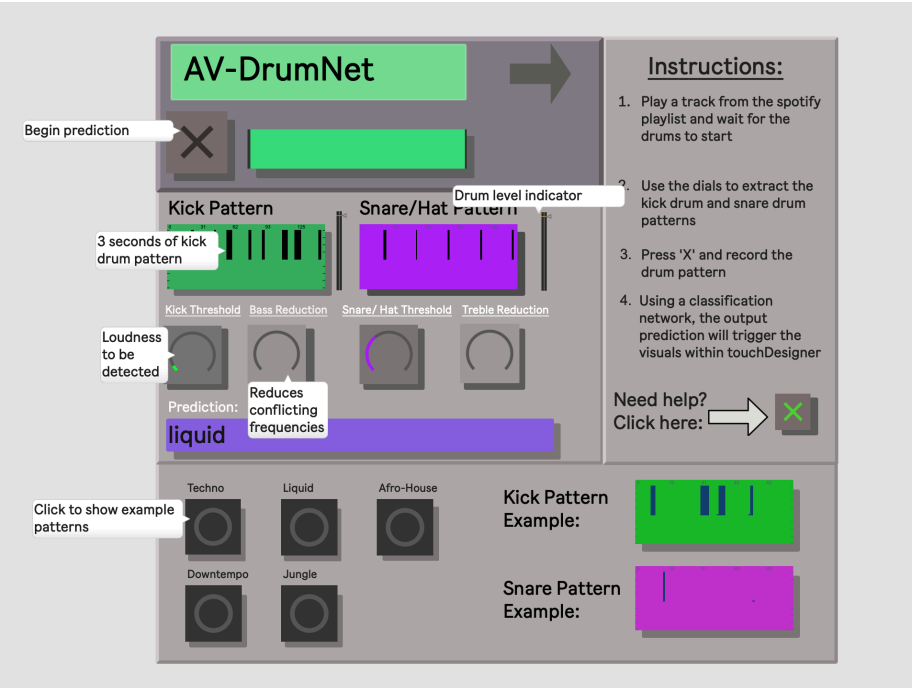
Develop

Another way to make the project more user friendly was to develop a javascript file that is triggered using a help button. The features I wanted to include were:

- Labels being added to each element of the tool, explaining how it works.
- Providing examples of drum patterns that commonly occur in specific genres.

I also began to experiment with different layouts and ways of displaying the data. I wanted the user to easily see what they are detecting using objects like live.scope~ and live.gain~. To achieve this, I converted the TD bangs into a signal using click~ which resulted in a more engaging and informative interface.

Deliver



After creating a more appealing UI, I began work on the javascript file that dynamically adds and removes elements of the interface. When the script is toggled, it registers a number of components that exist within the patch by using the function ‘this.patcher.getnamed(object scripting name)’, and assign a variable to this. It will then add and remove certain elements from presentation mode, and place them at suitable positions in the presentation window. The design of the UI encourages the user to toggle this button through the use of panel objects in the shape of an arrow.

When the user loads the patch, the largest font is the title of the project, followed by an arrow that diverts the users eyes to the instructions. This is a simple piece of text that instructs the user what to do. At the bottom of the instructions, another arrow is created that directs the user to the help button. I am very happy with this addition as I feel it is a vital tool that almost all plug-ins contain. As a result, it gives the user a greater understanding of the software. I would've liked to have added a song example file, so that the user could listen to a small portion of a track from each genre. However, due to the audio driver routing and the disconnect between Max MSP and TouchDesigner I felt that this may confuse the users more than it would assist them.

Accuracy

"The network has detected the wrong genre"(User3,2024)

The biggest flaw with my project is also the most obvious. Despite training the network many times in multiple ways, the system is not 100% accurate. I believe that this is partly due to the audio analysis tool and its inconsistency. If there is a 'wrong' prediction, it's most likely because the user has not dialled in the parameters accurately enough. However I understand that it is my responsibility to engineer the tool in a way that gives users the freedom to make small mistakes.

It took many hours to manually add the data as carefully and as accurately as possible. Eventually, I had to realise that I was spending too much time obsessing over the accuracy of the network, which is hard to measure. I believe it is also normal for inaccuracies to occur. Drum pattern conventions exist but they are guidelines. Every artist has external influences from a variety of genres, and this often makes it harder for listeners to categorise their music. The AV-DrumNet has the same struggle. This is because it is virtually impossible to 'perfectly' categorise drum patterns, as genre is subjective.

Debugging

Training the network

To my surprise, an important variable affecting the overall success of training the network was not just the amount of training data, the data flow or the programming aspects. A major portion of my time was spent finding tracks that roughly follow the same drum conventions. There are examples on the internet for traditional drum patterns in certain styles, although here is what I found after countless hours of listening:

Techno:

Tempo: 120-150bpm

Kick drum: Beats 1,2,3,4 in crotchets.

Hat/snare: On the offbeats in between each kick

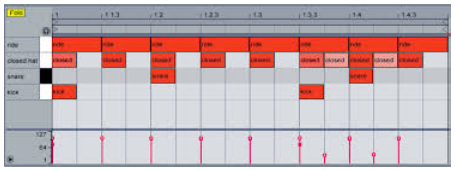
It is important to mention that as techno very rarely contains a snare drum, when training the network I am using the hat as the snare.

Liquid/Drum & Bass:

Tempo: 170-175 bpm

Kick drum: Beat 1 and beat 3.5

Snare drum: Beat 2 and beat 4



Liquid drum and bass pattern(Attack Magazine, 2012)

Jungle:

Tempo: 150-175bpm

Kick drum: 1-2 hits every bar, usually at the beginning of the bar

Snare drum: High amount of snare hits, often rolled with an accent on offbeats.

Downtempo:

Tempo: 90-115bpm

Kick drum: Beat 1 and Beat 3

Snare drum: Beat 2 and 4

Afro-House

Tempo: 115-130bpm

Kick drum: Beats 1,2,3,4

Snare drum: Triplets and syncopated offbeats

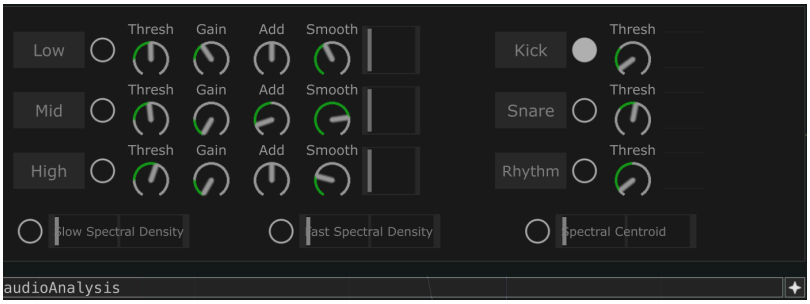
I found that my song selection was very important during training and testing. My project is drum pattern recognition, so when choosing a track, I had to ignore every element other than the drums. My display buffers helped me with this, as I could visualise the patterns when extracting new ones. This was very useful in training as I could determine whether the song was going to be effective in reinforcing these conventions or whether it was going to throw off the network. After training the first few networks, I realised that the system was not working. This is because every pattern was too different, as I was training too many portions of a song, which would sometimes

include silence or drum fills. To counter this I began importing the audio files into Touchdesigner and looping a single section of the track.

In addition to this, I included a feature in my recording system that meant recording would only start after a kick drum is detected. This meant that for a genre such as techno for example, there were only 4 possible variations of the same loop (due to the kick pattern). By starting recording on a kick, it meant that the 'phase' of the pattern was quantised. This greatly improved my results and the network was significantly more accurate.

Within my project files I have a diary I kept during testing. This is separate from my weekly diary/ blog and instead goes into detail about my findings during every training phase of my network.

Audio Analysis



Although being a key feature of my project, it is clear that there is a lot of work to be done with the audio analysis tool. It uses filters and fft's to detect a kick, snare or rhythm hit from a piece of music. The tool has difficulty differentiating from the bass, hi hats, synthesisers and even vocals. The result of this is that it is quite difficult to use for certain tracks. At times, some tracks simply cannot be used, despite adding my own filters and spending time changing threshold parameters by small amounts. As this is a pre-made tool from Derivative, there is not much I can do to change these features. If I were to take this project further I would attempt to create my own system for kick and snare detection. Although, with the resources that Derivative have, I am confident that they are not incapable, it is simply an incredibly difficult task.

A note to user: As stated before, if the network is not detecting the correct genre, make sure that the threshold parameters are dialled in correctly as this is the most common reason for a false reading.

Multichannel buffers

During development, a lot of time was spent experimenting with the best ways to input data into the network. In my first prototype, I used a matrix controller and converted it to a list. As I continued development, I wanted to find a way to place the kick and snare into different channels of a buffer. I believed that this would give the network a better chance at differentiating patterns. This led me to experiment with many different forms of data input.

I needed to tackle the challenge of inputting the data in two parts. In order to do this I needed to create a multichannel buffer, one for the kick and another for the snare.

I began by converting the bangs from the audio analysis tool into a continuous signal. I could then record the signal into the buffer using the record~ object as a stereo input, with each channel representing the kick or snare pattern. I was successful in achieving this, and it made the recording process easier as I did not have to create a makeshift recording system with zl.stream.

However, when using signal, every attempt at fitting the data would cause the computer to either crash or take over 5 minutes to succeed, as there were +192,000 elements in each datapoint (which made multiple fittings a very long process). This resulted in me attempting to 'downgrade' the signal so that the shape remained the same, but it contained less samples. I tried to lower the sample rate in TouchDesigner but this meant that the patterns were inaccurate. I also attempted to use the degrade~ object in Max MSP, but this made no difference. Because of these problems I avoided converting the drum detection to signal.

Eventually, I managed to input the data from zl.stream into a multi-channel buffer using specific messages. Despite this, it seemed that the network was only registering one channel. I did some research and found no information on multichannel buffers and mlp.classifier. Instead, I decided to revert back to my original idea and input the data as a single list, creating my own metro-based system to record the data.

Conclusion

Successes

I am very confident in the overall success of my project. I tackled many problems through development and created unique ways to transport and format data to train a neural network. Comparing my project to online tutorials such as "*Controlling a synth using a neural network in max (Manipulation, n.d.)*", I believe that I have pushed boundaries and taken a unique approach in my use of the 'Fluid corpus manipulation toolkit'. This is due to the amount of data I used to train the network, and the many techniques I've used to format it correctly. In addition to this, I have demonstrated how to seamlessly send data back and forth between various applications, in order to make the best use of all the tools that Max MSP and TouchDesigner have to offer.

I want to clearly state that this is a drum pattern recognition system and is not to be judged on its ability to predict genre, which through my research I appreciate is an incredibly complex task. From the beginning I set out to create a tool that reacts to data in real time, and does not incorporate deep learning. I stand by my belief that drum pattern recognition is one of the most effective ways to predict a genre in real-time.

Goals, were they properly scoped?

I believe that my original goals were properly scoped, as any further development of this project would most likely include creating my own audio analysis tool. This would've been an entire final project in itself due to the complexity of fft analysis and bespoke filter design which is more suitably scoped for a masters or PhD student. I feel that it is fair to make this claim due to the

existing flaws of the audio analysis tool used in the project, a tool which has been designed by multi-million dollar company ‘Derivative’.

My goals for this project did not significantly change throughout, as I am lucky to have not encountered any issues that could not be fixed. Although, my biggest change in direction was when I opted to use an audio signal to extract the pattern instead of the matrixctrl object. This greatly improved the project and I would like to credit my personal tutor for the suggestion.

Testing

The training and testing of the project was as big of a task as the software development. I learnt through this process the importance of correct data selection, making sure I am consistent in the way that I loop and test tracks. I was often met with genres where the conventions simply weren’t strong enough to be used, or that they conflicted with each other. It has resulted in a newfound respect for those that correctly format data for large models with millions of datapoints. I throughly enjoyed the process of training a network in different ways, experimenting with different tracks, genres and testing techniques. The most important thing I have learnt through the training process is the importance of consistency, so as to not create a bias for a certain label (genre).

Creative process

The TouchDesigner visuals for this project have been mostly successful. I’m happy as I feel that they greatly represent my personal representation of the 5 genres. If I was examining the scope of the project as a whole, the neural network and max patch is where 80% of the work has been conducted. Although a nice addition to the project, I am happy that I have spent most of my time focusing on making the system as accurate and as user-friendly as possible. The visuals displayed are mainly an example of how the tool can be used.

That being said, I am not as satisfied with the look of the afro-house visual as I believe it is probably the least visually appealing. This makes sense, as this was a genre I discovered during the project, as opposed to the others, which I was previously familiar with. Because of this, I would’ve liked to do more research into afro-house and the culture that surrounds it.

Problem solving

Although my problem-solving skills were successful throughout the project, I believe that at times I underestimated my abilities. When beginning the project, I knew about some potential problems with the audio analysis tool, which is why I originally opted for the matrix controller. I was partly correct as I did still have issues, however I underestimated my own ability to adapt to these problems and instead began a project which was too simple.

At times I also became transfixed with problems such as creating multi-channel buffers. It is important to explore these ideas, however I feel that I wasted time exploring avenues that were simply not going to work. I should have known this beforehand through more extensive research

and planning. This would've meant that I could have devoted more time into the development of the visual aspects, or training more networks.

Future development

In future, I would love to develop this project within a music production context by using the system to control parameters within a DAW. This could be achieved by adapting the AV-DrumNet into a 'max for live' plug-in. To do this, I would instead use a regression network, so that the drum pattern can influence a number of values. The user would then map these parameters however they decide.

For artists that enjoy modular synthesis but also like the process of sampling, this tool would create an interaction between the two techniques. By using the AV-DrumNet, it could bridge the gap between the synthesised and the sampled, thus creating a new way to utilise audio features and apply them to music production techniques.

References

Anadol, R. (n.d.). Machine Hallucinations — Nature Dreams. [online] Refik Anadol. Available at: https://refikanadol.com/works/machine-hallucinations-nature-dreams/.

Attack Magazine. (2012). Raw Drum & Bass. [online] Available at: https://www.attackmagazine.com/technique/beat-dissected/raw-drum-bass/.

Condon, D. (2024). *The Chemical Brothers' live show is a terrifying and euphoric explosion of light and sound.* [online] Double J. Available at: https://www.abc.net.au/listen/doublej/music-reads/features/chemical-brothers-brisbane-australia-live-review-2024/103523624# [Accessed 15 Apr. 2024].

Czapiga, K. (n.d.). Cosmodernism. [online] Cosmodernism. Available at: https://www.cosmodernism.com.

Derivative (2022). *TouchDesigner Documentation*. [online] Derivative. Available at: https://docs.derivative.ca/Palette:audioAnalysis [Accessed 16 Apr. 2024].

existential.audio. (n.d.). *BlackHole: Route Audio Between Apps*. [online] Available at: https://existential.audio/blackhole/.

Highmore, B. (2011). *Ordinary lives : studies in the everyday*. London: Routledge.

Keita, Z. (2022). Classification in Machine Learning: A Guide for Beginners. [online] Datacamp. Available at: https://www.datacamp.com/blog/classification-machine-learning.

Manipulation, F.C. (n.d.). *Controlling a Synth using a Neural Network in Max.* [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=XfNZzQPdPG0 [Accessed 15 Apr. 2024].

Quilez, I. (n.d.). *Inigo Quilez.* [online] iquilezles.org. Available at: https://iquilezles.org/articles/distfunctions/.

Terra, J. (2022). Regression vs. Classification in Machine Learning for Beginners | Simplilearn. [online] Simplilearn.com. Available at: https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article.

Tremblay, P.A., Green, O., Roma, G., Bradbury, J., Moore, T., Hart, J., & Harker, A. (2022) **The Fluid Corpus Manipulation Toolbox (v.1).** Zenodo. doi: /10.5281/zenodo.6834643

tschepe, B. (n.d.). *Dynamic Texture Grids – TouchDesigner Tutorial 62.* [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=Eq5amV7obwg&t=1090s [Accessed 16 Apr. 2024].

tschepe, bileam (n.d.). *Circuit Bending – TouchDesigner Tutorial 42.* [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=mAp_wxuuw_U [Accessed 16 Apr. 2024].

Vanta (2020). *GLSL 102 Raymarching : Beyond Copy and Paste Shadertoy - VanTa.* [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=h-xIuHyhQGE [Accessed 15 Apr. 2024].

Velardo, V. (2020). *This AI Recognises Moods in Songs and Explains How it Does it. [*online] www.youtube.com. Available at: https://www.youtube.com/watch?v=doUTqWUAuDw [Accessed 15 Apr. 2024].

www.reasonstudios.com. (n.d.). *Reason Studios.* [online] Available at: https://www.reasonstudios.com/devices/redrum [Accessed 20 Apr. 2024].

Appendix

Audio training files = Documentation/Training audio files

Feedback Diary = Documentation/Diary/Feedback Diary

Github Link(Diary/Blog Entries) - https://github.com/jaakestevens/AVdrumnet

Iterations of previous max projects = Documentation/Past logs/Max Prototypes

Spotify playlist = https://open.spotify.com/playlist/1hSD97amRTvoevrSVNNFhy?si=53fed56353124c98

Survey Results(from prototype 1) = Documentation/Past logs/Max prototypes/Version(1)/Results.txt.csv

Trained Networks = Documentation/Past logs/Networks

Training Diary = Documentation/Diary/Training Diary

Version(1) = Documentation/ Past logs/ Max prototypes/Version(1)

Video Demo = Documentation/Video Demo

Glossary

Buffer - A container that stores data.

CHOP - A TouchDesigner family of operators that assist with audio processing.

DAW - Digital audio workstation such as logic or Ableton.

GEO Comp - Renders 3d geometry.

GLSL TOP - Allows the user to create shaders using GLSL.

LFO - Low frequency oscillator.

OSC - A way of transmitting data between applications.

Raymarching - A technique used to create 3d objects from code.

SOP - A TouchDesigner family of operators that assist with 3d processing.

TOP - A TouchDesigner family of operators that assist with 2d processing.

UI - User interface.